
university of groningen
faculty of science and engineering
computing science
sc@RUG 2021 proceedings
18th SC@RUG
2020-2021
Rein Smedinga, Michael Biehl (editors)
rug.nl/research/bernoulli

SC@RUG 2021 proceedings

Rein Smedinga
Michael Biehl
editors

2021
Groningen

ISBN (e-pub): 978-94-034-2926-7
Publisher: Bibliotheek der R.U.
Title: 18th SC@RUG proceedings 2020-2021
Computing Science, University of Groningen
NUR-code: 980

About SC@RUG 2021

Introduction

SC@RUG (or student colloquium in full) is a course that master students in computing science follow in the first year of their master study at the University of Groningen.

SC@RUG was organized as a conference for the 18th time in the academic year 2020-2021. Students wrote a paper, participated in the review process and gave a presentation. Due to the corona virus the conference itself was done completely online.

The organizers Rein Smedinga and Michael Biehl would like to thank all colleagues who cooperated in this SC@RUG by suggesting sets of papers to be used by the students and by being expert reviewers during the review process. They also would like to thank Dick Toering for giving additional lectures and workshops on presentation techniques and speech skills.

Organizational matters

SC@RUG 2021 was organized as follows:
Students were expected to work in teams of two. The student teams could choose between different sets of papers, that were made available through the digital learning environment of the university, *Nestor*. Each set of papers consisted of about three papers about the same subject (within Computing Science). Some sets of papers contained conflicting opinions. Students were instructed to write a survey paper about the given subject including the different approaches discussed in the papers. They should compare the theory in each of the papers in the set and draw their own conclusions, potentially based on additional research of their own.

After submission of the papers, each student was assigned one paper to review using a standard review form. The staff member who had provided the set of papers was also asked to fill in such a form. Thus, each paper was reviewed three times (twice by peer reviewers and once by the expert reviewer). Each review form was made available to the authors through *Nestor*.

All papers could be rewritten and resubmitted, also taking into account the comments and suggestions from the reviews. After resubmission each reviewer was asked to re-review the same paper and to conclude whether the paper had improved. Re-reviewers could accept or reject a paper. All accepted papers[1] can be found in these proceedings.

In his lecture about communication in science, Rein Smedinga explained how researchers communicate their findings during conferences by delivering a compelling storyline supported with cleverly designed graphics. Lectures on how to write a paper, on scientific integrity and on the review process were given by Michael Biehl

Dick Toering gave tutorials about presentation techniques and speech skills.

Students were asked to give a short presentation halfway through the period. The aim of this so-called two-minute madness was to advertise the full presentation and at the same time offer the speakers the opportunity to practice speaking in front of an audience.Dick Toering, Rein Smedinga and Micheal Biehl were present during these presentations.

Dick Toering gave tutorials in small groups to further practice presentation skills.

The final online conference was organized by the students themselves (from each author-pair, one was selected to be part of the organization and the other doing the chairing of one of the presentations). Students organized the conference using zoom and added online gaming and other social events during the breaks. They also found a keynote speaker, **David Smits** from **IBM** who spoke about *Blockchain for Enterprises*. The organizing students also created a website for this years conference, to be found on **https://www.studentcolloquium.nl/2021/**

The overall coordination and administration was taken care of by Rein Smedinga, who also served as the main manager of *Nestor*.

Students were graded on the writing process, the review process and the 2 minute madness presentation, the presentation during the conference and on their contribution in the organization of this conference.

For the grading of the 2 minute mandess presentations we used the assessments from the audience and calculated the average of these. For the presentations during the conference we also used the assessments of the audiencee (for 50%) and the assessments of Dick Toering, Michael Biehl and Rein Smedinga (also for 50%).

The gradings of the draft and final paper were weighted marks of the review of the corresponding staff member (50%) and the two students reviews (25% each).

The complete conference was also recorded and this recording was published on *Nestor* for self reflection.

The best 2 miniute madness presentation, the best conference proesentation and the best paper were awarded with a voucher and mentioned in the hall of fame.

Website

Since 2013, there is a website for the conference, see **www.studentcolloquium.nl**.

[1] this year, all papers were accepted

Thanks

We could not have achieved the ambitious goals of this course without the invaluable help of the following expert reviewers:

- Lorenzo Amabili
- George Azzopardi
- Kerstin Bunte
- Majid Lotfian Delouee
- Mostafa Hadadan
- Bas van der Heuvel
- Boris Koldehofe
- Jiri Kosinka
- Michel Medema
- Fadi Mohson
- Saad Saleh
- Estefania Talavera Martinez
- Fatih Turkmen
- Maria Leyva Valina

and all other staff members who provided topics and sets of papers.

Also, the organizers would like to thank the *Graduate school of Science and Engineering* for making it possible to publish these proceedings and sponsoring the awards for best presentations and best paper for this conference.

Rein Smedinga
Michael Biehl

Since the tenth SC@RUG in 2013 we added a new element: the awards for best presentation, best paper and best 2 minute madness.

Best 2 minute madness presentation awards

2021
Niels Bügel and Albert Dijkstra
Mining User Reviews to Determine App Security
2020
Andris Jakubovskis and Hindrik Stegenga
Comparing Reference Architectures for IoT
and
Filipe R. Capela and Antil P. Mathew
An Analysis on Code Smell Detection Tools and Technical Debt
2019
Kareem Al-Saudi and Frank te Nijenhuis
Deep learning for fracture detection in the cervical spine
2018
Marc Babtist and Sebastian Wehkamp
Face Recognition from Low Resolution Images: A Comparative Study
2017
Stephanie Arevalo Arboleda and Ankita Dewan
Unveiling storytelling and visualization of data
2016
Michel Medema and Thomas Hoeksema
Implementing Human-Centered Design in Resource Management Systems
2015
Diederik Greveling and Michael LeKander
Comparing adaptive gradient descent learning rate methods
2014
Arjen Zijlstra and Marc Holterman
Tracking communities in dynamic social networks
2013
Robert Witte and Christiaan Arnoldus
Heterogeneous CPU-GPU task scheduling

Best presentation awards

2021 Niels Bügel and Albert Dijkstra
Mining User Reviews to Determine App Security
2020
none, because of corona virus measures no presentations were given
2019
Sjors Mallon and Niels Meima
Dynamic Updates in Distributed Data Pipelines
2018
Tinco Boekestijn and Roel Visser
A comparison of vision-based biometric analysis methods
2017
Siebert Looije and Jos van de Wolfshaar
Stochastic Gradient Optimization: Adam and Eve
2016
Sebastiaan van Loon and Jelle van Wezel
A Comparison of Two Methods for Accumulating Distance Metrics Used in Distance Based Classifiers
and
Michel Medema and Thomas Hoeksema
Providing Guidelines for Human-Centred Design in Resource Management Systems
2015
Diederik Greveling and Michael LeKander
Comparing adaptive gradient descent learning rate methods
and
Johannes Kruiger and Maarten Terpstra
Hooking up forces to produce aesthetically pleasing graph layouts
2014
Diederik Lemkes and Laurence de Jong
Pyschopathology network analysis
2013
Jelle Nauta and Sander Feringa
Image inpainting

Best paper awards

2021
Ethan Waterink and Stefan Evanghelides
A Review of Image Vectorisation Techniques
2020
Anil P. Mathew and Filipe A.R. Capela
An Analysis on Code Smell Detection Tools
and
Thijs Havinga and Rishabh Sawhney
An Analysis of Neural Network Pruning in Relation to the Lottery Ticket Hypothesis
2019
Wesley Seubring and Derrick Timmerman
A different approach to the selection of an optimal hyperparameter optimisation method
2018
Erik Bijl and Emilio Oldenziel
A comparison of ensemble methods: AdaBoost and random forests
2017
Michiel Straat and Jorrit Oosterhof
Segmentation of blood vessels in retinal fundus images
2016
Ynte Tijsma and Jeroen Brandsma
A Comparison of Context-Aware Power Management Systems
2015
Jasper de Boer and Mathieu Kalksma
Choosing between optical flow algorithms for UAV position change measurement
2014
Lukas de Boer and Jan Veldthuis
A review of seamless image cloning techniques
2013
Harm de Vries and Herbert Kruitbosch
Verification of SAX assumption: time series values are distributed normally

Contents

Comparing Different Approaches to Parallelizing Constraint Satisfaction Problems

Radu Catarambol and Richard Westerhof

Abstract— Parallelization is a matter of great interest in the research area of constraint programming. This task can be addressed by parallelizing the filtering algorithms or by parallelizing the search process. This paper aims to gather some popular approaches for accomplishing the latter and compare them. To this end, we analyse three different parallelization approaches and their respective research papers: the embarrassingly parallel search, the confidence-based work stealing and the mixed static and dynamic partitioning work stealing. We compare the approaches on performance and efficiency, based on the results presented in the aforementioned papers, more specifically on the speedup factor of each approach, since the experiments were originally conducted on different hardware and with different problem sets. Our comparison yields the embarrassingly parallel search as the best-performing approach, offering a speedup of 21.3 and 13.8 with OR-Tools and Gecode, respectively, on a 40 core machine. We conclude with a proposition for future research by combining the approaches presented throughout the paper, as we think their combination can lead to a new, even better-performing method.

Index Terms—Constraint satisfaction problems, Constraint programming, Search trees, Parallelization, Work stealing.

1 Introduction

In the field of constraint programming, concerned with solving combinatorial optimization problems, parallelization represents a topic of great interest. Examples of a constraint satisfaction problem could be solving a sudoku puzzle, which has the simple constraints that a number must be unique in the row, column, and block that it is in, or scheduling appointments or meetings, which has the constraint that appointments cannot overlap. From the current state of the art, two main sub-fields for parallelizing a constraint satisfaction problem solver emerge: parallelizing the filtering algorithms or parallelizing the search process.

The goal of this paper is to compare the performance and the efficiency of new methods of parallelizing the search process — mixing static and dynamic partitioning and confidence-based work stealing — with the other methods previously mentioned. Towards accomplishing this goal, the results and the comparison process presented by Hamadi et al. [7] will be used. Thus, by carrying out this research, the authors expect to contribute to the current state-of-the-art by providing an updated perspective on the performance of the currently popular parallelization methods for constraint satisfaction problems.

In this paper, we will dive deeper into the state of the art by investigating existing methods in section 2. We will explain how we compared these approaches and showcase the results of each approach in section 3, after which we will discuss these results in section 4. Finally we will conclude the research and this paper in section 5, and also propose some ideas for potential future work in section 6.

2 State of the art

There are multiple ways in which a constraint satisfaction problem can be parallelized. In this section we will go over some of these methods.

A general condition that every approach discussed in this paper relies on is that the resolution time, the time that it takes to solve a problem, for all individual sub-problems must be equivalent to the total resolution time for the complete problem. If this is not the case, the speedup numbers will not be as indicative of the efficiency of the approach, and may even lead to situations where almost no speedup can be achieved.

- *Radu Catarambol is with the University of Groningen, E-mail: r.catarambol@student.rug.nl.*
- *Richard Westerhof is with the University of Groningen, E-mail: r.s.westerhof.2@student.rug.nl.*

2.1 Work stealing

Distributing search trees over a set of workers for parallelization is not a recent problem. An old but still relevant method to attempt to distribute a tree over a set of workers is work stealing, going as far back as a paper published by Burton et al. in 1981 [5]. In this method, instead of trying to split the tree in an even way from the start of the search, workers are allowed to steal work from other workers when they have finished their own work, to prevent them from staying idle while a few workers are left solving the remaining problems.

2.1.1 Downsides

This method remains as a simple way to distribute the work quite evenly. However, it does have some downsides. The first downside is the fact that the communication between workers to exchange problems between each other takes time, which could have been spent on actually solving the problem if this communication had not been needed. The second downside is also related to communication, because near the end of a search, a large number of workers will become idle since they finished their work and there isn't much work left to steal either. However, this means that the communication between workers will skyrocket, as all the idle workers will try to steal from the same few workers that are still busy. This causes a large amount of overhead, which may actually slow down the search. Especially this last problem was already recognized and addressed by Burton et al. [5], in their case by allowing each node in the tree to be stolen only once.

2.2 Embarassingly parallel search

The paper by Malapert et al. [9] proposes the embarrassingly parallel search (EPS), a new method for achieving parallelization of the search process. According to the authors "a computation that can be divided into completely independent parts, each of which can be executed on a separate process(or), is called embarrassingly parallel" and "an embarrassingly parallel computation requires none or very little communication". It functions by splitting the initial problem into a large number of sub-problems which will be successively and dynamically distributed among the idle workers. The main idea behind their approach is to decompose the initial problem into a set of sub-problems, such that their total resolution time can be shared in an equivalent manner by the workers, rather than splitting the initial problem into a set of equally difficult problems.

The EPS method is presented as an alternative for the work stealing method [9], which is based on the similar idea that a large number of small sub-problems can be distributed evenly more easily than a hand-

ful of larger sub-problems, though the way this distribution is achieved differs between these two approaches. Work stealing allows the work to be split dynamically, whereas the EPS method creates all of its sub-problems at the start.

Malapert et al. describe three main stages in their embarrassingly parallel search algorithm. The first stage is known as the *task definition* stage, in which all of the sub-problems are generated. The second stage is known as the *task assignment* stage, in which the tasks are divided over all of the workers. The third and final stage is known as the *task result gathering* stage, in which the results from all workers are combined into a final result, which will be the solution to the original problem.

2.2.1 Different ways to perform task assignment

Generally, task assignment can be done either statically or dynamically. Static task assignment means each worker is assigned a set of tasks before execution, whereas dynamic task assignment means that workers take a new task from a communal *work pool*. In the context of EPS, a work pool was chosen as the way to distribute the work. This was done in an attempt to minimize the communication that needs to occur, since workers only need to query one source for new work instead of all other workers. This should also make the approach more scalable on a large number of workers.

According to the authors, their approach is based on the fact that "the active time of all the workers may be well balanced even if the resolution time of each subproblem is not well balanced". What they mean by this is that a problem can be split into several sub-problems that are not equal in resolution time, but they can be grouped into groups that have a total resolution time that is well balanced. This grouping becomes easier when there are significantly more problems to solve than workers, which is why the number of sub-problems is often specified as a multiple of the number of workers.

It is important to note that the resolution time cannot be known before the task is executed, which is why this approach uses dynamic task assignment. This will ensure that all workers will have to wait at most the time it takes to solve the longest sub-problem when they are waiting for the last worker to finish, and since the sub-problems are intentionally quite small, this should not take long.

2.2.2 Different ways to generate sub-problems

The authors present three ways to generate sub-problems, each of which builds on the previous one. The first method is simply referred to as a "simple method". This method simply generates all possible combinations of values for all of the variables, and splits the set of combinations into q subsets of combinations. The issue with this method is that a large number of combinations may be trivially inconsistent.

The next method that is presented is finding "not detected inconsistent (NDI) subproblems". The idea behind this strategy is to find the first q sub-problems that are not detected as inconsistent by propagation of the constraints. For example, if there is an `alldiff` constraint, meaning all variables must have different values, a large number of sub-trees that assign the same value to two or more variables are trivially inconsistent. To obtain these NDI sub-problems, an iterative deepening depth-first search approach is used, which is optimized in three ways. First, instead of only incrementing the depth by one at each step, an estimate for how many more layers of depth are needed to reach the desired number of sub-problems is made. Second, the layers of the search tree that have already been explored are kept in a lookup table to avoid checking those again each time the depth is increased. Finally, the search process itself is parallelized as well.

The final method that is presented has to do with large domains. Whereas normally the sub-problems would be split into groups of variables, if the domain of a variable is very large, this may result in sub-problems that are still quite large. In such a case, the domain of a variable can be split up into multiple smaller domains and submitted as separate jobs.

2.2.3 Final observations

The authors touch on an important observation: since there is no communication between workers, information discovered by one worker cannot be used by other workers to improve their current job. However, there is communication between the workers and the work pool, so the work pool can store this useful information, which may for example include a value for a variable that always violates a certain constraint. This in combination with the fact that jobs are small and should be quick to complete, means that not a large amount of time gets wasted on the current job if the information is not received by the worker, but future jobs can still be optimized when the worker does receive the information.

The two methods are extensively tested and compared alongside other similar methods in another paper by Malapert et al. [9], which reveals that the EPS and the work stealing methods produce the best results. They also discuss how the communication affects work stealing. They mention that there are multiple ways to try to mitigate the communication overhead as much as possible. One option is to make sure that workers don't ask for work too often and not too many times in a row. Another option is, as discovered before, to have a central work pool, also known as a queue, that all workers can take work from. However, this may still be costly on a large cluster of machines where the queue has to be polled over a network. In such a case, Machado et al. [8] proposed a hybrid configuration, where there is a small number of workers on each machine, who will first try to steal from other workers on the same machine before going to the queue.

2.3 Confidence-based work stealing

Another approach to work stealing is presented by Chu et al. [6], which uses a confidence function to estimate the probability of the solution being along a certain path in the search tree, and explores the most promising branches first. Because of this, the primary use case of this approach is for finding the best solution, rather than all solutions, provided there are multiple solutions. This approach was quite successful, being capable of super-linear speedup in some instances, due to the fact that information discovered by one worker can be used by other workers to speed up their search process as well. It is also worth noting that this approach works in a sequential, or single worker, search as well, as it focuses on optimizing the order in which work is completed, rather than the distribution of work like in the regular work stealing approach.

2.3.1 Important observations

There are some observations that can be made about the confidence-based work stealing approach by Chu et al. [6]. First, any estimate of the solution density of any branch will have a high error, since it is likely the real solution density is actually zero. Second, branches that are close together should have roughly the same solution density, since they are based on the same decisions up to that point. Third, the solution density estimate of a sub-tree should decrease as more nodes are explored. This is because the most promising sub-trees are searched first, and if they turn out not to be fruitful, the confidence should decrease. Another reason is similar to the second observation, because if one part of the sub-tree was not successful, the other parts are less likely to be successful as well since they are based on the same decisions.

2.3.2 Explanation of the algorithm

Initially, all nodes in the tree are configured to have $confidence = 0.5$, since we have no knowledge beforehand. The workers are divided in the same ratio as confidence at any split in the tree. For example, when the confidence of branch 1 is 0.8 and branch 2 is 0.2, 80% of the workers go down branch 1, while the other 20% of the workers go down branch 2. As problems are starting to get solved, we can update the estimated solution densities of each node. When a worker is finished, it starts traversing the tree from the top again, going down whichever path that has the biggest deviation in the number of workers actually working on it compared to the number of workers that should be working on it.

2.3.3 Issues

There are some issue with this method, however. Currently, workers would steal leaves of the tree, which are so quick to solve that communication costs would be a large part of the total runtime. To solve this, a minimum height for a sub-tree to be stolen is defined, which will extend the size of the job and therefore the time it takes to solve, meaning communication is a smaller part of the runtime. Of course, sub-trees should not be too large either, so some balancing is required here.

Another issue is that workers may take very long to find the solution of a given sub-tree. This is why a restart time is defined. After a worker has been working on the sub-tree for a certain amount of time, it abandons its current sub-tree and starts from the root of the tree again to steal a new sub-tree. Because the confidence values have been updated by the non-solutions found by other workers and itself, it is very likely it will steal a different sub-tree this time. This makes sure that a worker does not keep spending its time in a sub-tree whose confidence value has dropped significantly since the worker started working on it.

2.4 Mixing static and dynamic partitioning

Another idea, proposed by Menouer et al. [10], is to combine two methods of partitioning the search tree, namely static partitioning and dynamic partitioning. Let us first look at what each method entails.

2.4.1 Static partitioning

Static partitioning is a partitioning strategy that occurs before the workers start working on the problem. It splits the search tree, which represents the entire search space, up into smaller sub-trees which represent part of the search space. It then distributes these generated sub-trees over the workers. Because distributing sub-trees evenly across workers is a difficult problem, as also found by Burton et al.[5] and Malapert et al. [9], it is likely that static partitioning on its own would lead to imbalance in the workload of the workers. However, the benefit of static partitioning is that it is relatively simple, and, as we will see later, can be used as a good starting point for an approach that aims for a more even distribution of work.

2.4.2 Dynamic partitioning

Dynamic partitioning occurs during the time that the workers are working on the problem, in contrast to static partitioning. In dynamic partitioning, there is a global queue which contains the sub-trees to be solved. Workers will split their current sub-tree into two smaller sub-trees when they notice that there are idle workers. They then keep one of the sub-trees for themselves to work on, and submit the other sub-tree to the queue, where the idle worker can take it from. This strategy ensures a more even distribution of work, since work is split on demand, but it also forces the workers to spend some extra time either splitting their current work and submitting it to the queue in case they are busy, or waiting for new work and retrieving it from the queue in case they are idle. This causes the efficiency of the workers to slightly decrease, leading to a potentially slower total resolution time even though the work was distributed more evenly.

2.4.3 Combining the two partitioning strategies

In short, both static- and dynamic partitioning have their own advantages and disadvantages, and the goal of mixing these partitioning strategies is to preserve the effect of the advantages of both, while trying to minimize the effect of their disadvantages. In practice this is done by applying static partitioning as a preprocessing step, since this strategy can be performed before the workers begin working, and then applying dynamic partitioning towards the end of the search, when some workers are finishing their own batch of work, and receiving new work from other workers, whose sub-trees turned out to take longer to solve.

Using this new method, the authors were able to achieve a noticeably higher speedup than using either of the methods individually. This method also led to a more equal distribution of work across threads and a lower average idle time per core, which will be elaborated on further in sections 3 and 4.

3 Results

In this section we present the results of the aforementioned algorithms as showcased in their respective papers [6, 7, 9, 10]. The results will be further discussed in section 4.

While we would have liked to perform some experiments with these new methods ourselves, both to see if our results would be consistent with the original authors' results and to test the approaches on an even playing field, the resources that the authors used to test their methods and the implementations of their methods themselves were not available, and therefore this was not possible. It is because of this reason that we used the results provided by the original authors themselves to compare the approaches instead.

3.1 Embarrassingly parallel search

Table 1 shows that the regular work stealing implemented in Gecode [2] obtains an average speedup of 7.7 (7.8 for the satisfaction problems and 7.6 for the optimization problems), while EPS yields an average of 13.8 (18.0 for the satisfaction problems and 12.3 for the optimization problems). Furthermore, it can be seen that the EPS method outperforms the regular work stealing on all but the last sub-problem of the test set.

The results presented in Table 2 show that the OR-Tools [4] implementation of EPS, which obtained a speedup average of 21.3 outperforms Gecode [2], which, as previously mentioned, obtained an average of 13.8.

Table 1. The performance of EPS and Gecode work-stealing; 40 workers and 30 sub-problems per worker. As seen in Table 1 of [9].

Instance	Seq	Work-stealing		EPS	
	t	*t*	*s*	*t*	*s*
Satisfaction problems:					
allinterval_15	262.5	9.7	27.0	8.8	29.9
magicsequence_40000	328.2	529.6	0.6	37.3	8.8
sportsleague_10	172.4	7.6	22.5	6.8	25.4
sb_sb_13_13_6_4	135.7	9.2	14.7	7.8	17.5
quasigroup7_10	292.6	14.5	20.1	10.5	27.8
non_non_fast_6	602.2	271.3	2.2	56.8	10.6
Optimization problems:					
golombruler_13	1355.2	54.9	24.7	44.3	30.6
warehouses	148.0	25.9	5.7	21.1	7.0
setcovering	94.4	16.1	5.9	11.1	8.5
2DLevelPacking_Class5_20_6	22.6	13.8	1.6	0.7	30.2
depot_placement_att48_5	125.2	19.1	6.6	10.2	12.3
depot_placement_rat99_5	21.6	6.4	3.4	2.6	8.3
fastfood_ff58	23.1	4.5	5.1	3.8	6.0
open_stacks_01_problem_15_15	102.8	6.1	16.9	5.8	17.8
open_stacks_01_wbp_30_15_1	185.7	15.4	12.1	11.2	16.6
sugiyama2_g5_7_7_7_7_2	286.5	22.8	12.6	10.8	26.6
pattern_set_mining_k1_german-credit	113.7	22.3	5.1	13.8	8.3
radiation_03	129.1	33.5	3.9	25.6	5.0
bacp-7	227.2	15.6	14.5	9.5	23.9
talent_scheduling_alt_film116	254.3	13.5	18.8	35.6	7.1
total (t) or geometric mean (s)	**488.2**	**1174.8**	**7.7**	**334.2**	**13.8**

3.2 Confidence-based work stealing

Chu et al. [6] decided to implement the algorithm using Gecode. The machines used for experimenting with the confidence-based work stealing approach were a Mac with two 2.8 GHz Intel Xeon Quad Core E5462 processors with 4 GB of RAM and a Dell PowerEdge 6850 with four 3.0 GHz Intel Xeon Dual Core Pro 7120 processors with 32 GB of RAM. The experiments were conducted using three optimization problems — the Traveling Salesman Problem, Golomb Ruler and Queens-Armies — and three satisfaction problems — n-Queens, Knights and Perfect-Square. The tests tracked the following metrics: wall clock runtime, number of steals, total numbers of nodes searched and number of nodes explored to find the optimal solutions. The results of those experiments can be seen in Tables 4 and 3.

Table 2. The performance of EPS implemented in OR-Tools; 40 workers and 30 sub-problems per worker. As seen in Table 2 of [9].

Instance	Seq	EPS	
	t	*t*	*s*
Satisfaction problems:			
allinterval_15	2169.7	67.7	32.1
magicsequence_40000	—	—	—
sportsleague_10	—	—	—
sb_sb_13_13_6_4	227.6	18.1	12.5
quasigroup7_10	—	—	—
non_non_fast_6	2676.3	310.0	8.6
Optimization problems:			
golombruler_13	16210.2	573.6	28.3
warehouses	—	—	—
setcovering	501.7	33.6	14.9
2DLevelPacking_Class5_20_6	56.2	3.6	15.5
depot_placement_att48_5	664.9	13.7	48.4
depot_placement_rat99_5	67.0	2.8	23.7
fastfood_ff58	452.4	25.1	18.0
open_stacks_01_problem_15_15	164.7	7.1	23.2
open_stacks_01_wbp_30_15_1	164.9	6.3	26.0
sugiyama2_g5_7_7_7_7_2	298.8	20.5	14.6
pattern_set_mining_k1_german-credit	270.7	12.8	21.1
radiation_03	416.6	23.5	17.7
bacp-7	759.7	23.8	32.0
talent_scheduling_alt_film116	575.7	15.7	36.7
total (t) or geometric mean (s)	**25677.2**	**1158.1**	**21.3**

For the optimization problems, the results show that the runtime is proportional to the number of nodes searched, and highly correlated to the amount of time taken to find the optimal solution. This is only natural, as the quicker the optimal solution is found, the fewer nodes need to be searched, ensuing in a lower total runtime. Using a confidence of 1 achieves near perfect algorithmic efficiency, for the strong heuristic TSP, while the lower confidence values cause a decrease in the efficiency, to 0.81 and 0.80, respectively. The algorithm efficiency is defined by the authors as the total number of nodes searched in the parallel algorithm compared to the sequential algorithm. It can also be seen that the opposite is true for the weak heuristic TSP, where the confidence of 0.5 outputs the best result.

For the Golomb Ruler, a greedy branching heuristic was used that selects the minimum possible value for the variable at each stage. The results in Table 3 show that for Golomb Ruler 12 and 13, the optimal

Table 3. Results of the confidence-based work stealing approach for optimization problems. As seen in Table 1 of [6].

conf	Runtime	Speedup	RunE	Steals	Nodes	AlgE	Onodes	SFE
TSP with strong heuristic, 100 instances (Mac):								
Seq	313.3	—	—	—	5422k	—	1572k	—
1	38.2	7.25	0.91	708	5357k	1.01	1589k	0.99
0.66	47.2	5.88	0.74	319	6657k	0.81	5130k	0.31
0.5	48.0	5.77	0.72	467	6747k	0.80	5275k	0.30
TSP with weak heuristic, 100 instances (Mac):								
Seq	347.8	—	—	—	7.22M	—	1.15M	—
1	46.7	7.45	0.93	1044	6.96M	1.04	1.09M	1.06
0.66	45.8	7.60	0.95	379	7.02M	1.03	1.10M	1.05
0.5	41.6	8.36	1.08	259	8.42M	1.15	0.66M	1.63
Golomb Ruler, 2 instances (n = 12, 13) (Mac):								
Seq	562	—	—	—	9.71M	—	1.07M	—
1	69.0	8.15	1.02	572	8.96M	1.08	0.81M	1.33
0.66	59.0	9.54	1.19	346	7.58M	1.28	0.49M	2.21
0.5	65.2	8.63	1.08	259	8.82M	1.15	0.66M	1.63
Queen Armies, 2 instances (n = 9, 10) (Mac):								
Seq	602	—	—	—	13.6M	—	845k	—
1	87.1	6.91	0.86	1521	14.5M	0.94	1878k	0.45
0.66	86.3	6.98	0.87	1143	14.5M	0.96	2687k	0.31
0.5	86.0	7.00	0.87	983	14.5M	0.95	2816k	0.30

Table 4. Results of the confidence-based work-stealing approach for constraint satisfaction problems. As seen in Table 2 of [6].

conf	Solved	Runtime	Speedup	RunE	Steals	Nodes	AlgE
n-Queens, 100 instances (n = 1500, 1520, ..., 3480):							
Seq	4	2.9	—	—	—	1859	—
1	4	10.4	—	—	2	1845	—
0.66	29	18.0	—	—	9	15108	—
0.5	100	2.9	—	—	8	14484	—
Knights, 40 instances (n = 20, 22, ..., 98):							
Seq	7	0.22	—	—	—	213k	—
1	7	0.26	—	—	2	1150	—
0.66	13	0.50	—	—	8	8734	—
0.5	21	0.66	—	—	8	8549	—
Perfect-Square, 100 instances							
Seq	15	483.1	—	—	—	231k	—
1	13	72.3	6.68	0.83	419	216k	0.99
0.66	14	71.2	6.78	0.85	397	218k	0.98
0.5	82	8.9	54.02	6.75	21	32k	6.64

solution does not lie directly in the left-most branch and that a degree of non-greediness leads to a super-linear solution finding efficiency.

The results for Queens-Armies are very similar, regardless of the confidence level used.

Switching to the satisfaction problem results, we can see a large difference in the number of instances solved between the parallel algorithm and the sequential one. Due to the fact that n-Queens and Knights have very deep subtrees to search, once the sequential algorithm fails to find a solution in the leftmost subtree, it will often get stuck. Using a confidence level of 0.5 produces a super linear speedup and enables the parallel algorithm to solve all 100 instances of n-Queens and 21 instances of Knights. This is a large imrovement, compared to the sequential one which solved 4 and 7 instances, respectively.

A similar trend can be observed in the case of Perfect Square as well, with the parallel algorithm obtaining a super linear speedup again, at 0.5 confidence, managing to solve 82 instances compared to 15 for the sequential variant.

3.3 Mixing static and dynamic partitioning

Experiments of this approach were conducted on two Linux machines, tagged `M1` and `M2`. The machines were both equipped with an Intel Xeon X5650 processor with 12 cores and 48 GB of RAM. The algorithm itself was implemented and tested in OR-Tools [4], so the results depend directly on this software.

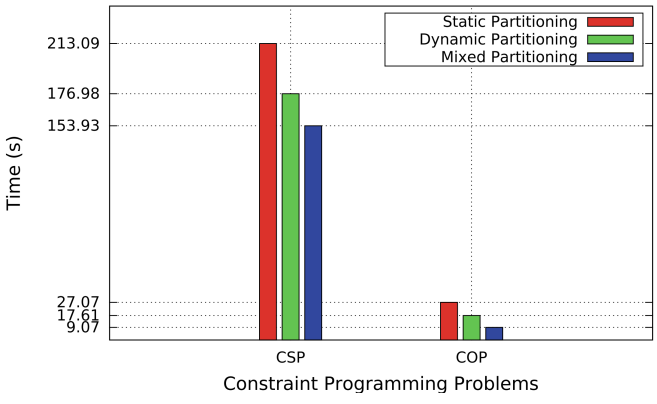


Fig. 1. Computation times of three partitioning strategies for the constraint programming problems. As seen in Figure 7 of [10].

Figure 1 shows a comparison between the performance of the static, dynamic and mixed partitioning strategies for solving two constraint programming problems. The first problem is the Quasi Group problem [3], a constraint satisfaction problem labelled "CSP" and the second one is the Level Packing problem [3], a constraint optimization

problem, labelled "COP". The results, as seen in Figure 1 show a clear improvement in the computation times for the mixed partitioning strategy for both optimization problems and constraint satisfaction problems.

The mixed partitioning approach is further compared against the `Gecode CP` [2] solver for the same 6 constraint satisfaction problems. The resulting execution times can be seen in Table 5 and Table 6, respectively.

Table 5. Execution times of the mixed partitioning strategy for 6 constraint satisfaction problems. As seen in Table 1 of [10].

Problems	Mixed partitioning run times (s)			
	1 core	4 cores	8 cores	12 cores
quasigroup7_10.fzn	1377.25	432.43	227.55	162.71
sb_sb_13_13_5_1.fzn	653.58	157.42	93.83	56.65
sb_sb_14_14_6_0.fzn	152.25	51.23	29.63	23.81
sb_sb_15_15_7_4.fzn	284.73	91.86	57.47	49.13
sb_sb_12_12_5_4.fzn	64.51	14.34	8.51	7.46
sb_sb_15_15_7_3.fzn	248.95	71.52	52.12	44.27

Table 6. Execution times of the Gecode CP solver for 6 constraint satisfaction problems. As seen in Table 2 of [10].

Problems	Gecode solver run times (s)			
	1 core	4 cores	8 cores	12 cores
quasigroup7_10.fzn	239.69	73.66	37.46	24.74
sb_sb_13_13_5_1.fzn	301.44	107.80	60.75	50.85
sb_sb_14_14_6_0.fzn	151.69	57.2	28.23	22.15
sb_sb_15_15_7_4.fzn	411.24	139.56	78.84	66.26
sb_sb_12_12_5_4.fzn	42.33	16.8	9.24	6.90
sb_sb_15_15_7_3.fzn	335.73	175.97	81.38	68.48

The aforementioned tables show the run time in seconds of both approaches for each of the problems so we can directly compare their speedup factors. The average speedup obtained using the mixed partitioning is higher than the one obtained by the Gecode CP [2] solver for the optimization problems as well as for the satisfaction problems.

4 Discussion

Each of the parallelization approaches presented so far throughout this paper were tested on vastly different machines. Furthermore, the tests were ran on different problem sets. Thus, unfortunately for the goal of this paper, we cannot directly compare the performance of the three strategies — EPS, confidence-based work stealing and work stealing with mixed partitioning. As a result, we will be focusing on the speedup factor of each approach in trying to determine which one performs the best and under what circumstances.

Based on the results presented in the reference papers, the EPS method for solving constraint programming problems in parallel seems to perform the best, as it frequently gives linear speedups and outperforms the rest of the approaches. More specifically, the results show that splitting the initial constraint programming problem into 30 sub-problems per thread yields speedup factors of 21.3 and 13.8 with OR-Tools [4] and Gecode [2], respectively, on a machine with 40 cores, as shown in Tables 2 and 1.

The confidence-based work stealing approach comes next, yielding an approximate speedup of 7 for 8 threads, as the results presented in Tables 3 and 4 show, for optimization problems as well as satisfaction problems. Using confidence values that have only a small bias towards the real value is enough to produce super linear speedup. Naturally, moving away from the real value results in a substantial decrease in performance.

The mixed partitioning strategy shows promising results as well for the speedup factor, but falls short in the runtime compared to the regular work stealing approach for certain test cases. Tables 5 and 6 show that the mixed partitioning obtains speedup factors higher than the average speedup factors of the Gecode [2] solver. The mixing strategy gains a speedup of 7.02 for 12 cores, while the Gecode [2] solver gains only a 4.3 speedup. Furthermore, Figure 1 shows that the mixing partitioning method has a lower computation time than the other two variants — static and dynamic — for constraint satisfaction problems as well as for constraint satisfaction problems. The mixing partitioning strategy benefits from the fact that, as an external parallelization, it is possible to combine it with the portofolio parallelization. Moreover, it is designed for shared and distributed memory architectures alike, while the other parallel constraint programming solvers are designed for only one of those memory types. However, probably the biggest benefit of this strategy is that it does not change the original OR-Tools [4] code, so it can be used directly with other versions of the code. On the other hand the mixing partitioning approach also has some drawbacks, namely: it's not deterministic, so the resolution of the problems depends on the threshold of static and dynamic partitioning and the fact that it can only attempt to solve problems modelled using the FlatZinc [1] format.

5 Conclusion

The main goal of this paper was to contribute to the current state-of-the art of the constraint programming research field by offering a broader comparison between the currently popular approaches and to further determine the method which performs the best and under what circumstances. Thus, we compared the EPS method, presented as an alternative to the work stealing approach and two methods that augment the work stealing — confidence-based work stealing and mixing dynamic and static partitioning. Comparing the aforementioned methods was not easy, since they were tested on different hardware and problem sets and there was no possibility of making our own experiments, since source code is not publicly available for every approach. However, as presented in section 4, we mainly relied on the speedup factor and tried accounting for the performance difference of the machines the tests were ran on. Based on the results gathered, we concluded that the embarrassingly parallel search represents the most performant option for parallelizing constraint programming problems. The two other methods performed admirably as well and, as they tackle different parts of the process of solving constraint programming problems, we think they can be combined. For example, using the high speedup factor of EPS and its constant number of subproblems per worker together with the low communication overhead and the external parallelization nature of the mixed-partitioning approach. Thus, we conclude with the idea that combining the three aforementioned approaches would form something greater than the sum of their separate parts and thus may be a topic worth researching.

6 Future work

As mentioned in section 5, the information gathered suggests the possibility of combining the three approaches mentioned throughout the paper. We think that this might be a way to overcome some of their drawbacks and obtain a new approach, with increased performance. The three approaches offer many combination possibilities, as they tackle different parts of the constraint programming solving process. For example, the threshold of static and dynamic partitioning that the mixed partitioning approach depends on can benefit from the confidence-based strategy that was introduces in the confidence-based work stealing approach.

Another topic suitable for future work would be attempting a direct comparison between the three parallelization methods. They were all tested on different hardware and with mostly different problem sets.

Thus, in this paper, we only managed to compare them based on their speedup factor. A more direct comparison would yield a clearer perspective on the benefits and drawbacks of each approach and where they perform best.

REFERENCES

[1] FlatZinc. https://www.gecode.org/flatzinc.html. Accessed: 15-03-2021.

[2] Gecode. https://www.gecode.org/. Accessed: 15-03-2021.

[3] MiniZinc Challenge 2012. https://www.minizinc.org/challenge2012/challenge.html. Accessed: 15-03-2021.

[4] OR-Tools. https://developers.google.com/optimization. Accessed: 15-03-2021.

[5] F. W. Burton and M. R. Sleep. Executing functional programs on a virtual tree of processors. In *Proceedings of the 1981 Conference on Functional Programming Languages and Computer Architecture*, FPCA '81, page 187–194, New York, NY, USA, 1981. Association for Computing Machinery.

[6] G. Chu, C. Schulte, and P. J. Stuckey. Confidence-based work stealing in parallel constraint programming. *Principles and Practice of Constraint Programming*, 15:226–241, Sept. 2009.

[7] Y. Hamadi and L. Sais, editors. *Handbook of Parallel Constraint Reasoning*. Springer International Publishing, Cham, 2018.

[8] R. Machado, V. Pedro, and S. Abreu. On the scalability of constraint programming on hierarchical multiprocessor systems. *ICPP*, pages 530—-535, 2013.

[9] A. Malapert, J.-C. Régin, and M. Rezgui. Embarrassingly Parallel Search in Constraint Programming. *Journal of Artificial Intelligence Research*, 57:421–464, Nov. 2016.

[10] T. Menouer, M. Rezgui, B. Le Cun, and J.-C. Régin. Mixing Static and Dynamic Partitioning to Parallelize a Constraint Programming Solver. *International Journal of Parallel Programming*, 44(3):486–505, June 2016.

Comparison of Novel Software Defined Networking Approaches for Improving Data Centre Networking

Daan Raatjes and Sjouke de Vries

Abstract— Modern data centres must scale to a large number of servers, while offering flexible placement and migration of virtual machines. Traditional approaches suffer from shortcomings that may impair both their scalability and flexibility. Several novel approaches to the design of large scale data centres are discussed and evaluated. An overview of newly developed methods with their advantages and disadvantages is provided. Subsequently, each of the methods are compared to each other for their feasibility, flexibility and scalability.

Index Terms—— Software Defined Networking, PARIS, Iris, OpenFlow, RSDN, data centre interconnect

1 Introduction

Recently, the Covid-19 pandemic has led to unprecedented changes in the way people interact with each other. Consequently, this shift has increased the overall pressure on the internet. For Facebook, the pandemic caused a spike in traffic and significant change in behaviour [1]. The latter point shows the rising demand of flexibility, requiring Internet Service Providers (ISPs) to configure their networks such that they can adopt to sudden traffic spikes. Typically, internet traffic increases at a rate of 30% annually [2]. Feldmann et al. [3] show that an increase in traffic in the order of 15-20% was reported within *days* after the lockdown started, again stating the need for highly flexible and scalable networks.

With the recent surge in demand for cloud computing, the efficient design and implementation of data centres becomes increasingly important. Nowadays networks are part of the most critical infrastructure of our businesses, homes and schools. As the infrastructure has become increasingly critical, the barrier for testing new ideas also raised alongside with it.

Virtualized programmable networks could lower the barrier to entry for new ideas, increasing the rate of innovation in the network infrastructure [4]. The networking community is hard at work developing programmable networks, such as GENI [5], a proposed nationwide research facility for experimenting with new network architectures and distributed systems. However, the current state of network architectures can not keep up with the current growth rate.

Several software defined networking (SDN) approaches have recently been proposed that may be able to deal with this upsurge of network traffic. In this paper, a dynamic network scheduler, Iris, PARIS and a recursive SDN (RSDN) framework are discussed. These methods are analysed and compared for their feasibility, flexibility and scalability. The discussed approaches are found to be highly scalable in the initial analysis. However, the flexibility and feasibility differ per method. The recursive SDN and the dynamic network scheduler will be shown to lack in feasibility. In addition, our analysis suggests that PARIS outperforms Iris in terms of flexibility.

The rest of the article is organized as follows. Firstly, software defined networking is introduced along with the necessary background information. Subsequently, the novel approaches for network routing in cloud data centres are presented. After introducing these methods, a comparison is given that highlights their pros and cons. Furthermore, several general problems and shortcomings surrounding software defined networks are discussed. Finally, the paper is concluded with a summary of our findings.

- *Daan Raatjes (s2953854) is with University of Groningen, E-mail:d.h.a.raatjes@student.rug.nl.*
- *Sjouke de Vries (s3186520) is with University of Groningen, E-mail:s.de.vries.44@student.rug.nl.*

2 Background

Before examining possible solutions, it important to examine the different components that make up this new type of networking. In this paper, the focus lies not only on networking inside a single Data Centre (DC), but also into inter-DC network connectivity.

2.1 Traditional Hierarchical Network Structure

In traditional large data centers, the network structure is typically divided into three tiers; access layer (switches on top of a rack that are physically connected to servers), aggregation layer (connects access layers and provides service such as firewall) and core layer (provides high-speed forwarding for packets that enter and leave the data centre).

The collection of devices and underlying infrastructure that is used to transmit data is sold as a commodity to the end user is called a (telecommunications) carrier network. Traditionally, data centre connectivity is also less geographical disperse in comparison to the data centres that are being managed by carriers [6]. Since traditional hierarchical networking is not sufficient to match the increasing network traffic, there is a need for software defined networking.

2.2 Software Defined Networking

Software defined networking is a new approach to networking that attempts to solve the shortcomings of traditional networking where network hardware is manufactured to support a fixed set of protocols. Whenever it is desirable to extend this set of supported features, it is necessary to replace existing hardware with new hardware. Not only will this be less cost-efficient it also takes a lot of time as physical operations are required. In short, the aim is to separate the control plane (protocols) from the data plane (the actual switch hardware).

OpenFlow is a network protocol consisting of one or more flow tables and a group table, which perform packet lookups and forwarding, and one or more OpenFlow channels to an external controller as shown in Fig. 1 [7]. With the OpenFlow switch protocol, the controller can add, update and delete flow entries in flow tables, both reactively (in response to packets) and proactively (predefined routes). Each flow table in the switch contains a set of flow entries; each flow entry consists of match fields, counters, and a set of instructions to apply to matching packets. Matching starts at the first flow table and may proceed with additional tables in a pipelined fashion as shown in Fig. 2.

Most studies concerning SDN lack multiple orders of magnitude from today's carrier networks. Therefore, McCauley et al. [6] present a recursive SDN framework that combines the programmability of SDNs with the scalability of a hierarchical network structure. Each level of the route computation acts on a set of aggregates (called logical cross-bars, or LXBs) and they communicate a summary of the re-

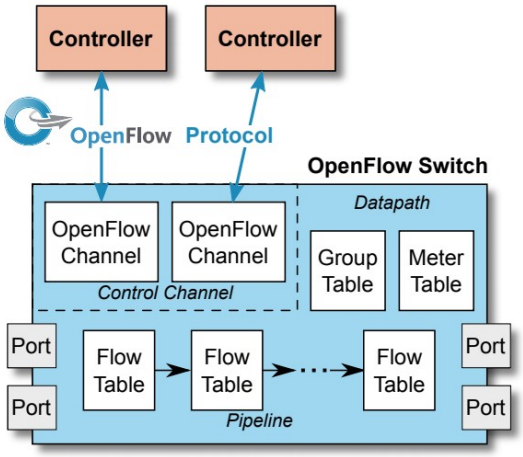


Fig. 1. Main components of an OpenFlow switch. [7]

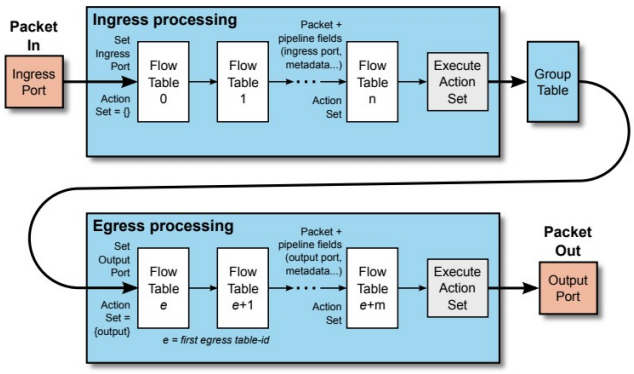


Fig. 2. Packet flow through the processing pipeline. [7]

sults to their parent and child LXBs. Effectively, this approach limits the number of route computations a single node has to handle as it can re-use summaries from its children and/or parents. The framework not only facilitates route computation but also incorporates a mechanism for rapid and localized recovery from failures.

Several manufacturers also offer software for extensive monitoring of network traffic to enhance security. *HPE VAN SDN* controller software [9] can monitor ARP, DHCP, and IP packets from edge ports. This provides a cache of MAC and IP addresses for each end point, which provides identification of devices or users attached to a network. In general the impact on security is positive. Nevertheless, with the introduction of new technology, a new attack surface can be introduced as one is now able to experiment with self made protocols that might not behave as intended. The challenge remains in enforcing security correctly in software.

3 Methodology

Several methods to improve the scalability and flexibility have recently been proposed. In this section we discuss four of these methods and highlight their technical aspects.

3.1 Iris

Iris is an optical-circuit-switched architecture that lowers infrastructure cost and complexity barriers, making a richer topology design space more accessible to operators of regional (data centre) networks [8]. Using optical ports is the biggest contributor of costs. Reducing the costs of these ports makes distributed topologies more accessible as their cost becomes comparable to centralized topologies. In addition, by reducing network ports, the complexity in network management and configuration is reduced. Iris essentially optimises the network design between data centres in a region, known as a regional Data Centre Interconnect (DCI).

Fig. 3 shows some of the specifications when designing a DCI. Given the data centre sites, the topology, capacity and switching implementation must be determined.

3.2 Recursive SDN

A recursive routing computation framework is presented that balances the programmability of software defined networks with the scalability of traditional hierarchical structure by using a combination of both Course Grained Routing (CGR) and Fined Grained Routing (FGR) [6]. It mainly exploits the fact that in almost all networks parts can be aggregated into so called logical cross-bars (LXBs). They intuitively behave the same as switches. This process of aggregation will form a tree-like hierarchical structures where each LXB is connected with the LXB of the same depth (see Fig. 4).

Its recovery approach offers 99.999% of network repair under heavy link failure scenario. Common practice for network availability is to implement alternative paths that are computed beforehand. This will work perfectly fine for majority of cases, except when the alternative route(s) also becomes unavailable. Key difference with the RSDN algorithm is that it can recover from an arbitrary amount of failures as long as a path exists. Every node will start with an internal table of where to route each packet. After applying the algorithm every node will have a set of internal tables to virtually route packets through that node. Consider for example when node A wants to send packets through node B to node C. When node B becomes unavailable node A will virtualize the routing of node B itself, by including B's routing table into A, so that node C can still be reached. One can see that this process can be applied recursively covering any amounts of failures as long as a path from source to destination remains.

3.3 Dynamic Network Scheduler

A dynamic network scheduler technique has been examined that maximises fairness in resource sharing while minimising unutilized resources.

Bandwidth can be allocated naively by dividing the total amount available by the number of consumers. Consequently, an idle instance will have unutilized bandwidth which leads to suboptimal performance. It is increasingly common to over allocate resources in order to not only increase profit, but also energy efficiency. Another problem arises whenever all instances would require maximum bandwidth at the same time. A dynamic scheduler [10] can be used in this case to not only help tackle this problem, but it will also optimize the overall bandwidth utilization. For example, say we have a theoretical network uplink of 1 Gbit/s and two Virtual Machines (VMs). The naive approach would allocate both VMs an equal amount of bandwidth. In order for this allocation to be fair, the bandwidth usage of both VMs should be similar, but in practice this is rarely the case.

The dynamic network scheduler constructs a graph from data collected from software switches on machines, top-of-rack switches and routers (depicted in Fig. 5) by means of using software like open vSwitch [11].

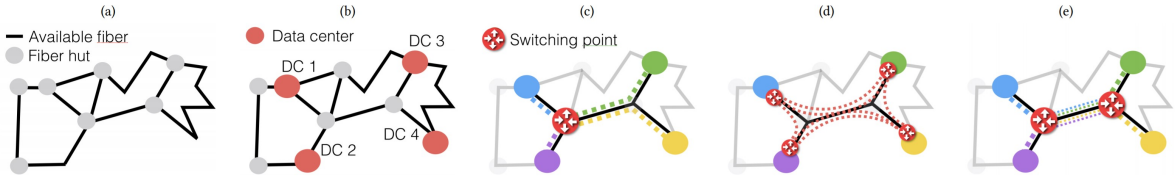


Fig. 3. DCI design example: (a) The fiber map, which contains all available fiber ducts and huts. (b) The region has 4 DCs for which DCI connectivity is to be determined. (c) The centralized approach uses a hub to which all DCs connect; in practice 2 hubs are used for resilience, but for clarity only one is shown. (d) An extreme version of the distributed approach, with all pairs of DCs connected directly to each other. (e) A sparser distributed approach, with two pairs of DCs – each pair connects to a hub, and the two hubs connect to each other. [8]

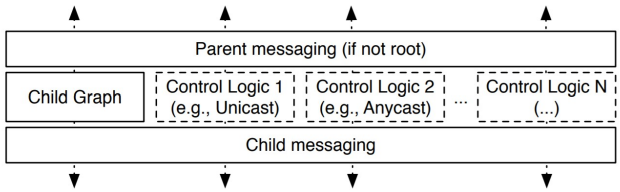


Fig. 4. Software structure in a normal (non-leaf) LXB. [6]

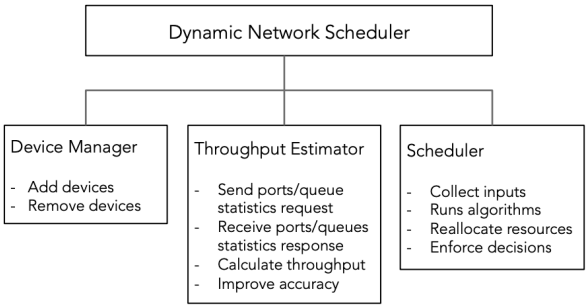


Fig. 6. The components and tasks of the Dynamic Network Scheduler. [10]

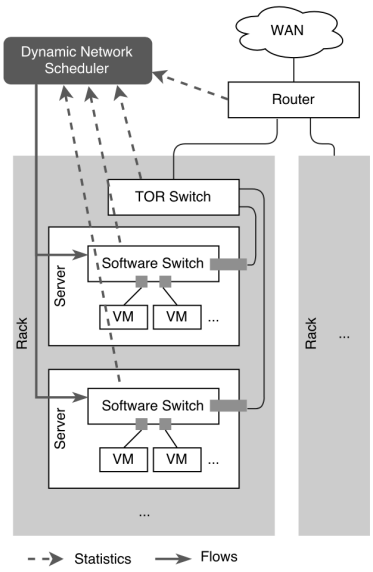


Fig. 5. Dynamic Network Schedule in a cloud data centre. [10]

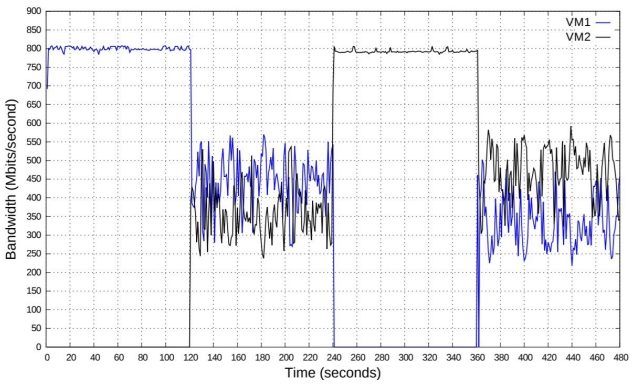


Fig. 7. Network Fairness for two VMs without Dynamic Network Scheduler. [10]

A Directed Graph of the entire Data Centre network is constructed and fed to the algorithm. Fig. 6 shows the three different parts of the dynamic network scheduler: the device manager, the throughput estimator and the scheduler.

In order to evaluate the Dynamic Scheduler, Hauser et al. [10] conducted an experimented with two physical machines (HP Proliant Microserver Gen8) with Linux and KVM as a hypervisor. On each machine they hosted two virtual machines and an additional machine was used to host the switch controller that controls the Open vSwitch which in turn connects both machines physically. The iPerf3 [12] tool was used to produce and measure the throughput of the machines. Important to note is that the TCP protocol has a built in congestion control, compromising the results of the experiment [13]. Therefore, the number UDP packets are used as measure of throughput.

Fig. 7 and Fig. 8 show that with the dynamic scheduler the bandwidth is shared more fair, efficient and with a better behaviour pattern, in the way it gives back bandwidth immediately to a donor when needed.

3.4 PARIS

Finally, we examine PARIS [14]; an SDN architecture that prepositions IP forwarding entries in the switches of a network. Switches within a network pod are entirely aware of the virtual machines that reside underneath them. Every core switch in the network maintains a forwarding state. PARIS has the advantage that it allows for complete flexibility of choosing a topology that satisfies latency and bandwidth requirements during the design of the network. The controller in the PARIS architecture has complete transparency over the topology of the network and is aware of all the virtual machines their addresses and locations. This knowledge is subsequently used to achieve three goals. Firstly, it allows for ideal placement of forwarding information

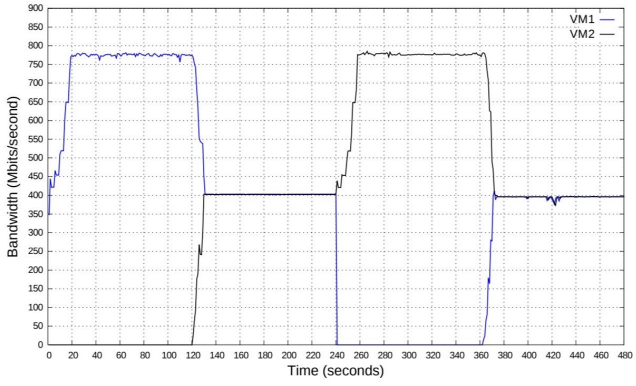


Fig. 8. Network Fairness for two VMs with Dynamic Network Scheduler. [10]

in switches after initialisation. Secondly, the controller keeps track of switch-level topology and the location of the host. Finally, the controller monitors the network traffic such that traffic engineering can be enabled.

The switches in the architecture must support neighbor discovery via LLDP protocols and inform the controller to learn about topology changes. The switches should at least be able to transform an IP prefix to an outgoing link. This approach allows us to use SRAM/DRAM-based switches which are significantly less expensive to the more common TCAMS switches that some other architectures require [15].

The hosts in the PARIS architecture are placed in their individual subnet with a route to its edge switch by default. This architecture has the advantage that it does not need to perform unnecessary lookups of directory servers nor does it need to spend additional time on dealing with host ARP broadcasts. Host DHCP messages can directly be forwarded to the controller. Subsequently, the controller can assign any free IP address to any host.

4 Analysis

The previously mentioned methods are compared and evaluated for their feasibility, flexibility and scalability. Although the analysis is limited because quantitative data is lacking, we attempt to provide a qualitative and objective analysis. The analysis is summarised in Table 1. In the next sections, each property is discussed in-depth for every SDN method.

Approach	Scalability	Flexibility	Feasibility
Iris	++++	+++	++++
Recursive SDN	+++++	++++	+
Dynamic Network Scheduler	++	+++	++
PARIS	+++	+++++	++++

Table 1. Comparison of SDN methods

4.1 Feasibility

Feasibility is an important aspect when considering viable networking solutions in data centres. Although some methods may theoretically achieve promising results, if their implementation in practice is too challenging, then the method should not be considered to be a viable option. The design of the recursive SDN seems to be difficult to be deployed in the current state of carrier networks because of their scale. For this approach to be feasible, software defined networking will have to be used more in carrier networks. Conceptually, the dynamic network scheduler shows promising results. However, to reliably determine the scheduler's feasibility, the prototype would first have to be expanded and tested in an actual data centre. Both Iris and PARIS seem to be feasible options. Iris's ease of implementation combined with the fact that it can be implemented using off-the-shelf hardware makes it a compelling improvement. The proactive routing approach introduced by PARIS also seems to be a very feasible solution in practice.

4.2 Flexibility

The dependencies of a solution on existing systems should also be taken into account when reasoning about the flexibility of proposed solutions. The Recursive SDN allows for a flexible way to implement numerous designs over existing hierarchical structures. Being able to decorate existing structures rather than having to replace them entirely, makes the Recursive SDN very flexible. The Dynamic Network Scheduler approach is also flexible in the way it is placed at data centres sites. Moreover, tweaks regarding the algorithm for scheduling are easy to make as they do not affect the metrics collection. The constraints of Iris are less rigid when compared to a distributed design in order to maximize deployment flexibility. PARIS has a dynamic switch, link and host configuration that allow for a lot of flexibility in the aforementioned sections of the network.

4.3 Scalability

Based on the presented experiments, the evaluated methods all seem to scale well in their own way. The dynamic network scheduler has sub-optimal bandwidth performance due to its distance from the virtual machines but performs excellent in ensuring a fair allocation of resources. The PARIS architecture shows promising results in terms of the bandwidth scalability as depicted in Fig. 9. However, its scalability can be limited by the OpenFlow controller because it handles numerous tasks. Preliminary research has shown that by distributing the OpenFlow controller it can scale to 2M virtual machines and 100k servers [16]. Simulations of the recursive SDN framework show that the network can easily deal with 10K nodes. Furthermore, the recursive SDN makes it trivial to implement scalable versions of other routing designs. Finally, we find that Iris is an excellent solution to scalability problems for the inter-connectivity of regional data centres.

5 Discussion

In the following sections, the results of the analysis will be discussed and put into context. The dilemmas a network designer faces, will be discussed as well as some additional considerations that should be taken into account when designing the network of a data centre.

5.1 Flexibility versus Complexity

One of the characteristics of cloud computing according to the NIST definition [17] is resource pooling that allows for dynamically allocating of virtual resources. SDN contributes to this characteristic of cloud computing. Nevertheless, this separation in layers adds more complexity as defining very large infrastructures in software is not trivial. All the previously introduced systems have this trade-off between flexibility and complexity to some extent. Arguably, solutions that build on top of traditional systems tend to tackle the complexity better as solutions already exist and can be re-used or consulted when writing new variations.

5.2 Cost versus Speed

In general, the perception about programmable switches is that they are 10-100 times slower fixed-function switches and they cost more and consume more power. With a high throughput switch like the Intel Tofino 2 the programmability of the switch is preserved while the disadvantages are mitigated [18]. One could argue that a data centre consisting of programmable switches is significantly faster as the overall congestion can be handled a lot better.

5.3 Repair capabilities

Traditional systems often do not guarantee more than 99,99% up-time as there is a limited amount of physical back-up links that can be established. Physical connections remain tedious to repair as they require human interaction. Besides failure in hardware, we should also take into account software failures. Wrongly forcing routes for packets can have terrible outcomes and quickly cause congestion in a data

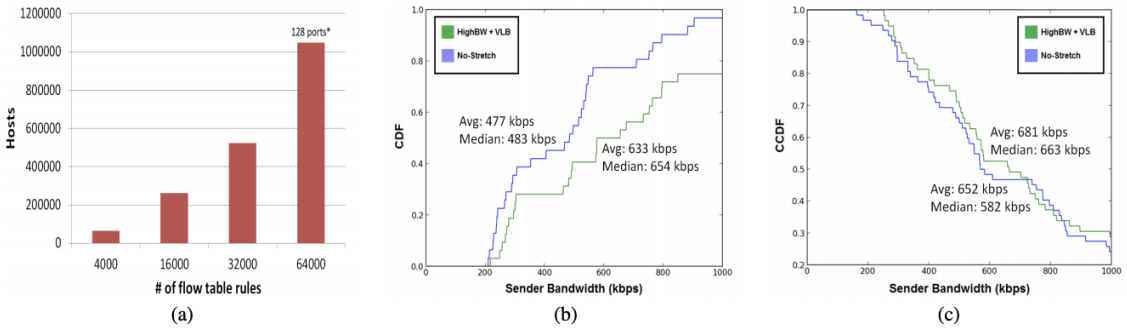


Fig. 9. (a) Number of table entries at the core switches. (b) CDF of sender bandwidth for No-Stretch and High-Bandwidth PARIS. (c) CCDF of sender bandwidth for No-Stretch and High-Bandwidth PARIS. [14]

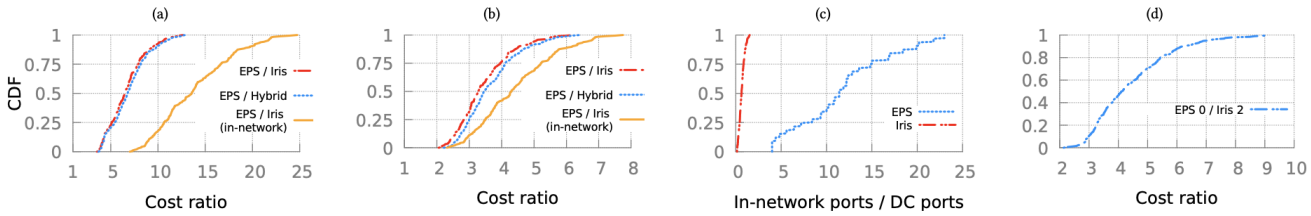


Fig. 10. Iris is substantially cheaper: (a) Relative cost of Iris, EPS, and hybrid networks across all 240 scenarios. (b) Same as (a) but with DCI transceiver cost assumed (unrealistically optimistically) equal to SR transceivers. (c) EPS uses many more in-network ports, as shown by the ratio of in-network to DC ports across designs. (d) Relative cost of an EPS supporting no failures vs. Iris, which handles up to 2 failures. [8]

centre network. Basically, nullifying all the benefits of using a network completely defined in software. Extra physical hardware links are relatively costly compared to the extra memory needed to store more routes in software. Moreover, methods such as recursion can be exploited in software to ensure infinite fail-over scenarios as long as a path exists.

5.4 Network Statistics

Throughput Iris shows for shorter intervals a maximum slowdown of 2% across all flows at the 99^{th} percentile when compared to the Evolved Packet System (EPS). This result is as expected, since the probability of a short flow ($< 50KB$) being affected is small given that the interval of reconfiguration is much larger than short flow completion times. Nevertheless, large flows see only a short and negligible drop in throughput. All test cases executed for the Dynamic Network scheduler which were measured throughput using iPerf3, the results of test cases with and without the dynamic network scheduler can be used to show its success in guaranteeing better throughput. In PARIS, a packet may travel four hops instead of two in the core layer. Hence, there is a trade-off between stretch for throughput. This is a reasonable trade-off in data centres, since they have very low network latency.

Latency Iris is capable of achieving a 6× latency reduction and in more than 20% of the cases the reduction is more than 2×. The dynamic network scheduler attempts to manage the resources either by dropping the packets that arrive beyond the buffer limits or by reordering the packets that are already present in the buffer for reducing the latency for Quality of Service (QoS) measures. In the RSDN experiments, approximately fewer than 5% of the pairs have a stretch over 10% for latency. PARIS installs forwarding-table entries before the traffic arrives which reduces packet latency and avoids the overhead of learning the information reactively.

5.5 Implications of network functions

Load balancing Iris' internal routing to tier-2 switches can be achieved using standard mechanisms like ECMP or anycast, such that traffic for each (external) destination arrives at the right tier-2 switches in a load balanced fashion. If a corelayer switch fails in PARIS, the virtual prefix can be sub-divided into smaller sub-prefixes and stored on other core-layer switches until a new core-layer switch is available. This provides load balancing properties to the architecture.

Congestion control The dynamic network scheduler control network congestion with the build-in congestion control of the TCP protocol. There are many traffic engineering designs for carrier networks, but the one concerning congestion control is used in conjunction with multipath routing. A feedback system relays congestion information about a path to its source, and the source relays traffic over paths with less load.

5.6 Security

Regarding security rapid control loops for the detection and mitigation of cyber-attacks, schemas for e.g., DDOS detection have been shown in [19]. Traditional devices require exchanging a lot of information and waiting for a certain time in order to partially infer the state of the remaining parts, and where only a few devices are logging statistics (if any) [20]. The global network view of SDN allows for network-wide intrusion detection that analyzes traffic from all switches in order to detect malicious network. SDN also comes with conditional rules that are an example of a self-healing mechanism.

One of the main drawbacks of using SDN is the limited memory capacity which can lead to Denial-of-Service (DoS) attacks. Moreover, it is (yet) undefined how a software defined network switch should deal with encrypted packets as the headers (and payload) it acts on become inaccessible. As a final remark, access control of the control plane needs to be properly handled as an attacker could decide to drop all incoming traffic to certain hosts or use the switches to perform a DDoS attack elsewhere.

6 Conclusion

The efficient design of data centres has become paramount to ensure scalability in the cloud computing domain. Four methods for improv-

ing the design of data centres have been examined: recursive SDN, PARIS, Iris and the Dynamic Network Scheduler. Each of these methods have been assessed for their feasibility, flexibility and scalability. Furthermore, several dilemmas that a designer has to consider when constructing a data centre have been discussed. All of the mentioned approaches show promising results but they each have their own drawbacks. Both the recursive SDN and the dynamic network scheduler seem too difficult to deploy and lack feasibility. Iris and PARIS appear to be realistic options, where PARIS has superior flexibility and Iris scales better. More research has to be done to reliably determine whether the theoretical scalability and feasibility uphold their expectations in real-life scenarios.

7 Future Work

Future work that should be done is integrating the dynamic network scheduler into an OpenStack cluster such that it has integration with a stable component of a Cloud data centre. This research would increase the feasibility of this method. Further planned extensions are continuous monitoring and profiling of the network traces for VMs in a cluster. This could lead to a better understanding of network usage patterns such that it can include predictions and give feedback to the Cloud middleware.

Without a doubt, SDN will keep evolving and become more and more the industry standard in the (near) future. As an example, we can take the P4 programming language [21]; the first version of $P4_{14}$ was released in March 2015 and it successor $P4_{16}$ had its first release in May 2017 introducing a number of significant, backwards-incompatible changes to the language in a time span of roughly two years.

Therefore, additional future work can be done in this field to further enhance the capabilities, improve performance and mitigate any existing issues of solutions that use software defined networks.

References

[1] T. Böttger, G. Ibrahim, and B. Vallis, "How the internet reacted to covid-19: A perspective from facebook's edge network," in *Proceedings of the ACM Internet Measurement Conference*, IMC '20, (New York, NY, USA), p. 34–41, Association for Computing Machinery, 2020.

[2] "Cisco annual internet report (2018–2023) white paper." https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. Last accessed on 15-03-2021.

[3] A. Feldmann, O. Gasser, F. Lichtblau, E. Pujol, I. Poese, C. Dietzel, D. Wagner, M. Wichtlhuber, J. Tapiador, N. Vallina-Rodriguez, O. Hohlfeld, and G. Smaragdakis, "The lockdown effect: Implications of the covid-19 pandemic on internet traffic," in *Proceedings of the ACM Internet Measurement Conference*, IMC '20, (New York, NY, USA), p. 1–18, Association for Computing Machinery, 2020.

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, p. 69–74, Mar. 2008.

[5] "GENI exploring networks of the future." https://www.geni.net/. Last accessed on 02-03-2021.

[6] J. McCauley, Z. Liu, A. Panda, T. Koponen, B. Raghavan, J. Rexford, and S. Shenker, "Recursive sdn for carrier networks," *SIGCOMM Comput. Commun. Rev.*, vol. 46, p. 1–7, Dec. 2016.

[7] The Open Networking Foundation, "OpenFlow Switch Specification," Jun. 2012.

[8] V. Dukic, G. Khanna, C. Gkantsidis, T. Karagiannis, F. Parmigiani, A. Singla, M. Filer, J. L. Cox, A. Ptasznik, N. Harland, W. Saunders, and C. Belady, "Beyond the mega-data center: Networking multi-data center regions," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, (New York, NY, USA), p. 765–781, Association for Computing Machinery, 2020.

[9] "HPE VAN SDN controller software." https://support.hpe.com/hpesc/public/docDisplay?docId=emr_na-c03967699. Last accessed on 02-03-2021.

[10] C. B. Hauser and S. R. Palanivel, "Dynamic network scheduler for cloud data centres with sdn," in *Proceedings of The10th International Conference on Utility and Cloud Computing*, UCC '17, (New York, NY, USA), p. 29–38, Association for Computing Machinery, 2017.

[11] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, "The design and implementation of open vswitch," in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, NSDI'15, (USA), p. 117–130, USENIX Association, 2015.

[12] "iperf - the ultimate speed test tool for tcp, udp and sctp." https://iperf.fr/. Last accessed on 11-03-2021.

[13] J. Wang, Y. Jiang, Y. Ouyang, C. Li, Z. Xiong, and J. Cai, "Tcp congestion control for wireless datacenters," *IEICE Electronics Express*, vol. 10, no. 12, pp. 20130349–20130349, 2013.

[14] D. Arora, T. Benson, and J. Rexford, "Proactive routing in scalable data centers with paris," *DCC 2014 - Proceedings of the ACM SIGCOMM 2014 Workshop on Distributed Cloud Computing*, 08 2014.

[15] T. Mizrahi, O. Rottenstreich, and Y. Moses, "Timeflip: Scheduling network updates with timestamp-based tcam ranges," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 2551–2559, IEEE, 2015.

[16] A. Tavakoli, M. Casado, T. Koponen, and S. Shenker, "Applying nox to the datacenter," in *HotNets*, 2009.

[17] P. Mell and T. Grance, "The nist definition of cloud computing," Tech. Rep. 800-145, National Institute of Standards and Technology (NIST), Gaithersburg, MD, September 2011.

[18] A. Agrawal and C. Kim, "Intel tofino2 – a 12.9tbps p4-programmable ethernet switch," in *2020 IEEE Hot Chips 32 Symposium (HCS)*, pp. 1–32, 2020.

[19] M. Dimolianis, A. Pavlidis, and V. Maglaris, "A multi-feature ddos detection schema on p4 network hardware," in *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pp. 1–6, 2020.

[20] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Software-defined networking security: pros and cons," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 73–79, 2015.

[21] "P4 language consortium." https://p4.org/. Last accessed on 15-03-2021.

An Overview of Privacy-preserving Genomic Data Processing Methods

Xiya Duan (s3700283) Fabian Prins (s3460509)

Abstract—The development of genetic analysis allows people to better predict or diagnose diseases through genetic testing. However, there might be privacy leakage on processing genomics data, exposing a person and their relatives, which may also cause a series of social problems such as genetic discrimination. Therefore, a privacy-preserving method is necessary for testing, storing and sharing genomics data. This is referred to as GWAS (genome-wide association studies).
We take a look at state-of-the-art cryptographic methods that allow computations on genetic data, without exposing the genetic contents of the original data. One possible method for this is homomorphic encryption (HE). HE allows a secure transformation from the original data to encrypted data, while still allowing for computations to be made without decrypting it [3].
Various protocols have been introduced that aim to make this possible: *Private, Authorized, and Fast Personal Genomic Testing* (PAPEETE) [12], HEAAN (an approximate homomorphic encryption scheme) [10], and Bonte et al.'s HE and secure multiparty computation (MPC) implementations [5]. These methods among others aim to make privacy-preserving GWASs possible with sufficient precision and speed. In this paper, we review Perillo et al.'s PAPEETE protocol and Kim et al.'s HEAAN implementation of semi-parallel GWAS. We discuss their strengths and weaknesses, which provide an idea of the present and future state of homomorphic encryption for GWAS.

Index Terms—Homomorphic encryption, genome-wide association studies.

1 Introduction

With the rapid development of computing and communication technology in recent years, cheap and fast gene sequencing has become possible. The required time for whole-genome sequencing has been reduced from the original 13 years to only one day now, and its cost has also been reduced from 3 billion US dollars to less than 1 thousand [17]. As a result of the reduced cost of genome sequencing, it is possible to use genome data in other domains, including personalized healthcare, clinical prediction, Direct-To-Customer, and allergen testing [4, 18]. A typical application scenario is the analysis, detection, and vaccine research of COVID-19, a coronavirus which started a global pandemic. In China, through rapid batch nucleic acid testing of millions of people, the Chinese Centers for Disease Control and Prevention (CCDC) can rapidly determine the scope of COVID-19 infection and cut the chain of transmission [2, 13].

However, the widespread usage of genomic data also raises the concern of privacy issues. An example of genomic data in GENOME-WIDE ASSOCIATION STUDIES (GWAS) are SINGLE-NUCLEOTIDE POLYMORPHISMS (SNPs). An SNP represents a genome variation at a particular position in DNA sequence(as shown in Figure 1) and shared by at least 1% of a population [14]. In GWAS algorithms, a SNP is a value indicating whether a certain variation is present (`1` for presence in one chromosome, `2` for both chromosomes) or not (`0`) [15]. Consequently, an SNP vector represents a list of genome variations and their presences. An individual's genome, e.g. represented by SNPs, provides information about what genetic diseases they are carrying, their susceptible diseases, and personality traits amongst other information. Additionally, because genomic data is unique, it can be used to identify individuals in forensic science [11]. However, this means that leakage of genomic information may lead to employment or social discrimination. Moreover, a data leak affects the victim's relatives as well, because of the similarity in genomic data [11]. It is important to note that a genomic data leak has permanent consequences, since people's genomic data does not change. This is different from other sensitive information such as usernames and passwords. Consequently, as science learns more about genomes and what information they have on an individual, attackers can extract this new information as well from already leaked genomic data. In the U.S., health records are considered to be privacy protected when there is no information about identifying attributes such as name and date of birth [11]. However, due to the identifying nature of genomic data and the progress in genomic research, this is clearly not enough. In the future, genomic data might even be used to reconstruct an individual's face with high accuracy. Hence, it is problematic if the genomic data is not used securely.

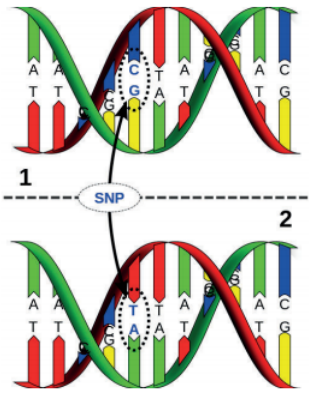


Fig. 1. SNP with alleles C and T [1].

In this paper, we aim to bring more awareness to privacy preservation of genomic data and compare different processing methods. Following this introduction, this paper is constructed as follows:

- Section 2: background. Here we provide context about current privacy-preserving methods for genomic data computations, their advantages and disadvantages. We further explain the concept of HOMOMORPHIC ENCRYPTION (HE) as we will be looking at state-of-the-art HE solutions in the following section.
- Section 3 is a case study of two novel HE works in the field of privacy-preserving genomic data processing.
- Section 4: discussion. In this section we discuss the strengths and weaknesses of these methods.

- *Xiya Duan is a MSc Computing Science student at the University of Groningen, E-mail: x.duan.1@student.rug.nl.*
- *Fabian Prins is a MSc Computing Science student at the University of Groningen, E-mail: f.l.prins@student.rug.nl.*

- Section 5: conclusion. We conclude our case study with a few words about the present state and the future of HE for GWAS.

2 Background

Berger et al. discuss the current state of privacy-preserving genomic computing in their 2019 editorial paper [3]. They mention three different frameworks that are currently used for the securely sharing of biomedical data, namely secure multiparty computation (MPC), homomorphic encryption (HE) and hardware-based approaches.

2.1 MPC

Secure multiparty computation (MPC) allows multiple entities to cooperatively perform computations on genomic data without exposing the original input data [3]. Each entity receives a securely shared part of the input data, it performs the requested computations and securely returns the result to another entity. During this process none of the computation entities are in possession of the complete data, hence they can not reconstruct it and therefore the data remains secure. The problem with these systems is the complexity of organizing a secure multiparty system [3], as well as the reliability on these parties.

2.2 Homomorphic encryption

The second framework, homomorphic encryption (HE), is an encryption form which allows computation on encrypted data without having access to the private key, while the computed result remains encrypted. Compared to MPC, HE is a lot easier to set up, as all encrypted data can be send to a single computation entity. However, encrypted data introduces significant computational overhead [3].

According to the allowed types and numbers of operations, homomorphic encryption schemes are classified to three main types [6]:

- *Partially Homomorphic Encryption*, which supports only one type of operation, either addition or multiplication, to be performed on encrypted data for unlimited times. One of the Partially Homomorphic Encryption schemes is known as the ElGamal cryptosystem, which is proposed by ElGamal in 1985 [8].

- *Somewhat Homomorphic Encryption*, which allows both addition and multiplication on encrypted data within limited times before the result becomes inaccurate by the noise produced during each operation. One of the Somewhat Homomorphic Encryption schemes is HEAAN (Homomorphic Encryption for Arithmetic of Approximate Numbers) proposed by Cheon, Kim, Kim and Song (CKKS) which implements an approximate HE [7].

- *Fully Homomorphic Encryption*, which supports diverse operations to be performed on encrypted data for unlimited times. It was first proposed by Craig Gentry in 2009, but it is still in the development stage [9].

2.3 Hardware-based approaches

Lastly, there is the concept of hardware-based approaches. Here, data is isolated into a protective enclave. They have the advantage over MPC and HE in terms of speed, as it uses unencrypted data during the computations. Nevertheless, it is not completely secure, as demonstrated by successful security attacks on these systems [3].

3 Privacy-preserving genomic data processing methods

In this section, we will elaborate two state-of-the-art homomorphic encryption protocols or algorithms that can be used for privacy-preserving genomic data processing. These methods aim to provide faster and/or more secure methods than conventional methods. PAPEETE is a secure and efficient protocol for personal genomic tests. Modified semi-parallel GWAS for HEAAN is a protocol used to find which Single Nucleotide Polymorphisms survive testing corrections, i.e. which SNPs give information about an individual's phenotypes, such as drug response.

Work	Privacy	Authorization	Efficiency	Weighted Avg
(Baldi et al., 2011)	✓	✓	✓	✗
(Ayday et al., 2013)	✓	✗	✗	✓
(Danezis and De Cristofaro, 2014)	✓	✗	✓	✓
PAPEETE	✓	✓	✓	✓

Fig. 2. Comparison between PAPEETE and other protocols[12]

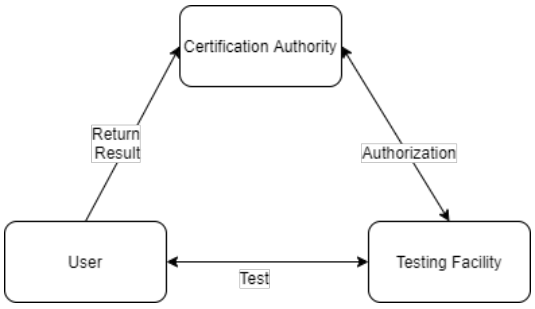


Fig. 3. PAPEETE architecture [12]. The test is authorized by Certification Authority (CA) and performed by User locally. The result is returned to CA for decoding and finally decrypted by the Test Facility.

3.1 Private, Authorized, and Fast Personal Genomic Testing (PAPEETE)

Based on additively Homomorphic Encryption, Angelo Massimo Perillo and Emiliano De Cristofaro proposed the PAPEETE (Private, Authorized, fast PErsonal gEnomic TEsting) protocol applied for personal genomic tests. PAPEETE aims to establish a personal genome test mode, performed as weighted average computing on target SNPs, that can simultaneously achieve privacy, authenticity, and efficiency. As shown in Figure 2, compared with other protocols, PAPEETE is the first protocol which achieves all these requirements[12].

- *Privacy*. The privacy of both the user and the test facility should be protected concurrently. On the one hand, the test should be performed only on the objective genomic data instead of the entire genome. Additionally, only the test result instead of the raw genomic information should be exposed to the testing facility. On the other hand, the test specifics should also be kept secret as they might be relevant to intellectual property [12].

- *Authenticity*. Due to the sensitivity of genomic information, the objective weight and positions of a test should be authorized by a trusted third party in order to ensure that the user's genomic information is treated appropriately.

- *Efficiency*. As the number of genes on chromosomes might reach a number of hundreds of thousands or even millions, the test mode should be able to process such a large size dataset.

3.1.1 PAPEETE Architecture

The high level architecture of the PAPEETE protocol is illustrated in Figure 3. There are three entities involved in PAPEETE:

- a User, who takes the authorized test from the testing facility without exposing their genomic information.

- a Testing facility, which wants to perform genomic tests without exposing the target testing position and the relevant weights.

- a Certification Authority, which is trusted to verify the test from a testing facility and process the user's testing result.

3.1.2 AH-ECC cryptosystem

Since the PAPEETE protocol is based on Additively Homomorphic Elliptic Curve based ElGamal Cryptosystem (AH-ECC), it is necessary to introduce the AH-ECC before presenting the PAPEETE protocol. The original ElGamal encryption scheme, proposed by ElGamal[8], is multiplicative homomorphic and therefore implemented in elliptic curves to obtain the additively homomorphic property[16]. Generally, the AH-ECC consists of three parts, which are explained as follows:

- *KeyGen*. At first, a proper elliptic curve E of the order q is selected and the point generator G is determined accordingly. Then, randomly pick a value x from the original order q as the private key. The public key P_k can be generated by $P_k = xG$.
- *Encryption*. Given a plaintext m and a random $k \in [1, n-1]$ where n is the order of E, the corresponding ciphertext C can be obtained by $C = (R, S) = (kG, mG + kP_k)$.
- *Decryption*. Firstly, we can obtain the mapped value of m by $mG = -xR + S$. Then, a reverse mapping function can be used to extract m from mG.

3.1.3 PAPEETE protocol

In this section, we will present the workflow of PAPEETE protocol, which consists of two main parts: authorization and test.

First of all, we assume the tests can be expressed as a weighted average of the SNPs (Single Nucleotide Polymorphisms), which is formed as:

$$R(X) = \frac{\sum_i w_i * P[X|SNP_i]}{\sum_i w_i} \tag{1}$$

where $R(X)$ represents the result of test X, w_i represents the weight and $P[X|SNP_i]$ represents the appearance of the SNP in chromosomes [12].

Authorization Before starting a test, the testing facility is obligated to acquire authorization from the certification authority (CA) which not only verifies whether the test is credible, but also encrypts the weight, ensuring that the test facility can only get an extracted result from certification authority instead of the user. The detailed process is explained step by step as follows:

1. Given an order q, the certification authority needs to choose a pair of parameters (e, d) subject to $e = 1/d (\text{mod } q)$ and keeps them confidential.
2. The testing facility sends all weights w_i at position i to CA for authorization.
3. The authorized weights are transformed by an exponentiation defined by $W_i = G^{i \cdot e} \cdot G^{w_i \cdot e} \cdot G^e$, where G is the generator in public parameters.
4. The authorized weights are sent back to the testing facility.
5. The testing facility performs the encryption of AH-ECC to the authorized weights and outputs the encryption ct_i.

Test After the authorization, the test facility is allowed to run the test on the user's genomic data. We assume the user already has the sequenced genomic data $SNP_1, SNP_2, \ldots, SNP_n$, while the testing facility holds the encrypted and authorized weights $ct_1, ct_2, \ldots, ct_n$ corresponding to each SNP. The test is performed in the following steps:

- At the beginning, the user initializes the output variables: the encrypted genomic data ct_{res}, the sum of the positions of the SNPs p_{res}, and the sum of all the SNPs s_{res} to 0.
- The testing facility then computes the ct_{res}, p_{res}, and s_{res} over all SNPs as:

$$ct_{res} = \sum_{i=0}^{n} ct_i \cdot SNP_i \tag{2}$$

$$p_{res} = \sum_{i=0}^{n} i \cdot SNP_i \tag{3}$$

$$s_{res} = \sum_{i=0}^{n} SNP_i. \tag{4}$$

- The user sends the obtained result to the certification authority.
- The certification authority decodes the test result with the secrete key d and sends the decoded information to the testing facility.
- The testing facility decrypts the decoded result and gets the final result.

3.1.4 Experiment and results

Angelo Massimo Perillo and Emiliano De Cristofaro made an experiment where they evaluated the perfomance of the PAPEETE protocol and compared it with Fast and Private Genomic Testing for Disease Susceptibility (FPGTDS), which is proposed by Danezis and De Cristofaro in 2014. Since the FPGTDS also has a weights encryption step, the comparison is divided into two parts: offline operations and online steps. The offline operations part represents the weights encryption or authorization, and the online steps part represents the test and decryption. Perillo et al. compared their PAPEETE protocol with FPGTDS, looking at time and bandwidth consumption for both offline and online parts. The result is shown in Table 1.

As can be observed from the Table 1, although added the authorization step, the PAPEETE achieves almost the same efficiency as FPGTDS and is feasible to be used in real world. Considering the time consumption for PAPEETE and FPGTDS are linearly related to the amount of SNPs, we conjecture that the measured time cost of offline operation in FPGTDS(3.85ms) might have a mistake on the units of measurement.

3.2 Privacy-preserving Approximate GWAS computation based on Homomorphic Encryption

Kim et al. (2020) developed a novel privacy-preserving GWAS algorithm which makes use of HEAAN (Homomorphic Encryption for Arithmetic of Approximate Numbers) as proposed by Cheon et al. (2017) and a modified version of the semi-parallel GWAS algorithm originally developed by Sikorska et al. (2013) [10, 7, 15]. The GWAS algorithm calculates the p-value for each single-nucleotide polymorphism (SNP) data, while correcting for covariates such as height, weight and age. That is, if a p-value for some SNP is very small (less than some threshold θ), then that SNP influences some target phenotype (e.g. physical trait or carried disease) with high confidence. Kim et al. use binary SNP instead of the 2-bit representation: $SNP \in (0, 1, 2)$. Hence, $SNP_i = 1$ indicates that genomic variation i is present in one or both chromosomes, while $SNP_i = 0$ indicates that the genomic variation is not present.

The dataset consists of an $n \times m$ input matrix consisting of $n = 245$ vectors (samples) with $m = 25{,}484$ binary SNP data each, and a binary vector $\boldsymbol{y}$ of length $n = 245$ where each column indicates whether a sampled individual has the phenotype of interest or not. This dataset is provided by the 2018 secure genome analysis competition hosted by Integrating Data for Analysis, Anonymization and SHaring (IDASH).

In practice, a data holder encrypts their data and sends it to a computation server. The server then does all but the last few steps in the algorithm, at which point it returns encrypted output data to the data holder, which decrypts it and performs the last few steps to compute the p-values with the output data. The algorithm proved to be scalable, fast, secure and accurate [10].

SNPs	Offline		Online		Bandwidth
	PAPEETE	FPGTDS	PAPEETE	FPGTDS	
1×10^3	3.88s	3.85ms	0.83s	0.82s	64.51KB
1×10^4	37.77s	37.40s	7.04s	7.03s	645.12KB
1×10^5	6.27m	6.22m	1.31m	1.31m	6.3MB
1×10^6	62.77m	62.21m	18.89m	18.88m	63MB

Table 1. Execution times and bandwidth consumption of PAPEETE and FPGTDS [12].

3.2.1 HEAAN encryption

Kim et al. encrypt the input data, a matrix of n samples (the rows) by m binary SNPs (the columns), column-wise with HEAAN encryption. HEAAN is a scheme developed by Cheon et al. [7] that uses a polynomial representation of vector data to do homomorphic computations. It consists of five functions: KeyGen(1^λ), Encrypt$_{pk}(m)$, Decrypt$_{sk}(\boldsymbol{c})$, Add$(\boldsymbol{c}_1, \boldsymbol{c}_2)$ and Multiply$(\boldsymbol{c}_1, \boldsymbol{c}_2)$.

- *KeyGen*. KeyGen(1^λ) takes as input the security parameter λ. It generates secret key sk, public key pk and evaluation key evk such that security attacks should take $\Omega(2^\lambda)$ bit operations [7].
- *Encryption*. Encrypt$_{pk}(m)$ takes as input a polynomial $m \in R$ and outputs ciphertext $\boldsymbol{c}$, which introduces small error e.
- *Decryption*. Decrypt$_{sk}(\boldsymbol{c})$ takes as input the ciphertext $\boldsymbol{c}$ and outputs an approximation of original polynomial m, due to the error introduced during encryption.

To be able to use the HEAAN scheme for integer-valued SNP vectors, Kim et al. use canonical embedding to transform these vectors into an integer-rounded polynomial m for encryption to ciphertext $\boldsymbol{c}$. Rounding the polynomial to an integer representation introduces a large error, therefore Kim et al. first scale the real-valued polynomial by p bits to control the error [10]. Decryption is done with the inverse of the canonical embedding and scaling to get back an integer array from the polynomial m. In the case of GWAS with HEAAN, decryption is done at the end of the algorithm when the p-values are calculated. This means that the last steps are performed on the data holder's end instead of the computation server.

3.2.2 Modification of Sikorska et al.'s semi-parallel GWAS

Kim et al. made use of the semi-parallel GWAS algorithm by Sikorska et al. with modifications that reduce the amount of expensive matrix computations, while allowing for homomorphic encrypted input data. This has the effect of speeding up the process while maintaining sufficient accuracy. Kim et al. modified the Fisher Scoring algorithm and the parts of the algorithm where a large, computationally expensive matrix S^* is used. These steps are outlined in Figure 4. The algorithm is called 'semi-parallel' as it is not the same as parallel computation on multiple processors. Instead, it does calculations on the full input SNP matrix, which is faster than doing these calculations for each SNP individually [15].

Fisher Scoring First we consider the modification of Fisher Scoring in the original semi-parallel GWAS algorithm; step 1 in Figure 4. The original Fisher Scoring algorithm can be seen in Figure 5, where the steps that Kim et al. will modify are outlined in red. Fisher Scoring is an algorithm to solve maximum likelihood equations. It takes as input a covariate matrix X, phenotype vector $\boldsymbol{y}$, and total iterations constant **iter**. As can be seen in step 4 of the Fisher Scoring algorithm, the matrix inverse $(X^T W^{(t)} X)^{-1}$ is computed. Note that the inverse of a matrix A is calculated with $\frac{1}{\mathbf{det}(A)} \cdot \mathbf{adj}(A)$, where $\mathbf{det}(A)$ is the determinant of A, and $\mathbf{adj}(A)$ is the adjugate of A. This is an expensive operation when done in HE, as it is non-polynomial [10]. Instead, Kim et al. make an approximation using just the adjugate of the original matrix $X^T W^{(t)} X$ multiplied by a constant $\alpha > 0$ instead of the inverse of the matrix' determinant. This works because the term with the inverse matrix approaches zero during the iterative process of the

Algorithm 1 The Original Semi-Parallel GWAS

Input: SNP matrix $S \in \{0,1\}^{n \times m}$, covariate matrix $X \in \mathbb{Q}^{n \times k}$, phenotype vector $\boldsymbol{y} \in \mathbb{Q}^n$, and # iteration iter.
Output: p-value vector $\overrightarrow{\text{pval}} = (\text{pval}_1, \cdots, \text{pval}_m) \in \mathbb{Q}^m$.
1: $(\boldsymbol{\beta}, \boldsymbol{p}, W) \leftarrow \text{FisherScoring}(X, \boldsymbol{y}; \text{iter})$
▷ FisherScoring$(\cdot)$ is described in Algorithm 2
2: $\boldsymbol{v} \leftarrow \log(\boldsymbol{p}/(\mathbf{1} - \boldsymbol{p})) + (\boldsymbol{y} - \boldsymbol{p})/\text{diag}(W)$
3: $S^* \leftarrow S - X(X^T W X)^{-1} X^T W S$
4: $\boldsymbol{v}^* \leftarrow \boldsymbol{v} - X(X^T W X)^{-1} X^T W \boldsymbol{z}$
5: $\boldsymbol{c} \leftarrow {S^*}^T W \boldsymbol{v}^* \in \mathbb{Q}^m$
6: $\boldsymbol{d} \leftarrow \text{diag}({S^*}^T W S^*) \in \mathbb{Q}^m$
7: **for** $i = 1$ to m **do**
8: $\quad a_i \leftarrow -|c_i/\sqrt{d_i}|$
9: $\quad \text{pval}_i = 2 \cdot \int_{-\infty}^{a_i} \rho(x)dx$ ▷ $\rho(x) := \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right)$
10: **end for**
11: **return** $\overrightarrow{\text{pval}}$

Fig. 4. The original semi-parallel GWAS [10], developed by Sikorska et al. [15].

Fisher Scoring algorithm, and can therefore have a slightly different value. However, this comes at the cost of a few additional iterations for convergence. As the modified Fisher Scoring introduces this new constant α, it is added as additional input parameter.
Kim et al. replace step 3 and rewrite step 4 of Fisher Scoring before applying their α replacement of $\frac{1}{\det(X^T W^{(t)} X)}$ [10].
Firstly, Kim et al. use step 3 to store matrix $X^T W^{(t)} X$ as $U^{(t)}$:

$$3: \quad U^{(t)} \leftarrow X^T W^{(t)} X. \tag{5}$$

Secondly, $\boldsymbol{v}^{(t)}$ is reformulated as:

$$\boldsymbol{v}^{(t)} = \log\left(\frac{\boldsymbol{p}^{(t)}}{\mathbf{1} - \boldsymbol{p}^{(t)}}\right) + \frac{\boldsymbol{y} - \boldsymbol{p}^{(t)}}{\text{diag}(W^{(t)})} = X\boldsymbol{\beta}^{(t)} + \frac{\boldsymbol{y} - \boldsymbol{p}^{(t)}}{\text{diag}(W^{(t)})}. \tag{6}$$

The rewritten $\boldsymbol{v}^{(t)}$ replaces the old $\boldsymbol{v}^{(t)}$ in step 4 of the Fisher Scoring algorithm to get:

$$\boldsymbol{\beta}^{(t+1)} = (U^{(t)})^{-1} X^T W^{(t)} \left(X\boldsymbol{\beta}^{(t)} + \frac{\boldsymbol{y} - \boldsymbol{p}^{(t)}}{\text{diag}(W^{(t)})}\right) \tag{7}$$

$$= \boldsymbol{\beta}^{(t)} + (U^{(t)})^{-1} X^T (\boldsymbol{y} - \boldsymbol{p}^{(t)}) \tag{8}$$

$$= \boldsymbol{\beta}^{(t)} + \frac{1}{\det(U^{(t)})} \cdot \text{adj}(U^{(t)}) X^T (\boldsymbol{y} - \boldsymbol{p}^{(t)}). \tag{9}$$

Lastly, α replaces the inverse determinant such that step 4 becomes:

$$4: \quad \boldsymbol{\beta}^{(t+1)} \leftarrow \boldsymbol{\beta}^{(t)} + \alpha \cdot \text{adj}(U^{(t)}) X^T (\boldsymbol{y} - \boldsymbol{p}^{(t)}). \tag{10}$$

This concludes the modifications made to the Fisher Scoring algorithm.

Removal of large matrix S^* The second optimisation Kim et al. make is a modification in steps 2—10 of the original semi-parallel GWAS algorithm, as seen in Figure 4. The problem with the original algorithm is the computations involving the large matrix S^*. Because of its size, it takes a lot of computations to perform any operation, which slows down the algorithm. The objective is to calculate $\boldsymbol{c}$ from step 5 and $\boldsymbol{d}$ from step 6 without using S^*.

$$5: \quad \boldsymbol{c} \leftarrow {S^*}^T W \boldsymbol{v}^* \in \mathbb{Q}^m \tag{11}$$

$$6: \quad \boldsymbol{d} \leftarrow \text{diag}({S^*}^T W S^*) \in \mathbb{Q}^m. \tag{12}$$

Algorithm 2 FisherScoring

Input: Covariate matrix $X \in \mathbb{Q}^{n\times k}$, phenotype vector $\boldsymbol{y} \in \mathbb{Q}^n$, and # iteration iter.
Output: Coefficient vector $\boldsymbol{\beta} \in \mathbb{Q}^k$, fitted vector $\boldsymbol{p} \in \mathbb{Q}^n$, weight matrix $W \in \mathbb{Q}^{n\times n}$

1: $\boldsymbol{\beta}^{(0)} = \mathbf{0} \in Q^k$, $\boldsymbol{p}^{(0)} = \vec{0.5} \in \mathbb{Q}^n$, $W^{(0)} = 0.25 \cdot I_n$
2: **for** $t = 0$ to iter $- 1$ **do**
3: $\quad \boldsymbol{v}^{(t)} \leftarrow \log(\boldsymbol{p}^{(t)}/(\mathbf{1} - \boldsymbol{p}^{(t)})) + (\boldsymbol{y} - \boldsymbol{p}^{(t)})/\mathsf{diag}(W^{(t)})$
4: $\quad \boldsymbol{\beta}^{(t+1)} \leftarrow (X^T W^{(t)} X)^{-1} X^T W^{(t)} \boldsymbol{v}^{(t)}$
5: $\quad \boldsymbol{p}^{(t+1)} \leftarrow \sigma(X\boldsymbol{\beta}^{(t+1)})$
6: $\quad W^{(t+1)} \leftarrow$ diagonal matrix of $\boldsymbol{p}^{(t+1)} \odot (\mathbf{1} - \boldsymbol{p}^{(t+1)})$
7: **end for**
8: **return** $(\boldsymbol{\beta}^{(\text{iter})}, \boldsymbol{p}^{(\text{iter})}, W^{(\text{iter})})$

Fig. 5. The original Fisher Scoring [10] with steps that Kim et al. modify outlined in red.

Algorithm 4 Modified Semi-Parallel GWAS

Input: SNP matrix $S \in \{0,1\}^{n\times m}$, covariate matrix $X \in \mathbb{Q}^{n\times k}$, phenotype vector $\boldsymbol{y} \in \mathbb{Q}^n$, # iteration iter, and constant $\alpha > 0$.
Output: p-value vector $\overrightarrow{\mathsf{pval}} = (\mathsf{pval}_1, \cdots, \mathsf{pval}_m) \in \mathbb{Q}^m$.

1: $\boldsymbol{\beta}, \boldsymbol{p}, W \leftarrow \mathsf{ModifiedFisherScoring}(X, \boldsymbol{y}; \text{iter}, \alpha)$
2: $U \leftarrow X^T W X$, and compute $\mathsf{adj}(U), \det(U)$
3: $V \leftarrow X^T W S$
4: $\boldsymbol{c} \leftarrow S^T(\boldsymbol{y} - \boldsymbol{p})$
5: $\boldsymbol{d} \leftarrow \det(U) \cdot \mathsf{diag}(S^T W S) - \mathsf{diag}(V^T \mathsf{adj}(U) V)$
6: **for** $i = 1$ to m **do**
7: $\quad z_i \leftarrow \det(U) \cdot c_i^2/d_i$
8: $\quad \mathsf{pval}_i = 2 \cdot \int_{-\infty}^{-\sqrt{z_i}} \rho(x)dx$ ▷ Done in unencrypted state
9: **end for**
10: **return** $\overrightarrow{\mathsf{pval}}$

Fig. 6. Kim et al.'s modified semi-parallel GWAS [10].

Kim et al. first show that it is possible to calculate $\mathbf{diag}(S^{*T} W S^*)$ without using S^*. They start by rewriting how S^* is declared in step 3 of the original algorithm. They introduce two new matrices: $U = X^T W X$ and $V = X^T W S$. Hence, $S^* = S - XU^{-1}V$. With this in mind, they rewrite $S^{*T} W S^*$ as:

$$S^{*T} W S^* = (S - XU^{-1}V)^T W (S - XU^{-1}V) \tag{13}$$
$$= S^T W S - V^T U^{-1} V. \tag{14}$$

Kim et al. use the determinant and adjugate matrices of matrix U without having to inverse any term. Step 6: $\boldsymbol{d} \leftarrow \mathrm{diag}(S^{*T} W S^*)$ from the original algorithm is then calculated as follows:

$$\boldsymbol{d} \leftarrow \det(U) \cdot \mathrm{diag}(S^T W S) - \mathrm{diag}(V^T \mathrm{adj}(U) V). \tag{15}$$

Kim et al. continue by showing that it is possible to approximate $S^{*T} W \boldsymbol{v}^*$, which is used to calculate $\boldsymbol{c}$ in step 5. Their first observation is the fact that $S^{*T} W \boldsymbol{v}^*$ is equal to $S^T W \boldsymbol{v}^*$ due to the definitions of S^* and $\boldsymbol{v}^*$. With this rewrite they are able to derive the following step:

$$\boldsymbol{c} \leftarrow S^T(\boldsymbol{y} - \boldsymbol{p}) - S^T W X U^{-1} X^T (\boldsymbol{y} - \boldsymbol{p}). \tag{16}$$

Kim et al. note that the long term on the right is very small, since $X^T(\boldsymbol{y} - \boldsymbol{p})$ is close to zero due to Fisher Scoring. Therefore, they omit this term to speed up the algorithm, which is especially helpful due to the computationally expensive inverse matrix U^{-1} in that term. The finalised modified semi-parallel GWAS algorithm can be seen in Figure 6.

3.2.3 Homomorphic evaluation

In the evaluation section, Kim et al. show how their modification of the semi-parallel GWAS algorithm works for HEAAN encrypted SNP data. The original GWAS algorithm uses non-polynomial functions such as matrix operations and the use of a sigmoid (σ) function. However, they are able to replace these with polynomial based operations, or find a close approximation [10].

3.2.4 Experiments

Kim et al. already give the suggestion of a protocol for their secure GWAS algorithm in their experiments. The 'data holder' sends the encrypted data to a 'server' which performs the semi-parallel GWAS algorithm until either step 5 or 7, as can be seen in Figure 6. This choice depends on the need for more speed, or accuracy and security respectively [10]. If the server stops at step 5, the encrypted numerator $\boldsymbol{c}$ and denominator $\boldsymbol{d}$ are returned as output. Kim et al. mention these variables contain more information than just p-values, however, it is considered "very hard to extract any important information" out of them [10]. On the other hand, if the algorithm stops at step 7, the server outputs the encrypted squared z-scores, which have the same information as the p-values [10]. The data holder receives these outputs and decrypts it with its private key and does the final steps to calculate the p-values.

Kim et al. perform two different experiments:

- Exp I: The server returns output after step 5.
- Exp II: The algorithm returns output after step 7.

Data	Params Exp	iter	Comp. time	F_1 Score TH: 10^{-2}	TH: 10^{-5}	TH: 10^{-12}
iDash_Test	I	1	13 min*	0.960	0.969	0.243
	I	2	27 min	0.985	0.985	0.955
	I	**4**	**32 min**	**1.000**	**0.999**	**0.997**
	II	**4**	**52 min**	**1.000**	**0.999**	**0.998**
iDash_Eval	**I**	**4**	**38 min**	**0.998**	**0.995**	**0.992**
	II		**62 min**	**0.998**	**0.996**	**0.994**

*: We used more streamlined parameter; $\log N = 16, L = 950, p = 50, \mathsf{h} = 91$.

Fig. 7. Results of experiments I and II [10].

Both experiments are performed on two different subsets from the IDASH dataset: iDash_Test, which has a reduced number of SNPs (namely 10,643), and iDash_Eval, which is used to evaluate the algorithm in the IDASH competition [10]. Both subsets have 245 samples and 3 covariates. The results are show in Figure 7. **iter** denotes the number of iterations in Fisher Scoring, **Comp. time** denotes the running time of the algorithm in encrypted state, and **TH** denotes the threshold of p-values for classification [10]. As can be seen, Exp I is around 20 minutes faster than Exp I, but it is slightly less accurate. It is therefore up to the user to decide what they consider to be most important: speed, or accuracy and security.

4 Discussion

Perillo and Cristofaro have provided a protocol for calculating SNP weighted averages. Since the calculation is performed by the user with encrypted weights, the increasing number of samples would not affect the time cost of the test, which increase the scalability of the protocol. Due to the additively homomorphic property of the AH-ECC cryptosystem, the obtained weighted average stays confidential. Therefore, the test can be performed without exposing the raw genomic data.

Compared with FPGTDS, PAPEETE introduces the certification authority, which guarantees the test can only be performed on allowed SNPs and protects the privacy of test facilities. Furthermore, the test step in FPGTDS can only be done on trusted hardware such as smart card, while PAPEETE allows users to take tests on their smartphone. We think it makes PAPEETE more convenient and more likely to become popular. However, although the test is performed by users locally, all test results need to be decoded by authority which is not scalable with the increasing amount of test. Thus, the efficiency of the decoding process may become the main factor limiting the efficiency of the test.

In a similar vein, Kim et al. are the first to provide a secure HEAAN implementation of the semi-parallel GWAS algorithm from Sikorska et al. [10]. Kim et al. have shown that the HEAAN scheme still produces highly accurate results, even with its error introducing nature.

The HEAAN implementation has the same scalability benefit as Sikorska et al.'s original semi-parallel GWAS algorithm, due to its use of the complete SNP matrix.

Kim et al. note that their approximation of $\alpha \approx 1/\det(U^{(t)})$ in their modification of Fisher Scoring requires further research to speed up the convergence rate as much as possible [10]. They have pointed out that α merely depends on the size of covariate matrix X, however, they do not go into detail about how they derived their value of α. All that is mentioned is that it was derived through experimenting, which presumably implies trial and error. Moreover, it is not clear how much of an impact α has on both the accuracy and speed of the algorithm.

5 Conclusion

In this paper we discussed why privacy-preserving is important in genomic research and how it can be achieved. We investigated several privacy-preserving genomic data processing methods like secure multiparty computation (MPC), homomorphic encryption (HE) and hardware based approaches. We also elaborate two HE implementations, PAPEETE and privacy-preserving GWAS algorithm based on HEAAN.

PAPEETE is the first personal genome test protocol which simultaneous achieves efficiency, authenticity and privacy. However, the efficiency of decoding test result limits the performance of the whole system with the increasing amount of test. As future work, the decoding step can be improved by supporting parallel computing or other possible ways to increase the scalability of the protocol.

Kim et al. have shown that it is possible to apply HEAAN to modern algorithms such as Sikorska et al.'s semi-parallel GWAS. The resulting, scalable and secure algorithm gives the user freedom to choose between optimal security and speed. However, the algorithm can be further improved by optimizing approximations such as α in Fisher Scoring.

Nevertheless, both works have given a positive outlook for the use of homomorphic encryption, particularly in the field of genomic studies.

References

[1] E. Ayday, J. L. Raisaro, J.-P. Hubaux, and J. Rougemont. Protecting and evaluating genomic privacy in medical tests and personalized medicine. In *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, WPES '13, page 95–106, New York, NY, USA, 2013. Association for Computing Machinery.

[2] BBC News. Coronavirus: Wuhan draws up plans to test all 11 million residents. https://www.bbc.com/news/world-asia-china-52629213, May 2020.

[3] B. Berger and H. Cho. Emerging technologies towards enhancing privacy in genomic data sharing. *Genome Biology*, 20(1):128, Jul 2019.

[4] S. J. Bielinski, J. E. Olson, J. Pathak, R. M. Weinshilboum, L. Wang, K. J. Lyke, E. Ryu, P. V. Targonski, M. D. Van Norstrand, M. A. Hathcock, et al. Preemptive genotyping for personalized medicine: design of the right drug, right dose, right time—using genomic data to individualize treatment protocol. In *Mayo Clinic Proceedings*, volume 89, pages 25–33. Elsevier, 2014.

[5] C. Bonte, E. Makri, A. Ardeshirdavani, J. Simm, Y. Moreau, and F. Vercauteren. Towards practical privacy-preserving genome-wide association study. *BMC Bioinformatics*, 19(1):537, Dec 2018.

[6] P. Chaudhary, R. Gupta, A. Singh, and P. Majumder. Analysis and comparison of various fully homomorphic encryption techniques. In *2019 International Conference on Computing, Power and Communication Technologies (GUCON)*, pages 58–62, 2019.

[7] J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In T. Takagi and T. Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham, 2017. Springer International Publishing.

[8] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology*, pages 10–18, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[9] C. Gentry et al. *A fully homomorphic encryption scheme*, volume 20. Stanford university Stanford, 2009.

[10] D. Kim, Y. Son, D. Kim, A. Kim, S. Hong, and J. Cheon. Privacy-preserving approximate gwas computation based on homomorphic encryption. *BMC Medical Genomics*, 13, 07 2020.

[11] M. Naveed, E. Ayday, E. W. Clayton, J. Fellay, C. A. Gunter, J.-P. Hubaux, B. A. Malin, and X. Wang. Privacy in the genomic era. *ACM Computing Surveys (CSUR)*, 48(1):1–44, 2015.

[12] A. M. Perillo and E. De Cristofaro. Papeete: Private, authorized, and fast personal genomic testing. pages 650–655, 07 2018.

[13] Reuters Staff. China's qingdao orders city-wide testing after new covid-19 infections. https://www.reuters.com/article/health-coronavirus-china-cases-idUSKBN26X042, Oct 2020.

[14] Scitable by nature education. single nucleotide polymorphism (snp).

[15] K. Sikorska, E. Lesaffre, P. F. Groenen, and P. H. Eilers. Gwas on your notebook: fast semi-parallel linear and logistic regression for genome-wide association studies. *BMC Bioinformatics*, 14(1):166, May 2013.

[16] O. Ugus, A. Hessler, and D. Westhoff. Performance of additive homomorphic ec-elgamal encryption for tinypeds. *6. Fachgespräch Sensornetzwerke*, page 55, 2007.

[17] K. A. Wetterstrand. The cost of sequencing a human genome. https://www.genome.gov/about-genomics/fact-sheets/Sequencing-Human-Genome-cost. Last updated: 2020-12-07.

[18] M. S. Williams, C. O. Taylor, N. A. Walton, S. R. Goehringer, S. Aronson, R. R. Freimuth, L. V. Rasmussen, E. S. Hall, C. A. Prows, W. K. Chung, et al. Genomic information for clinicians in the electronic health record: Lessons learned from the clinical genome resource project and the electronic medical records and genomics network. *Frontiers in genetics*, 10:1059, 2019.

State-of-the-Art Fuzzing: Challenges, Limitations and Improvements

Nik Dijkema, *s3230007* and Zamir Amiri, *s2780542*

Abstract—Over the last decade, fuzzing has become a widely-used and very effective method for vulnerability detection in the field of cybersecurity. Fuzzing is the intuitive process of repeatedly executing a program with potentially malformed input to detect undesired program behaviour.
In this paper, we present a review of the background of fuzzing and some of the recent advances in the fuzzing domain, such as transformational and compositional fuzzing. We discuss different fuzzing approaches, focus on their strengths, limitations and the current challenges in the fuzzing domain. We found that the body of research on fuzzing has become so large, that it has become difficult to obtain a broad, coherent view. We argue that, from a broad perspective, the domain of fuzzing has become too vast to be able to propose concrete solutions for the existing challenges, and that these challenges have become research topics in and of themselves. Furthermore, when looking at the topic from an accessibility point of view, we argue that there is significant room for improvement. We suggest that a more user-friendly approach to documentation for fuzzers should be introduced, such that this field of research may blossom.

Index Terms—Fuzzing, cybersecurity, vulnerability detection.

1 Introduction

With the ever-continuing rise of digitization, the importance of information security increases as well. While anti-virus software has long fought a battle against malware, there are still no viable defence programs against cyber attacks. An example of an attack strategy used by cyber criminals is to scan for software bugs, such as buffer-overflows, and exploit them. Therefore, locating and fixing bugs in software, before attackers discover them, is crucial. In recent years, vulnerability detection through fuzzing has gained a large foothold in the field of cybersecurity. Fuzzing is a technique in which programs are tested by repeated execution with generated inputs. These inputs may be malformed syntactically or semantically, and might induce incorrect program behaviour, crashes or other correctness policy violations [16]. Over the years, the primary application of fuzz testing has been finding security-related bugs, however, nowadays fuzzing has also found its way into other fields, such as finding performance issues in programs [13].

The domain of fuzzing has become very large, and many different strategies and applications exist. Through our research, we strive to answer the following research questions:

RQ1 What are the 'hard' fuzzing challenges for which new solutions are needed?

RQ2 Can we propose innovative solutions to address these hard challenges?

We present a review of four different fuzzing paradigms; (1) *mutational fuzzing*, (2) *generational fuzzing*, (3) *transformational fuzzing* and (4) *compositional fuzzing*. We review different approaches of fuzzing algorithms within each paradigm, the strengths and weaknesses of different techniques and we discuss the challenges that arise. The contribution of this paper is, after establishing the state-of-the-art, to propose improvements to the field of fuzzing, for future work to build upon.

- *Nik Dijkema, MSc Student Computing Science at University of Groningen, E-mail: n.p.dijkema@student.rug.nl.*
- *Zamir Amiri, MSc Student Computing Science at University of Groningen, E-mail: a.z.amiri@student.rug.nl.*

1.1 Terminology and definitions

Throughout the rest of this paper, we will use the terminology and definitions as established and described by Manès et al. in their 2019 fuzzing survey [16].

We define fuzzing as follows; an instance of *fuzzing* is performed on a *Program Under Test (PUT)*. Fuzzing is the execution of the PUT with inputs generated from the *fuzz input space*. *Fuzz testing* is the testing of a PUT through fuzzing. A *fuzzer* is a program that performs fuzz testing on a PUT. We refer to a single instance of fuzz testing as a *fuzz campaign*. The parameter value(s) of a fuzzer that control the fuzzer's algorithm in a fuzz campaign is a *fuzz configuration*. The program that decides whether a fuzzing execution of the PUT violates correctness policies is called the *bug oracle*, and may be part of the fuzzer itself. *Code coverage* refers to the percentage of the PUT's code that has been activated over the course of a fuzz campaign.

1.2 Paper structure

In the next section (Section 2) we provide relevant background knowledge regarding fuzzing. After that, in Section 3, we provide an overview of the different fuzzing paradigms, we lay out current challenges in fuzzing, and discuss potential solutions to these challenges. The discussion is covered in Section 6, and finally, we present our conclusions in Section 7.

2 Background

Before we proceed to our detailed review of fuzzing techniques from the different fuzzing paradigms, we first provide some background information about fuzzing. Fuzzing was first proposed in the early 1990s by Miller et al. [17] and has been increasingly used since then.

2.1 Fuzzer Taxonomy

Fuzzing techniques have been categorised in three classes, namely *black-* [28], *grey-* [21] and *white-box* fuzzers [6]. The performance of fuzzers from each class can differ vastly, therefore it is important to mention their differences.

White-Box The term white-box denotes techniques in which a user or program has full knowledge of the inner workings of a target, the PUT in the case of fuzzing. The fuzzer is able to perform program analysis and instrumentation (see Section 2.3), and is therefore better capable of fine-tuning the fuzzing process. An important thing to note is that white-box approaches almost always have a larger overhead and are slower than other fuzzer types.

Black-Box On the other end of the spectrum, black-box fuzzers are fuzzers that have no knowledge of the internals of the PUT. In general, these types of fuzzers only look at the input and output behaviour of the PUT. Black-box techniques are much more inline with real-world attacks, as most cyber-attacks are executed by attackers with no internal knowledge of the target. Therefore, a vulnerability found by a black-box fuzzer should always be given a higher severity level then an equal vulnerability found by a white-box fuzzer.

Grey-Box Some fuzzers take a middle-ground approach by taking some limited degree of internal information about the PUT into account. These fuzzers are faster in execution than white-box fuzzers, as they have less fine-grained control over the fuzzing process, but instead approximate the internals of the PUT and tune the test cases accordingly.

2.2 General Fuzzing Model

The fuzzing process can be split into several components, as described in more detail in [16]. These components are; `Preprocess`, `Schedule`, `InputGet`, `InputEval`, `ConfUpdate` and `Continue`. An illustration of the general fuzzing process is shown in Figure 1. A short explanation of each component follows.

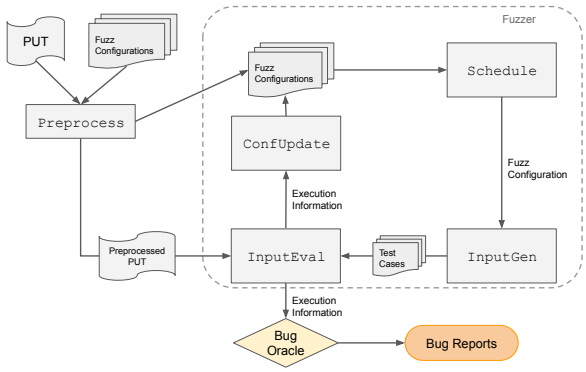


Fig. 1. The general fuzzing process.

Preprocess In the `Preprocess` stage, a fuzzer may perform various preprocessing actions, depending on the class of the fuzzer (see Section 2.1). In this step, the user supplies the fuzzer with a set of fuzz configurations as input. These configurations usually consist of (a) valid input(s) for the PUT, called *seed(s)*, and a set of fuzzer parameters. The fuzzer can perform different operations such as *seed selection* and *seed trimming*, in order to optimize the seeds for the next step. Other operations include instrumentation of the PUT (see Section 2.3), generating driver programs or constructing a model for the `InputGen` phase [23]. In some cases, the fuzzer may measure the execution time of the PUT when executing seeds. With this information, the fuzzer can perform numerous other preprocessing actions, such as calculating the time needed to execute the request and a time limit, $t_{elapsed}$ and t_{limit}, respectively. The preprocessing step returns a potentially modified set of fuzz configurations $\mathbb{C}$.

Schedule Given the set of fuzz configurations $\mathbb{C}$ (and potentially $t_{elapsed}$ and t_{limit}), the fuzzer can select a fuzz configuration to be used in the current fuzzing iteration. In many fuzzers, the `Schedule` step is a lot more sophisticated, such that the fuzzer is better able to steer the fuzzing campaign by cleverly prioritizing certain fuzz configurations, for example by using code coverage information (see Section 3.1.1).

InputGen `InputGen` takes the selected fuzz configuration and returns a set of new test cases that are based on the configuration. The exact input generation strategy depends on the class of the fuzzer and the technique it uses.

InputEval `InputEval` executes the test cases on the PUT, and evaluates the output and program behaviour. This could mean that it catches crashes, measures the execution time, measures code coverage, and checks if a bug was found using the bug oracle.

ConfUpdate `ConfUpdate` takes the current fuzz configuration(s) $\mathbb{C}$ and the execution information gathered from `InputEval`, and then decides whether $\mathbb{C}$ has to be updated. Not all fuzzers update their fuzz configuration(s) during the fuzzing process.

Continue `Continue` determines whether another iteration of the fuzz program should be performed.

2.3 PUT Instrumentation

As described by Huang et al. in [10], the essence of program instrumentation is the insertion of statements into a program, such that execution information is gathered. In the case of PUT instrumentation in fuzzing, the purpose is to enable the collection of much more detailed execution information, rather than simple output and/or crash behaviour. Detailed execution information includes, for example, stack- or call-traces, code coverage measurements, profiling or other event logging. PUT instrumentation is performed in the `Preprocess` step, and the resulting execution information is then used in the `InputEval` and/or `ConfUpdate` step. The collected execution information allows the fuzzer perform more complex analysis and fuzz more effectively. However, it is important to note that program instrumentation often adds (significant) execution overhead, depending on the instrumentation.

2.4 Symbolic and Concolic Execution

Symbolic [12] and/or *concolic execution* [25] are strategies that are nowadays commonly used in fuzzing methods. In the context of fuzzing, symbolic execution is the **white-box** process of executing the PUT with symbolic rather than concrete input values. Starting out, these symbolic inputs symbolize all possible input values. During the evaluation, the possible values are reduced based on the constraints in the PUT. The resulting path predicates can be used to generate concrete test-cases, for example through an SMT solver [5]. In fuzzing, dynamic symbolic execution is used, which is a combination of symbolic and concrete execution, hence the term concolic execution. In concolic execution, concrete execution states are used to reduce the complexity of the symbolic constraints, because symbolic execution is expensive [1]. Most symbolic execution engines prioritize path exploration such that unexplored paths have a higher priority [18]. However, symbolic execution faces a challenge known as the path explosion problem (see Section 4.2), where the number of paths or execution states increases drastically as the program scale gets larger [4, 14]. One of the ways in which symbolic execution engines address this problem, is through targeted symbolic execution [15], where symbolic execution is focused on paths that lead to interesting areas of code, which are usually specified by the user.

3 Fuzzing Paradigms

Now that we have established the relevant background knowledge regarding fuzzing, we will discuss different fuzzing approaches and algorithms.

3.1 Mutational Fuzzing

The first technique we discuss, is mutational fuzzing. Mutational fuzzers may belong to any of the *black-*, *grey-* or *white-box* classes. In mutational fuzzing, the test cases are generated from provided seeds, which are usually a correctly formed input, by partially mutating the seed. The input may be mutated by flipping bit values. The number of bits that are mutated is called the mutation ratio r. The bits can be flipped at random, or a more sophisticated method might be used. The performance of mutational fuzzers is highly dependent on the quality of the provided seed(s). Furthermore, as mentioned in [2], the classic random mutation testing does not perform well. The challenge here is that mutational fuzzers are often capable of obtaining only a low code

coverage [23]. Therefore, modern mutational fuzzers use an evolutionary algorithm to steer the mutation process [30, 22], which allows them to have better performance.

3.1.1 Instrumentation/Coverage Guided Fuzzing

In order for evolutionary mutation algorithms to be effective, a metric for 'seed fitness' is needed, this is where coverage-guided fuzzing comes in. Coverage-guided fuzzing was developed for *white-* and *grey-box* fuzzers, in order to increase code coverage by prioritizing seeds that explore new execution paths in the PUT. In coverage-guided fuzzing, test cases are generated by analysing the previously executed test case(s). The test cases that are identified to increase coverage, i.e. cover an unexplored execution path, are called *interesting seeds* (i.e. fitness). If a test-case is not marked as interesting, then it might be ignored or be given a lower priority in the next execution iteration. This feedback mechanism allows the fuzzer to effectively and autonomously exclude test-cases that cover previously explored paths, which leads to a more efficient use of time in a fuzzing campaign. An example of such a fuzzer, is the American Fuzzy Lop (AFL) fuzzer, which is one of the leading fuzzers in the field today [30, 9].

3.2 Generational Fuzzing

After mutational fuzzing, we discuss another fuzzing technique that focuses on input generation, generational fuzzing. Generation-based fuzzers create test cases that are based on a model that is provided by the user or inferred from the PUT [16]. The provided model usually contains some information about the structure of input that is accepted by the PUT. Therefore, generation-based fuzzers are almost never *back-box* methods. One example of a generation-based fuzzer is T-Fuzz [11], which takes in a protocol specification from the user. In short, one could say that generation-based fuzzing is like mutational fuzzing with constraints. This partially solves the problem of large numbers of possible mutations. However generation-based fuzzing also has its limitations. The model provided to the fuzzer might, for example, constraint the fuzzer by partially blocking paths that the mutational fuzzer would have been able to explore. The fuzzer may also require manual effort in order to define the model on which the test cases are built, which reduces automation.

3.3 Transformational Fuzzing

The next type of fuzzing algorithms we discuss are transformational fuzzers, sometimes called patch(-based) fuzzers. Transformational fuzzers belong to the *white-box* fuzzer category, and were introduced to overcome a significant drawback in the previously discussed traditional fuzzing; if inputs are rejected by the PUT at an early execution stage, for example in a parsing step, then the fuzzer is ineffective at reaching deeper program parts [26]. Figure 2 illustrates such a situation. This problem, for example, arises in mutational fuzzers when the PUT verifies input integrity using a checksum mechanism. If the fuzzer mutates parts of the included checksum value, or the rest of the input, the input will not pass such a checksum step early in the execution of the PUT. The main idea behind transformational fuzzing, is to attain better code coverage of deeper program execution paths, by mutating both the input and the PUT. Figure 3 provides an illustration of the model of a transformational fuzzer, T-Fuzz. These PUT transformations do not only apply to (early) checksum mechanisms, but also to hard-to-reach code deeper in the PUT. Such hard-to-reach code is often protected by difficult conditional checks, such as magic bytes, for which fuzzers have difficulty generating valid inputs that pass them [24, 26]. Transformational fuzzers solve this challenge by locating such checks, and transforming the program such that a given check is easily bypassed, allowing the fuzzer to fuzz the underlying deeper code paths. Transformational fuzzers use different strategies to discover these difficult checks and bypass or disable them. We will now discuss some transformational fuzzers in more detail.

TaintScope [26] The idea of bypassing checksum integrity tests was introduced by Wang et al. in 2010. Their checksum-aware fuzzer TaintScope locates bytes in well-formed inputs that are associated with security checks. When these checks are discovered in the PUT through

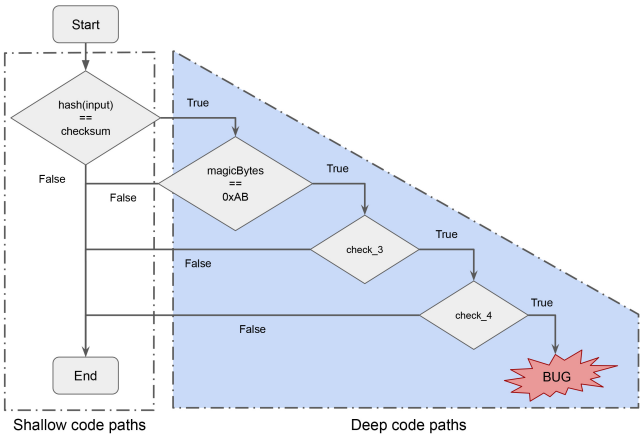


Fig. 2. An illustration of hard-to-reach code protected by difficult checks.

taint analysis, it patches the PUT to bypass the check. If, after patching and proceeding, a bug is found, the correct passing checksum value is calculated and corrected in the input, such that the crashing input is able to pass the check in the unmodified PUT, to verify the bug [16]. A clear limitation of TaintScope, is the fact that it is limited to only bypassing checksum mechanisms.

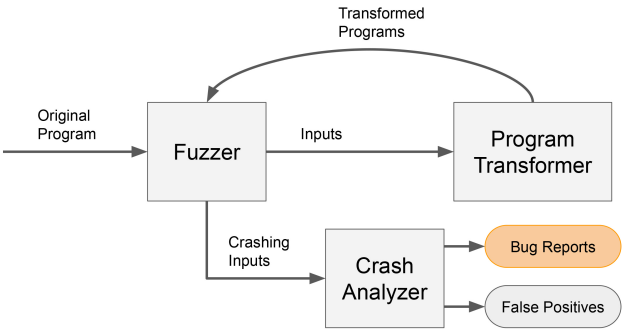


Fig. 3. An example of the workflow of a transformational fuzzer (T-Fuzz). *Figure adapted from [19].*

T-Fuzz [19] Next, we consider T-Fuzz. Unlike TaintScope, T-Fuzz does not limit its bypassing to checksum mechanisms, instead it extends this idea to bypassing a broader class of difficult-to-pass conditional checks [20]. T-Fuzz determines which checks do not affect the program logic of the PUT, which the authors call *Non-Critical Checks (NCCs)*. These NCCs may then be bypassed when a fuzz campaign gets stuck and is unable to further increase coverage by discovering new paths. A challenge here is NCC selection, to determine which NCC(s) should be prioritized to be transformed, otherwise *transformation explosion*, as Peng et al. call it [19], may occur. We further discuss transformation explosion in Section 4.3. T-Fuzz disables NCCs by flipping the checks through fast static binary patching. T-Fuzz uses the formal method of constraint solving in its bug validation process. It keeps track of the path constraints that lead to discovered bugs. After a bug is found, T-Fuzz solves this constraint trace to determine whether it is satisfiable or not. If the constraints are not satisfiable, the reported bug is a false positive. The fact that T-Fuzz only has to solve constraints when a bug is found, instead of when generating input, is one of its optimizations when compared to other methods (e.g. concolic execution). However, constraint solving is still expensive and not always possible in reasonable time, especially when traces get longer and/or contain more constraints.

DeepDiver [24] The last transformational fuzzer we discuss is DeepDiver, a hybrid fuzzing approach that combines patching with concolic execution to overcome the limitations of T-Fuzz. DeepDiver refers to the complex checks that it negates as *Roadblock Checks (RCs)*, examples of RCs are the previously described checksums, magic bytes, and, especially, nested complex sanity checks. DeepDiver locates RCs using coverage information and trace analysis. DeepDiver uses concolic execution to generate relevant concrete inputs based on the negated RCs, which are then fed back to the fuzzer, such that the fuzzer is able to explore deeper parts of the binary. Similar to T-Fuzz, DeepDiver also needs to validate found bugs to eliminate false positives. It does this using a similar strategy as T-Fuzz.

3.4 Compositional Fuzzing

Now, we move on to the last fuzzing paradigm we discuss; compositional fuzzing. The main idea behind compositional fuzzing is to fuzz program components (e.g. functions) individually. When a bug is found in one of the components, the fuzzer traces the program to determine whether or not the bug can be triggered from a program entry-point, such as the `main` function [18].

Wildfire [18] The compositional fuzzing algorithm we discuss, is Wildfire. Wildfire is claimed to be the first hybrid fuzzer that combines fuzzing with compositional analysis. Wildfire fuzzes C programs' individual (parameterized) functions to find bugs. It then uses targeted symbolic execution to determine if bugs can be triggered from a top-level function. Figure 4 depicts the general workflow of Wildfire. It works as follows [18]; after instrumenting and compiling the source code, it generates function drivers and respective seed inputs for all parameterized functions. These drivers are simple wrappers around the respective function, which are then fuzzed to generate test-cases. Since, during the fuzzing of individual functions, the calling context is ignored, the fuzzing of individual functions can be performed in parallel. Having obtained test-cases for the individual functions, Wildfire minimizes the sets of test-cases by removing those that execute redundant paths, and also removes any leading and trailing null bytes that do not affect the execution path. The minimized test-case inputs are then replayed to an instrumented version of the compiled object, to obtain crash reports and determine vulnerable functions. It then proceeds to use stack-trace matching and targeted symbolic execution to determine whether vulnerabilities can be triggered from an entry-point function and generate a concrete crashing test-case. In order to address the path-explosion problem (see Section 4.2), Wildfire first replaces vulnerable functions by summaries of their crash reports (value assertions that would trigger the vulnerability). This step restricts the paths of vulnerable functions to only those that lead to the vulnerability.

Like any fuzzer, Wildfire also has some limitations. Firstly, it only supports the analysis of functions that strictly have *non-pointer* and/or *single-pointer* type arguments. It is unable to fuzz functions that have double- or more pointer type arguments. Secondly, even though Wildfire cleverly reduces the chance of path-explosion by replacing vulnerable functions with value assertion summaries, path-explosion may still occur in higher-level functions, which could lead to exploitable vulnerabilities not being traced back to an entry-point, and therefore not being reported as exploitable (i.e. false negatives).

4 Fuzzing Challenges

Various challenges exist in the domain of fuzzing and many solutions for them have been proposed. Fuzzers from different paradigms have different strengths, but also have different limitations, as we have seen in the previous section(s). In most cases, new fuzzers are designed in an attempt to solve some of these challenges, but these solutions often incur new challenges and limitations. In this section, we discuss the current challenges that occur in the fuzzing sub-domains.

4.1 Code Coverage

The primary limitation for fuzzers in general, lies in the fact that their effectiveness is inherently constrained by the execution path coverage they achieve. It is well-known that attaining good code coverage becomes significantly more difficult in deeper parts of programs.

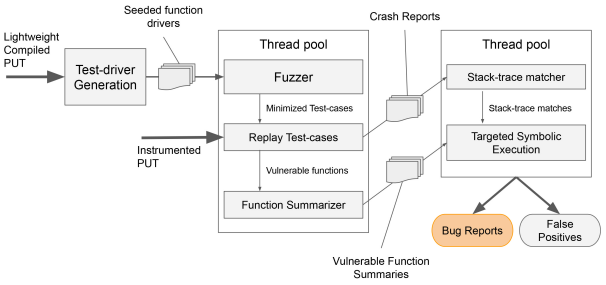


Fig. 4. A high-level illustration of the workflow of Wildfire. *Figure adapted from [18].*

Hard-To-Reach Code As discussed in Section 3.3, some code is often protected by hard sanity checks such as input file checksum mechanisms (Figure 2). It is difficult for fuzzers to generate input that passes such tests, which may block the fuzzer from fuzzing underlying program code.

4.2 Constraint Solving

As we have seen in the (*white-box*) fuzzing approaches we discussed in Section 3, modern fuzzers often require constraint (satisfiability) solving in some part of their process. This can either be to generate concrete inputs that execute certain program paths, or to verify bug candidates by determining whether they can be triggered in the PUT with appropriate input.

Path-explosion The challenge regarding constraint solving, is that it becomes extremely expensive (time consuming), if not impossible, when the set of constraints becomes large, which is likely to occur in large-scale programs. As program scale increases, the number of execution states grows explosively and is likely to exceed the capabilities of constraint solvers [4, 14].

4.3 Transformation Explosion

Transformation explosion is a problem that arises in transformational fuzzing. Transformation explosion refers to a rapid increase in the number of transformed PUT versions. It can limit the efficiency with which different program paths are explored and can also deplete hard-disk memory resources [24].

4.4 Automation

Another challenge in the field of fuzzing, is the reduction of manual effort in order to make fuzzers more accessible and user-friendly. Generally, this means that the fuzzers have to become more autonomous, such that they can function without manual input from a user with great expertise. One of the challenges with regards to automation is input generation. These days, most software make use of more complex structured input, such as Data Transfer Objects (DTO) in programs written in an object oriented language [3], for which input generation is more complex to effectively automate.

4.5 Scalability

Even though fuzzing, in many cases, is an embarrassingly parallel workload, the scalability of state-of-the-art fuzzers, such as AFL [30], on larger numbers of cores, has been shown to be surprisingly poor [29]. In the case of AFL, Xu et al. determined that this to be due to file system contention and the scalability of system calls such as `fork()`.

5 Potential Solutions and Improvements

Now that we have established the major challenges in the domain of fuzzing, we will discuss how these have been addressed in the past, and how further improvements to the field of fuzzing may be achieved.

5.1 Code coverage

Improving code coverage is one of the main focuses of fuzzing research and new fuzzing methods, because fuzzers are only as effective as the code coverage they achieve. Due to the volume of research, many approaches to increase code coverage have already been studied. Interesting approaches such as combining grammar- or model-based generational fuzzing with machine learning techniques, like neural networks to learn the input model, are being explored in recent years [7].

5.2 Constraint Solving

Many fuzzers use constraint-solving in one way or another. Therefore, in order to increase fuzzer performance, methods to solve constraint satisfaction problems more effieciently are required. However, this is an NP-complete [27] problem, and while research into better methods continues, this topic is beyond the scope of this paper.

5.3 Automation

The need for further development towards automation is clear. One interesting research area to help achieve this, may also be machine learning. Machine learning has been used in fuzzing in the past, with promising results [7]. There are existing fuzzers that are already decent with regard to automation [30], however, these fuzzers can still be further improved. Proper development of (fuzzing) software requires widespread use, so the fuzzers should be accessible to less-experienced developers. One of the ways in which the widespread use of fuzzers may be increased, is by extending these often complex software packages with a user-friendly UI, and distributing them as plugins for projects. This will make fuzzers more accessible to the less-experienced developer, which will likely result in more issue reports, feature requests, and ideally, further improvements.

5.4 Scalability

Reducing the time needed by fuzzers to obtain good code coverage and discover bugs, allows fuzzers to cover more code in the same amount of time and likely find more bugs. Increasing the speed of fuzzers is also important when we consider the increasing scale of programs and code-bases [29]. Scaling fuzzers to multiple processing cores or even distributed systems is a very promising way to achieve such performance increases. An example of scalability efforts in fuzzing is Google's ClusterFuzz [8]. ClusterFuzz is Google's scalable fuzzing infrastructure, which they also use to fuzz all their own products. Due to the nature of fuzzing, scalability is a very viable and important way to improve fuzzer performance.

6 Discussion & Future Work

During our research, we found that, due to the momentum that fuzzing has gained over the past years, the body of research on fuzzing has become very vast and diverse. Many different fuzzers exist and more are still being developed. Even though determining the current general challenges in the fuzzing domain was doable in the broad scope of this paper, proposing worthwhile approaches to potentially address these challenges proved to be more difficult than we anticipated. The active research on fuzzing is a sign of a lively scientific field, with much potential for progress, but it also makes it difficult to obtain a coherent view of the field and the research that has already been done. Knowing this, we now believe that, due to the abundance of different approaches in fuzzing, the research field would stand to benefit from review papers with a smaller scope, allowing for a more detailed compilation of the current knowledge and progress in the world of fuzzing.

During our research on this topic, it also became apparent to us that finding appropriate fuzzer tools for specific applications was rather difficult. Even though abundant information is available, we often found that, instead of finding answers to our questions, we ended up with more questions raised. When looking at this experience, we can only imagine how lost the average developer must feel when they attempt to explore the world of fuzzing and try to find a fuzzer that is suitable for their specific purposes. Therefore, we propose that a more user friendly approach to documentation of available fuzzer software be introduced, such that the usage of fuzzers increases rather than decreases.

7 Conclusion

Over the past years, fuzzing has become one of the main tools used for vulnerability detection in the field of cybersecurity. Many different fuzzing approaches have been proposed and developed, with a vast body of scientific literature. The purpose of our research was to answer **RQ1** and **RQ2** (see Section 1).

To answer the first research question, the current 'hard' challenges in fuzzing include *increasing code coverage*, *path- and transformation explosion*, *fuzz campaign guiding*, *constraint solving*, *automation*, and *scalability*. Different fuzzers are strong with respect to some of these challenges, but are lacking with regard to others.

Unfortunately, we found that the scope of our research was too broad for our goals. Therefore, we were unable to propose novel, innovative solutions to existing challenges within the time-frame of our research. Most of the challenges in fuzzing require a deep, detailed dive into existing literature and research, and should be considered to be research topics in and of themselves.

Acknowledgements

The authors wish to thank TNO for proposing this research subject. The authors also wish to thank Dr. Fatih Turkmen for taking the role of expect reviewer. Finally, the authors wish to thank the organization of the SC@RUG2021 symposium.

References

[1] S. Anand, E. K. Burke, T. Y. Chen, J. Clark, M. B. Cohen, W. Grieskamp, M. Harman, M. J. Harrold, and P. Mcminn. An orchestrated survey of methodologies for automated software test case generation. *J. Syst. Softw.*, 86(8):1978–2001, Aug. 2013.

[2] A. Arcuri, M. Z. Iqbal, and L. Briand. Random testing: Theoretical results and practical implications. *IEEE Transactions on Software Engineering*, 38(2):258–277, 2012.

[3] M. Boehme, C. Cadar, and A. ROYCHOUDHURY. Fuzzing: Challenges and reflections. *IEEE Software*, pages 0–0, 2020.

[4] C. Cadar and K. Sen. Symbolic execution for software testing: Three decades later. *Commun. ACM*, 56(2):82–90, Feb. 2013.

[5] L. De Moura and N. Bjørner. Satisfiability modulo theories: Introduction and applications. *Commun. ACM*, 54(9):69–77, Sept. 2011.

[6] P. Godefroid, M. Levin, and D. Molnar. Automated whitebox fuzz testing. 01 2008.

[7] P. Godefroid, H. Peleg, and R. Singh. Learn & fuzz: Machine learning for input fuzzing. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 50–59, 2017.

[8] Google. Clusterfuzz. `https://google.github.io/clusterfuzz/`.

[9] C.-C. Hsu, C.-Y. Wu, H.-C. Hsiao, and S.-K. Huang. Instrim: Lightweight instrumentation for coverage-guided fuzzing. 01 2018.

[10] J. Huang. Program instrumentation and software testing. *Computer*, 11:25–32, 1978.

[11] W. Johansson, M. Svensson, U. E. Larson, M. Almgren, and V. Gulisano. T-fuzz: Model-based fuzzing for robustness testing of telecommunication protocols. In *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation*, pages 323–332, 2014.

[12] J. C. King. Symbolic execution and program testing. *Commun. ACM*, 19(7):385–394, July 1976.

[13] C. Lemieux, R. Padhye, K. Sen, and D. Song. Perffuzz: Automatically generating pathological inputs. ISSTA 2018, page 254–265, New York, NY, USA, 2018. Association for Computing Machinery.

[14] J. Li, B. Zhao, and C. Zhang. Fuzzing: a survey. *Cybersecurity*, 1(1):6, Jun 2018.

[15] K.-K. Ma, K. Yit Phang, J. S. Foster, and M. Hicks. Directed symbolic execution. In E. Yahav, editor, *Static Analysis*, pages 95–111, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[16] V. J. M. Manès, H. Han, C. Han, S. K. Cha, M. Egele, E. J. Schwartz, and M. Woo. The art, science, and engineering of fuzzing: A survey. *IEEE Transactions on Software Engineering*, pages 1–1, 2019.

[17] B. P. Miller, L. Fredriksen, and B. So. An empirical study of the reliability of unix utilities. *Commun. ACM*, 33(12):32–44, Dec. 1990.

[18] S. Ognawala, F. Kilger, and A. Pretschner. Compositional fuzzing aided by targeted symbolic execution, 2019.
[19] H. Peng, Y. Shoshitaishvili, and M. Payer. T-fuzz: Fuzzing by program transformation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 697–710, 2018.
[20] H. Peng, Y. Shoshitaishvili, and M. Payer. T-fuzz: Fuzzing by program transformation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 697–710, 2018.
[21] V.-T. Pham, M. Böhme, A. E. Santosa, A. R. Căciulescu, and A. Roychoudhury. Smart greybox fuzzing. 2018.
[22] S. Rawat, V. Jain, A. Kumar, L. Cojocar, C. Giuffrida, and H. Bos. Vuzzer : Application - aware evolutionary fuzzing. In *NDSS'17*, NDSS'17, 2017.
[23] A. Rebert, S. K. Cha, T. Avgerinos, J. Foote, D. Warren, G. Grieco, and D. Brumley. Optimizing seed selection for fuzzing. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 861–875, San Diego, CA, Aug. 2014. USENIX Association.
[24] F. Rustamov, J. Kim, and J. Yun. Deepdiver: Diving into abysmal depth of the binary for hunting deeply hidden software vulnerabilities. *Future Internet*, 12(4), 2020.
[25] K. Sen, D. Marinov, and G. Agha. Cute: A concolic unit testing engine for c. 30(5):263–272, Sept. 2005.
[26] T. Wang, T. Wei, G. Gu, and W. Zou. Taintscope: A checksum-aware directed fuzzing tool for automatic software vulnerability detection. In *2010 IEEE Symposium on Security and Privacy*, pages 497–512, 2010.
[27] Wikipedia contributors. Np-completeness — Wikipedia, the free encyclopedia, 2021. [Online; accessed 17-March-2021].
[28] M. Woo, S. Cha, S. Gottlieb, and D. Brumley. Scheduling black-box mutational fuzzing. pages 511–522, 11 2013.
[29] W. Xu, S. Kashyap, C. Min, and T. Kim. Designing new operating primitives to improve fuzzing performance. CCS '17, page 2313–2328, New York, NY, USA, 2017. Association for Computing Machinery.
[30] M. Zalewski. American fuzzy lop. `https://lcamtuf.coredump.cx/afl/`.

Comparing Strategies for Mining User Reviews to Determine App Security

Albert Dijkstra and Niels Bügel

Abstract—In the past few years there has been an increase in apps published to app stores. The combination of this increase in apps, low entry barriers for publishing to app stores and pressure on developers to rapidly develop apps, results in an increase of security risks for the users. While code analysis and behaviour testing have been used to identify security risks in the past, a different approach has emerged that analyses user reviews. User reviews can be used to identify possible security risks. However, with this, three challenges emerge: the issue of semantics, extracting only security relevant reviews and the issue of user credibility. In this paper we analyze four different algorithms that can be used to assess app security risks based on user reviews. The algorithms in question are: AUTOREB, MARS, SRR-Miner and the method proposed by T. Hadad et al. The goal is to provide an overview of how the different algorithms perform and what their advantages and disadvantages are. While all algorithms are useful in their own right, we found that MARS provides the most reliable results for assessing security risks of apps.

Index Terms—App, Risk, Security and Privacy, User Review, Mining

1 Introduction

In recent years, the number of applications on app stores such as Google Play have shown strong growth. The growth can be attributed to the increasing popularity of software engineering studies [1] and low entry barriers. Due to these low entry barriers and the fact that there is high pressure on the developers to quickly develop apps [2], the security and privacy risks increase for the end user [3]. Numerous different methods have been proposed for analysing the security impact of applications. These methods include code analysis [3, 4], testing the behaviour of an app [5] or a combination of both [2]. Another, more recent, approach is to study the user reviews of apps in these app stores and infer useful information about security and privacy aspects of the corresponding app from this. User reviews provide a unique point of view on these aspects and, as a result, can provide useful information.

From the state-of-the-art [6, 7, 8, 9] we see that there are a number of challenges to overcome when mining user reviews. The first challenge concerns the semantics. Since most reviews are typed on smartphones, the reviews tend to be short and they can contain misspelled words [8]. In addition to this, different words can be used to express the same meaning [6]. The second challenge is finding all the reviews that are relevant for assessing security risks. App reviews are diverse and not all reviews are security related [8]. The third challenge concerns user credibility. Paying attention to user credibility can prevent the reviews of experts from being overshadowed by those of less trustworthy reviewers [6]. In general, the methods used for mining these user reviews make use of either supervised learning or unsupervised learning. However, not all of the approaches tackle the aforementioned challenges in the same way and there are multiple ways of dealing with these problems.

In this paper, we aim to analyse how four different methods of analysing user reviews perform in determining the privacy and security risks of an app. The goal is to highlight the different advantages and disadvantages and provide a meaningful discussion about this. We expect that each approach has their specific use case and our paper aims to provide an overview of this.

In section 2 we discuss the different algorithms and how they work. In section 3 we go through the different challenges and see how each algorithm deals with them. In section 4 we provide an analysis and overview of the various aspects of the aforementioned methods. Lastly, concluding remarks and future work are discussed in section 5.

- *Albert Dijkstra of the Faculty of Science and Engineering, University of Groningen, E-mail: a.dijkstra.38@student.rug.nl.*
- *Niels Bügel of the Faculty of Science and Engineering, University of Groningen, E-mail: n.a.bugel@student.rug.nl.*

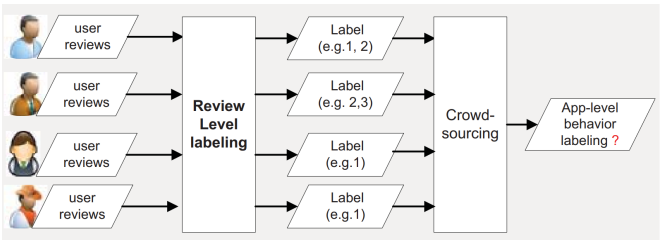


Fig. 1. Inferring app-level labels via crowdsourcing [6].

2 Background

Before a comparison analysis can be done on the different methods, we first provide a basic overview of them. The four methods that are discussed in this paper are AUTOREB [6], MARS [7], SRR-Miner [8] and the method proposed by T. Hadad et al. [9] which we refer to as UFA from this point on. These particular methods were chosen due to the variety of techniques they employ. While the intend of these methods is to perform security risk assessment, the manifestation of this measure differs significantly between them. To illustrate the techniques and measurements these methods use, we discuss each of them in more detail next.

2.1 AUTOREB

The goal of AUTOREB is to infer security behaviour from reviews using machine learning. Each review is given one or more of the following security behaviours: `Spamming`, `Financial Issue`, `Over-privileged Permission` and `Data Leakage`. The method distinguishes two stages. The first stage is the review-level security behavior inference (RLI) stage. This stage is concerned with assigning the correct labels to each review using a machine learning algorithm. The second stage is the app-level security behaviour inference (ALI) stage. This stage uses crowdsourcing to combine the reviews from the RLI stage to obtain the security labels associated with the application. The two stages of AUTOREB can be seen in Figure 1.

The RLI stage consists of a training phase during which the classifier is trained and a testing phase during which new user reviews are automatically labeled by feeding them into the classifier. The training phase itself is composed of three steps. During the first step, the security-related features are extracted by determining those features

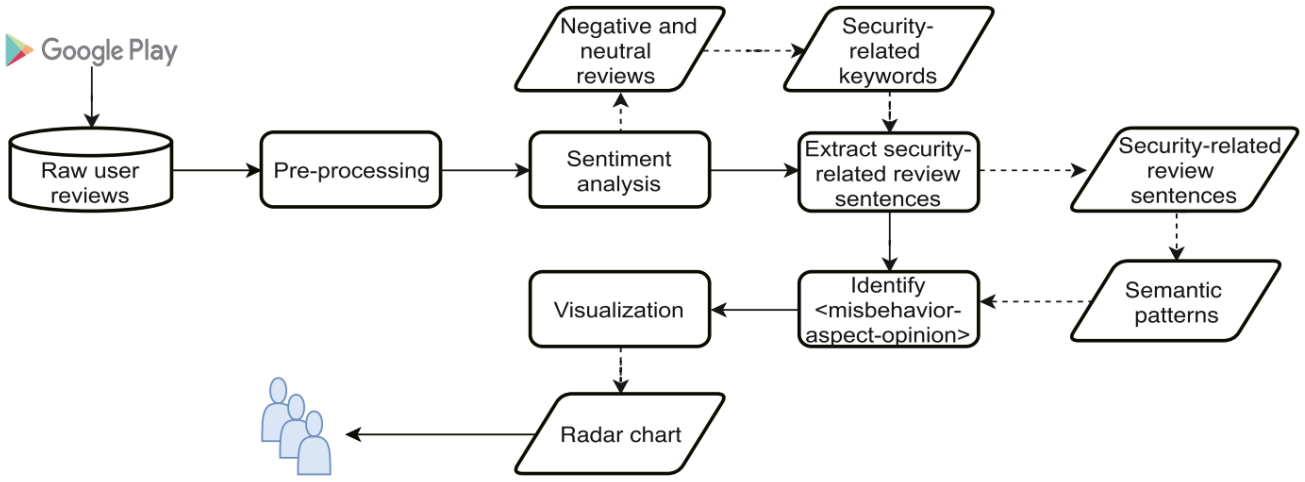


Fig. 2. An overview of the framework used by SRR-Miner [8].

that have a close relation to one of the aforementioned security behaviours. This is done using a keyword-based approach.

The next step is referred to as *semantic expansion* and attempts to deal with reviews that have similar meaning, but are described differently. It does this using a technique called *feature augmentation*, during which additional relevant words or phrases are added. At the end of this step, each user review is represented as a feature vector called a bag of words (BOW). In the last step of the training phase this bag of words is used for the actual training of a sparse Support Vector Machine (SVM). To classify a user review during the testing phase of the RLI stage, a BOW is first generated similar to the testing phase. Once this BOW has been obtained, the sparse machine learning classifier can determine the corresponding security behaviour category.

After the RLI stage has finished, the results are combined to obtain the security labels associated with the app itself. This is the ALI stage. It is done using *crowdsourcing*; a technique that aims to to combine the labels of multiple works to find the ground truth, i.e. the true labels of the item. Crowdsourcing assumes the labels to be noisy, of low quality and possibly contradictory. In contrast to majority voting, the two-coin model [10] used by AUTOREB does not treat each user equally and tends to weigh the opinions of more trustable users higher. This attempts to reduce the issue of the crowd's less valuable opinions potentially overshadowing the contributions of the experts. The two-coin model is explained further in subsection 3.3.

The result of AUTOREB is a collection of app-level security label indicators. These indicators are not to be interpreted as the probability of the application having a particular security issue, but rather as a security-risk ranking score that can be used for comparing the security risk against other applications. Through manual annotation, a threshold can be set on these indicators to determine if an application has a particular security behaviour or not [6].

2.2 MARS

MARS stands for Mobile App Reviews Summarisation. It makes use of natural language processing, sentiment analysis and machine learning to summarise user reviews. The goal of MARS is to provide insights in the relation between the real behaviour of an app and the privacy and security-related reviews.

At the start of the process, a filtering and pre-processing step is performed that tokenises the review, removes stop words and applies stemming to reduce the number of words. Similar to SRR-Miner, it uses Vader SentimentAnalyser [11] to discriminate between positive and negative reviews. The automatic classification of the user reviews makes use of supervised learning. The feature extraction makes use of a Term Frequency-Inverse Document Frequency (TF-IDF) feature for the representation of each user review. For the classification algorithm, logistic regression was selected due to its simplicity and the fact that it is less prone to over-fitting. In the end, MARS assigns a number of labels to each user review. For the end-user of MARS, it presents a web interface that provides an overview of the security risks and most relevant reviews of the application in question [7].

2.3 SRR-Miner

In contrast to AUTOREB, SRR-Miner does not make use of machine learning. Instead, it uses a keyword-based approach. In contrast with MARS and AUTOREB, the goal is not to classify reviews or the application itself, but rather to provide a summary of the user reviews. It does this by extracting <misbehaviour, aspect, opinion> triples from the user reviews. For example, a user review sentence such as "This app is leaking GPS location, it is very annoying." corresponds to the triple <leaking, GPS location, annoying>. The entire process consists of five steps: pre-processing, sentiment analysis, sentence extraction of security-related reviews, triple extraction and lastly a visualisation of the summarisation. An overview of the framework as a whole can be seen in Figure 2.

The first step in this technique is the pre-processing of the user reviews. This step consists of removing emojis, discarding non-English reviews, correcting abbreviations, expanding contractions and attempting to correct typos. The second step is the sentiment analysis, where the negative reviews are extracted for further analysis using Vader SentimentAnalyser similar to MARS. The next step is extracting the security-related reviews by constructing a list of security-related keywords that allow the system to extract reviews based on those keywords. The fourth step is extracting the <misbehaviour, aspect, opinion> triples from the security-related reviews with the use of a semantic dependency graph. The final step is the visualisation of the summarisation using a radar chart [8].

2.4 UFA

The goal of the method proposed by T. Hadad et al. is to explicitly detect malicious applications and label them as such. The system consists of four main steps. The first step is generating a domain specific lexicon using a domain-specific corpus. Both the domain-specific corpus and the user reviews are examples of textual corpora. These are represented in natural language form and require pre-processing. The domain-specific corpus can be used to generate a Domain Lexicon (DL). The specific pre-processing steps for these textual corpora used in this method are further discussed in subsection 3.1.

The second step is using this DL in combination with the user reviews to extract application features. The third step is referred to as

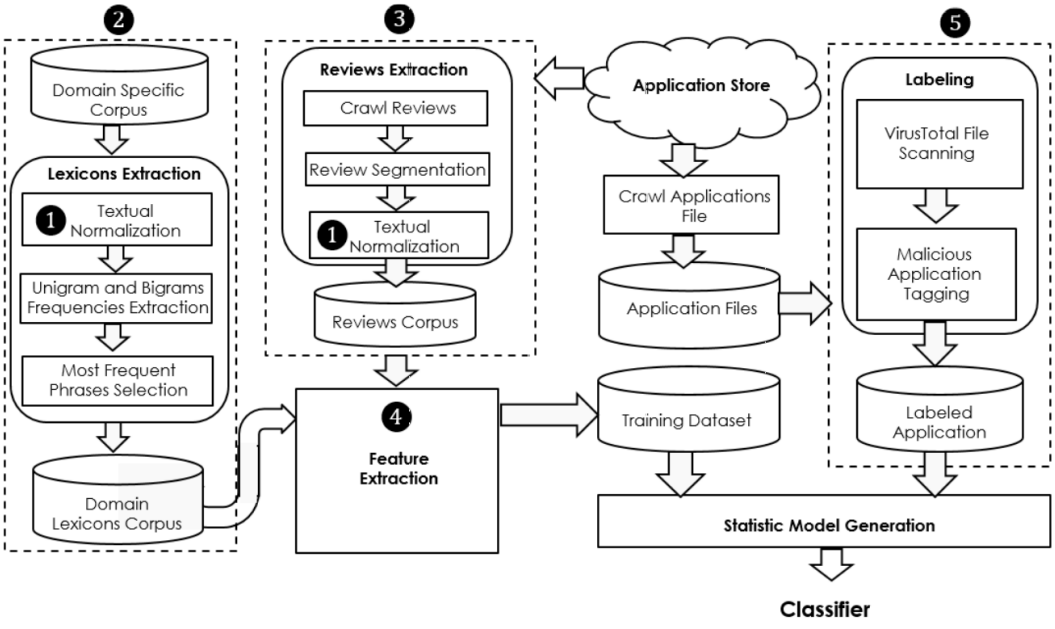


Fig. 3. A flowchart of the malware detection system proposed by T. Hadad et al. [9].

feature engineering; in this step, two features are generated for each application. The last step is generating a classification model using supervised learning. The features generated in the previous step are used here as the training dataset. A flowchart of the full process can be seen in Figure 3.

All the different supervised learning classifiers mentioned in their paper need labelled data for the testing, training and validation sets. As such, T. Hadad et al. use several virus scanners and combined their outputs to obtain this labelled data. They tested three different classifiers: C4.5 decision tree learner [12], random forest [13] and logistic regression [14]. Among these three classifiers, logistic regression performed the best [9].

3 Analysis

To determine the app security from reviews there are a number of challenges to overcome. In this particular paper, we discuss three of these challenges in more detail: semantics, relevance and user credibility. Not all of these challenges are addressed by each method in the same way; this may include ignoring it all together. Nevertheless, we discuss each of these challenges in more detail next and how the methods approach them. In addition to this, we also discuss how the different methods obtained their data sets.

3.1 Semantics

Instead of the often-used static code analysis or behaviour testing, the methods discussed in this paper make use of user reviews. The challenge here is that the reviews are written in a natural language. This presents issues such as the fact that users may use emojis, abbreviations or non-existing words[6, 8]. This means that there are a number of challenges to overcome here. For the four different algorithm, dealing with the semantics is generally done in a pre-processing step. In this step, a number of these issues are addressed. The different problems and their possible solutions within this pre-processing step are discussed next.

Noise removal. Noise removal concerns itself with removing irrelevant data from the user reviews. Different algorithms classify different things as noise. For noise removal, MARS and AUTOREB remove stop words [7, 6]. UFA removes, in addition to stop words, numbers and punctuation [9]. SRR-Miner only removes Emojis and instead of removing punctuation, SRR-Miner uses it to split it into subgroups in order to process these as independent clauses. It should be noted that SRR-Miner explicitly filters out non-English reviews in this step [8]. The other methods make no mention of the language assumptions.

Character replacement. There are a number of things that can be addressed using character replacement. The first is removing character continuity. If a character repeats itself for more than two times, the extend of characters are removed, e.g. “sleeeep” becomes “sleep”. SRR-Miner also ensures that text and slang language - including abbreviations and contractions - are replaced by the corresponding word. For example, “fav” will become “favourite” [8].

Since most of these reviews are written on mobile, spelling mistakes and expressions occur relatively often. As such, both UFA and SRR-Miner replace expressions and frequent spelling mistakes which are known beforehand [8, 9]. In addition to this, missing apostrophes are added; e.g. “dont” is replaced by “don’t”. Afterwards, predetermined words with apostrophes are expanded, which means that “don’t” would be expanded to “do not”. It is important to note that, while AUTOREB does not fix spelling mistakes in the pre-processing step, it indirectly circumvents this issue in the feature augmentation step by using a search engine that still works with misspelled queries.

Stemming. MARS, UFA and AUTOREB stem all words to their roots. After stemming, words such as “walk”, “walking” and “walked” all match the common root “walk” [6, 7, 9]. Each of the previously mentioned steps aim to normalise the words so that the next step in the main process becomes easier.

3.2 Relevance

In contrast to the pre-processing step, where only individual words were addressed, the relevance step concerns itself with the meaning behind the reviews. This can be done with a variety of different methods.

For AUTOREB, the relevance stage is equivalent to the RLI stage. AUTOREB uses a keyword-based approach to select from a review the security relevant words or phrases. AUTOREB find words or phrases of similar meaning by submitting a query to a search engine. The advantage of this approach is that the query can be misspelled, short or varied in the choice of words. The relevant reviews are then extracted from the top documents that were obtained by submitting the query. The resulting reviews are then added to the original feature vector: this is called feature augmentation. This feature vector describing the

review is referred to as a Bag-of-Words (BOW). Specifying how applicable the predetermined labels are to a specific review is then done using a sparse machine learning classifier [6].

MARS uses supervised learning to determine the threat described in the user review. In order to compare a user review with the different threats it uses word vectors. For each threat there is a set of words that are expanded with additional words related to the security threat. This is done by text expansion. MARS searches in the GloVe [15] vocabulary for the N most similar words and returns those words along with their corresponding match percentage. This is then used in combination with a supervised classification algorithm to obtain the privacy and security relevant reviews [7].

UFA creates a domain specific corpus by first extracting data from computer and network security books. It then transforms it to canonical form, which corresponds to the normalized form as described in subsection 3.1. Afterwards, it extracts unigrams and bigrams of the phrases along with the frequencies from the corpus so that the frequently used phrases can be selected. As a result, only the security relevant phrases are left [9].

SRR-Miner uses three steps to extract security threats mentioned in user reviews. It first extracts key words in the user reviews. For this it uses Natural Language Processing in order to discover verbs and nouns and lemmenises them. In the next step an iterative process is used to find keywords related to security. First a number of keywords are manually selected, which are then expanded using a co-occurrence matrix. If there are enough user reviews that use a particular word, said word is added as keyword. Last, three sub-steps are used to make sure that non-security related keywords are not added. In the first substep, words such as app, phone etc. are removed. In the second substep, words are removed in which the verb is the user itself, such as uninstall etc. In the last sub-step, all the non-security related words are removed. The final step of the relevance step is to use these keywords to extract security relevant reviews [8].

In addition to the steps mentioned above, SRR-Miner and MARS add another pre-processing step by looking at the sentiment of the user review [7, 8]. A review can be positive, negative or neutral. However, negative, and sometimes neutral reviews tend to reflect security issues more compared to positive reviews [8, 9]. SRR-Miner and MARS use Vader SentimentAnalyser to give a normalised probability a review is positive or negative. A threshold can be selected to determine the useful reviews. In contrast with this, UFA does not use a sentiment analyser. Instead, it uses the fact that negative reviews often have a one star rating and only uses those reviews [9].

3.3 User Credibility

An extensive study towards detecting fake reviews [16] indicates the difficulty to differentiate between fake and authentic reviews. Fake reviews are defined as non-spontaneous reviews by people who get paid by the developer or are otherwise related, such as friends. They showed an initial result that over a third of reviews are fake. This means that user credibility is something that should be taken into consideration.

Determining the results on app-level is typically done by combining the results on review-level to an app-level classification. It should be noted that SRR-Miner returns a report with possible threats and lets the user infer conclusions on it [8]. This means that, in contrast to AUTOREB, MARS and UFA, SRR-Miner focuses on providing results on review-level. Therefore, user credibility is not relevant for SRR-Miner.

The most straightforward way of combining results on review-level to app-level is to use majority voting. This means that, for example, an application will receive the "spamming" label if most users mention "spamming". A label d is defined as follows:

$$d_{u,i}^{l} = \begin{cases} 1 & \text{if user } u \text{ labels app } i \text{ as label } l \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Where u is the user, i is the app and l is the label. Given that there are m users, majority voting can be calculated as defined in Equation 2.

$$y_i^l = \frac{1}{m} \sum_{j=1}^{m} d_{j,i} \tag{2}$$

The issue with this approach is that the contribution of every user weighs equally heavy. This means that the contribution of experts would be under appreciated. To rephrase this issue: some users can be trusted more than other users. Of the four algorithms discussed in this paper, AUTOREB is the only one that explicitly addresses this issue of user credibility [6].

To combat this issue, AUTOREB uses a two-coin system [6]. This system gives more credit to trustworthy users compared to more untrustworthy users. The two-coin system assumes that the chance a user classifies the label of the app correctly follows the Bernoulli distribution. For this we denote two cases: α_u^l for sensitivity and β_u^l for specificity. Sensitivity is the chance that the worker would correctly label the app with security threat l, under the condition that the security threat l exists. Specificity is the chance that the worker would correctly label the app without security threat l, under the condition that the security threat l does not exists. Formally this is defined as follows:

$$\alpha_u^l = Pr(d_{u,i}^l = 1 \mid y_i^l = 1) \tag{3}$$

$$\beta_u^l = Pr(d_{u,i}^l = 0 \mid y_i^l = 0) \tag{4}$$

Here i refers to a particular app, u to a user and l to a specific label. The parameters α_u^l and β_u^l can be computed according to the Maximum Likelihood Estimation (MLE). With this information, y_i^l can be computed with use of the Bayesian Rules.

In contrast to majority voting, D. Kong et al. derive y_i^l using Equation 3 and Equation 4 to obtain the formula seen in Equation 5.

$$y_i^l = \frac{a_i^l \cdot Pr(y_i^l = 1 \mid \theta)}{a_i^l \cdot Pr(y_i^l = 1 \mid \theta) + b_i^l \cdot Pr(y_i^l = 0 \mid \theta)} \tag{5}$$

Where $\theta = \{\alpha, \beta\}$, a_i^l is the likelihood that app i is getting label l and b_i^l is the likelihood that app i is not getting label l. Moreover, $Pr(y_i^l = 0|\theta) = 1 - Pr(y_i^l = 1 \mid \theta)$ and $Pr(y_i^l = 1 \mid \theta)$ is the prior probability. It should be noted that y_i^l cannot be used directly as a classifier. This is because the annotations that are used are not given by the users, but rather by the algorithm. Thus the prior probability of finding such an annotation for the security risk are low. Moreover, some users may not encounter the issue, or not give a review after encountering the issue. Additionally, the algorithm may not derive the correct semantic meaning. This all means that y_i^l should not be considered as a probability in itself, but as a ranking score. However, a threshold can be used for each different label to assign the security labels. With this threshold there are two different categories. Which then can determine how much weight you assign to this label, before aggregating all user-reviews together for labels on app-level.

All in all, the two-coin model ensures that experts are given more credibility. By extensions, users that untrustworthy are given less credibility. The combination of this aims to more truthfully represent the potential security risks of an application.

3.4 Data sets & Ground truth

The four methods use data sets obtained from different sources and of different sizes. Additionally, the way each method determines its ground truth also varies between them. To provide an overview of this, we briefly describe how each algorithm obtained their data sets and how they define their ground truth.

AUTOREB randomly selects apps from Google play. In total they use 194,13 reviews of 3,174 apps. The obtained dataset is annotated manually by three experts and this is used by AUTOREB as their ground truth [6].

UFA uses 2,506 different apps and 128,863 user reviews on the Amazon application store. These were obtained throughout a 2-month

	Noise removal	Character replacement	Stemming	Spelling correction
AUTOREB	X	-	X	-
MARS	X	-	X	-
SRR-Miner	X	X	-	X
UFA	X	X	X	X

Table 1. Overview of the actions within the pre-processing steps for each of the four algorithms. Her 'X' denotes that it is addressed in some extent and '-' that it is not addressed.

period. The apps are a random subset of all the reviewed apps in this period. In order to determine the ground truth, a tool called VirusTotal is used. It uses a collection of different virus scanners to determine if an app is malicious or not. The labels assigned to the app by VirusTotal are used as the ground truth [9]. This is also the only algorithm that does not have a manual inspection of the assigned labels.

MARS uses 812,899 reviews of 200 different apps, with up to 4,500 reviews per app. These apps are chosen in two steps. First 981,075 apps are obtained and their corresponding permissions are extracted. This results in 142 unique permissions. The authors focused on applications that had both internet connection access and requested at least 2 security sensitive permissions. From these apps, the 10 categories with the most apps associated with them are chosen. Of these 10 categories the 20 top apps are chosen based on search results from the Google Play Store. The ground truth is then determined in two steps. First the GloVe algorithm is used to find reviews which are possibly privacy relevant. Then three security experts manually check the 2,896 reviews found by GloVe. Of these 2,896 reviews, 2,412 were labeled correctly. These were given to the classifier [7].

The data set that SRR-Miner used consists of 17 apps. In order to create some variety in the apps, the apps were chosen from 15 different categories. Of these 17 apps, 12 were popular apps and had a rating of more than 3.5 stars. The rest were less popular apps with a rating of 3.5 stars or lower. For the ground truth they decided to label 1,500 negative and neutral review sentences of the popular apps. Similarly, all negative and neutral reviews of the less popular apps were also labeled. In total 18,668 of the 44,859 reviews were manually labeled and used as the ground-truth [8].

4 Discussion

In the following section we give a comparison of the algorithms. We provide a discussion of how the different methods handle the three challenges, how their statistics compare and inspect whether there are any additional issues with them. The pre-processing step and the issue of user credibility are discussed in subsection 4.1. The challenge of finding the reviews which are relevant to security shows overlap with the goal of some of the algorithms. As such, this is combined with the overall performance of the algorithms discussed in subsection 4.2.

4.1 Pre-processing & User Credibility

The four algorithms all start with a pre-processing step. The result of this step is a normalised and higher quality data set. As such, the degree of thoroughness is important for the quality of the classification later on in the process. A summarisation of which pre-processing steps are done can be found in Table 1. It should be noted that this specific table provides only a surface level comparison and does not go into detail of how well each specific method tackles each challenge. First and foremost, we can see that all the four methods have some form of noise removal. Both SRR-Miner and UFA make use of character replacement and all the methods except for SRR-Miner make use of stemming. Of the four methods, UFA takes the most actions in this step with the removal of stop words, numbers and punctuation.

It is important to note that not all methods are equally reliant on the quality of the normalisation in the pre-processing step. AUTOREB for example, uses a search engine for which the queries do not need to be normalised. As such, it is for AUTOREB less important to have extremely thorough pre-processing compared to the others, since the search engine indirectly solves this problem. Nevertheless, it would be interesting to see how much the methods would improve by introducing more thorough pre-processing.

Of the four methods, only AUTOREB addresses the issue of user credibility. While the performance statistics of AUTOREB, MARS and SRR-Miner show good results, these concern the performance on review-level. Apart from UFA, none of the papers disclose the performance in correctly classifying applications as malicious, i.e. on application-level. The output of AUTOREB is not a set of labels, but rather as a ranking score. A binary output for each label can be produced by setting a threshold, but the accuracy of this strategy is not discussed. As such, it is not possible to compare this against UFA. Nevertheless, given that more than a third of the reviews cannot be trusted [16] and the results shown by D. Kong et al. it is likely that the results of UFA could be improved by taking user credibility into account.

4.2 Performance

In order to compare the performance between the different algorithms we look at the precision, recall, F1-Score and accuracy reported in the corresponding papers. These scores can be calculated using the true positive, false positive, true negative and false negative rates. A True Positive (TP) indicates that an algorithm correctly identifies a review as a security issue. On the other hand, a False Positive (FP) indicates that an algorithm incorrectly identifies a review as a security issue. A True Negative (TN) indicates that an algorithm correctly identifies a review as a non-security issue, while a False Negative (FN) indicates that an algorithm incorrectly identifies a review as a non-security issue. Then precision, recall, F1-score and accuracy are defined as follows:

$$Precision = \frac{TP}{TP+FP} \tag{6}$$

$$Recall = \frac{TP}{TP+FN} \tag{7}$$

$$F1 = \frac{TP+TN}{TP+FP+TN+FN} \tag{8}$$

$$Accuracy = \frac{TP}{TP+FP+TN+FN} \tag{9}$$

Due to the slightly different objectives of the methods, the results cannot be compared directly in a meaningful way for all four methods. The exception is AUTOREB and MARS, which use the same measurements. The measurements used by both AUTOREB and MARS look at how well they are able to classify user reviews. In other words, these particular metrics are not concerned with the performance of the method on an application level, but rather on review level. While the performance on application-level is mentioned for AUTOREB, no explicit performance measurements are provided. The comparison between AUTOREB and MARS can be seen in Table 2. We observe that for all the different metrics, MARS scores considerably higher than AUTOREB. If we compare the data sets that these two algorithms use,

X	Precision	Recall	F1	Accuracy
AUTOREB	80,1	82,46	81,26	94,05
MARS	94,84	91,30	92,79	N/A

Table 2. Overview of the performance of AUTOREB and MARS.

we see that the data set used by MARS contains a significantly smaller number of applications. However, the total number of reviews that MARS is higher compared to AUTOREB. From the way the data sets were obtained, we can see that both are of similar quality.

SRR-Miner gives good results on the extraction of security-related review sentences with an F1-score of 89%. One major difference with other algorithms is that it provides a summarisation report instead of a classification for the app. Meaning that instead of providing a label to end users without them knowing how this label was arrived at, end users receive a summary of the reviews and can draw that conclusion themselves. A survey with a test group of 20 people indicate 18 found the report useful in understanding the security issues of the app, while the other 2 were neutral about the app [8]. This indicates that SRR-miner provided a successful approach to solving the given problem. That said, SRR-Miner only tested its algorithm on 17 different, hand-selected mobile apps. The majority of these apps are well known and very popular downloaded. While the number of reviews was large, the small number of apps might cause a bias in the results. The authors are aware of this and want to give more insight in future work.

UFA only reports the recall and accuracy. However, these metrics do not concern individual reviews, but rather the classification of an application as a whole. They reported their recall to be 23%, which means that only a relatively small percentage of the applications that are classified as malicious are indeed malicious. The authors of UFA also identified this problem and concluded: "TPR results show that it is not feasible for using our method as a sole detection method, and further research should be performed in this aspect." [9] Even though further work might solve this problem, in its current state, the method does not prove to be a feasible technique in reliably identifying malicious applications. Additionally, the data set that UFA uses was not manually inspected. As such, the data set is more likely to contain errors compared to the data sets of the other methods.

5 Conclusion

In this paper we compared four different algorithm: AUTOREB, MARS, SRR-Miner and UFA. While the individual goals of the methods differed slightly, the fundamental objective is the same: provide information on the security risks of applications based on user reviews. The three main challenges that these methods had to deal with were semantics, relevance and user credibility.

UFA can classify apps as malicious. While it is an interesting choice, its low recall rate make it untrustworthy. In contrast with this, SRR-Miner provides a summary of the user reviews. It has practical value for the end users, since it leaves the final decision of the security risk up to the user. However, its research might be biased, due to only testing with a handful of well-known apps.

MARS and AUTOREB both provide classification on review-level. Of these two, MARS clearly provides more reliable results. MARS also provides a ranking score on application-level. For this it takes user credibility into consideration; a strategy that UFA might benefit from. All in all we provided an overview of the different algorithms and have shown the advantages and disadvantages of each of them.

5.1 Future Work

There are a number of things that could be done in future research. First, the results of UFA as of right now are not very usable due to the low recall rate. Its results might improve by integrating a system that addresses the user credibility problem. Second, each method used a different set of pre-processing techniques. As such, it would be interesting to see how the performance of the methods change when the pre-processing techniques are expanded. Third, the data set that the authors of SRR-Miner used to validate its performance was very small compared to the others. As such, it is not immediately clear how it would perform when using a larger data set.

In general, the methods used vastly different data sets and varying methods of establishing the ground truth. To better compare the methods in questions, a single large data set could be established. Not only would this allow researchers to better compare existing methods, but it would also be useful for future research. Last of all, the results and performance of some of these methods are already useful for the end-users. As such, the next step would be making the results easily accessible for the end users.

Acknowledgements

The authors wish to thank the supervisor, reviewers and organisers of the 18th student colloquium.

References

[1] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A survey of app store analysis for software engineering," 2017.

[2] B. F. Demissie, M. Ceccato, and L. K. Shar, "Security analysis of permission re-delegation vulnerabilities in android apps," *Empirical Software Engineering : An International Journal*, vol. 25, no. 6, pp. 5084–5136, 2020.

[3] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, "A study of android application security," in *Proceedings of the 20th USENIX Conference on Security*, SEC'11, (USA), p. 21, USENIX Association, 2011.

[4] J. Chen, C. Wang, K. He, Z. Zhao, M. Chen, R. Du, and G. J. Ahn, "Semantics-aware privacy risk assessment using self-learning weight assignment for mobile apps," 2021.

[5] S. A. Gorski and W. Enck, "Arf: Identifying re-delegation vulnerabilities in android system services," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '19, (New York, NY, USA), p. 151–161, Association for Computing Machinery, 2019.

[6] D. Kong, L. Cen, and H. Jin, "Autoreb: Automatically understanding the review-to-behavior fidelity in android applications," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, (New York, NY, USA), p. 530–541, Association for Computing Machinery, 2015.

[7] M. Hatamian, J. Serna, and K. Rannenberg, "Revealing the unrevealed: Mining smartphone users privacy perception on app markets," *Computers Security*, vol. 83, pp. 332–353, 2019.

[8] C. Tao, H. Guo, and Z. Huang, "Identifying security issues for mobile applications based on user review summarization," *Information and Software Technology*, vol. 122, p. 106290, 2020. ID: 271539.

[9] T. Hadad, B. Sidik, N. Ofek, R. Puzis, and L. Rokach, "User feedback analysis for mobile malware detection," pp. 83–94, 0001 2017.

[10] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy, "Learning from crowds," *Journal of Machine Learning Research*, vol. 11, no. 43, pp. 1297–1322, 2010.

[11] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *ICWSM*, 2014.

[12] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[13] T. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, pp. 832–844, 1998.

[14] S. H. WALKER and D. B. DUNCAN, "Estimation of the probability of an event as a function of several independent variables," *Biometrika*, vol. 54, pp. 167–179, 06 1967.

[15] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1532–1543, Association for Computational Linguistics, Oct. 2014.

[16] D. Martens and W. Maalej, "Towards understanding and detecting fake reviews in app stores," *Empirical software engineering : an international journal*, vol. 24, pp. 3316–3355, Dec 2019.

An Overview of Evaluation Metrics for Video GANs

Robbin de Groot and Max Verbeek

Abstract— Generative adversarial networks (GANs) are used to generate high-quality synthetic images and video. These networks are trained by optimising a generative component and a discriminative component separately. The error for both components is, by nature, not objective and hence we cannot tell from the error which of two GANs performs best. To assess the performance of a GAN overall, several *objective evaluation methods* exist. These methods are less established for video GANs than for image GANs. Here we present a number of existing evaluation methods that have seen success in the application of video generation. In this paper, we consider a selection of state-of-the-art video GAN evaluation methods. We aim to discover how they compare in various applications. To achieve this, we provide an overview of five criteria that these evaluation methods can adhere to, with their corresponding performance in each criterion. To aid our comparison, we propose a simple *generalised evaluation method pipeline* to classify differences between methods. The contribution of our paper is to aid researchers in assessing the performance of their video GAN by providing an overview of methods to consider. This will improve the quality of future papers by shining light on existing, yet scarcely used methods.

Index Terms—GAN, Video Generation, Evaluation Metric, Comparison, Objective Function

1 Introduction

Video generation with Generative Adversarial Networks (GANs) is a newfound technique to generate video material. This technique has been made feasible on consumer-grade computers due to a recent increase in computational power. Formerly, GANs have been used to generate images with great accuracy and good realism [1]. Video GANs expand on this idea by generating sequences of temporally related images. This has applications such as generating video from a description [2] or transforming video captured on a rainy day into sunny bright blue skies [3].

Goodfellow et al. originally proposed GANs that consist of two components, a generator and a discriminator, that are optimised separately [4]. We illustrate these components in Fig. 1. The generator aims to generate an output that looks indistinguishable from the class of input that it is trained on. Meanwhile, the discriminator tries to differentiate between the real samples and the generated samples. The GAN is considered to be well-trained when the generator can produce data that the discriminator cannot confidently distinguish from real data. However, because the discriminator is trained on synthetic data, the output depends on the generator as well. The networks play a zero-sum game in an attempt to find the *Nash equilibrium*: neither network can improve their own performance, as modifying the model will also affect the other party leading to performance decrease. [5]

Generation of video proves to be more difficult, due to the higher dimensionality of videos. The temporal dimension adds an additional requirement for video material to look realistic, that is, the continuity between frames should be fluent. Assigning a numeric score to the similarity of images or video is very difficult to do objectively for machines, as this is a problem that requires human intuition of what real video looks like [6]. Some papers resort to similarity scores made by humans for their final result set [7, 8], which illustrates the difficulty in deriving an accurate, numeric metric. Additionally, by the nature of GANs, the accuracy of the discriminator cannot be used to evaluate the training progress. Many evaluation methods are developed for images. However, these methods do not extend well to video material, largely due to the more sophisticated and diverse requirements that video GANs have. Because of the scarcity of well-defined methods in the literature, we aim to explore existing evaluation metrics for video generation. These video methods are in some cases similar to the methods that can be found in the image domain, with a small extension to account for continuity in the temporal domain. In other cases, a completely new technique has been established using trainable classifiers. Although this paper is aimed at video and image sequences, we will often return to the image domain. This domain is better established and therefore relevant to understand the effects of adding the time dimension.

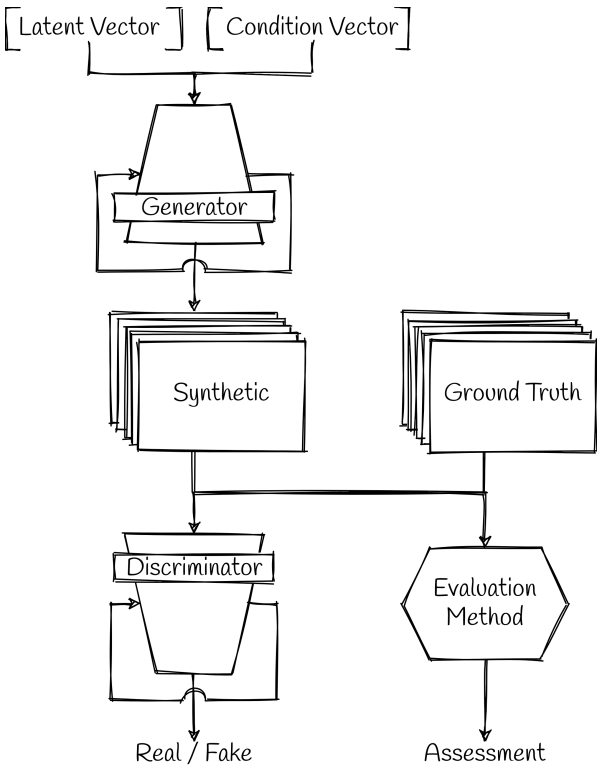


Fig. 1. A graphical depiction of the general shape of a video GAN and an evaluation method.

In this paper, we will assess five categories for evaluation methods, namely: efficiency, objectivity, continuity, quality, and diversity. All of these categories except for continuity are also applicable to the image domain. A common approach is to combine local similarity metrics with temporal similarity metrics. In these cases, the local

- *Robbin de Groot of the Faculty of Science and Engineering, University of Groningen, E-mail: r.s.de.groot@student.rug.nl.*
- *Max Verbeek of the Faculty of Science and Engineering, University of Groningen, E-mail: m.j.verbeek.2@student.rug.nl.*

approach can be reused from the image domain and only the temporal similarity needs to be taken care of. With this paper, we aim to answer the question: how do existing evaluation methods perform in the five categories mentioned above, and in which application is each method most suitable?

We will first outline the state of the art to provide context and background. We will then present the approach to our comparison. We introduce the measures with which we will compare the methods under consideration. We then concretely list the said methods, outlining their origins, past uses, strengths, and weaknesses. With that, we commence our comparison. We finally conclude by collecting and summarising our findings.

2 State of the Art

To evaluate the *quality* and *diversity* of GAN generated data, we first require a definition of these terms. Quality is closely related to the accordance of generated data with the ground truth. A common approach involves statistical density estimation. This paper adopts the common notation for densities; p_z for the latent density of the generator G, p_G for the data generated by $G : \mathbb{E}_{z\sim p_z}[G(z)]$, and the ground truth density p_T. One can compute several metrics on these. For comparing generated images, the Fréchet Inception Distance (FID) [5] is a popular choice. The FID is defined as

$$\text{FID}((\mu_G,\Sigma_G),(\mu_T,\Sigma_T)) = |\mu_G-\mu_T|^2+\text{tr}(\Sigma_G+\Sigma_T-2(\Sigma_G\Sigma_T)^{\frac{1}{2}}), \tag{1}$$

where the inputs are the means μ and covariances Σ from samples drawn from p_G and p_T. Compared to pre-existing methods, such as the Kullback-Leibler Divergence [9] and Inception Score [6], the FID is robust against common patterns in images such as shifting, scaling, and rotating [5]. These patterns arise because pixels in images are not generally statistically independent. A comparison between the FID and other methods for evaluating image GANs are outlined in [1]. Hence, we will not go into further detail. The FID has been proven useful in evaluating images produced by GANs on their realism compared to the ground truth [5, 10, 11].

We now extend to the time domain. By introducing recurrent elements to GANs, we obtain GANs that produce sequences of images: videos. Video GANs come in many different flavours. Often, video GANs are conditional GANs, meaning they take additional structured and meaningful input (e.g. the desired class or previous output). Video GANs, therefore, differ in the conditioned data or the (temporal) relationship with the conditioned data. We may distinguish between several kinds:

- **Sequence extension GANs** that generate video material from a starting image or video [8, 12, 13].
- **Sequence-conditioned GANs** that generate video using a variable-length sequence as a condition, such as text or speech [2, 14, 15, 16].
- **Video translation GANs** that translate e.g. the scene in existing video or cycle between representations [3, 17].
- **Probabilistic video GANs** that are capable of generating possible futures from a starting image of sequence [18, 19].

These different types of GANs require different evaluation methods for proper assessment, although some methods may be universally applicable.

To illustrate the need for a proper method, we turn to recent works [7] and [8]. These papers display the use of Preference Opinion Scores and Amazon Mechanical Turk, respectively. Both these methods involve human labour to judge the realism of generated video. Clearly, quantitative and on-demand assessment of high-dimensional, temporal data is a challenge of today.

However, if we move away from visual forms of data (images/video), we may discover better-established methodologies. Work by Yoon, Jarrett, and Van der Schaar [20] has shown that autoregressive, recurrent GANs are successful at generating time series data. This data does not necessarily consist of images. The paper describes 3 distinct measures to evaluate their generated time-series data. We will briefly outline those here.

1. **Visualisation**: By flattening the temporal dimension, one can use t-SNE [21] or PCA [22] to visualise samples drawn from p_G and p_T alongside each other in 2 dimensions. This allows an observer to assess distributions quantitatively and qualitatively.
2. **Discriminative Score**: A classifier is trained to distinguish between p_G and p_T. The error on a left-out test quantitatively assesses the generated data. Unlike the discriminator, this model is not coupled to the performance of the generator.
3. **Predictive score**: A separate, recurrent model is trained on synthetic data. This model is used for single next-in-sequence predictions on the ground truth. The error defines this metric. This tests whether the predictive capabilities of both p_G and p_T align.

Though these methods are reliable, they are not useful for frequent (i.e. once per epoch) evaluation. The visualisation method requires human inspection, and the discriminative score and predictive score require the training of a recurrent model, which is computationally expensive. Moreover, neither of these metrics are inherently objective.

Fortunately, work by Zhang et al. shows how simple statistical inference and clustering methods may be employed to estimate the closeness of time-series densities [23]. These approaches are simple, interpretable and fast to compute. They apply their techniques to evaluate a predictive GAN for power consumption and solar panel production. One key difference is that their data is 1-dimensional for every time step. Therefore, these simple methods may not scale to raw, high-dimensional data like video.

We may also attempt to apply the FID and other such metrics on time-series data. Because this data may vary in length and therefore dimensionality, we cannot compute these metrics directly. However, by using a feature extraction method, we may reduce all data sequences to an equal dimensionality and compute the FID on the feature space [12, 18]. For video data, this may be done by training a recurrent model like an auto-encoder on the ground truth. Though more involved, this metric is objective and its computational load depends for the majority on the feature reduction method used. Wang et al. use a state-of-the-art, pre-trained action detector called Inception3D (I3D) for this purpose. [12] By removing the last few layers, they obtain a fixed-sized video feature vector for any length of video. These feature vectors are used for evaluation.

3 Approach to Comparison

We will compare the methods described in section 2. We consider only methods that operate in identical data spaces; we compare images to images, video to video, and features to features. This eliminates cycling methods, as these are often assessed by comparing the quality of translating from one space to another and back. We assess the performance of evaluation methods in a handful of categories:

1. **Computational Efficiency**: for methods that are to be frequently evaluated on a test set, there is a need for computational efficiency. Methods that take relatively long to compute may not be desirable as they will adversely affect the total train time.
2. **Objectivity**: methods that always yield the same output for the same input are called objective. This is desirable, as results are one-to-one comparable and thus there is no need to accommodate for noise or discrepancies.
3. **Continuity**: the temporal dimension that video adds on top of single images is more than simply increased data complexity. Human viewers also expect the content to believably follow from

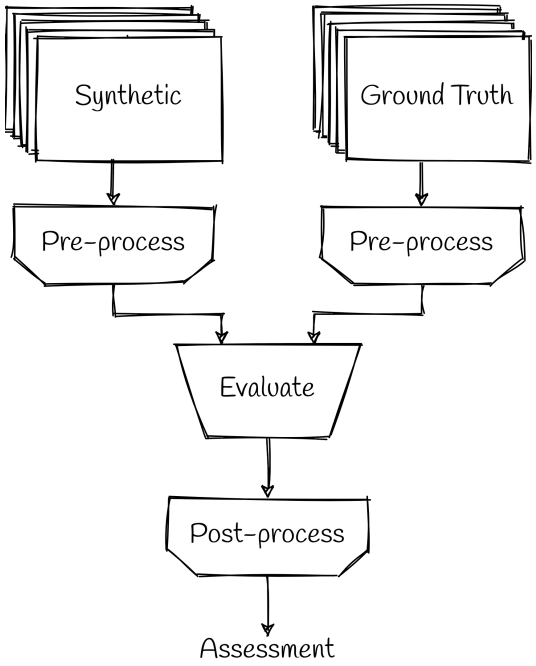


Fig. 2. A pipeline of a general evaluation method.

one frame to the next. The generated video should be context-aware. Moreover, the context displayed should be realistic with respect to the ground truth as well.

4. **Quality**: as the data in question is generated by GANs, we expect a certain level of realism as seen in the ground truth. Regardless of whether or not an image sequence 'makes sense' if the images that it consists of are not realistic (i.e of high-quality) the resulting video will not be either.

5. **Diversity**: as with any generative model, we expect the model to not simply learn and reproduce a single or handful of examples. This is a special case of overfitting called *mode collapse*. Common causes are oversized models and class imbalance. We expect a video GAN to learn and be able to reproduce the entire density, not just a portion of it. Note that good diversity entails that the density must be skewed towards the ground truth in case of class imbalance.

Note that categories 3, 4, and 5 are subjective measures to some extent. In this paper, the term *realism* is therefore considered with respect to the provided ground truth. Unfortunately, this implies that selection between two performant methods may come down to the preference of the researcher. Therefore, we will not attempt to order or score methods, rather mentioning the presence or absence of compliance with these categories.

We will compare GAN generated video to the respective ground truth using the methods presented in section 2. We observe that most methods can be split up into a general pipeline. We introduce this *generalised evaluation method* as seen in Fig. 2. Pre- and post-processing are optional steps. For images, it is common to use a technique that extracts features from images as a pre-processing step. The methods we present are often distinguished in the 'Evaluate' part of the pipeline. For example, when using the Inception3D, we may apply any technique to reduce the feature-space representation produced by the network into a value indicative of the performance or assessment.

We will base our comparison on methods sourced from state-of-the-art literature. These techniques do not encompass all techniques currently known. Instead, we focus on methods that have been successful as claimed by their respective authors. Moreover, these methods excel in different yet comparable applications. Hence, we expect that the comparison offers meaningful insights into the trade-offs and the general strengths and weaknesses of each method. Note that these techniques are, generally speaking, complementary in nature. The deficits of one technique can be made up for by using another. For example, one may choose to evaluate the image quality using the FID, then use another technique to assess the attention to context like in [24]. The question of how to combine methods into a meaningful value or summary is beyond the scope of this paper. For further reading, one can find examples of such use cases in [13, 18, 24].

3.1 Single image FID

One approach to obtain a similarity score is to reduce the problem to the equivalent problem in the image domain, where there already exists a solution. This solution in the image domain is the FID score [5, 10]. One can treat every frame of the video as an individual image and combine the FID scores of each frame into a final score. The FID can be computed using (1). An obvious drawback of doing this is that this method does not impose any requirements or constraints on the continuity between frames. Another disadvantage is that the FID cannot prevent the generator from generating video that does not have any coherent context. This is because the FID does not do any form of image recognition.

3.2 Time-Series Discriminative Score

Following the research presented by Yoon, Jarrett, and Van der Schaar [20], one can train a recurrent classifier to discriminate between real and synthetic data.[1] The error emitted by the classifier scores the network, according to

$$\mathrm{DS}(S_G, S_T) = 1 - \frac{1}{2}\left[acc(C(S_G), fake) + acc(C(S_T), real)\right], \quad (2)$$

where S_G and S_T are equally sized sample sets drawn from p_G and p_T respectively. C is a binary classifier trained on sample sets S_G and S_T. $acc(\cdot,\cdot)$ is the training accuracy function that returns the probability of correct classification at *train* time. A higher score is better. This method, however, is only objective as long as the classifier model is constant. If it is retrained every epoch, we cannot objectively describe progress. Nonetheless, we may deduce that a higher quality synthetic dataset will yield a higher error in the classifier, as it is harder to distinguish real from synthetic. This method captures temporal evolution well and therefore yields excellent attention to context on feature reduced images.

3.3 Clustering

A popular use for GANs is the generation of extra data from a limited dataset. Zhang et al. use GANs to create synthetic datasets [23]. Their approach uses a statistical model that extracts the probability distribution of a given dataset and then use a GAN to generate data that statistically lies within the same distribution as the original dataset. Their validation method, therefore, requires measuring the statistical difference between two clusters of data. To this end, they apply the F_1 score to perform an analysis of variance. Alternatively, the data can be clustered and their centroid points can be compared. For example, we may use a nearest neighbour (NN) classifier as

$$\mathrm{CS}(c_G, c_T) = 1 - \frac{1}{2}\left[acc(\mathrm{NN}(c_G), fake) + acc(\mathrm{NN}(c_T), real)\right], \quad (3)$$

where, like with (2), *acc* is the function of the accuracy of classification evaluated at train time. c_G and c_T are the clusters obtained from clustering on samples of p_G and p_T using a clustering algorithm of choice. Again, higher is better. The results of the experiments done in this paper show that the synthetic data is indistinguishable from the ground truth dataset.

[1] The difference with this network and the discriminator is the dependence on the generator. This additional classifier simply classifies between real and synthetic datasets, however these datasets can hypothetically be drawn from any source.

3.4 Inception3D FID

Due to the complex nature of video similarity, it is often infeasible to define a function that will define a numeric similarity between two videos. This is because such a function would require knowledge of the contents of the video which is highly dependent on the structure of the pixels, rather than the value of the pixels themselves. A much easier approach would be to define a similarity function on the features of a video, rather than its contents.

This technique has been explored in a few papers [2, 12, 25] with successful results. The downside of this method is that a feature extractor would have to be trained, which can be time-consuming and is a learning problem itself. To this end, modified image and video classifiers have been used to serve as a feature extractor. Zhang et al. [23] use two modified pre-trained video recognition CNNs, namely Inception3D [25] and ResNeXt [26], and strip the last few layers to turn these networks into feature extractors. The FID can be calculated on the extracted feature vectors to get a numeric result. We denote this mathematically as

$$\text{I3DFID}(S_G, S_T) = \text{FID}(\text{I3D}(S_G), \text{I3D}(S_T)), \tag{4}$$

where FID is computed using (1) and I3D denotes the feature extraction of input samples using the Inception3D network. We assume that I3D returns a mean and covariance pair instead of raw features.

3.5 Specialised methods

In many cases, the evaluation of the performance of a GAN cannot be well described using generic existing methods, as these methods cannot deal with the complexity of the data that is generated by a GAN. We list a handful of interesting use cases (according to the authors).

- Larsen et al. argue that the evaluation itself can be seen as a problem that can be solved with a learnable model [27]. In their experiments with image generation, they train a regression network that should predict features from their generated images.

- Duarte et al. create a GAN to generate video from a given speech sample [15]. Their evaluation method aims to re-identify speakers from their generated faces, which is a very niche and specific technique.

- Chen et al. translate video to video by first converting their source video to a label mapping, and then converting the labels to a new video [28]. For both of these steps a GAN is used, each with its evaluation methods. The video-to-label GAN uses three standard methods defined in [29]: Mean Pixel accuracy, Average Class accuracy, and Intersection over Union. For the label-to-video step, the generated video is fed to a fully convolutional network, which has been pre-trained to generate a similar label mapping. The labels of the generated video frames are compared to the source labels to retrieve an accuracy measure.

- Hu et al. work with a non-deterministic, predictive model. Hence, they use a distance diversity metric to capture both the prediction accuracy and diversity of their results [18].

To reiterate, these methods are commonly only applicable to the model they were created for. However, studying them may give insight and inspiration for the development of new, more general methods.

4 Results and Discussion

We will now discuss the various methods and their qualities. We will first provide an overview of the performance in the criteria mentioned in section 3. Then, we collect all findings and provide suggestions for various applications based on the discussion prior. In Table 1, we show a table containing the methods under consideration categorised for the provided measures. We will briefly elicit the categories and their respective values.

4.1 Efficiency

For efficiency, we look at the computational load required to evaluate real versus generated samples using each method. The aim is to evaluate once after every epoch to measure training progress. Therefore, we compare the compute time of the method to that of the epoch itself. As such, methods labelled as fast induce close to negligible overhead, whereas slow methods are performance bottlenecks. Slow metrics are likely not suitable for this kind of evaluation, so fast methods are recommended. From our results, we observe that a better efficiency score is a trade-off.

4.2 Objectivity

The objectivity of a method is easily determined from its definition. Most methods are therefore easily categorised. Clustering requires additional explanation. The source paper for this method, [23], describes a triplet of approaches using different choices for train and test sets. When the train set consists of solely ground truth data, this method is objective, as the clusters may be computed in advance. However, in the remaining two cases, synthetic data is used for training which is only available during the training phase. Therefore, this metric is not objective during training using these setups.

4.3 Continuity

Continuity between frames is arguably the most important category for video GANs in comparison to image GANs. Being able to quantitatively measure it is therefore paramount. The FID by itself evidently does not capture any relationships between images. We label methods between none and excellent, the former indicating no continuity and the latter indicating attention to context akin to human inspection. In our results, we notice that methods that have good continuity perform inadequately in at least one other category. This reintroduces the need for using multiple methods to account for these deficiencies. Continuity is also a subjective measure. In practice, if a generated video is shown, it may or may not appear like a real video to a human observer. Metrics that capture this accurately therefore commonly operate in feature space. Intuitively, some features should persist throughout multiple frames to indicate some visual context. The Inception3D FID method achieves this by detecting actions (e.g. playing cricket), which illustrates this intuition. Clustering also shows the capability to capture continuity. However, this only applies to sequence extending GANs, as it compares against the ground truth sequence. The discriminative score method shines in this category. It is effectively designed to measure context awareness, at the expense of high compute time. This method is therefore only suitable for post-training evaluation.

4.4 Quality

GANs commonly work with visual data. One of the selling points of GANs is their ability to generate highly realistic images with as uncanny resemblance to real images. We scale our results relative to human inspection. A rating of excellent indicates that images are hard to distinguish by humans, and poor ratings are trivially distinguished by humans.[2] As mentioned prior, the quality of images produced by GANs has been an important area of research. This extends to video GANs. If a video GAN does not produce realistic-looking frames, the entire footage will be considered unrealistic. The FID score can be used to evaluate the quality outside the temporal context. The other methods rely on the technique that is used to reduce the images to feature space. If the feature reduction method is simplistic or inadequate for the task, the quality assessment of these methods will suffer. This is because the feature space representation does not properly represent the image contents. The Inception3D FID method is a known good feature reducer, meaning this is not a problem.

4.5 Diversity

When generating images, we want every image to be different. We rate our methods between poor and excellent, the former given when the

[2] None of the methods we considered were assigned this rating. However, we include this definition to indicate the scale we use for our judgement.

Table 1. A comparison overview of video GAN evaluation metrics on five different criteria.

Section	Name	Efficiency	Objectivity	Continuity	Quality	Diversity
3.1	FID	Fast	Objective	None	Excellent	Good
3.2	Disc. Score	Slow	Not Objective	Excellent	Adequate*	Poor
3.3	Clustering	Average	Not Objective†	Good	Adequate*	Excellent‡
3.4	I3D-FID	Slow	Objective	Good	Good	Good

* May vary depending on implementation of pre-processing methods like a feature extractor.
† May be objective when clusters are trained solely on ground truth.
‡ Selection of samples depends on the picked ground truth samples, so diversity is not controlled by the method.

method is insensitive to mode collapse, the latter when it is sensitive to exactly the ground truth density. In GANs, this is achieved using latent input noise and optional condition vectors. A proper evaluation metric should consider whether the distribution of classes (for a multiclass generator) in generated data is equivalent to that of the ground truth. As mentioned in section 3, it is common in poorly trained or constructed GANs to observe mode collapse. Ideally, methods should judge the degree to which this happens. Distribution-based methods usually have this 'built-in', which is what we observe for both the FID and Inception3D FID methods. The discriminative score, however, does not measure diversity. After all, if a generator produces the same output every time, a classifier will also classify it as the same. The clustering method can achieve any level of diversity, as it is dependent on the selection of ground truth examples to compare against. The method operates on extended base sequences and, therefore, the selection of base sequences determines the diversity.

5 Conclusion

The advent of Generative Adversarial Networks introduced a great tool to generate synthetic data or content to the field of machine learning. Sometimes, however, they prove to be difficult to train. An objective evaluation method is crucial to assess the performance of a GAN and these methods are scarcely present in the domain of video generation. We have listed a few methods, each with advantages and drawbacks.

One of the methods that we found simply applies the FID to single images. This method chooses to solve the problem in the image domain rather than the temporal domain. This provides an intuitive solution, however, the continuity between frames will be lost. Therefore, we conclude that this method is suitable to judge the realism of individual frames, however not suitable for evaluating video footage.

Other methods acknowledge the difficulty of similarity measurements and solve this problem with additional machine learning or statistical inference techniques. The clustering technique aims to assess the difference in probability density of a real and synthetic dataset. It does this through statistical tests or clustering algorithms. The performance of this method is considerably worse than that of the FID score but much faster than training an entire network.

Further methods employ feature extractors to gain a representation of images or the entire video. The time-series discriminative score method trains a network to discriminate between two images or extracted feature vectors and outputs a score to measure the difference. In case the network is retrained every epoch, this will not be objective and slow. This method is however precise, making it a good option to generate high-quality realistic content.

Lastly, there we covered the Inception3D FID method, which strips the last few layers of a pre-trained video classification CNN to end up with a network that generates features instead of class labels. Rather than using the FID on the raw image data, the features extracted from the video will be used to obtain an FID score. This method has proven to be powerful but can be rather slow depending on the feature extractor that is used.

In this paper, we have derived several criteria to assess the performance of evaluation metrics. These criteria are efficiency, continuity, quality, objectivity and diversity. Existing methods may be classified under these criteria, which provides an immediate overview of when a method can be beneficial to a researcher. Future methods may also be assessed under these same criteria to add to the growing collection of evaluation techniques.

All in all, we observe that many techniques exist and excel in various applications that employ video GANs. This paper structures the search for a suitable method or ensemble of methods by presenting the strengths and weaknesses of a selection of methods. From here, research can be done in search of more appropriate methods, and methods that fill the use cases that are not yet covered by the current state-of-the-art.

6 References

[1] Z. Pan et al. "Recent Progress on Generative Adversarial Networks (GANs): A Survey". In: *IEEE Access* 7 (2019), pp. 36322–36333. DOI: 10.1109/ACCESS.2019.2905015.

[2] Y. Li et al. "Video Generation From Text". In: *arXiv e-prints*, arXiv:1710.00421 (Oct. 2017), arXiv:1710.00421. arXiv: 1710.00421 [cs.MM].

[3] Y. Chen et al. "Mocycle-GAN: Unpaired Video-to-Video Translation". In: *Proceedings of the 27th ACM International Conference on Multimedia*. MM '19. Nice, France: Association for Computing Machinery, 2019, pp. 647–655. ISBN: 9781450368896. DOI: 10.1145/3343031.3350937. URL: https://doi.org/10.1145/3343031.3350937.

[4] I. J. Goodfellow et al. "Generative Adversarial Networks". In: *arXiv e-prints*, arXiv:1406.2661 (June 2014), arXiv:1406.2661. arXiv: 1406.2661 [stat.ML].

[5] M. Heusel et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

[6] T. Salimans et al. "Improved Techniques for Training GANs". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf.

[7] W. Xiong et al. "Learning to Generate Time-Lapse Videos Using Multi-Stage Dynamic Generative Adversarial Networks". In: *arXiv e-prints*, arXiv:1709.07592 (Sept. 2017), arXiv:1709.07592. arXiv: 1709.07592 [cs.CV].

[8] C. Vondrick, H. Pirsiavash, and A. Torralba. "Generating Videos with Scene Dynamics". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/04025959b191f8f9de3f924f0940515f-Paper.pdf.

[9] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1 (Mar. 1951), pp. 79–86. DOI: 10.1214/aoms/1177729694. URL: https://doi.org/10.1214/aoms/1177729694.

[10] T. Karras et al. "Analyzing and Improving the Image Quality of StyleGAN". In: *CoRR* abs/1912.04958 (2019). arXiv: 1912.04958. URL: http://arxiv.org/abs/1912.04958.

[11] N.-T. Tran et al. "On Data Augmentation for GAN Training". In: *IEEE Transactions on Image Processing* 30 (2021), pp. 1882–1897. DOI: 10.1109/tip.2021.3049346. URL: https://doi.org/10.1109/tip.2021.3049346.

[12] T.-C. Wang et al. "Video-to-Video Synthesis". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[13] Y. Kwon and M. Park. "Predicting Future Frames Using Retrospective Cycle GAN". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 1811–1820. DOI: 10.1109/CVPR.2019.00191.

[14] D. Kim, D. Joo, and J. Kim. "TiVGAN: Text to Image to Video Generation With Step-by-Step Evolutionary Generator". In: *IEEE Access* 8 (2020), pp. 153113–153122. DOI: 10.1109/ACCESS.2020.3017881.

[15] A. Duarte et al. "Wav2Pix: Speech-conditioned Face Generation using Generative Adversarial Networks". In: *arXiv e-prints*, arXiv:1903.10195 (Mar. 2019), arXiv:1903.10195. arXiv: 1903.10195 [cs.MM].

[16] S. Alexanderson et al. "Style-Controllable Speech-Driven Gesture Synthesis Using Normalising Flows". In: *Computer Graphics Forum* 39.2 (2020), pp. 487–496. DOI: https://doi.org/10.1111/cgf.13946. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13946. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13946.

[17] M.-Y. Liu, T. Breuel, and J. Kautz. "Unsupervised Image-to-Image Translation Networks". In: *arXiv e-prints*, arXiv:1703.00848 (Mar. 2017), arXiv:1703.00848. arXiv: 1703.00848 [cs.CV].

[18] A. Hu et al. "Probabilistic Future Prediction for Video Scene Understanding". In: *arXiv e-prints*, arXiv:2003.06409 (Mar. 2020), arXiv:2003.06409. arXiv: 2003.06409 [cs.CV].

[19] T. Kurutach et al. "Learning Plannable Representations with Causal InfoGAN". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/08aac6ac98e59e523995c161e57875f5-Paper.pdf.

[20] J. Yoon, D. Jarrett, and M. Van der Schaar. "Time-series Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019, pp. 5508–5518. URL: https://proceedings.neurips.cc/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf.

[21] L. Van der Maaten and G. Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[22] S. Wold, K. Esbensen, and P. Geladi. "Principal component analysis". In: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), pp. 37–52.

[23] C. Zhang et al. "Generative Adversarial Network for Synthetic Time Series Data Generation in Smart Grids". In: *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. 2018, pp. 1–6. DOI: 10.1109/SmartGridComm.2018.8587464.

[24] Q. Chen et al. "Scripted Video Generation With a Bottom-Up Generative Adversarial Network". In: *IEEE Transactions on Image Processing* 29 (2020), pp. 7454–7467. DOI: 10.1109/tip.2020.3003227. URL: https://doi.org/10.1109/tip.2020.3003227.

[25] J. Carreira and A. Zisserman. "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4724–4733. DOI: 10.1109/CVPR.2017.502.

[26] S. Xie et al. *Aggregated Residual Transformations for Deep Neural Networks*. 2017. arXiv: 1611.05431 [cs.CV].

[27] A. B. L. Larsen et al. *Autoencoding beyond pixels using a learned similarity metric*. 2016. arXiv: 1512.09300 [cs.LG].

[28] Y. Chen et al. "Mocycle-GAN". In: *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, Oct. 2019. DOI: 10.1145/3343031.3350937. URL: https://doi.org/10.1145/3343031.3350937.

[29] E. Shelhamer, J. Long, and T. Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (Apr. 2017), pp. 640–651. DOI: 10.1109/tpami.2016.2572683. URL: https://doi.org/10.1109/tpami.2016.2572683.

Graph Neural Networks for Pattern Analysis from Time Series

Ayça Avcı, Jeroen de Baat

Abstract— Many real world relations can be expressed as graphs, such as a social network or the state of traffic at a certain time. These relations often involve a spatial component, and are not static but evolve over time. While many deep learning models have been proposed to extract patterns from static graphs, fewer models exist for pattern analysis on spatiotemporal graphs because of the increased complexity data. Yet, the latter has many applications in domains such as behavior prediction, computer vision and robotics. Here, we present related work on the application of neural networks on graphs in general, give an overview of the current state of the field, and provide a comparison and discussion of the described methods.

Index Terms—Artificial Intelligence, Neural Networks, Deep Learning, Graph Neural Network, Time series, Pattern analysis/recognition.

1 Introduction

Pattern analysis is the detection of regularly occurring events in data. There are various methods for this, of which deep learning approaches have gained significant popularity as these have shown to be effective in several disciplines, such as image classification [17] and natural language processing [1]. Recently, there has been much interest in extending pattern analysis to graphs, as many real world relations can be represented as such, for instance, traffic flow [11, 18, 30, 34, 36] and social networks [28, 38]. Applications include the prediction of traffic flow [11, 18, 30, 34, 36], demographic attribute prediction, content recommendation, and targeted advertising [28]. Using deep learning methods on graphs is more challenging due to the increased structural complexity of the data. Even more challenging is the modeling of graphs in time series (i.e. a sequence of graphs which evolve over time), and spatiotemporal graphs (i.e. graphs in times series where the data also has a location component), which is what we will focus on in this paper.

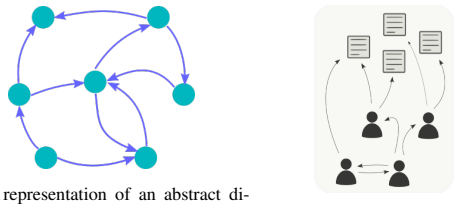
An overview of the current state of the field does, to the best of our knowledge, not exist. In this work, we aim to provide a brief introduction to deep learning and Graph Neural Networks (GNNs), followed by a survey of several graph neural network models for the analysis of time series for pattern recognition. We will focus on how the various approaches work and how these compare to one another, based on their specific applications and effectiveness.

This work is structured as follows: In Section 2, we describe the datasets on which the models are tested. In Section 3, we describe the models themselves, and in particular their networks, data representation methods and goals. The metrics used to compare the models are described in Section 4, and the results obtained form each method are in Section 5. The results are discussed in Section 6, and our findings are concluded in Section 7. First, we will provide definitions and context for the topic at hand.

A graph G is defined as a tuple $G = (V,E)$, with the vector V representing the *vertices* (also called *nodes* or *points*), and the vector E representing the *edges* i.e. the connections between the vertices. A graph can either be directed or undirected, weighted or unweighted, and signed or unsigned. In the context of GNNs, we mainly consider unsigned graphs without self-loops and without multiple edges. Figure 1a shows such a graph. Depending on the data and model,

- *Ayça Avcı, MSc, is a student Computing Science at the University of Groningen, E-mail: a.avci@student.rug.nl.*
- *Jeroen de Baat, MSc, is a student Computing Science at the University of Groningen, E-mail: j.de.baat@student.rug.nl.*

Manuscript received 24 February 2021; posted online 15 April 2021.
For information on obtaining reprints of this article, please contact the CS Student Colloquium of the University of Groningen, faculty of Science and Engineering.



(a) A representation of an abstract directed graph, with unlabeled vertices and edges.

(b) A social network represented as a graph.

Fig. 1: Two examples of graphs representations.

this definition of a graph can be extended. For example, feature vectors can be added for the vertices and edges to express more complex information. Figure 1b shows an example of a more complex graph of a social network, where some vertices represent users and others represent posted messages. Here, an edge between two users could mean that the users know each other, and an edge between a user and a message could mean that the user has posted that message.

Deep Learning is a field within Artificial Intelligence which is defined by the use of artificial neural networks with multiple layers between the input layer and the output layer. These so called *Deep Neural Networks* (DNNs) have been used for a variety of applications with data of different natures, such as image processing [20] and audio signal processing [22].

Extensive work has been done on the use of DNNs on non-spatiotemporal graphs [39], with applications ranging from social networks [28, 38] to biology networks. Zhang et al. [39] have identified several challenges in using DNNs to model graphs in particular:

- Graphs often have irregular structures which makes the application of common operations in neural networks, such as convolution, more difficult.
- The heterogeneity and diversity of graphs: they can be heterogeneous or homogeneous, weighted or unweighted, and signed or unsigned. In addition to this, the learning in graphs can be node-focused (e.g. node classification, node prediction, link prediction) or graph-focused (i.e. graph classification, graph generation), depending on the nature of the data and the application. Consequently, there is no general modeling approach for all graphs. Instead, each type of graph and application requires its own approach.
- Graphs can be very large, requiring the algorithms operating on them to be very efficient.
- Graphs often represent data from disciplines such as biology, chemistry, and the social sciences. Understanding the nature of

Data set	Size	Min	Max	Mean	SD
Japan-Prefectures	47×348	0	26635	655	1711
US-Regions	10×785	0	16526	1009	1351
US-States	49×360	0	9716	223	428

Table 1: "Dataset statistics in terms of min, max, mean, and standard deviation (SD) of patient counts; dataset size means the number of locations multiplied by number of weeks" [8].

the data is often essential to modeling, making the modeling process require interdisciplinary knowledge.

Despite these challenges, many models have been proposed, categorized by Zhang et al. [39] into Graph Recurrent Neural Networks (GRNNs) [24], Graph Convolutional Networks (GCNs) such as [2], Graph Autoencoders (GAEs) such as [27], Graph Reinforcement Learning, and Graph Adversarial Networks. For each of these models, variants exist depending on the nature of the data and modeling task.

As indicated previously, many real world relations — which can be represented using a graph — change over time. The change in relations can therefore be represented as a *sequence* of graphs, also known as *temporal* graphs. The patterns in these graphs may not only extend to the static nodes and edges themselves, but also to their relation to the temporal dimension. Therefore, specialized models are required to perform pattern analysis on these graphs.

2 Datasets

In this section, we describe the datasets used by the models surveyed.

2.1 CalendarGNN: Calender Graph Neural Networks

The CalendarGNN model has been trained and tested using large-scale user behavior logs which have been collected from two real portal websites. The data contains articles and news updates on several topics. The two spatiotemporal datasets are created as $B^{(w1)}$ and $B^{(w2)}$ [28]. These datasets provide spatiotemporal behavior logs of the browsing behavior of users from these two websites, both ranging from January 1, 2018 to June 30, 2018. Each dataset is filtered to 10000 users, who have most clicks, after users have been anonymized [28].

The 3 user attributes used for prediction tasks are as follows:

- Gender: The user's binary gender, where the gender is {"f", "m"}, "f" represents female, and "m" represents male.
- Income: The user's categorical income level, where the income is in $\{0, 1, \ldots, 9\}$. Larger values represent a higher annual income and 0 represents unknown.
- Age: The user's age according to their registered birthday.

2.2 Cola-GNN: Cross-location Attention based Graph Neural Networks

Deng et al. [8] made use of the following datasets as shown in Table 1 for their experiments: The Infectious Disease Weekly Report (IDWR) in Japan to obtain Japan-prefectures data, the Center for Disease Control (CDC) in the United States to obtain influenza data about the US-states, and the ILINet portion of the United States Department of Health and Human Services (US-HHS) to obtain data about the US-region.

2.3 Examining COVID-19 Forecasting using Spatio-Temporal Graph Neural Networks

Kapoor et al. [15] took advantage of the following datasets for the modeling: The New York Times (NYT) COVID-19 dataset for common node features such as day, past cases, and past deaths; the Google COVID-19 Aggregated Mobility Research Dataset to obtain inter-county flows and intra-county flows to establish the graph neural network; and the Google Community Mobility Reports to obtain summarized mobility trends at places which are aggregated at the county level.

2.4 Traffic Flow Prediction via Spatial Temporal Graph Neural Network

Wang et al. [30] tested the framework on two real-world traffic datasets as seen Table 2:

Dataset	Sensors	Length	Unit	Size
META-LA	207	4 month	5 min	34,272
PEMS-BAY	325	6 month	5 min	52,116

Table 2: "Dataset statistics in terms of sensors, length, unit and size [30].

- METR-LA: A traffic dataset which is centered around the LA county road network [14] and includes high-resolution spatio-temporal transportation data. Loop-detectors in the network supply traffic speed or volume data as well.
- PEMS-BAY: A traffic dataset which is supplied by the California Department of Transportation (Caltrans) Performance Measurement System (PeMS) [4] to measure traffic in the Bay Area.

3 GNN Methods

In this section we will describe the networks of several spatiotemporal GNN models.

3.1 CalendarGNN: Calender Graph Neural Networks

Wang et al. have developed the CalendarGNN model [28] which predicts specific user features (e.g. binary gender, income and age) based on spatiotemporal behavior data. Various methods exist [7, 13, 16, 19, 26] which aim to predict features using merely temporal (sequential) data, however, the authors state that user behavior often follows a spatiotemporal pattern which can be modeled and used to generate more accurate predictions than the existing methods. The prediction of user behavior has applications in content recommendation and targeted advertising.

3.1.1 The CalendarGNN network

The CalendarGNN model takes as input a network consisting of three sections: locations, timestamps, and items (e.g. reading a news article, clicking on a website, posting a message on social media). Depending on the type of data, the raw features are transformed using a Multilayer Perceptron (i.e. a variation on the original perceptron [23] which uses multiple layers) or Bidirectional Long Short-Term Memory [25] encoder into dense hidden representations. These representations are subsequently concatenated into a vector. The set of all item and location vectors and then embedded in their respective layers.

The item embeddings are then aggregated together with the temporal data, resulting in session embeddings, which in turn are aggregated into separate embeddings for the hour, week, and weekday time units. The session embeddings are also combined with the location embeddings. The embeddings per time unit and location are then aggregated into patterns embeddings, which are fused (by concatenation) into a final user embedding. This user embedding is the input for a dense layer resulting in a prediction. Note that the authors chose hour, week, and weekday time units based on their data and application, however other time units can be used as well.

The aggregation layers use the Gated Recurrent Unit (GRU) [5] for the aggregation function, and a non-linear activation function, e.g. ReLU [21]. The temporal aggregation layer partitions the continuous timestamps of the sessions into discrete time units, aggregate sessions of the same time unit, and aggregate time unit embeddings into a temporal pattern embedding. For the mathematical definitions of each operation, please consult the original paper.

The CalendarGNN model has two limitations. First, the different time units (i.e. week, hour, weekday) are all considered equally important in the aggregation step into patterns, while this may not reflect the input data. Second, the spatial and temporal data are processed separately, while the relation between the two should be captured by the model as well to achieve true modeling of spatiotemporal patterns. To achieve this, the authors propose a variation on the CalendarGNN model called CalendarGNN-Atnn [28]. This model is identical to CalendarGNN, except that it uses pattern definitions which allow for location-time interactions.

3.2 Cola-GNN: Cross-location Attention based Graph Neural Networks

The forecasting of influenza-like illnesses (ILI) is of high importance to epidemiologists in terms of resource allocation and the intervention of outbreaks. Deng et al. [8] focused on the long-term forecasting of ILI by using influenza surveillance data from several locations. It was difficult to accurately forecast long-term epidemics due to limited accountability of short-term input data and the change in the influence of other locations on any location. Deng et al. [8] aimed to construct a long-term prediction of the spread of ILI by accounting for a limited time range of data with deep spatial representations inside a graph propagated model. They implemented a graph neural network framework to model epidemic propagation at the level of the population. Furthermore, they explored capturing sequential dependencies in local time-series data through recurrent neural networks; and identify short- and long-term patterns through dilated temporal convolutions [8].

The framework, as shown in Figure 2, is comprised of: location-wise interactions (node attributes) to be caught through location-aware attention, short-term and long-term local temporal dependencies (node attributes) to be caught through dilated convolution layer, and the combination of the temporal features and the location-aware attentions through global graph message passing to make forecasts on newly learned hidden location embeddings [8].

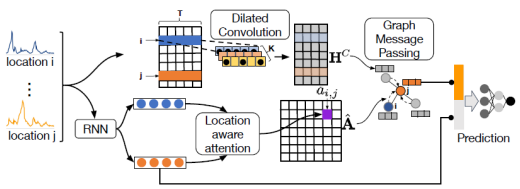


Fig. 2: The Cola-GNN framework [8].

3.2.1 Direct spatio influence learning

Deng et al. [8] constructed a dynamic model to assess the impact ILI in one location can have on the ILI of another location. They first utilized a Recurrent Neural Network (RNN) to learn the hidden states of each location from a certain period. Then, using the data from the RNN, they defined a general attention coefficient to measure to what extend two locations impact each other [8]. Finally, they include the spatial distance between the two locations in their calculations. The feature fusion gate is dynamically learned, and then models the influence that two locations have on each other by weighing the geographic and historic information [8].

3.2.2 Multi-scale dilated convolution

Convolutional Neural Networks (CNN) have proven very accurate in determining grid data, sequence data, and other local patterns. Deng et al. [8] aimed to use CNN for graph message passing. Hence, they adopted a multi-scale dilated convolutional module consisting of multiple parallel convolutional layers with different dilation rates, but the same filter and stride size. They then used multiple filters to produce different filter vectors [8]. They proceeded to concatenate these filter vectors to get the final convolution output. This output encodes local patterns into short-term and long-term trends [8].

3.2.3 Graph message passing

Using Graph Neural Networks (GNN), they designed a flu propagation model. They modeled ILI propagation among different locations, where each location is a node in a graph [8]. In their calculations, the dilated convolved features are used instead of the original time series since multiple levels of granularity can be captured in the hidden temporal features. Using the RNN hidden states and the graph features, they send their combination to the output layer to obtain their prediction [8].

3.3 Examining COVID-19 Forecasting using Spatio-Temporal Graph Neural Networks

During the COVID-19 pandemic, being able to accurately forecast caseload is highly necessary for numerous reasons, such as controlling outbreaks. Currently, two approaches of modeling COVID-19 outbreak are most commonly used: the mechanistic approach, and the time series learning approach [15]. Both approaches usually only depend on information from a single location or nearby locations where a pattern emerged, in forecasting for that location. Utilizing the widespread use of GPS-enabled mobile devices, Kapoor et al. [15] believe that they can build a better model by using more accurate real-time data and developing an approach that combines both the above approaches. They proposed a spatio-temporal graph neural network that uses precise mobility data to forecast daily new COVID-19 cases. The key discernment of the GNN-model is that the input node's signal transformation can be associated with the information propagation of a node's neighbors. This serves to better notify the future hidden state of the original input. The messaging framework designed by Gilmer et al. [10] is a great example of this. The messages are first propagated at the neighboring nodes and then aggregated to obtain new representations [15].

3.3.1 Modeling the COVID-19 graph

Multiple time-series sequences are most often used in the modeling of infectious diseases. However, this method does not take the human mobility across locations into account. Kapoor et al. [15] created a graph with different edge types to model both spatial and temporal dependencies. The edges in the spatial domain represent inter-location movement and are weighted by normalizing the mobility flow against the intra-flow. In Figure 3, edges in the temporal domain represent connections to the past days [15].

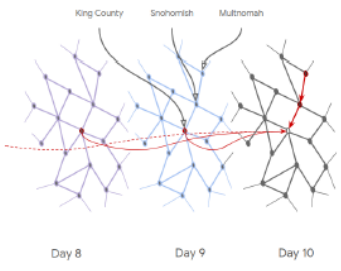


Fig. 3: "A slice of the COVID-19 graph showing spatial and temporal edges (highlighted in red) across three days." [15].

3.3.2 Skip-Connections Model

Concerning graph convolutions, Kapoor et al. [15] integrated skip-connections between layers in the spectral graph convolution model designed by Kipf and Welling [16] to avoid diluting the self-node future state, represented in Figure 4. A learned embedding from the temporal node features is concatenated to the output of each layer [15].

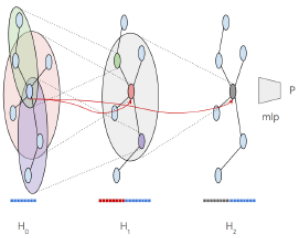


Fig. 4: 2-hop Skip-Connection model [15].

3.4 Traffic Flow Prediction via Spatial Temporal Graph Neural Network

The analysis and predictability of dynamic traffic conditions are of key importance in the planning and construction of roads and future city expansion. The problem lies in the increasing difficulty of traffic flow predictability [30]. This is due to the volatility of vehicle flow in the temporal dimension in the short-term, as well as the complex relationship between the vehicles and the roads in the spatial dimension. The inclusion of road crossings and vehicle lanes, which come with increased complexity, further decreases the predictability of traffic. A time series can be implemented on a road network to represent traffic data, where the spatial proximity of separate roads can be used to connect them [30].

Wang et al. [30] propose a new Graph Neural Network layer with a position-wise attention mechanism such that the traffic flow from connected roads can be better aggregated. Local and global temporal dependence is captured using the combination of a recurrent network and a Transformer layer. A new Spatial-temporal Graph Neural network (STGNN) is specifically designed to model series data with topological and temporal dependency. This new framework is finally tested on real traffic datasets to obtain results about short-term traffic speed predictions [30]. The experiments prove the proposed model is significantly better than other previously used methods. They plan to predict future traffic flow by utilizing historical traffic flow data. This data can be represented on a traffic network of connected traffic sensor nodes with proximity weighted edges [30].

Figure 5 illustrates the proposed spatial-temporal Graph Neural Network framework. There are three main components to the framework: Spatial Graph Neural Network (S-GNN) [30] layers that use the traffic network to represent the spatial relations between different roads, the Gated Recurrent Unit (GRU) layer which serves to represent the temporal relation sequentially, and the Transformer layer which serves to directly represent the long-term temporal dependence on the sequence. The S-GNN layer models the spatial relation between the nodes. As in Figure 5, the S-GNN layer is applied to both the input and the hidden representations of the GRU. Although they represent different perspectives, both the GRU layer and Transformer layer represent the temporal dependency of each node individually [30].

4 Evaluation Metrics

In the experiments, the following evaluation metrics are used to assess the performance of the methods:

- *Pearson's correlation coefficient (PCC)*: Measures the linear dependence between two variables [8].

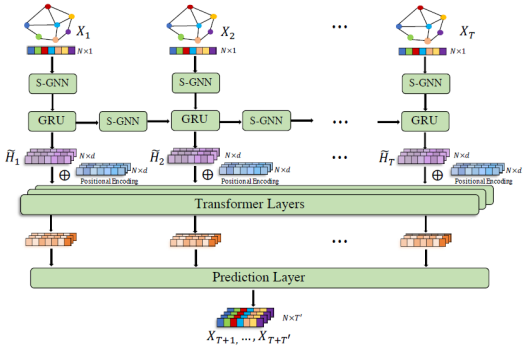


Fig. 5: Spatial Temporal Graph Neural Network Framework [30].

$$PCC = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{1}$$

- *Root Mean Squared Error (RMSE)*: Measures the difference between two values (true values and the predicted values) after the projection of normalized values to the real range [8].

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{2}$$

- *Root Mean Squared Logarithmic Error (RMSLE)*: Measures the difference between the logarithms of two values (the true values and the predicted values) after the projection of normalized values to the real ones [9].

$$RMSLE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2} \tag{3}$$

- *Mean Absolute Error (MAE)*: Measures the absolute difference of two continuous variables [30].

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{4}$$

- *Mean Absolute Percentage Error (MAPE)*: Measures the absolute difference divided by true value of two continuous variables (the true value and the predicted value) [30].

$$MAPE = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{5}$$

- *R-squared (R^2)*: Represents the proportion of the variance for a dependent variable which is explained by an independent variable [6].

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{6}$$

Here, we list the surveyed models and the metrics used:

- CalendarGNN: Calender Graph Neural Network: *PCC, RMSE, MAE, R^2*.
- Cola-GNN: Cross-location Attention based Graph Neural Networks: *PCC, RMSE, MAE.*

- Examining COVID-19 Forecasting using Spatio-Temporal Graph Neural Networks: *PCC, RMSE, RMSLE.*
- Traffic Flow Prediction via Spatial Temporal Graph Neural Network: *MAE, MAPE.*

5 Findings

5.1 CalendarGNN: Calender Graph Neural Networks

The CalendarGNN and CalendarGNN-Attn models have been tested using large-scale user behavior logs from two real portal websites providing news updates and articles on various topics. The models were used to predict the user's binary gender, income and age. Compared against various baseline methods as well as logistic/linear regression (LR), LearnSuc [29], and SR-GNN [31], the proposed models outperform all other models based on nearly all different metrics used.

5.2 Cola-GNN: Cross-location Attention based Graph Neural Networks

The results of their methods are evaluated in terms of the Root Mean Square Error (RMSE) and Pearson's Correlation (PCC) metrics [8]. Leadtime represents the number of weeks that is predicted by the model in advance. They also showcase a relative performance advantage of their method to the best baseline model. The variance in their data is the cause of the large differences in RMSE values across the different datasets. Considering a relatively small leadtime window, their proposed method performs the best and the most stable for all the datasets. This is also true for most datasets if they consider a long lead time window [8]. The performance results of Vector Autoregression (VAR) and RNN suggest the necessity to control model complexity when only insufficient data is available, as well as that long-term ILI forecasts require a better design to capture the spatial and temporal dependencies. All methods perform relatively equally well when a short leadtime window is used, but the simpler methods quickly degenerate into severe inaccuracy as the leadtime window is increased [8].

5.3 Examining COVID-19 Forecasting using Spatio-Temporal Graph Neural Networks

Kapoor et al. [15] represent the forecasting performance of the spatio-temporal GNN in comparison to some baseline models. The Root Mean Squared Log Error (RMSLE) and Pearson Correlation (PCC) [15] evaluation metrics are represented for the predicted caseload and the case deltas. The results are that the graph neural network surpasses the baselines and obtains the best score on nearly every evaluation metric. Furthermore, inserting further mobility data improves performance for all the deep models, yet weakens the performance of the Autoregressive Integrated Moving Average (ARIMA) baseline [15]. It is understood this to be due to ARIMA assuming fixed dynamics and a linear dependence on county-level mobility.

5.4 Traffic Flow Prediction via Spatial Temporal Graph Neural Network

Wang et al. [30] measure the forecasting performance of different methods using Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) as metrics [30]. Since diverse traffic conditions can be expected from different areas, an absolute error could be useful to indicate where the model is overfitting to relatively simple samples. Meanwhile, in volatile areas, a square error is more scrutinizing and can therefore give better performance under complex situations [30].

A time-scale forecasting of 15, 30, and 60 minutes was averaged for the results. The STGNN proposed framework is superior to all other methods across all timescales, errors, and datasets. The superiority of STGNN was more profound in PEMS-BAY than on METR-LA [30]. Two variations of STGNN were tested as well; one without gated recurrent units and the other without transformer layers. These are also mostly superior to all the baseline methods across all timescales, errors, and datasets, and yet were still inferior to the STGNN with all the components present. This indicates how all components discussed are key to optimal results but are still significant even without GRU and Transformer [30].

6 Discussion

In the previous sections, we have described four models for pattern analysis on graphs in time series. Our observations are that this is indeed a relatively new field, with proposed models that are somewhat 'isolated', meaning that each model has a unique approach, application and is trained on a very specific dataset. We do not see any convergence towards one particular approach that performs the best, although we do observe some very general patterns.
We see that the proposed models:

- operate on spatiotemporal graphs,
- use frameworks with multiple sequential processing steps, and
- are composed of existing models, e.g. CNNs, GRUs, MLPs.

It is difficult to directly compare the models as the approaches (and thus the frameworks) are significantly different. Any differences in effectiveness cannot easily be attributed to any specific minor or major difference in the framework. In addition to this, the datasets used in the experiment are all different, as are the goals. Also note that all proposed models substantiate their results by performing an experiment on a very limited number of datasets. Analytical analysis, for example of time and space complexities, of the proposed models was largely absent.

When choosing a model with a particular application in mind, having domain knowledge will be very useful, if not essential, as the proposed models have been developed with a specific goal in mind, and may not perform well under different circumstances.

For more context, a broader look at the field is summarized in Table 3, listing the (abbreviated) model names, network types, datasets used and metrics. We see here, again, that most models use some form of convolution and that there is very little overlap in the datasets used.

7 Conclusion

In this literature review, we discussed how different Graph Neural Networks are used for analyzing time series data. All the proposed models are effective in producing results for the application they were designed for. However, lacking information, little can be said about which model is better in extracting patterns from spatiotemporal graph data in general. We concluded that since each method use different dataset and evaluation metrics, comparison between proposed methods is not trivial.

To reach definite conclusions about the effectiveness and generalization ability of the proposed models, more research is needed in which the models with the same goal are directly compared using the multiple identical datasets. In addition to this, more general variations of the models could be designed with the goal of making them suitable for a broader range of applications. These variations could then be directly compared in the general ability to extract patterns from spatiotemporal graph data, perhaps leading to more fundamentally relevant results. However, the models' respective usefulness related to their specific application is certainly valuable on a practical level.

In their survey of Deep Learning on Graphs, Zhang et al. [39] have categorized four possible future directions, which includes the study of dynamic graphs. The other directions are: new models for unstudied graph structures, compositionality of existing models, and interpretability and robustness. All three can be studied on dynamic graphs as well.

Acknowledgements

The authors wish to thank expert reviewer dr. Estefanía Talavera Martínez and reviewers Nitin Paul and Floris Westerman.

Model name(s)	Network type(s)	Data set(s)	Metric(s)
ASTGCN [12]	Convolutional, attention.	PeMSD4, PeMSD8.	RMSE, MAE.
CalendarGNN, CalendarGNN-Attn [28]	Attention, aggregation.	Self-collected.	Mean accuracy, AUC, F1, MCC, Cohen's kappa, R^2, MAE, RMSE, PCC.
Cola-GNN [8]	Convolutional, recurrent.	Japan-Prefectures, US-States, US-Regions.	RMSE, MAE, PCC, Leadtime.
STGNN [15]	SGC, message passing.	NYT COVID-19, Google COVID-19 AMR, Google CMR.	RMSLE, PCC.
STGNN [30]	Convolutional.	METR-LA, PEMS-BAY.	MAE, MAPE.
DMVST-Net [35]	Convolutional, LSTM.	Self-collected.	RMSE, MAPE.
DCRNN [18]	Convolutional, recurrent.	METR-LA, PEMS-BAY.	RMSE, MAE, MAPE.
WD-GCN, CD-GCN [19]	Convolutional, LSTM.	DBLP (subset), CAD-120.	Monte Carlo Cross-Validation, Wilcoxon test.
Graph WaveNet [32]	Convolutional.	METR-LA, PEMS-BAY.	MAE, RMSE, MAPE.
STDN, STDN-Graph [34]	Convolutional, FGN, PSAM.	STDN: NYC-Taxi, NYC-Bike. STDN-Graph: Self-collected.	RMSE, MAPE.
STGCN [33]	Convolutional.	Deepmind Kinetics human action dataset, OpenPose, NTU-RGB+D.	Accuracy percentage.
STGCN [37]	Convolutional, gated convolutional.	BJER4, PeMSD.	MAE, MAPE, RMSE.
StemGNN [3]	Spectral, spectral convolution.	METR-LA, PEMS-BAY, PEMS07, PEMS03, PEMS04, PEMS08, Solar, Electricity, ECG5000, COVID-19.	MAE, MAPE, RMSE.

Table 3: Overview of models

REFERENCES

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[2] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs, 2014.

[3] D. Cao, Y. Wang, J. Duan, C. Zhang, X. Zhu, C. Huang, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang. Spectral temporal graph neural network for multivariate time-series forecasting. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17766–17778. Curran Associates, Inc., 2020.

[4] C. Chen, K. Petty, A. Skabardonis, P. Varaiya, and Z. Jia. Freeway performance measurement system: Mining loop detector data. *Transportation Research Record*, 1748(1):96–102, 2001.

[5] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[6] A. Colin Cameron and F. A. Windmeijer. An r-squared measure of goodness of fit for some common nonlinear regression models. *Journal of Econometrics*, 77(2):329–342, 1997.

[7] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering, 2017.

[8] S. Deng, S. Wang, H. Rangwala, L. Wang, and Y. Ning. Cola-gnn: Cross-location attention based graph neural networks for long-term ili prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 245–254, New York, NY, USA, 2020. Association for Computing Machinery.

[9] K. V. Desai and R. Ranjan. Insights from the wikipedia contest (ieee contest for data mining 2011), 2014.

[10] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry, 2017.

[11] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI*, 2019.

[12] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:922–929, 07 2019.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[14] H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi. Big data and its technical challenges. *Commun. ACM*, 57(7):86–94, July 2014.

[15] A. Kapoor, X. Ben, L. Liu, B. Perozzi, M. Barnes, M. Blais, and S. O'Banion. Examining covid-19 forecasting using spatio-temporal graph neural networks, 2020.

[16] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks, 2017.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[18] Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, 2018.

[19] F. Manessi, A. Rozza, and M. Manzo. Dynamic graph convolutional networks. *Pattern Recognition*, 97:107000, Jan 2020.

[20] S. Mohapatra, T. Swarnkar, and J. Das. 2 - deep convolutional neural network in medical image processing. In V. E. Balas, B. K. Mishra, and R. Kumar, editors, *Handbook of Deep Learning in Biomedical Engineering*, pages 25–60. Academic Press, 2021.

[21] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[22] H. Purwins, B. Li, T. Virtanen, J. Schluter, S.-Y. Chang, and T. Sainath. Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):206–219, May 2019.

[23] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[24] L. Ruiz, F. Gama, and A. Ribeiro. Gated graph recurrent neural networks. *IEEE Transactions on Signal Processing*, 68:6303–6318, 2020.

[25] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[26] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*, pages 362–373. Springer, 2018.

[27] R. van den Berg, T. N. Kipf, and M. Welling. Graph convolutional matrix completion, 2017.

[28] D. Wang, M. Jiang, M. Syed, O. Conway, V. Juneja, S. Subramanian, and N. V. Chawla. Calendar graph neural networks for modeling time structures in spatiotemporal user behaviors. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 2581–2589, New York, NY, USA, 2020. Association for Computing Machinery.

[29] D. Wang, M. Jiang, Q. Zeng, Z. Eberhart, and N. V. Chawla. Multi-type itemset embedding for learning behavior success. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2397–2406, 2018.

[30] X. Wang, Y. Ma, Y. Wang, W. Jin, X. Wang, J. Tang, C. Jia, and J. Yu. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of The Web Conference 2020*, WWW '20, page 1082–1092, New York, NY, USA, 2020. Association for Computing Machinery.

[31] C. Wu and M. Yan. Session-aware information embedding for e-commerce product recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 2379–2382, New York, NY, USA, 2017. Association for Computing Machinery.

[32] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang. Graph wavenet for deep spatial-temporal graph modeling, 2019.

[33] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition, 2018.

[34] H. Yao, X. Tang, H. Wei, G. Zheng, and Z. Li. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction, 2018.

[35] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li. Deep multi-view spatial-temporal network for taxi demand prediction, 2018.

[36] B. Yu, H. Yin, and Z. Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, Jul 2018.

[37] B. Yu, H. Yin, and Z. Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, 2018.

[38] C. Zang, P. Cui, and C. Faloutsos. Beyond sigmoids: The nettide model for social network growth, and its applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 2015–2024, New York, NY, USA, 2016. Association for Computing Machinery.

[39] Z. Zhang, P. Cui, and W. Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020.

A Comparative Analysis of Swarm Intelligence-Based Clustering Algorithms

Sjoerd Bruin and Jasper van Thuijl

Abstract— Unsupervised clustering is a difficult problem because in general we do not have a good idea about the number of expected clusters. As a result, we are at risk of underclustering or overclustering. Further problems with determining the shape of clusters compound these potential issues even more. One recent development in unsupervised learning is the development of algorithms based on the emergent behaviour of swarms analogous to natural swarms such as flocks of birds, colonies of ants, or schools of fish. These swarm-based clustering algorithms typically maximise a fitness function in order to guide positioning of individual agents in the artifical swarm. However, the DataBionic Swarm (DBS) framework does away with the fitness function optimisation and instead uses game-theoretical payoff functions to guide the positioning of individual agents. This represents a novel approach to implementing the behaviour of a swarm. In addition to this novel swarm implementation, the DataBionic Swarm framework also visualises a two-dimensional projection of a higher-dimensional dataset in order to facilitate understanding and manual clustering, which can then help inform the clustering process.
In this study, we compared the DataBionic Swarm with a more traditional state-of-the-art implementation of a swarm that uses a hybrid of Particle Swarm Optimisation and Genetic Algorithm, and looked at a more general modification for swarm-based algorithms that uses computational centroids instead of particle positions. We found that the modification of the DBS projection algorithm with the computational centroids showed significant promise as a potential improvement of the DBS framework. In addition, we looked at the Particle Swarm Optimisation with Genetic Algorithm performance. The main advantage of this algorithm over the DataBionic Swarm framework is that it has much faster computation times: its convergence is fast, whereas evaluating the DataBionic Swarm projection algorithm can take up to a day. The DataBionic Swarm framework provides a powerful clustering algorithm due to its ability to handle non-circular clusters well. We identified computational centroids as an enhancement that could potentially improve this property further by centering the clusters in a better way. Future research should focus on the applicability of this improvement in order to try and improve the DataBionic Swarm framework further.

Index Terms—Self-organised clustering, clustering, semi-interactive clustering, unsupervised learning, swarm intelligence, particle swarm optimization.

1 Introduction

The DataBionic Swarm (DBS) framework that [12] present, combines a number of aspects that are important in unsupervised learning. Firstly, it contains a projection algorithm that projects a dataset with a large dimensionality down to a two-dimensional space. Secondly, the algorithm provides an intuitive topographic map of this projection. With this visualisation, we can visually judge the number of clusters present in a dataset, and can judge the connectedness of the dataset by means of a topographic analogy. Lastly, the DBS framework provides two clustering options based either on compactness or connectedness. The visualisation aspect of the DBS framework is very useful for visually judging the number of clusters and for learning about the structure of the data. Naturally, we have to make the assumption that the two-dimensional projection algorithm does indeed map a high-dimensional dataset to a two-dimensional projection in a way that preserves the structure of the high-dimensional dataset as closely as possible. To this end, DBS uses a swarm-based algorithm that maps agents to a two-dimensional grid and moves them around on that grid in such a way that it preserves the high-dimensional properties of the original dataset as well as possible.

At the early stages of swarm intelligence research, [1] gave the first major description of swarm intelligence in the context of artificial intelligence, and identified several of the core principles that a swarm-based algorithm should display. They also give the main emergent behaviours that a swarm should exhibit in order to facilitate self-organisation.

There are two main types of swarm-based clustering algorithms: the first one is Ant Colony Systems (ACS), which communicate indirectly by using scent to inform future movement and, as [8] mention, is often used in data mining applications. The second type is Particle Swarm Optimisation (PSO), which was first proposed by [3]. PSO algorithms communicate directly rather than via indirect queues such as scent. The polar swarm algorithm that the DBS framework uses for projecting the dataset to a two-dimensional space is similar to the movements strategies of agents that PSO employs. The polar swarm algorithm also uses a scent function similar to the ones used in ACS algorithms.

The concept of an artifical swarm agent, sometimes called a DataBot, was proposed by [13]. The DBS framework uses DataBots in the polar swarm algorithm to create a two-dimensional projection of a larger-dimensional dataset. In this study, we reviewed the DBS framework and compared its functionality with other state of the art approaches to swarm-based learning. In particular, we looked at Particles Swarm Optimisation with the Genetic Algorithm that [7] presented in order to compare the performance of both swarm-based clustering approaches. In addition, we looked at an improvement to swarm-based algorithms proposed by [10]. This improvement, which uses computational centroids for distance measures rather than agent positions, has shown tangible benefits for many clustering approaches, and in this study we looked at the applicability of this technique to the projection algorithm that DBS uses.

The aim of this study was to assess the quality of the DBS framework, and to propose ways in which we could improve the underlying algorithm. This helps to determine how accurate this framework is for unsupervised learning tasks, and helps to improve the quality of this framework in order to advance the state of the art in unsupervised clustering. The DBS framework is useful because it combines projection, visualization, and clustering in a semi-interactive framework. The visualization that the framework produces succeeds in producing an easy-to-understand representation of the clusters. Furthermore, the clustering accuracy of the framework is also quite high, outperform-

- *Sjoerd Bruin is with University of Groningen, E-mail: s.bruin.5@student.rug.nl*
- *Jasper van Thuijl is with University of Groningen, E-mail: j.m.van.thuijl@student.rug.nl*

ing other unsupervised clustering techniques. However, the reliance of DBS on radial distance measures between data elements might impede its effectiveness for non-radially distributed datasets. Therefore, we propose that an approach using computational centroids could improve the accuracy of the algorithm even further. One downside of DBS is its computational complexity: it takes many hours to compute the result even for datasets with only a few thousand entries. Other PSO algorithms typically have much faster convergence, making them more convenient to use.

In section 2, we examine the theoretical underpinnings of the DBS framework and a PSO with Genetic Algorithm hybrid in detail, and we discuss the computational centroids modification. In section 3, we compare DBS and the other algorithms, and we highlight some approaches in these algorithms that could potentially improve DBS. Lastly, we discuss our findings and introduce future research opportunities in section 4.

2 METHODS

In this section, we discuss the different swarm-based learning algorithms that we are going to compare. We discuss the DataBionic Swarm (DBS), Particle Swarm Optimisation (PSO) combined with a genetic algorithm (GA), and Fitness Evaluation with Computational Centroids (FECC). We aim to highlight the difference between these approaches in order to make clear which improvements could benefit DBS.

The algorithms under consideration in this paper all belong to the class of particle swarms. A particle swarm is inspired by biological swarms, for example ants and schools of fish. Such swarms exhibit five significant properties, summarised by [12], that make them very useful as a template for clustering algorithms:

- **Homogeneity:** Every agent has the same behaviour.
- **Locality:** The motion of each agent is influenced only by its nearest neighbours.
- **Velocity Matching:** Each agent attempts to match the velocity of its closest neighbours.
- **Collision Avoidance:** Each agent avoids collisions with nearby neighbours.
- **Flock Centering:** Agents attempt to stay close to neighbouring agents.

The desirability of these five properties for a clustering algorithm is immediately obvious: the homogeneity principle ensures that data point behaviour depends only on the attribute values, not on any arbitrary differential treatment between otherwise similar data points. The locality principle puts a focus on the nearby data points, which are most likely to have a significant impact on the clustering result, in contrast to data points that are far from the current data point. The velocity matching principle makes sure that data points become organised at comparable speeds. This prevents the unfortunate situation where some data points reach a locally optimal position while other data points are still far from their local optimum, which potentially can cause some convergence issues. The collision avoidance principle prevents data points from converging on the same point in the attribute space. The flock centering principle, finally, encourages that data points which are close together try to stay close together, which allows for clusters to form.

2.1 Particle Swarm Optimisation with Genetic Algorithm

A difficulty in cluster analysis is determining the optimal number of clusters for unknown data. The DCPG algorithm proposed by [7] is a dynamic clustering technique that is capable of automatically determining the optimal number of clusters, which minimizes the need for user interactions. DCPG is a hybrid of Particles Swarm Optimization (PSO) and Genetic Algorithm (GA).

2.1.1 PSO

The key foundation of the DCPG algorithm is the PSO algorithm, which is based on swarm processes that can be observed in several animal species. For instance, if an individual bird knows the direction to food, it will transmit that information to its flock, which will in turn correct their direction. This will allow the other birds in the flock to move to the food location. Similarly to birds in a flock, PSO models both the individuals' knowledge of the search space and global knowledge of the search space. In the implementation of the algorithm, each potential solution is called a particle, and each particle keeps track of its own position and velocity [3]. One of the main advantages of PSO is high convergence velocity towards locally optimal solutions [7].

On initialization, a random number of particles is assigned a randomized velocity and direction. Iteratively, the particles move through the search space of the input problem's dimension N. In each iteration, an objective fitness function f is evaluated for each particle based on its position, producing a real number. The personal best value and location are tracked for each particle, and stored in $\mathbf{y}_i : \mathbb{R}^N$. This models the individuals' cognition of the search space. The overall best value and location obtained by any particle in the population is stored in $\hat{\mathbf{y}} : \mathbb{R}^N$, which can be interpreted as the social model. The velocity and direction of a particle is controlled by these values, but also depends on the previous velocity of the particle.

In Equation 1 the velocity update step is specified for an individual particle i for each dimension $j \in \{1,...,N\}$. In each iteration, individual particles accelerate towards both their previous best position and the global best position randomly by factors $r_{1,j} \sim U([0,1])$ and $r_{2,j} \sim U([0,1])$ respectively. Besides the random factors, hyperparameters c_1 and c_2 are used to influence the learning rates of both the cognition and social models respectively. The hyperparameter w for inertia is used to control the impact of the previous velocity of the particle on the new velocity. A low inertia value favours exploitation, while a high inertia value favors exploration around the best solutions found thus far [9]. Figure 1 shows the effects of the particles' inertia, cognitive model and social model on individual particles in the PSO update step.

$$\begin{aligned} v_{i,j}(t+1) =& w v_{i,j} + c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}) \\ &+ c_2 r_{2,j}(t)(\hat{y}_j(t) - x_{i,j}(t)) \end{aligned} \tag{1}$$

The new velocity $\mathbf{v}_i(t+1)$ is capped between $[-v_{max}, v_{max}]$, where v_{max} is a hyperparameter. After updating the velocity of a particle, its new position is calculated according to Equation 2.

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{2}$$

2.1.2 DCPG

The authors of the DCPG algorithm base their work on the Dynamic Clustering using Particle Swarm Optimisation (DCPSO) algorithm that was proposed by [9] for image segmentation. In DCPSO, a binary-PSO is used. This version of the PSO algorithm operates in binary space instead of continuous space, and is better suited towards discrete problems. To obtain the binary-PSO algorithm, the regular PSO algorithm only needs few minor modifications. The velocity update for a particle as described in Equation 1 remains the same, however, position $\mathbf{x}_i(t)$ and local best value $\mathbf{y}_i$ are restricted to the set $\{0,1\}$ in $\mathbb{Z}^N$. Furthermore, the velocity $\mathbf{v}_i(t+1)$ now represents a probability that a bit is flipped. The position update of a particle is modified by using a sigmoid function as shown in Equation 3, such that the new particle position is now calculated through Equation 4.

$$sig(\mathbf{v}_i(t+1)) = \frac{1}{1 + e^{\mathbf{v}_i(t+1)}} \tag{3}$$

$$\mathbf{x}_i(t+1) = \begin{cases} 1 & \text{if } rand() < sig(\mathbf{v}_i(t+1)) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

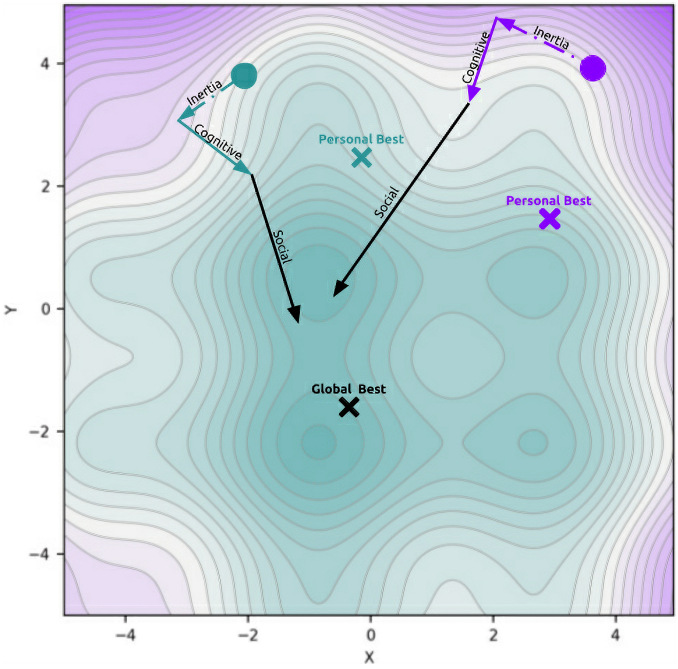


Fig. 1. Update steps of individual particles in the PSO algorithm in a two-dimensional search space. The effect of the social component and individual's knowledge of the search space are clearly shown. Free to use image adapted from [11].

In order to overcome the binary-PSO algorithm falling into the locally optimal solution instead of the global optimal solution, [7] implemented the crossover and mutation operators of GA into the DCPSO algorithm. The authors state that GA is especially suited towards large nonlinear space problems where solutions are unpredictable. Notable benefits of GA include increased search capability, high accuracy, and the ability to escape from locally optimal solutions. Other research also suggests that using a hybrid of PSO+GA is superior to using PSO or GA alone [6].

In every iteration of the DCPG algorithm, the population of particles is copied to a second population. The first population is kept intact, while on the second population two-point crossover is applied to $\mathbf{y}_i$ and $\hat{\mathbf{y}}$ and mutation for $\hat{\mathbf{y}}$. After this, population 1 and population 2 are combined, and the fitness values of all the particles are calculated. Elitist selection is applied to disregard the particles with the worst fitness values, and keep the particles with the highest fitness values for the population that will be used in the next iteration.

The resulting DCPG algorithm works as follows: In the first run, a set of cluster centroids $\mathbf{M}$ is randomly chosen from the input set of data points $\mathbf{Z}$, and a swarm of particles $\mathbf{S}$ is randomly initialized. Fitness values are calculated for all particles, after which binary-PSO is applied to the swarm of particles. Next, two-point crossover and mutation operations from GA are applied. Elitist selection selects the next population of particles and the steps are repeated for a predefined number of iterations. After the iterations have completed, the best set of cluster centroids, known as $\mathbf{M}_\tau$ remains. Subsequently, the k-means algorithm is applied to correct the particle centroids. Random components $\mathbf{M}_r$ are again chosen from the set $\mathbf{Z}$, and $\mathbf{M}$ is calculated as $\mathbf{M} = \mathbf{M}_\tau \cup \mathbf{M}_r$. The previous steps are repeated using the new $\mathbf{M}$ until a predefined number of iterations are met. On completion, $\mathbf{M}_\tau$ will contain the optimum number of cluster centroids, and n_τ the optimum number of clusters. A representation of the DCPG algorithm is shown in Algorithm 1.

In their research, [7] compared the DCPG algorithm to two binary-PSO and one GA-based clustering algorithms on four benchmark datasets and in one concrete case-study. In their benchmark study, DCPG converged in less iterations than the other algorithms, with similar computation times. Validation completed by the authors shows that DCPG achieved the lowest error rates in clustering results, and can accurately determine the correct number of clusters.

Algorithm 1 DCPG algorithm

INPUT: Set of data points $\mathbf{Z}$, maximum number of PSO+GA iterations a_{max}, maximum number of inner iterations b_{max}, population size s, inertia weight w, max velocity v_{max}, learning factors c_1, c_2, maximum number of clusters N_c, crossover rate and mutation rate
OUTPUT: Set of cluster centroids $\mathbf{M}$

Randomly choose set of data points $\mathbf{M}$ from $\mathbf{Z}$
Randomly initialize swarm $\mathbf{S} = \{\mathbf{x}_1, ...\mathbf{x}_i, ...\mathbf{x}_s\}$, where $\mathbf{x}_i = \{x_{i1}, ...x_{ik}, ...x_{iN_c}\}$ and $x_{ik} \sim U(0,1)$; // $x_{ik} = 1$ means that particle i belongs to cluster k, $x_{ik} = 0$ means the opposite
Randomly initialize particle velocities $\mathbf{v}_i$
for $a = 0 \rightarrow a_{max}$ **do**
  for $b = 0 \rightarrow b_{max}$ **do**
    $f = \sum_{k=1}^{N_c} \sum_{\forall \mathbf{Z} \in n_{ik}} \|\mathbf{Z} - \mathbf{M}_i\|$ // Calculate fitness value f for all particles
    for particle $i \in \mathbf{S}$ **do**
      Determine $\mathbf{y}_i$ and $\hat{\mathbf{y}}$
      Calculate new particle velocity $\mathbf{v_i}(t+1)$ using Equation 1
      Calculate new particle position $\mathbf{x_i}(t+1)$ using Equation 4
    end for
    Copy all particles to generate population 1
    Perform two-point crossover for $\mathbf{y}_i$ and $\hat{\mathbf{y}}$ and mutation for $\hat{\mathbf{y}}$ to generate population 2
    Combine populations and calculate fitness values
    Perform elitist selection to generate next population
  end for
  Apply K-means for correction of the particle centroids, resulting in $\mathbf{M}_\tau$
  Randomly choose a set $\mathbf{M}_r$ from $\mathbf{Z}$.
  $\mathbf{M} = \mathbf{M}_\tau \cup \mathbf{M}_r$
end for

2.2 DataBionic Swarm

DBS is not just a clustering algorithm but it is an entire clustering framework. This means that DBS tries to integrate the steps of cluster selection, classification, and visualization into a single approach. This becomes clear when we look at the polar swarm algorithm at the center of the DBS framework. The name polar swarm highlights that the algorithm evaluates the movement of agents in the swarm in log-polar coordinates. The aim of the polar swarm algorithm is to use the properties of swarm behaviour in order to produce a two-dimensional representation, denoted as the output space or projection space $O \subseteq \mathbb{R}^2$, of an N-dimensional dataset (where $N > 2$), denoted as the input space $I \subseteq \mathbb{R}^N$, using swarm properties to make sure that the two-dimensional representation maintains the inherent clusters that are present in the original N-dimensional representation.

The mechanics of a swarm lead to the concept of emergence due to four properties identified by [12]:

- **Randomness:** The update of agents contains a degree of randomness to prevent predictable movement patterns.
- **Temporal and structural unpredictability:** We do not make assumptions about the structure of the swarm or about the way in which the structure changes over time.
- **Multiple non-linear interactions among many agents**: Agents interact with one another in non-linear ways, leading to complex behaviour.
- **Irreducibility:** We cannot trace back the behaviour of the macroscopic swarm to the characteristics of individual agents.

Emergence produces swarm-level behaviour which we cannot accurately trace back to properties of the individual agents. This defini-

tion is suspicious, since it only tells us that there is some macroscopic behaviour for which we do not yet know the underlying microscopic dynamics, but emergence is useful as a concept nonetheless.

The polar swarm algorithm plays a game in the game-theoretical sense: it applies different strategies, which we understand to be different movements in the projection space, and it reaches a Nash equilibrium for the strategy with the best stability. The Nash equilibrium represents a state of the system where no change of an individual actor's strategy leads to a better payoff. We use a scent function similar to the one defined in equation 5 as the payoff function. We reproduce the polar swarm algorithm defined by [12] in algorithm 2 (we slightly changed the layout and included additional annotations, while preserving the core structure, so that the structure of the algorithm is clearer in our estimation).

Algorithm 2 Polar swarm projection algorithm

INPUT: Input space $I \subseteq \mathbb{R}^N$, output space $O \subseteq \mathbb{R}^2$, distance matrix $D(l, j)$
OUTPUT: Output space O with the projected positions of the data points

for all $z_i \in I$ **do**
  assign an initial random polar position $u_\phi(r) \in O$ on the grid to generate agents $b_i \in B$, where B is the set of agents
end for
for $R = \{R_{max} = Lines/2, ..., R_{min}\}$ **do**
  calculate chance $P(R)$ that an agent is allowed to jump
  repeat
    $s = \text{sample}(P(R), B)$ // take a distance-weighted random sampling of agents
    $m_k(s) = \text{uniform}(1, R_{max})$ for $k = 1, ..., \alpha$, with α the number of possible jump positions
    $l(s) = \text{argmax}_{j \in \{i, m_k(s)\}}(\lambda(b_j, B))$
    $l(!s) = i$ // Agents not selected for new positions stay in place
    $S = \sum_l \lambda_l(b_l, R)$ // Compute sum of scents
  until $\frac{\partial S(e, \lambda(R))}{\partial e} = 0$
end for
return Output space O with computed agent positions

Polar swarm randomly assigns all data points onto a position on a (toroidal) two-dimensional grid, and then attempts to take a number of movements for a random subset of the data points. It then evaluates the scent function to see whether the move improved the state of the agent. If so, it accepts the move. Otherwise, the move is reversed, and the agent stays in its current position. The algorithm starts by considering large jumps (i.e. a large value of R), in order to find large possible improvements in the beginning. Then, it decreases the size of R. For each value of R, polar swarm continues to make moves until the sum S of scents no longer changes during an epoch. During an epoch, the algorithm calculates a random sample which it will consider for a new position as a function of the jump size R, and then it determines a set of new positions $m_k(s)$ by moving a randomly chosen distance in the range $[1, R_{max}]$. After that, we determine whether one of the new positions has a better scent value than the current one. The agents that were not selected stay in their respective original positions.

In order to ensure that the data projection adheres to the swarm properties, [12] defines a general scent function $\lambda(b_j, R)$ given in equation 5.

$$\lambda(b_j, R) = \frac{\sum_{l \in I} h_R(d(j,l)) \cdot D(j,l)}{\sum_{l \in I} h_R(d(j,l))} \tag{5}$$

Here b_j is the current actor under consideration, $D(j,l)$ defines the distance between two locations in a neighbourhood I in the dataset space, and $d(j,l)$ defines the distance between two locations in the output space O. $h_R(x)$ is a monotonically increasing function in the range $[0,1]$. The scent function λ therefore defines a weighted sum of the distances of neighbouring agents.

The polar swarm algorithm relies on the scent function to make sure that the connectivity present in the N-dimensional dataset is translated to the two-dimensional projection. Polar swarm considers radial movements of the agents. Using this knowledge, [12] defines the scent function given in equation 5 more precisely using a radial monotonically decreasing function $h_R(x)$ as given in equation 6.

$$h_R(R_e) = \begin{cases} 1 - \frac{r(j,l)^2}{\pi R_e^2} & \text{iff } \frac{r(j,l)^2}{\pi R_e^2} < 1 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

In equation 6, the distance R_e depends on the epoch we are currently in, and the function $r(j,l)$ is a distance measure in the output space O. By using equation 6, [12] defines the scent function used in polar swarm, which we reproduce in equation 7.

$$\lambda_e(b_j, R_e, S_0) = \begin{cases} S_0 - \frac{\sum_{l \in I} h_R(r(j,l)) \cdot D(j,l)}{\sum_{l \in I} h_R(r(j,l))} & \text{iff } \sum_{l \in W} h_R(r(j,l)) > 0 \\ S_0 & \text{otherwise} \end{cases} \tag{7}$$

In equation 7, we define S_0 as the initial scent value, as given in equation 8.

$$S_0 = \sum_j |\lambda(b_j, R_{max}, 0)| \tag{8}$$

Thanks to the polar swarm algorithm, the DBS framework can produce very informative visualizations. We most likely lose some information because we do not have a guarantee that the two-dimensional projection accurately represents the N-dimensional connectivity, which was proven in [2]. This is not inherently a conviction of the method, of course: every clustering algorithm needs to reduce the dataset to a lower-dimensional description, and will lose some accuracy compared to storing the entire dataset. This is the fundamental trade-off between learning and storage.

The DBS framework includes a built-in visualization that displays the clusters, as shown in figure 2. Figure 2 shows that the DBS visualization very clearly shows the clusters and their separation. The visualization represents the connectivity of the two-dimensional projection by using topographical analogies: mountain peaks denote large distances between data points (i.e. low density), lakes denote small distances between data points (i.e. high density). This visualization is useful for visually determining clusters in the dataset. Thanks to the topographical description that the visualization uses, we can easily discern outliers and figure out how many clusters are appropriate for the clustering we are going to attempt. The accuracy of this method depends, of course, on the accuracy of the polar swarm projection: if the projection is not accurate, then we cannot figure out accurate information from the visualization, either.

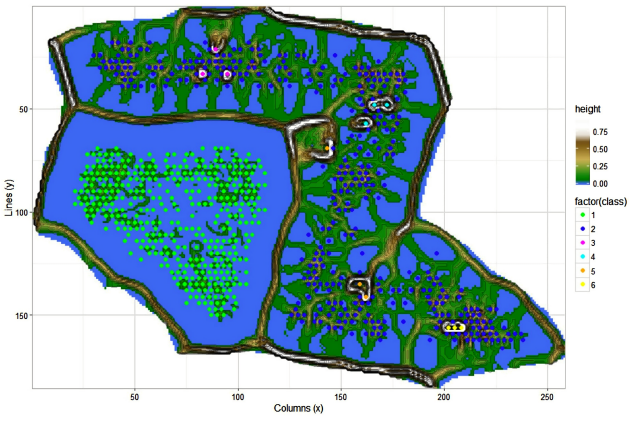


Fig. 2. Visualization of the projection produced by the Polar swarm algorithm in the DBS framework. This image was reproduced from [12].

The next step in the DBS framework is the actual clustering step. For this, we need to determine the number of clusters we want to find from the visualization we have just produced. Using this number of clusters, we can then apply a clustering algorithm. The clustering algorithm is applied on the two-dimensional projection, using the number of clusters we just indicated along with a choice of clustering method. DBS uses a shortest path calculation to evaluate a hierarchical clustering process which ultimately assigns the data points to a cluster. DBS includes two clustering approaches, based either on a compact or a connected representation of the clusters. The compact representation prevents the inclusion of data points that are too far out, but it can overestimate the number of clusters in the two-dimensional projection. The connected representation generally produces a lower number of clusters, but it can go over low-density regions of the visualization to find connections with other groups of data points. We need to make a choice based on the visualization which approach gives the best results for the given dataset.

2.3 Computational Centroids in PSO

As an improvement to the traditional PSO fitness evaluation, [10] propose using computational centroids, instead of centroids based on particle positions. Assigning centroids based on particle positions makes intuitive sense in PSO, because we want those particles to represent the clusters. However, [10] suggest not giving in to this assumption and instead using the computational centroids. The computational centroid is the familiar centroid that we can compute from the positions of the data points. At initialisation, we randomly initialise the centroid positions. Then, we compute the data points which are closest to the centroid, and recompute the centroid based the set of m_j data points $M_j \subseteq \mathbb{R}^N$ assigned to centroid $\mathbf{C}_j \in \mathbb{R}^N$. The new location of $\mathbf{C}_j$ at any point in the algorithm is computed using equation 9.

$$\mathbf{C}_j = \frac{1}{m_j} \sum_{\mathbf{x}_i \in M_j} \mathbf{x}_i \tag{9}$$

While intuitively we would expect that both approaches to determining the centroid would converge, [10] point out that this is only likely for non-overlapping clusters. For overlapping clusters, the methods are likely to yield different results.

In order to figure out whether the Function Evaluation with Computational Centroids modification is useful, [10] compare the FECC-modified versions of different Clustering Validity Indices (CVIs). These CVIs are measures for evaluating the accuracy of a given clustering. The PSO algorithm at some point requires these CVIs, and thus needs to compute the centroids of the dataset in question. As a result, we can directly compare the accuracy of the original CVIs with the FECC-modified version of the CVIs. The replacement is quite direct: we only need to modify the centroid computation procedure, so that it uses the computational centroids given by equation 9 instead of the particle positions of the swarm.

In their research, [10] find that the FECC-modified variants almost always perform better than the original versions. They explain this by noting that the computational centroids have a more natural tendency to center on the cluster than the centroids found by using the particle positions of the swarm, and therefore the clustering algorithms, which usually make use of radially symmetric distance measures, perform better when we use these computational centroids. For PSO in particular, the fitness evaluation function depends on the centroids, so the accuracy of these centroids is important in achieving a good criterion for validating the swarm behaviour.

FECC-modified versions have a tendency to overcluster, meaning that they find more clusters than the corresponding original versions do. This is not necessarily incorrect, of course. There are two options: firstly, the original versions might undercluster, which means that the larger number of clusters might bring the total number of clusters closer to the ground truth. Obviously, this is not always the case. However, [10] refer to the discussions by [5] and [14] when they say that the accuracy of a clustering algorithm amounts to more than just finding the correct number of clusters: an underclustered result can still have a better training and generalisation error if these clusters are more generally aligned with the ground truth clusters than a result with the same number of clusters as the ground truth. In short, we need to take into account not only the number of clusters, but also their accuracy.

3 Results & Discussion

In this section, we compare DBS with the other methods we described in section 2. In section 3.1, we compare DBS with PSO+GA in order to see how DBS compares and whether PSO+GA offers suggestions that could improve the DBS framework. In section 3.2, we look at the implications that the FECC modification has for the DBS clustering process.

3.1 DBS and DCPG

A fundamental difference between the DBS and DCPG approaches is that the DCPG algorithm aims to optimize an objective function, and therefore makes implicit assumptions about the underlying structure of the data [7, 12]. The DBS algorithm takes a different approach by omitting optimizing a global objective function, and relying on the concept of emergence instead.

The DCPG algorithm is characterized by its quick convergence, and despite its relative complexity it has short computation times and low computational costs. In comparison, the DBS algorithm suffers from one main limitation in its computation times. For example, the analysis of a dataset with more than 4000 cases would require more than a day of computation time [12]. This is due to the fact that every case needs its own agent, and the implementation of the polar swarm algorithm can only utilize a single core. The authors of the DBS algorithm also mention that if prior knowledge of the data set is available, a clustering algorithm with appropriately chosen parameter settings can outperform DBS.

The DCPG algorithm does come with some disadvantages however. [12] argue that the application of the k-means algorithm for clustering imposes a spherical cluster structure on the dataset, rendering it less practical for the detection of elongated clusters. Furthermore, the k-means algorithm is sensitive to noise and outliers. Besides these reasons, [12] also reason that it is difficult to determine a correct stopping criterion for the PSO algorithm, which is usually a fixed number of maximum iterations.

3.2 DBS and FECC

DBS can potentially benefit from the use of computational centroids. We explained in section 2.3 that FECC brings an improvement to the traditional PSO implications which rely on particle positions. This is due to the tendency of the particle positions to be more off-center for the cluster, which implies a lower compactness. The authors of [10] showed quite clearly that this optimisation had a positive impact on the classification performance of the PSO algorithms for which they tested the FECC modification. Although [12] do not refer to the centroids, they do rely on the same implicit assumption that [10] refer to, which is that the particle position is assumed to represent the centroid. We can see this in equation 6. The radial monotonic function centers on the current particle j under consideration, and the value of the scent function is dependent on this function from the definition in equation 7. The minus sign in equation 7 tells us, since we want to maximise the value of λ, that we want to minimise the fraction $\frac{\sum_{l \in I} h_R(r(j,l)) \cdot D(j,l)}{\sum_{l \in I} h_R(r(j,l))}$.

Before we can consider what a computational centroid could add to the polar swarm algorithm, we note that the centroid aims to position itself such that the average distance of the centroid to all associated data points is minimised. This means that we minimise $\langle D(j,C_j) \rangle$, where C_j is the centroid to which we associate data point j. In the projection space O, we have to replace the distance $r(j,l)$ with the distance $r(j,C_{j,O})$, where $C_{j,O}$ is the centroid computed for the set of agents in the region under consideration. By definition of the centroid, we minimise the average value $\langle r(j,C_{j,O}) \rangle$. As a consequence of minimising $\langle D(j,C_j) \rangle$ and $\langle r(j,C_{j,O}) \rangle$, we expect to have maximised

equation 7, although the product in the sum of the numerator indicates that we cannot assume this result to be true. If it is true, then we have found the strongest possible scent function value. Since the scent function represents the payoff function of the game-theoretical game, an attractive conclusion would be that we have produced a configuration that is maximally stable, but this assumption does not necessarily follow. Indeed, we only use the scent function to determine the best current configuration, but we consider the best possible configuration over a moveset, which includes a certain amount of randomness due to the random choice of movement that we take. As such, further research is necessary in order to test the hypothesis that we present above with respect to the FECC modification for DBS.

The results that [10] present for the use of computational centroids clearly show that using computational centroids can improve clustering performance, so there is good reason to study this effect for DBS. We explained in this section that it is possible to introduce computational centroids into DBS as well.

We must note that the visualisation aspect of the DBS framework does not impact the discussion we have given above. What sets DBS apart from the PSO algorithms that [10] studies is that DBS contains a semi-interactive visualisation which can be used to determine the number of clusters we expect the final result to have. However, this visualisation does not change the polar swarm algorithm that produces the projection. Indeed, the suggestion that we presented in this section proposes a modification to the polar swarm algorithm, and thus the semi-interactive nature of the visualisation does not ultimately affect this discussion.

4 Conclusion

The aim of this study was to assess the quality of the novel DBS framework proposed by [12], and present ways in which the underlying algorithm can be improved. To this end, we compared the DBS algorithm to other state-of-the-art clustering techniques. Literature study was performed to find suitable algorithms for comparison.

Frequently used hybridization techniques in clustering algorithms include PSO, GA and k-means [4]. The dynamic clustering technique DCPG presented by [7] utilizes all of these techniques, and is able to automatically determine the optimal number of clusters. By combining the PSO, GA and k-means algorithms, it offers fast convergence and better clustering results than by using PSO or GA-based algorithms alone [7]. We have highlighted the inner workings of the DCPG algorithm, and compared the algorithm to the DBS framework. In contrast to the DCPG algorithm, the DBS framework is a good candidate for data sets where no prior knowledge is available. For known data sets however, the DCPG algorithm with appropriately specified parameters could potentially outperform DBS. The choice between DCPG and DBS is therefore application-specific and mostly depends on whether knowledge of the data set is available. Further research is necessary to establish clustering results for both algorithms and provide a clear insight in their clustering accuracy. The Iris, Wine, Glass and Vowel data sets mentioned in [7] are commonly used benchmark data sets, and could be well-suited for this research project.

In an attempt to improve the accuracy the PSO clustering algorithm, [10] propose a technique that uses computational centroids instead of using the particle positions directly. The authors find that FECC-based algorithms almost always outperform the original versions, even though the FECC algorithm tends to overcluster in comparison. This suggests that not only the determined number of clusters is important, but also the accuracy of the found clusters. For this reason, the implementation of computational centroids in the DBS algorithm could possibly improve its performance.

At the present time, the main limitation of the DBS framework is its computation time. As the framework is currently only capable of using a single core, the DBS frameworks' performance might significantly increase from a multi-threaded implementation of the polar-swarm algorithm.

References

[1] Eric Bonabeau et al. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, New York, 1999.

[2] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures and Algorithms*, 22:60–65, November 2002.

[3] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. IEEE. URL: http://ieeexplore.ieee.org/document/494215/, doi:10.1109/MHS.1995.494215.

[4] Elliackin Figueiredo, Mariana Macedo, Hugo Valadares Siqueira, Clodomir J. Santana, Anu Gokhale, and Carmelo J.A. Bastos-Filho. Swarm intelligence for clustering — a systematic review with new perspectives on data mining. 82:313–329. URL: https://linkinghub.elsevier.com/retrieve/pii/S0952197619300922, doi:10.1016/j.engappai.2019.04.007.

[5] Ibai Gurrutxaga et al. Towards a standard methodology to evaluate internal cluster validity indices. *Pattern Recognition Letters*, 32(3):505–515, February 2011.

[6] C.-F. Juang. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. 34(2):997–1006. URL: http://ieeexplore.ieee.org/document/1275532/, doi:10.1109/TSMCB.2003.818557.

[7] R.J. Kuo, Y.J. Syu, Zhen-Yao Chen, and F.C. Tien. Integration of particle swarm optimization and genetic algorithm for dynamic clustering. 195:124–140. URL: https://linkinghub.elsevier.com/retrieve/pii/S0020025512000400, doi:10.1016/j.ins.2012.01.021.

[8] David Martens et al. Editorial survey: swarm intelligence for data mining. *Machine Learning*, 82:1–42, September 2010.

[9] Mahamed G. H. Omran, Ayed Salman, and Andries P. Engelbrecht. Dynamic clustering using particle swarm optimization with application in image segmentation. 8(4):332–344. URL: http://link.springer.com/10.1007/s10044-005-0015-5, doi:10.1007/s10044-005-0015-5.

[10] Jenni Raitoharju et al. Particle swarm clustering fitness evaluation with computational centroids. *Swarm and Evolutionary Computation*, 34:103–118, February 2017.

[11] Axel Thevenot. Particle swarm optimization visually explained. URL: https://towardsdatascience.com/particle-swarm-optimization-visually-explained-46289eeb2e14.

[12] Michael C. Thrun and Alfred Ultsch. Swarm intelligence for self-organised clustering. *Artificial Intelligence*, 290(1):103237, January 2020.

[13] Alfred Ultsch. Clustering with databots. In *Int. Conf. Advances in Intelligent Systems Theory and Applications*, 2000.

[14] Lucas Vendramin et al. Relative clustering validity criteria: A comparative overview. *Statistical Analysis and Data Mining*, 3(4):209–235, July 2010.

A Review of Image Vectorisation Techniques

Ştefan Evanghelides (s2895323), Ethan Waterink (s3417611)

Abstract— Most available and captured images are raster (bitmap) images, which are, by definition, resolution dependent. In this paper, we look at three image vectorisation methods available today for converting a raster image to a vector one. First, we look at optimized gradient meshes, which semi-automatically create gradient meshes from raster images by minimizing an energy function. Second, we consider diffusion curves, which partition the space through which it is drawn, defining different colors on either side. These colours are then diffused over the image. Third, we look at thin-plate splines, which create a hybrid vector structure, using parametric patches and detailed features for localized and parallelized thin-plate spline interpolation. Finally, we quantitatively and qualitatively compare the three methods, deriving strengths and weaknesses with respect to vectorisation complexity, reconstruction accuracy, image editing and animation.

Index Terms—Image vectorisation, optimized gradient mesh, diffusion curve, thin-plate spline, editing

1 Introduction

Image vectorisation is the process of converting a raster (bitmap) image into a vector image. Raster images typically consist of a rectangular grid of pixels, holding the colours. This means that raster images are dependent on the scale they were created in. If such an image is zoomed in, then blocky artifacts appear for round/shaded objects.

Vector images, on the other hand, are 1) scale independent, meaning they remain sharp when zoomed in. Their inner structure consists of a set of shapes, which determine the resulting pixel values usually by some interpolation technique. Such an image is 2), most of the time, more efficient in terms of space than the bitmap images. These are two of the reasons why it is useful to vectorise a raster image. The process of vectorising an image has been greatly improved over the years. While image designers are able to manually vectorise an image, nowadays it is possible to achieve near-perfect results using automatic methods.

The aim of this paper is to provide a review analysis of the current techniques used to automatically vectorise bitmap images. This paper gives a guideline in the field of image vectorisation that could possibly assist professionals and artists in choosing the most suitable approach for image vectorisation on a case-by-case basis. While a wide collection of techniques are available, we focus on analysing the following methods:

1. Image vectorisation based on optimized gradient meshes (OGM) [21],
2. Image vectorisation based on diffusion curves (DC) [16],
3. Image vectorisation using real-time thin-plate splines (TPS) [3].

These three methods represent different approaches, with different primitives used in image vectorisation, namely mesh-based, curve-based and patch-based vectorisation methods, respectively. As the names suggest, the first method aligns meshes with edges encoding colour information with interpolation inside each primitive for rendering. The second method uses curves and lines as colour constraints to ensure smooth colouring and rasterization. Lastly, the third method encodes colour and geometric information in parametric patches to facilitate editing and flexibility.

During our analysis, the proposed methods are considered quantitatively, by judging their performances based on metrics such as accuracy, efficiency and storage, as well as qualitatively, by analysing their performances and limitations in the context of complexity and editability. The implementation and the results achieved by the methods cannot be judged independently of their experimental setup, therefore the scope and limitations of the conducted experiments must also be considered.

This paper is structured as follows: Section 2 addresses related work on image vectorisation. Section 3 describes the preliminary information of the image vectorisation techniques. Then, Section 4 provides an overview of the state-of-the-art techniques. Section 5 presents a detailed comparison and discussion of the presented approaches. Finally, Section 6 offers the conclusions of our investigation, and Section 7 suggests possible extensions for the research conducted in this paper.

- *Ştefan Evanghelides is an MSc Computing Science student of the University of Groningen, E-mail: s.evanghelides@student.rug.nl.*
- *Ethan Waterink is an MSc Computing Science student of the University of Groningen, E-mail: e.waterink@student.rug.nl.*

2 Related Work

Numerous vector image (or vector graphic) representations and vectorisation techniques have been developed over the last 60 year, see e.g. [7, 18, 17]. In the field of image vectorisation, fundamental methods are the optimized gradient mesh [21] and the diffusion curves [16], making them representative methods in our analysis. These served as a basis for more advanced extensions. Barendrecht et al. [1] extended on the gradient mesh primitive by introducing a method for local refinement, while Lieng et al. [13] and Svergja et al. [23] created gradient meshes of arbitrary manifold topology. Li et al. [12] proposed temporal diffusion curves, which represent not only the graphics but also their evolution over time using continuous functions. For other extensions we refer to [25, 14, 22].

Considering the publishing dates of [21, 16], we included the contemporary real-time thin-plate splines method [3]. Lai et al. [9] began with an initial mesh created manually by artists, whereas Xia et al. [24] attempted to automate the vectorisation process using triangular meshes. However, both methods result in a large number of patches, decreasing the performance, as opposed to TPS.

3 Preliminaries

Vectorising binary (black and white) images can be done by extracting the edges, which can then be represented using existing primitives, such as curves [5]. This can be realized by basic thresholding to separate the objects. Vectorising coloured images poses certain challenges, most notable being the combination between sharp and smooth colour transitions [16] and diffusion constraints [2]. In this section, the main components of the vectorisation methods are generally defined, namely Bézier splines, Ferguson patches, gradient meshes, diffusion curves and thin-plate splines.

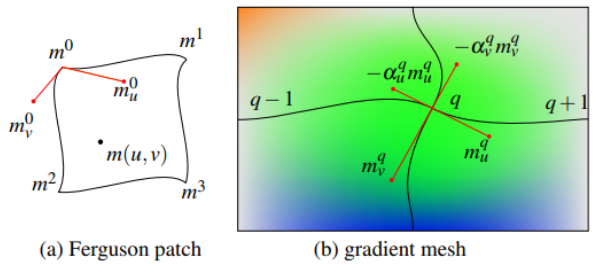
Bézier splines

Understanding gradient meshes and diffusion curves relies on understanding Bézier splines. A spline is a special function defined piece-wise by polynomials. When each polynomial piece of the spline is represented by Bernstein polynomials we speak of Bézier splines. A Bézier curve is a parametric curve, defined by the interpolation of a set of control points. The first and final control

points are always the endpoints of the curve; however, the intermediate control points (if any) generally do not lie on the curve. Bézier curves can be combined to form Bézier splines. Hence, they are piece-wise Bézier curves that are at least continuous [20].

Gradient meshes A gradient mesh is a topological lattice containing vertices grouped in quad primitives. The vertices have colours associated to them, which are interpolated across the mesh. Vertices are connected using 4 mesh-lines, which are defined as Bézier splines. The user therefore can control the vertex positions, the splines derivatives at each vertex and the RGB colour [21].

Ferguson patches Gradient meshes can be regarded as topologically planar rectangular Ferguson patches with mesh-lines. A Ferguson patch consists of a grid of bicubic-interpolated control points [6]. Figure 1 shows a simple mesh consisting of 4 Ferguson patches, along with the mesh-lines.

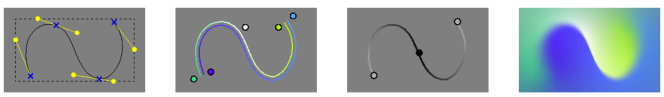


(a) Ferguson patch (b) gradient mesh

Fig. 1. The gradient mesh in (b) consists of 4 Ferguson patches (a). Image taken from [21].

Diffusion curves Diffusion curves are cubic Bézier splines which contain colours on both sides of the curve. Besides the control points of the curve, the curve contains a set of *blur control points*, which allow to control the smoothness of the transition of colours between the two halves [16].

Rendering images with diffusion curves is a 3-step process: first, the curves are defined, along with the colours on each side of the curves. Secondly, the colours are diffused over the image. Thirdly, the image is *reblurred* using the blurring attributes. Figure 2 illustrates the three components of a diffusion curve (a)–(c) and the final result is shown in (d).



(a) (b) (c) (d)

Fig. 2. Diffusion curve components: (a) geometric curve, (b) colours on either side, linearly interpolated along the curve, (c) blur amount linearly interpolated along the curve; (d) final image, obtained by diffusion and reblurring. Image taken from [16].

Thin-plate splines Thin-plate splines are spline-based techniques for data interpolation and smoothing, and are an important special case of a polyharmonic spline. The TPS interpolates a surface that passes through each control point. The fit resists bending, implying a penalty involving the smoothness of the fitted surface. The TPS arises from consideration of the integral of the square of the second derivative – this forms its smoothness measure [8].

4 Image Vectorisation Techniques

This section presents the three main image vectorisation techniques: optimized gradient meshes, diffusion curves and thin-plate splines. Improved diffusion curves models are also described.

4.1 Optimized gradient meshes

Optimized gradient meshes semi-automatically create gradient meshes from a raster image [21]. By formulating an energy minimization problem, an optimized gradient mesh is obtained. This energy term $E(M)$ is the reconstruction residual between the input image and the colour graphics rendered by the gradient mesh. Given a smooth raster image, the aim is to create a gradient mesh with a small reconstruction error. In this case, the gradient mesh is a set of connected Ferguson patches (Figure 1) which is fitted to the image.

4.1.1 Optimization

In fact, minimizing the energy function $E(M)$ is a non-linear least squares (NLLS) problem. To solve this, the Levenberg-Marquadt (LM) algorithm [10] is employed, as it uses an effective damping strategy that lends it the ability to converge promptly from a wide range of initial guesses.

Figure 3 shows an example of an optimized gradient mesh. First, a manually created gradient mesh in Adobe Illustrator is rasterized to an image as the input, see Figure 3(a). The mesh is initialized by a 4×4 evenly divided mesh. Then, Figure 3(b) is the optimized gradient mesh after 40 LM iterations. The optimized mesh-lines are similar to the manually created mesh-lines, and the rendering results are virtually identical. Notice that both smooth regions and sharp edges are faithfully reconstructed.

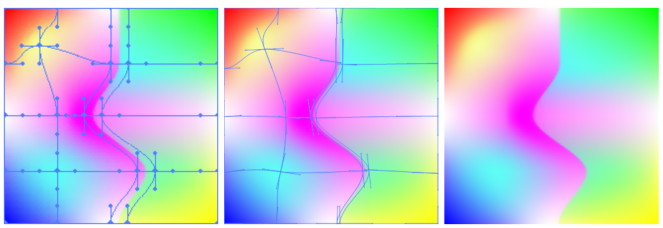


Fig. 3. Optimization. (a) A screen snapshot of a gradient mesh in Adobe Illustrator, which is rasterized as the input image. (b) and (c) Optimized gradient mesh. Image taken from [21].

In order to obtain more satisfactory results, Sun et al. (2007) enforce two constraints on the optimized gradient mesh [21].

Smoothness constraint The simple input image in Figure 3 was reconstructed faithfully. However, for complex examples or real images with noise, the optimization of energy function $E(M)$ often becomes stuck in a local minimum. As the resulting gradient mesh is not smooth, a smoothness term SC (smoothness constraint) is added into the energy, yielding the new energy term $E'(M) = E(M) + SC$. The new energy term is the smoothness of the gradient mesh. Smaller reconstruction errors are obtained by arriving at a better local minimum; see Figure 4.

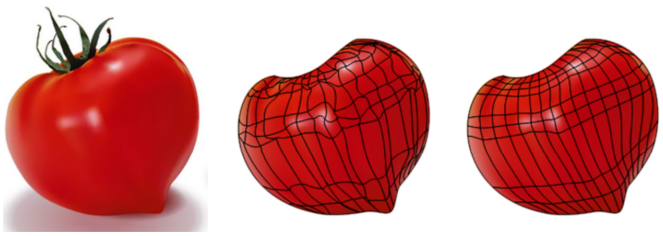


Fig. 4. Smoothness constraint. Input image and optimized results without and with the smoothness constraint. Image taken from [21].

Boundary constraint The boundary of a gradient mesh consists of four segments, see Figure 1, and each segment is one or more cubic Bézier splines. In the optimization, a boundary constraint is enforced in which the control points on the boundary only move along the splines.

4.1.2 Guided optimized gradient mesh

In most cases, the optimization can create satisfactory results automatically. To give the user more control, a few vector lines can be drawn in the image to control the mesh generation (i.e. the dominant direction of the mesh lines). This is formalized by adding a new term V to the energy function, to represent the user's input. Consequently, the new energy function $E''(M) = E'(M) + V$ is minimized. The new energy term encourages consistency between optimized mesh-lines and specified vector lines [21].

4.1.3 Mesh initialization

A complex object usually consists of several semantic parts. The input image object is first decomposed into several sub-objects using an interactive image cutout tool [11] or a free lasso tool. Then, the boundary of each sub-object is manually divided into four segments. In order to obtain a good result, it is better to have a division so that the segments follow the major and minor axes of the object. Each segment is then fitted by one or more cubic Bézier splines. Finally, the mesh-lines are initialized in two ways: evenly distributed or by interactive placement.

Each sub-object with four segments is treated as a Coons patch [4], which supports multiple splines on one segment. To create an evenly distributed mesh, the Coons patch is divided evenly in the parametric coordinates.

4.2 Diffusion curves

There are three ways to create a diffusion curve: manual, assisted and automatic. The artist can manually create diffusion lines from scratch, where she/he sketches the lines of the drawing and sets the colours. In most cases, however, the artist has an existing image as a starting point, which she/he manually traces and recovers the underlying colours from (assisted approach). The third approach automatically extracts and vectorises diffusion curve data from a bitmap [16].

The manual and assisted creation of the diffusion curves rely solely on the artist's skill to recreate the raster image. For the purpose of this paper, the automatic creation approach is much more relevant. This process involves two steps: data extraction and conversion to diffusion curves.

Data extraction Edges are extracted based on Orzan et al. (2007) [15]. The main point is to determine sharp changes of colours by blurring the image at a high scale (the higher the scale, the more blurred the edge), followed by determining the edges. The scale at which the edge is best represented will be used to identify the degree of the blur needed for the respective edge in the reconstructed image. In the same paper, Orzan et al. (2007) also designed a method to determine the edge's lifetime. This lifetime is used to adjust the preservation of details.

Conversion to diffusion curves Inspiration from Selinger (2003) [19] is taken for vectorisation of positions. This method generates smooth connected Bézier curves that approximate the appropriate pixel chains, where the user specifies the fitting error and the degree of smoothness. Several parameters influence the quality and complexity of the resulting image: for edge geometry: threshold for number of edges, despeckling parameter, smoothness and fitting error; for blur and colour: size of neighbourhood for eliminating outliers and the maximum error accepted when fitting the polyline. The paper also proposed values for each parameter. An example of the diffusion curve conversion is shown in Figure 5.

Fig. 5. Example of DC reconstruction: (a) original image; (b) result after conversion into DC representation; (c) automatically extracted diffusion curves; (d) RGB difference between original and reconstructed image (amplified by 4). Image taken from [16].

The article of Orzan et al. (2008) [16] served as a ground base for further development of the diffusion curves as vector primitives. These include: diffusion constraints [2], diffusion coefficients [14], inverse diffusion curves using shape optimization [25] and, finally, temporal diffusion curves [12]. We briefly describe diffusion constraints and coefficients.

4.2.1 Diffusion constraints

Diffusion constraints came as a theoretical extension to diffusion curves. As stated by Bezerra et al. (2010), the original diffusion curves do not allow the control of the diffusion in images and that the colour changes must be defined across the whole curves. When the diffusion constraints are taken into account, the user is able to control the strength and direction of the diffusion, by means of solving a linear system [2].

4.2.2 Diffusion coefficients

Lin et al. (2018) proposed an update of the diffusion curves primitives to take into account the diffusion constraints (strength and direction, described in Section 4.2.1) and diffusion points [22]. Such points are useful, for instance, in images with stars on a sky.

This model is based on a diffusion equation with coefficients to produce a vector image. It was therefore named "Diffusion equation with coefficients" (DCC). This model consists of two types of layers: colour layers and coefficients layers. The colour layer is the colour source curve from Orzan et al. (2008), but discretised into pixels. The coefficients layer is further divided in two: *strength coefficient layer* and *direction coefficient layer* (as described in Section 4.2.1). Typically, there is at least one colour layer, and each colour layer is accompanied by at least one strength layer and one direction layer.

The combination of these two layers allows for the following: feature preservation, e.g. a flower's yellow stamens; emissivity, e.g. increasing the strength layer on the contour of a moon object will make the moon glow; diffusion direction: e.g. a rainbow, created only using the initial line strip of colours and modelled/arched using the diffusion direction. Diffusion direction can also be used to produce and control shadows.

4.3 Real-time thin-plate splines

According to Chen et al. (2020) real-time thin-plate splines *encode global manipulation geometries and local image details within a hybrid vector structure, using parametric patches and detailed features for localized and parallelized thin-plate spline interpolation* [3]. Thin-plate splines provide direct control over derivative interpolation and help to maintain their smoothness, in particular smooth local extremes.

Parametric patches (see Figure 6(a)) represent object components to facilitate editing. Colour details are encoded as features (see Figure 6(b)) to achieve faithful rasterization using TPS interpolation. Real-time patch-wise TPS inversion and interpolation allows for several vector image editing operations, including image magnification, colour editing, and cross mapping (see Figures 6(c)–(e)).

Fig. 6. Illustration of the image vectorisation process. (a) Raster image encoded by object segments as parametric patches marked by green curves. (b) Detailed features as colour constraints. (c) Rasterization employs a biharmonic interpolation of detailed features for scalability and compactness. (d) Colour editing. (e) Cross mapping. Image taken from [3].

In short, the real-time thin-plate splines image vectorisation consists of two parts:

- A **novel hybrid vector representation** of detailed colour features embedded in parametric patches for localized GPU TPS

rasterization to enhance compressibility and scalability, while enabling interactive editing in real time.

- An **optimal feature selection scheme** using gradient intensity histogram of an image to balance the number of features and the reconstruction error based on a compression efficiency metric defined by Chen et al. (2020).

The proposed algorithm maintains scalability and editability by vectorising a photorealistic image and its corresponding labeling map to create a hybrid representation comprising of parametric patches and detailed colour features. The vectorisation process involves parametric patch construction, detail feature extraction, TPS inversion, and rasterization [3].

Figure 7 presents the vectorisation and rendering pipeline according to Chen et al. (2020). First, the user provides a raster image and labeling map of interesting object segments. The system then constructs Hermite patches from these segments. After computing the gradient distribution histogram, image characteristics are analyzed to select an initial set of detailed colour features using adaptive superpixel and Canny operators. The system embeds the extracted features into the Hermite patches and clusters them into localized groups for evaluation. It then packs these groups with neighboring features and applies TPS interpolation to compute the colours of the pixels in the group. Finally, the rasterization regions are extended to provide suitable overlap. A weighted average is applied to the overlapping groups to remove seams, i.e. improve continuity. This system enables editing in real time by adjusting the colour and location of features and then repeating the rasterization process in order to generate results with minimal distortion [3].

5 Comparison and Discussion

The different methods presented in Section 4 have their advantages and disadvantages. In order to give an overview of the cases for which each method is most suitable, the results obtained by each approach are considered and discussed from a quantitative and qualitative point of view. A direct comparison between the methods is often difficult, as much depends on the chosen image content, as well as the application of the vectorised image.

5.1 Quantitative comparison

First, we quantitatively compare the presented image vectorisation techniques with respect to their reconstruction accuracy, (re)construction efficiency and storage requirements. We use the pepper from Figure 8 for the quantitative comparison. Note that there is no reconstructed pepper for TPS, as it was not directly available, but only mentioned in tables.

5.1.1 Accuracy

The vectorisation methods aim at reconstruction the input raster image with minimal reconstruction error. Hence, we focus on their accuracy in terms of reconstruction error.

OGM specifies the reconstruction error in terms of error per pixel, which is defined as the average of the differences between values of pixels in the images. The reconstruction errors of the experimental results were all below 1.0/pixel, so likewise for the pepper in Figure 8(a,b), the tomato in Figure 4(c) and the mesh in Figure 3.

DC does not report any quantitative reconstruction errors. While the reconstruction error is low, the most visible error occurs along edges, most probably because, through vectorisation, their localization is changed; see Figure 5(d). The main inaccuracy of DC comes from the sharp changes in colours or from images rich in features. We note that the DC-reconstructed pepper in Figure 8(c,d) is missing certain light reflections when compared to OGM. Emmissivity is therefore lacking. This is, however, solved in the diffusion curves with coefficients, as proposed by Lin et al. (2018), making DC images much more accurate than OGM [14].

TPS reports a low mean error of $1.06 \cdot 10^{-5}$ (no scale mentioned) for the pepper image. On top of that, the representation allows for a high compression ratio of 62% for the same pepper image.

5.1.2 Efficiency

Next, we consider the efficiency of the vectorisation methods with respect to their (re)construction time.

The vectorisation of the pepper image in Figure 8 took 7.3 minutes using the optimized gradient mesh, according to the Sun et al. (2007). This is the total duration of three phases: 3 minutes for the boundary initialization, 1.5 minutes for the mesh initialization and 2.8 minutes for the optimization.

The most time-consuming operation of the DC method is the colour diffusion. DC does not report any quantitative timings. However, as the diffusion is implemented on the GPU, real-time performance can be achieved. When using the diffusion curves with coefficients, the pepper in Figure 8 took 16.72 seconds to generate, according to the Lin et al. (2018).

The thin-plate spline method renders the images in real-time. The TPS kernels are rendered in the CPU, which are then sent to the GPU for inversion [3]. This means that the speed of rendering is dependent on the performance of CPU and GPU, as well as the bus speed between the CPU and GPU.

5.1.3 Storage

Lastly, we consider the storage requirements of the vectorised image, e.g. in terms of number of lines and patches, or bytes.

The storage space of the optimized gradient meshes is influenced by the number of patches in the mesh(es). The OGM pepper image in Figure 8 contains three meshes with 276 patches in total. There is, however, no mention on the actual space used in this case. The optimized gradient mesh of a yolk image (see [21]) consists of 270 patches, which has a size of 7.7KB. It can be safely inferred that the pepper image would take roughly the same amount of storage space, as it has about the same amount of patches.

The storage space for diffusion curves is influenced by the number of curves, geometric control points, left and right colour control points and blur control points. As the storage method may appear sparse, it is important to note that each geometric curve can hold an arbitrary number of colour and blur control points. This means, for an image rich in features, the resulting storage space may be relatively high. The DC pepper in Figure 8 contains 38 diffusion curves, with 365 geometric, 176 left-colour, and 156 right-colour control points.

The storage space for thin-plate splines depends on three aspects. First, TPS records a feature using $(x, y, R, G, B) : 2 \times 2 + 3 = 7bytes$. Second, a parametric patch requires four corner points, and the two control parameters' derivatives for $4 \times (2+2+2+2) = 32bytes$. The third contribution comes from the labeling map. The pepper image from Figure 8 uses 0.66 Bits-per-pixel and the resolution is 750×800, which is $600,000$ pixels. From this we infer that the pepper image would have a resulting size of approximately 396KB.

We note that the storage requirements, naturally, change with the level of added detail, i.e. the number of primitives used.

5.2 Qualitative comparison

Next, we qualitatively compare the presented image vectorisation techniques with respect to the complexity of the vectorisation process and the ease of post-editing. On top of that, we consider some other aspects of the vectorisation methods.

5.2.1 Complexity

First, we consider the complexity of the methods with respect to the vectorisation process. To determine this, we look at the complexity of their components.

The OGM aims at optimizing an energy function, which is an NLLS problem. This is solved by the LM algorithm, which requires the computation of the block-sparse Jacobian matrix. In order to avoid local minima and make the solver robust, a Gaussian pyramid is built from the input image and applied a coarse-to-fine optimization for LM.

The DC starts with data extraction using the Gaussian scale space. It then extracts edges at all available scales using a classical Canny detector. Lastly, it converts polylines to curves by performing classical least square Bézier fitting based on a maximum user-specified fitting

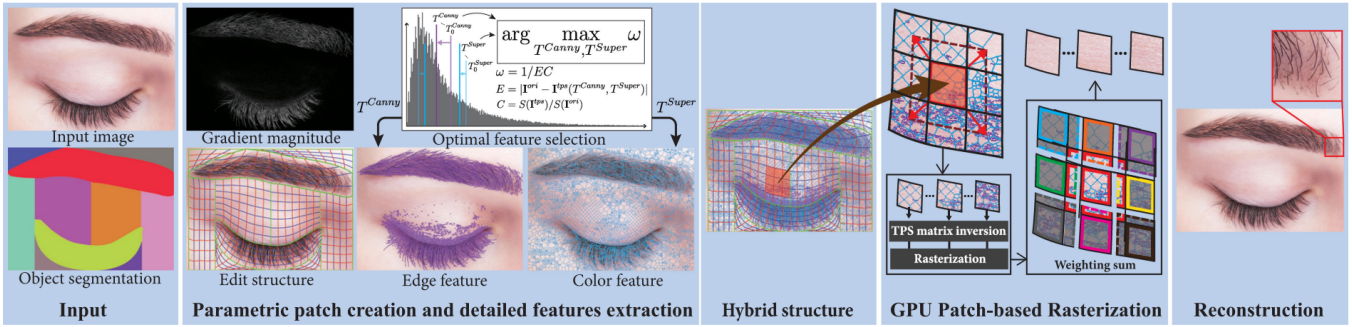


Fig. 7. Input raster image and its corresponding labeling map of object segments. The vectorisation pipeline involves parametric patch construction, optimal colour feature extraction, patch-based feature grouping, TPS kernel construction, and rasterization using GPU-based TPS colour interpolation. The red rectangle presents the eyebrow under 4X magnification. Image taken from [3]

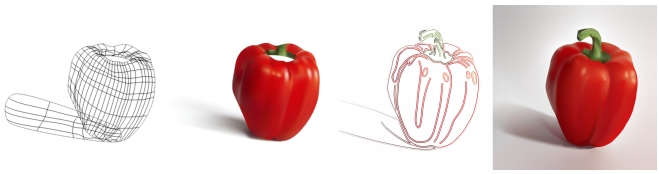


Fig. 8. Pepper reconstruction. (a,b) Optimized gradient Mesh. (c,d) Diffusion curves. Images taken from [21] and [16].

error and degree of smoothness. However, the DC model is more intuitive and easier to create, when compared against OGM, as noted by Orzan et al. (2008).

The most complex method is TPS. As described in Section 4.3 the TPS process consists of several steps, namely the parametric patch construction and detailed feature extraction, and creating the hybrid structure.

5.2.2 Editability

Next, we consider the editability of the resulting vector images. We define "editability" as the ease and flexibility of editing vector images.

The OGM is considerably simpler than other mesh-based image vectorisation methods. This simplicity allows users to more easily edit the mesh. For each control point in the mesh, three types of variables can be interactively edited: position, derivatives, and RGB colour. Sharp edges within an image object can be preserved by placing two closely-spaced mesh lines on either side of the edge. For each mesh point, its derivatives are manipulated by dragging four direction handles. Each mesh point's direction handles and paths define how this point's colour blends with other colours from other mesh points.

Gradient meshes produced by the OGM technique have several advantages: 1) Efficiency of use; the optimized gradient mesh makes it much faster for users to create gradient meshes from an input image. 2) Easy to edit; compared with other vectorisation tools, the optimized gradient mesh can produce a simpler mesh that the user can further edit and animate. 3) Scalability; The gradient meshes can be scaled in size with fewer artifacts. 4) Compact representation; The gradient mesh is an efficient representation for image objects with smooth transitions [21].

The DC provides great control and flexibility, as it allows any degree of control on a curve, without a topologically-imposed upper or lower bound on the number of control points. As artists commonly use strokes to sketch boundaries in an image, DCs are a more natural drawing tool than gradient meshes. Diffusion curves further allow an artist to evolve an artwork gradually and naturally. Gradient meshes, on the other hand, require careful planning and sound knowledge of the final composition of the intended art piece. Most gradient mesh images are a complex mixture of several individual gradient meshes, often overlapping. All these decisions have to be made before the relevant image content can be created and visualized.

In certain instances, the topology constraints of gradient meshes can be rather advantageous, for example when moving a gradient mesh to a different part of an image, or when warping the entire mesh. Such manipulations are also possible in the DC representation, but not as straightforward. For moving part of an image, the relevant edges have to be selected and moved as a unit. More importantly, without support for layering and transparency it is difficult to determine how the colours of outer edges should interact with their new surroundings. A mesh warp could be implemented as a space warp around a group of edges [16].

The TPS' hybrid vector representation uses efficient patch-wise TPS-based inversion and interpolation, which is ideally suited to editing in real time. It gives the flexibility required for image magnification, colour editing, and cross mapping with low reconstruction error in an intuitive manner. The ability to edit images directly in the vector space without for requiring intermediate raster representation and vectorisation would be beneficial to artists.

TPS enables natural magnification by directly scaling the parametric coordinate of all pixels, based on a given magnification ratio and rasterizing them based on these coordinates using the original TPS kernels. It also enables direct application of colouring operations to gradation and curvilinear features inside the desired region for the modification of appearance without altering the curvilinear features across the boundary, thereby maintaining important border characteristics. Lastly, it provides intuitive high-level object-based shape manipulation rather than low-level feature-based manipulation [3].

5.2.3 Other aspects

Finally, we consider some other aspects of the presented image vectorisation techniques.

First, we note some limitations of the three methods. For OGM, a simple gradient mesh is insufficient to capture the fine image details and highly textured regions. Another difficult case is when the boundaries of the object are too complicated, or the object has complicated topologies, very thin structures, or many small holes. The original DC system is single layered, but multiple, independent layers (i.e. DCC model [14]) offer more flexibility to artists. For TPS, although scatter data interpolation permits localized acceleration and manipulation while rasterizing with a very large magnitude of magnification, these point constraints become sparse and are prone to aliasing and blurring artifacts.

The vector representation can be used to create images in different styles. OGM, DC and TPS can best be used for designing photo-realistic images, depending on the level of details. TPS has the added benefit that it can realize fine detail. DC is best fitted for more cartoon-like images, although OMG and TPS can be used for this purpose as well. The DCC model has the flexibility needed for finer details, but that will result in more curves and potentially more time spent by the user for finer details.

OGM and DC allow for keyframe animation, enabling the user to easily create animations with the vector images. One advantage of the TPS method is that besides images, it also allows for video editing. A proposed improvement is adding a temporal parameter, which can be used to give a lifetime of an edited feature across multiple frames in video. Such temporal parameter would then make TPS a powerful vectorisation technique for videos.

6 Conclusion

In this paper, we have presented three image vectorisation methods: optimized gradient meshes, diffusion curves and thin-plate splines.

OGM allows artists to automatically vectorise a raster image by means of a gradient mesh. However, depending on the number of features, the resulting image may not be entirely accurate. Nonetheless, gradient meshes can be manually edited until the required level of detail is reached.

DC is more natural and provides, most often, more accurate results than the OGM method. The base DC model lacks certain properties, such as specularity, due to the reblurring at the end. The improved DCC model, however, fixes these problems and gives the users more flexibility: the extra layers allow, for instance, to control the emissivity of a shining moon and stars in a sky image.

TPS is better suited for direct editing of images, as changes can be generated in real-time by avoiding the intermediate raster representation. Objects can be manipulated directly with ease, which is highly beneficial for users.

All in all, our quantitative comparison showed that TPS outperformed OGM and DC on two metrics, achieving the highest accuracy and efficiency. The OGM, however, has the lowest storage requirements. Our qualitative comparison showed that DC has the least complex vectorisation process, followed by OGM and TPS. All three methods can be used successfully for editing images. We note that DC is more intuitive and thus easier to use for image creation, and that TPS can realize fine detail for photo-realistic images. Lastly, for creating keyframe animations, OGM and DC could be used, while TPS could be used for editing features and textures across frames.

7 Future Work

Although the aim of this paper was to provide a complete overview of the current image vectorisation methods, there is room for improvement. First, in this paper, we considered the main vectorisation methods: optimized gradient meshes, diffusion curves (and its variants) and thin-plate splines. The research carried out in this area is vast and we could not capture it in its entirety. It is therefore possible that certain (novel) methods were simply overlooked. A potential improvement of this paper is further analysis of other (possibly better) methods.

Secondly, this paper was also intended to provide a general overview of the advantages and disadvantages of the three methods in various contexts. These contexts, although broad, are limited. A more scoped, in-depth review could yield a better analysis with respect to certain domains, such as photo and video editing. One example of another application is storing vector images in `.pdf` format. Since most vector images are stored in the `.svg` format, it would be worth determining the compatibility of these methods between `.svg` and `.pdf` formats.

Lastly, given that each vectorisation method has advantages and disadvantages, a potential improvement would be to automatically determine the best vectorisation method in terms of accuracy or other criteria.

Acknowledgements

The authors wish to thank our expert reviewer Prof. J. Kosinka for his valuable feedback, and Prof. R. Smedinga and Prof. M. Biehl for giving us the required guidance in writing this paper.

References

[1] P. Barendrecht, M. Luinstra, J. Hogervorst, and J. Kosinka. Locally refinable gradient meshes supporting branching and sharp colour transitions. *The Visual Computing*, 34(6):949–960, 2018.

[2] H. Bezerra, E. Eisemann, D. DeCarlo, and J. Thollot. Diffusion constraints for vector graphics. *NPAR 10*, pages 35–42, 2010.

[3] K. Chen, Y. Luo, Y. Lai, Y. Chen, C. Yao, H. Chu, and T. Lee. Image vectorization with real-time thin-plate spline. *IEEE Transactions on Multimedia*, 22(1):15–29, 2020.

[4] S. A. Coons. Surfaces for computer-aided design of space form. Technical report, Massachusetts Institute of Technology, 201 Vassar Street, W59-200 Cambridge, MAUnited States, 06 1967.

[5] K.-C. Fan, D.-F. Chen, and M.-G. Wen. A new vectorization-based approach to the skeletonization of binary images. *ICDAR*, 2:627–632, 1995.

[6] J. Feguson. Multivariable curve interpolation. *Journal of the ACM*, 11(2):221–228, 1964.

[7] R. Grimsdale, F. Summer, C. Tunis, and T. Kilburn. A system for the automatic recognition of patterns. *Proc. IEE*, pages 210–221, 1959.

[8] W. Keller and A. Borkowski. Thin plate spline interpolation. *Journal of Geodesy*, 93:1251–1269, 2019.

[9] Y.-K. Lai, S.-M. Hu, and R. R. Martin. Automatic and topologypreserving gradientmesh generation for image vectorization. *ACM Trans Graph*, 28(3):85:1–85:8, 2009.

[10] K. Levenberg. A method for the solution of certain problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

[11] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Transactions on Graphics*, 23(3):303–308, 2004.

[12] Y. Li, X. Zhai, F. Hou, Y. Liu, A. Hao, and H. Qin. Vectorized painting with temporal diffusion curves. *IEEE Transactions on Visualization and Computer Graphics*, 27(1):228–240, 2019.

[13] H. Lieng, K. J., S. J., and N. A. Dodgson. A colour interpolation scheme for topologically unrestricted gradient meshes. *Computer Graphics forum*, 36(6):112–121, 2016.

[14] H. Lin, J. Zhang, and C. Xu. Diffusion curves with diffusion coefficients. *Computational Visual Media*, 4(2):149–160, 2018.

[15] A. Orzan, A. Bousseau, P. Barla, and J. Thollot. Structure-preserving manipulation of photographs. *NPAR '07*, page 103–110, 2007.

[16] A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, and D. Salesin. Diffusion curves: A vector representation for smooth-shaded images. *ACM Trans. Graph.*, 27(3):1–8, Aug. 2008.

[17] T. Pavlidis. A hybrid vectorization algorithm. *Proceeding of the Seventh International Conference on Pattern Recognition*, pages 490–492, 1984.

[18] T. Pavlidis and K. Steiglitz. The automatic counting of asbestos fibers in air samples. *IEEE Trans. Comput.*, 27:258–261, 1978.

[19] P. Selinger. Potrace: a polygon-based tracing algorithm, 2003.

[20] E. V. Shikin and A. I. Plis. *Handbook on Splines for the User*. CRC Press, 1995.

[21] J. Sun, L. Liang, F. Wen, and H.-Y. Shum. Image vectorization using optimized gradient meshes. *ACM Trans. Graph.*, 26(3):11–es, July 2007.

[22] T. Sun, P. Thamjaroenporn, and C. Zheng. Fast multipole representation of diffusion curves and points. *ACM Transactions on Graphics*, 3(4):1–12, 2014.

[23] J. K. Svergja and H. Lieng. A gradient mesh tool for nonrectangular gradient meshes. *SIGGRAPH '17: ACM SIGGRAPH 2017 Posters*, 2017.

[24] T. Xia, B. Liao, and Y. Yu. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Trans Graph*, 28(5):115:1–115:8, 2009.

[25] S. Zhao, F. Durand, and C. Zheng. Inverse diffusion curves using shape optimization. *IEEE Transactions on Visualization and Computer Graphics*, 24(7):2153–2166, 2018.

The role of inhibition in Deep Learning

Abdulla Bakija and Fatijon Huseini

Abstract—With the advancement of technology and machine learning, devices nowadays can identify voices, movements, and objects in images. Achieving the identification of objects in images, has passed through a lot of periods before the devices could figure out what was in the image. There are many methods like ReNet (Recurrent Neural Network), CapsNet (Capsule Neural Networks), the HMax Model and CNN (Convolutional Neural Networks). From all of the previous mentioned, CNN proved to be the state-of-art method in recognizing objects in images. Even though CNN was the best method when it came to identifying objects in images it had an accuracy of 70-85 percent. The reason behind this issue, was the background colour interfering with the colours of the objects that needed to be identified. To solve this issue at the AAAI-18 Conference, Chunshui Cao et al published their paper, as an attempt of modelling visual attention known as Lateral Inhibition Convolutional Neural Networks (LICNN), which proved to be the best-known method until now at identifying objects in images. With the presentation of LICNN, many individuals were interested and tried to see the advantages of LICNN. One of them was Filip Marcinek, who made a research paper, where he made some experiments to see how LICNN works on images.

Index Terms—Machine Learning, Object Identification, ReNet, CapsNet, HMax model, CNN, LICNN

1 Introduction

Visualizing objects is one of the most important activities the visual sense has, identifying objects in all sort of images, videos, daily life and all things seen by the human eye. This is done as a result of an activity brain does, which differentiates the object and its background, by making the object recognizable while lowering the background visibility level. This process is known as visual attention. This kind of mechanism in today's deep learning is known as LICNN (Lateral Inhibition Convolutional Neural Network). In deep learning the visual attention is done in a process of 5 steps:

1. Input an image.
2. Using CNN to create category-specific attention – in which with the use of CNN we can create gradient-based maps of the objects in the input image.
3. After creating those maps, we apply lateral inhibition in the hidden neurons of the gradient-based maps.
4. By doing the first 2 steps we obtain 5 category-specific attention maps.
5. With the sum of the specific attention maps we create salient objects. By creating salient objects, we obtain the object requested to be visualized.[2]

In our research paper we will explain what are CNN (convolutional neural networks), their structure and how do they recognize objects in images. We will continue to explain about saliency maps, how are they created and what is their purpose in object identification.
Then, we will explain lateral inhibition and their uses in different branches of science. In the paper we will also provide deeper explanation about LICNN(Lateral Inhibition CNN), which is our main topic, their algorithm, why LICNN is the state of art, how do they work and finally we will present some experiments done by the originators of LICNN and other researchers.
To finalize our research paper, we will provide how LICNN expands the future of Deep Learning and other sciences, in which LICNN is used.

2 CNN

Convolutional Neural Networks is a Deep Learning architecture used for identifying objects in images. This method tries to work like the part of the brain which identifies objects through our sense of sight. CNN are used a lot in Deep Learning and the reasons are:

· First, the key interest for applying CNN lies in the idea of using concept of weight sharing, due to which the number of parameters that needs training is substantially reduced, resulting in improved generalization [2]. Due to less parameters, CNN can be trained smoothly and does not suffer overfitting.

· Secondly, the classification stage is incorporated with feature extraction stage, both use the learning process.

· Thirdly, it is much difficult to implement large networks using general models of artificial neural network (ANN) than implementing in CNN. CNNs are widely being used in various domains due to their remarkable performance such as image classification, object detection, face detection, speech recognition, vehicle recognition, diabetic retinopathy, facial expression recognition and many more. [9]

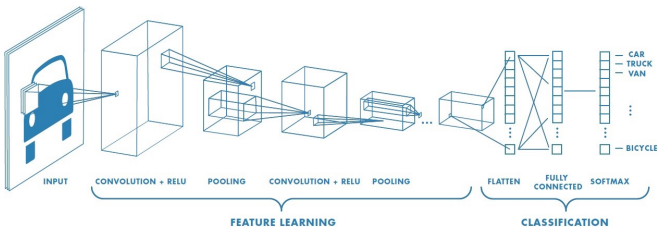


Fig. 1: CNN Structure Scheme [8]

The above figure (Fig.1) shows the structure scheme of CNN. The first part of the structure is the input layer, where the image given by the user is stored. The upcoming layers are convolution + ReLu layer, pooling layer which is the feature learning layer where the features of the input are learned by the machine.
The next layers are, flatten layer, fully connected layer and softmax layer which make the classification part of the structure which classifies different objects in the image.

2.1 Layers of CNN structure

Convolutional Layer is the first layer of the feature learning layers, which is made of convolution kernels that compute several outputs known as feature maps from the inputs[3], as seen in Fig.2.
The feature maps contain values that are changed in the Nonlinearity layer. This happens as a result of activation functions which allows neural networks to learn nonlinear dependencies [3]. The activation functions used in the nonlinearity layer are Relu, sigmoid or tanh.

- *Abdulla Bakija is with University of Groningen, E-mail: a.bakija@student.rug.nl.*
- *Fatijon Huseini is with University of Groningen, E-mail: f.huseini@student.rug.nl.*

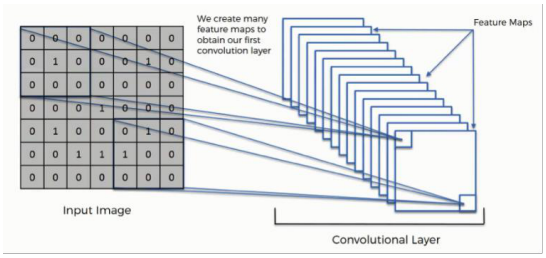


Fig. 2: Convolutional layer [2]

The formula used to calculate feature value y_{ij}, in the k-th featured map, can be calculated :

$$y_{ij,k} = sum\_all(\boldsymbol{w}_k \odot \boldsymbol{x}_{ij}) + b_k \quad [2]$$

where w_k and b_k mean k-th kernel containing weights and its bias term, x_{ij} is the input patch centered at location (i, j), $\odot$ means Hadamard product of two matrices, and sum-all() means summing all values from matrix.
After the feature maps are computed, pooling layer reduces the feature maps resolution. This computation is done by down-sampling the representation. The pooling function can be max or average.[3] (Fig.3).

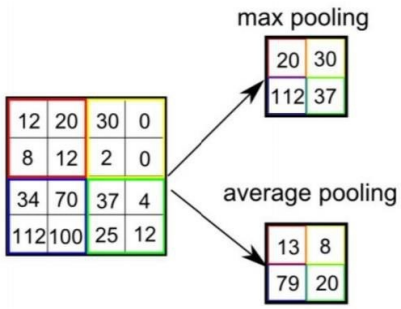


Fig. 3: Max Pooling and Average Pooling [10]

The aforementioned layers can happen many times in the process, dependable on the type of the image and the objects in it.
The next layer is the Flatten layer, in which the data given from the pooling layer is turned into a 1D array for inputting it into the fully-connected layer.[5] (Fig.4)

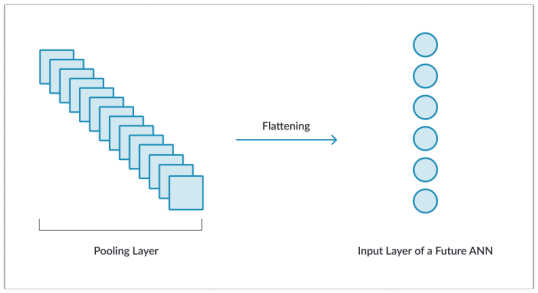


Fig. 4: Flatenning process [11]

The objective of a fully connected layer is to take the results of the convolution/pooling process and use them to classify the image into a label.
The fully connected layer of the CNN network goes through its own backpropagation process to determine the most accurate weights. Each neuron receives weights that prioritize the most appropriate label. Finally, the neurons "vote" on each of the labels, and the winner of that vote is the classification decision.[7]

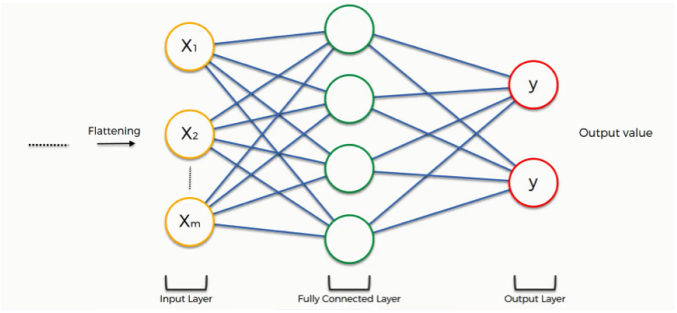


Fig. 5: Fully-connected layer [12]

The last layer of CNN is the loss layer, in which the global minimum of the loss function is searched. In this layer, the most used function is the softmax function which is an activation function that turns numbers into probabilities that sum to one. Softmax function outputs a vector that represents the probability distributions of a list of potential outcomes.[3]

2.2 Saliency Map

Saliency maps are otherwise known as attention maps, in which the important features of the input data are highlighted. In other words saliency maps, visualize the required objects in the image as topological representation. Creating saliency maps on images are done in different methods. For example, coloured images are converted to black-and-white images in order to analyse the strongest colours present in them. Other instances would be using infrared to detect temperature (red colour is hot and blue is cold) and night vision to detect light sources(green is bright and black is dark).[1]

3 LICNN(Lateral Inhibition CNN)

Lateral inhibition in top-down feedback is widely existing in visual neurobiology. LICNN authors are based on this concept from neurobiology which occurs in the human brain and is the process by which stimulated neurons to inhibit the activity of the nearby neuron.
Stimulated neurons inhibit the activity of nearby neurons, which helps sharpen our sense perception. What happens at LICNN, is that the neurons with higher signal block the ones that have a lower signal (mostly background neurons).[2] (Fig.6)
The lateral inhibition model is used for suppressing noise and increasing the contrast between targeted objects and the background.

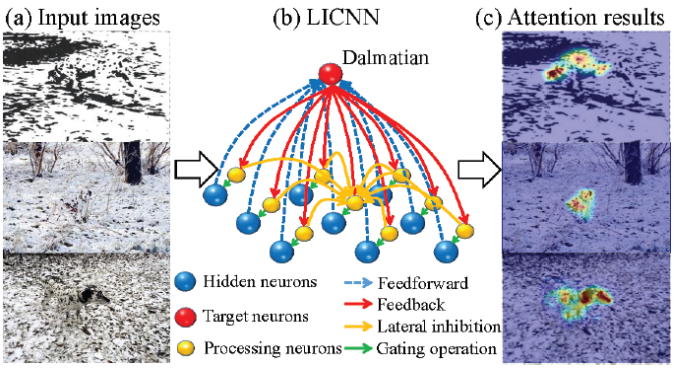


Fig. 6: Lateral Inhibition CNN [2]

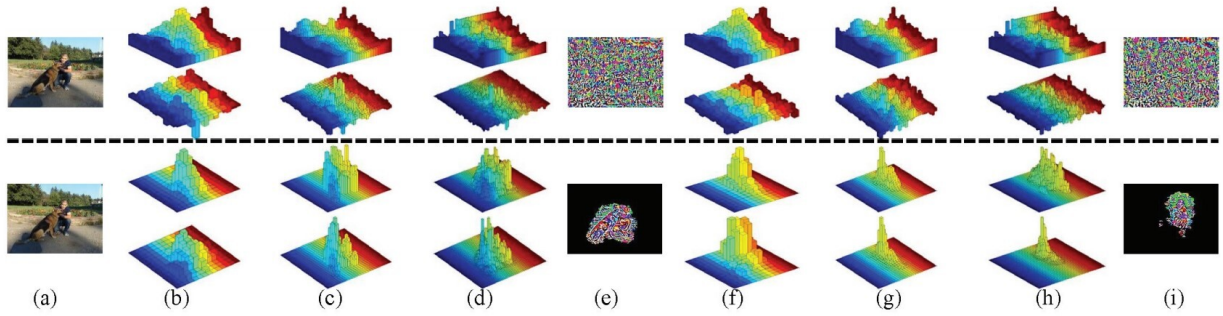


Fig. 7: Comparison between responses and gradients [3]

LICNN is such a powerful process when it comes to identifying salient objects from complicated images with only weak supervision cues. This is because CNN classification has learned many patterns of the objects and with the use of lateral inhibition these objects will be easily recognized and identified.

3.1 LI model implementation

To implement the lateral inhibition, we need to go through an equation so that we can compute the lateral inhibition value for each location. Both papers go through the implementation in the same way, they assume that there is a layer l which produces a cub of CWs(contribution weight) with dimension (W, H, C) where W stands for width, H for height and C for the channel. Primarily, maximum CW at each location should be selected, obtain a CW map, a matrix of dimensions (W, H) obtained by max operation, which is known as Max-C Map. Then lateral connections are constructed so that we can compute the lateral inhibition value for each location as mentioned before.[2]
This value is computed by this equation:

$$x_{ij}^{LI} = a * \underbrace{e^{-\overline{x_{ij}}}}_{average} + b * \underbrace{\sum_{uv}(d_{uv}e^{-d_{uv}}\delta(x_{uv} - x_{ij}))}_{differential} \quad [2]$$

where x_{ij} is a mean of all values in LIZ, x_{uv} is a neighbour of x_{ij} in LIZ, d_{uw} is the Euclidean distance between x_{ij} and x_{uv}, divided by k,(x) = max(0; x), a and b are the balance coefficients.

x_{ij}^{LI} is calculated in a square zone known as lateral inhibition zone (LIZ) which is formed by the k neighbouring points of x_{ij}. The x_{uv} is the neighbouring point of x_{ij}. The Euclidean distance between these points is denoted as d_{uw}. This distance is calculated by the equation:[2]

$$\delta(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \quad [2]$$

The lateral inhibition equation is constructed of 2 parts: Average term and Differential term. The average term protects the neurons within the high response zone. The differential term sharpens the boundaries of the object and increases the contrast between the object and the background in the protected zone created by the average term.[2]
The above equation shows that the furthest and nearest neighbours don't inhibit a lot of the central neuron and indicates that the inhibition is caused by the difference between the central and its neighbouring neurons.

3.2 Top-down Approach

Top-down Approach is a mechanism that proved to be better in terms of detecting an object and visualizing them. This came as a result that using top-down approach uses feedback gradient signals to estimate how much a pattern learned by the neuron, contributes to the given category.[3]
This means that if specific neurons are positively correlated to the neurons of a receptive field, they will be activated. Top-down approach is used to create specific category attention maps.

3.3 Category-Specific Attention Maps

Category Specific Attention Maps are maps that are created from LICNN as a result of doing hierarchical lateral inhibition, which is done by suppressing noise and interference in every layer[2]. This method helps a lot when it comes to detecting objects that are very close to each other and may interfere in visualizing those objects. For example, a person wearing a blue jacket trying to get to his/her blue car or a person with a dog as shown in the Fig.7.
In the fig.7, the first 2 rows show the response and gradient of SUM-C maps of the original VggNet. As it can be seen from the 2 rows, noise cancellation around the object is inconvenient since the object in column (e) and (i) are nowhere to be seen. On the other 2 rows where the response and gradient of SUM-C maps of the LICCN are shown, the noise cancellation is excellent. The image in column (e) shows the non-zero gradient of the dog and the image in column (i) shows the non-zero gradient of the human. For achieving these results in each layer, ReLu function will be used so that the object can identified and for noise cancellation around the object. The columns (b), (c), (d), (f), (g) and (h) are response and gradient of SUM-C maps.

3.4 LICNN Algorithm

Scheme of the algorithm for obtaining attention map:

1.Given an image and a pre-trained CNN classifier.

2.Perform feed-forward and gain predicted category.

3.Carry out gradient back-propagation of the predicted category to estimate contribution to the given category for all neurons.

4.In each ReLU layer, compute the Max-C map and apply the lateral inhibition model on the obtained Max-C map (save obtained suppression mask to the future feed-forward).

5.Perform feed-forward again, during which in each ReLu output layer erase through all channels locations which are 0 in suppression mask for this ReLu layer (erase means 'assign 0 value to it').

6.Calculate Sum-C map and normalize with L2 norm for each ReLu output layer (activation layer) obtained in this feed-forward.

7.Resize all Sum-C maps to the input image size, sum all together and normalize with L2 norm.

To gain a saliency map, one should perform the above algorithm

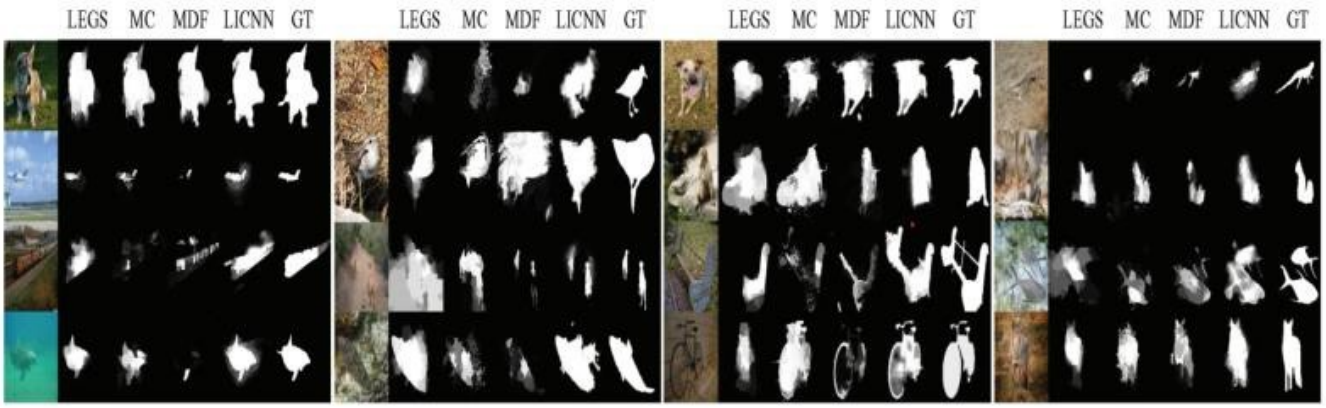


Fig. 8: Visual Comparison between saliency maps of LICNN and LEGS, MC, MDF. The last column is the GT (ground truth) [2]

Data Set	Metric	B1	B2	B3	DRFI	wCtr*	RC	BSCA	PISA	**LEGS**	**MC**	**MDF**	**Ours**
HKU-IS	maxF	0.510	0.759	0.798	0.776	0.726	0.726	0.723	0.753	0.770	0.798	0.861	0.841
	MAE	0.377	0.289	0.161	0.167	0.140	0.165	0.174	0.127	0.118	0.102	0.076	0.101
PASCAL-S	maxF	0.46	0.681	0.710	0.690	0.655	0.644	0.666	0.660	0.752	0.740	0.764	0.755
	MAE	0.395	0.322	0.212	0.210	0.201	0.227	0.224	0.196	0.170	0.145	0.146	0.162
ECSSD	maxF	0.477	0.748	0.805	0.782	0.716	0.738	0.758	0.764	0.827	0.837	0.847	0.831
	MAE	0.396	0.302	0.180	0.170	0.171	0.186	0.183	0.150	0.137	0.100	0.106	0.129
DUT-OMRON	maxF	0.360	0.534	0.613	0.664	0.630	0.599	0.617	0.630	0.669	0.703	0.694	0.677
	MAE	0.347	0.378	0.154	0.150	0.144	0.189	0.191	0.141	0.133	0.088	0.092	0.138

Fig. 9: Comparison of quantitative results including maximum F-measure (the larger is the better) and MAE (the smaller isthe better). The best three results are shown in red, blue, and green color, respectively. Note that LEGS, MC, and MDF are strongly supervised CNN based approaches, and our method is based on weak supervision cues.[2]

separately for each of top-5 predicted categories, then sum all these category-specific attention maps together and normalize with L2 norm.[3]

4 Comparison between methods

Chunshui Cao et al, made an experiment known as The Pointing game where they evaluated the discriminative power of attention maps created by LICCN and compared it to other methods like Excitation Backdrop (c-MWP), error back-propagation (Grad) and deconvolutional neural network for neuron visualization (Deconv). For achieving this experiment, they used PASCAL VOC 07 with 4952 images as a test set.
Excitation Backdrop is a backpropazation scheme which integrates both top-down and bottom up information to compute the winning probability of each neuron efficiently [6].Error back-propagation is the algorithm used along with an optimization algorithm such as Gradient Descent (GD) to learn the parameters of a neural network model. Error back-propagation produces gradients which are then used in optimization. [4] Deconvolutional neural network for neuron visualization is the opposite method of convolutional neural networks.
The experiment was done by extracting the maximum point on a category-specific attention map as the final prediction. For each of the methods mentioned above, the maximum points on a category specific attention map are counted as hits or misses, dependable if the points corresponds to the object category or not. The localization accuracy is measured by

$$Acc = \frac{Hits}{Hits + Misses}$$

for each category.[2]
By collecting the sum of all categories we find the mean accuracy which can be seen in the table below.

The table above shows that LICNN outperforms all the other methods. From the table shown it can be seen that LICNN as its close competitor has the Excitation Backdrop (c-MWP). The fig.11 shows the comparison between them which contain objects from two categories of Pascal VOC. It can be seen that LICNN has better accuracy in its attention maps, meaning less noise and frequency.

Table 1: Mean Accuracy

	Center	Grad	Deconv	c-MWP	LICNN (%)
ALL	69.5	76.0	75.5	80.0	85.3
Difficult	42.6	56.8	52.8	66.8	70.0

4.1 Salient Object Detection Experiment

LICNN has proved that even though there is not a CNN classifier applied to the input images, it can detect objects better than any other method.
To demonstrate the above-mentioned sentence, Chunshui Cao et al evaluated some methods by certain criteria. He and the rest compared LICNN with methods: LEGS, DRFI, wCtr, RC, BSCA, PISA, MC and MDF, to find out which was better at detecting salient objects. As datasets used at the salient object detection methods they used:
HKU-IS(4447 images, most of which have either low contrast or multiple salient objects), PASCAL-S(850 images and is built using the validation set of the PASCAL VOC 2010 segmentation challenge), ECSSD (1,000 structurally complex images collected from the Internet) and DUTOM-RON(5,168 challenging images, each of which has one or more salient objects with complex background).
To show the importance of LI (Lateral Inhibition) in the evaluation, for each dataset they compared the methods in maximum F-measure (the higher the better) and MAE (the smaller the better). Also to analyze the importance of LI they reported 3 baseline results: Baseline 1 (B1), LI is turned off; Baseline 2 (B2), they applied average denoising algorithm on the Max-C map and then handle the denoised Max-C map by thresholding with its mean value; Baseline 3 (b3), they turned off the optimization technique. [2]
Table 2 shows the results from the comparisons between methods in salient object detection by showing the best results coloured in blue, green and red. The values coloured in red are the best values, followed by blue and green as the third best method.
The reason why some methods outperform LICNN, is because LEGS, MC and MDF rely on the manually labeled segmentation masks of salient objects for model learning, while LICNN is based on pre-trained VGG classifier which only requires the image-level class label.
In the fig.8 we can see 4 tables that show the visual comparison between saliency maps of the methods previously mentioned. From the table, we can see that LICNN outperforms the other methods when it comes to creating saliency maps for objects where the background has

high interference with it, and make the objects harder to identify. The last column of the table shows the saliency map of the object required without any noise. The closest to it is LICNN, where its saliency maps are 90 percent identical with the real saliency map.
From all these results, we can say that with the help of feedback signals, LICNN merges all different activated patterns and create the final saliency map of different objects.

4.2 Sanity check of LICNN

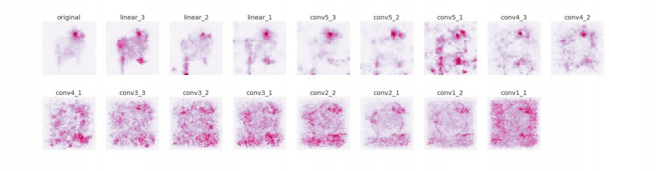


Fig. 10: Cascading randomization resulting saliency maps of the Junco bird image[3]

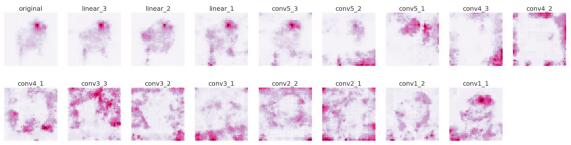


Fig. 11: Independent randomization resulting saliency maps of the Junco bird image.[3]

Filip Marcinek conducted a test based on Google Brain and Berkeley University, where he tested if he could find whether the saliency map creation method could be used to explaining the relationship between inputs and outputs that the model learned and debugging the model. (Marcinek, 2020)
He conducted 2 tests (cascading randomization test and independent randomization test) in which he tried to find out if the saliency maps created by the random values of heights, width and channel would be the same as the saliency maps created by a trained model. If the saliency maps would be the same, it would mean that saliency maps are insensitive to the model parameters.
In the cascading randomization test, the model parameters would be randomized gradually from the top layer to the bottom layer.
While in the independent randomization test, the weight would be independent of the other model parameters.
When the tests were done, the result proved that saliency maps are sensitive to the model parameters, and the saliency maps can change if the model parameters change.
The figures 10 and 11 show the saliency maps of the Jumbo bird created in the cascading randomization test and independent randomization test. We can see that the result, achieved from the cascading randomization show saliency maps, that are more clear than the ones achieved from independent randomization test.

4.3 Types of background influences

Since the background tends to seize the bigger part of the image, in some cases it may influence a lot when it comes to identifying selected objects.
Sometimes the object doesn't classify as an object that is part of that background, or other classified objects tend to occur more in that background.
Using salient maps those objects can be identified as not part of the background but not as the class requested to be seen. But if blurring is used then those objects can be easily identified and be put on the class they belong to.

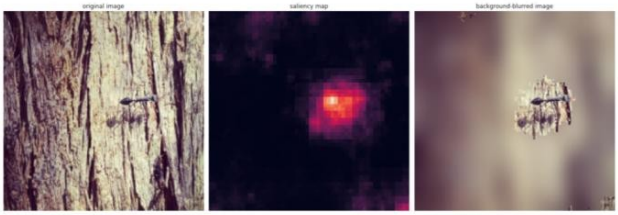


Fig. 12: Example of background impact on image classification. Normally classified as: (1) nail, (2) agama , (3) walking stick, (4) long-horned beetle, (5) foxsquirrel; whereas blurred image as: (1) barn spider, (2) ANT, (3) tick, (4) spaceshuttle, (5) cardoon. Ground truth is indicated by BIG letters.[3]

In the fig.12, it can be seen how blurring affects identifying targeted objects.

4.4 Blurring images

Before it was mentioned that LICNN was created to identify objects in an image, where the background interferes when it comes to identifying the specific object. Filip Marcinek tested this by blurring images in 2 ways. In the first test, he obtained the saliency maps for each image and then blurred the background using a python library Pillow.ImageFilter which applies Gaussian blurring. In the second test he did the opposite operation by blurring the object.
By doing these tests, he tried to find out if the background will interfere in the object detection. Meaning if he blurs the background, will it blur the object too. While in the second test, if blurring the object will blur the background.
For blurring the image, he used Gaussian blurring with the radius 2, 5 and 10.

Fig. 13: Blurring the background with Gaussian blurring with the radius 2, 5 and 10 [3]

Fig. 14: Blurring the salient object with Gaussian blurring with the radius 2, 5 and 10 [3]

From the test conducted with the Gaussian blurring, Filip Marcinek proved that using LICNN can identify objects very easy, even though the background may interfere with the object.

5 Conclusion

In this paper, we researched the role of inhibition in Deep Learning, by analyzing 2 research papers about LICNN. We analysed experiments given in both papers, explained them and their results, and based on what we read we can conclude that LICNN proves to be the state-of-art in the process of identifying specific objects in images.

With LICNN, we can create salient maps, that differentiate specific objects from the background, which has a huge impact when it comes to finding those objects. By the experiments done by different researchers, it could be easily seen that when using LICNN, specific objects are found even without using category-specific attention. Using LICNN has proved that even when blurring is used, either on the object or the background, the needed results are shown.
More research has to be done to reliably determine whether Lateral Inhibition is the best model used in different experiments. One could be integrating Lateral Inhibition in self driving cars, where it could be used for the network to automatically learn the maximum variable features from the camera input, without requiring any human intervention. Another experiment would be introducing Lateral Inhibition into the area of gravitational-wave (GW) data processing, which is strongly connected with the detection of black-holes. Lateral inhibition can show its power in medicine too, especially in creating prosthetic eyes, that can help blind people, see the world as it is.
LICNN is a powerful tool, that has made a huge step into the advancement stage of Deep Learning and AI. With LICNN the future is bright for technology, medicine and science as a whole.

REFERENCES

[1] Abhishek Sharma. What are saliency maps in deep learning. https://analyticsindiamag.com/what-are-saliency-maps-in-deep-learning/.

[2] Chunshui Shao, Zilei Wang, Yongzhen Huang, Liang Wang, Ninglong Xu, Tieniu Tan. Lateral inhibition-inspired convolutional neural network. *The thirty-second AAAI Conference on Artificial Intelligence - AAAI-18 (p. 8)*, 2018.

[3] Filip Marcinek. Reproduction of lateral inhibition - inspired convolutinal neural network. 2020.

[4] Hollan Haule. Understanding error backpropagation. https://towardsdatascience.com/error-backpropagation-5394d33ff49b.

[5] J. Jeong. The most intuitive and easiest guide for convolutional neural network. January 2019.

[6] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, Stan Sclaroff. *Top-down Neural Attention by Excitation Backprop*.

[7] MissingLink. Fully connected layers in convolutional neural networks. https://missinglink.ai/guides/convolutional-neuralnetworks/fully-connected-layers-convolutionalneural-networks-complete-guide/.

[8] Raghav Prabhu. Understanding of convolutional neural network cnn deep learning. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148.

[9] Sakshi Indolia. Conceptual understanding of convolutional neural network - a deep learning approach. In *International Conference on Computational Intelligence and Data Science*, 2018.

[10] Sumit Saha. A comprehensive fuige to cnn - the eli5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[11] SuperDataScience Team. Convolutional neural networks - flatening process. https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening.

[12] SuperDataScience Team. Convolutional neural networks - fully connected layer. https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection.

Facial Expression Recognition and its Impact on Athletes' Performance

Klaas Tilman and Robert Monden

Abstract—Using Facial Emotion Recognition (FER) systems, it is possible to detect the psychological state of athletes. There are various approaches that can be used for this purpose. In this paper we looked specifically at the deep learning and regular machine learning approaches proposed by [3] and [11] respectively, and how athletes' performance may benefit from using emotional recognition. Although both approaches resulted in models with promising accuracy, we found that using the regular machine learning approach as proposed by [11] delivered more accurate results. However, the deep learning approach had a wider range of detectable emotions. We also found that, in the case of soccer, there was a clear link between emotional state of players and team performance. We concluded that facial expression recognition can indirectly lead to improved team performance, since the psychological state of athletes could be monitored using artificial intelligence. This, in turn, would make it easier to take appropriate action at the right time.

Index Terms—Facial recognition, emotion recognition, artificial intelligence, machine learning, sports.

1 Introduction

The ability to predict the winning team of the next soccer world cup by simply looking at the participating players' faces may seem far fetched, but may soon become a reality through developments in the field of facial recognition. Research has shown that there is a clear link between emotions and how much a soccer team wins or loses [4], meaning advances in facial recognition could be used to give insight into the performance of athletes as well as possibly improve their performance.

In this paper we will discuss various facial recognition methods that are available and see what impact they can have on the performance of athletes. Part of deducing this impact is determining the relation between the performance of an athlete and their emotional state. From this, we will try to answer the following question: *"what is the impact of measuring psychological features from the faces of athletes?"*.

A typical FACIAL EMOTION RECOGNITION (FER) system consists of three main stages: *facial detection and pre-processing*, *feature extraction* and *classification* [3], as can be seen in Figure 1.

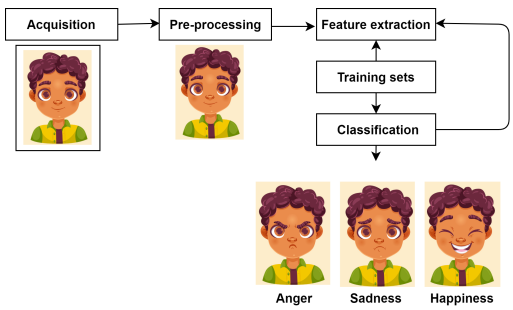


Fig. 1: Stages of a typical FER [5]

Facial detection and pre-processing can be challenging, for example due to insufficient or an abundance of lighting. In addition, it is necessary to filter out any information not needed for recognizing facial expressions. The image's background is a notable example of such information [3].

- *Klaas Tilman is a master's student of Computing Science at the University of Groningen. E-mail: k.tilman@student.rug.nl*
- *Robert Monden is a master's student of Computing Science at the University of Groningen. E-mail: r.monden.1@student.rug.nl*

In FER systems extracted features are used to classify one or more subjects portrayed in an image, based on their detected psychological (i.e. emotional) state. These two steps can be performed using, inter alia, DEEP LEARNING or regular machine learning.

Benamara et al. describe a method that makes use of an approach that involves a CONVOLUTIONAL NEURAL NETWORK (CNN). CNNs are frequently used in pattern recognition, although their widespread use is limited by the amount of resources required by a typical modern CNN architecture [9]. The other method, described by Zedan et al., involves regular machine learning.

After discussing the different possible methods of facial expression recognition we will be discussing its possible impact on athletes' performance on the field.

The paper is organized as follows:

- *Section 2: Methods and Materials* — Concerns the methods used for answering the research question.
- *Section 3: Results* — The core findings of the two different facial recognition approaches as well as the studies regarding impact on athletes.
- *Section 4: Discussion* — Discussion of the results presented in Section 3.
- *Section 5: Conclusion* — Answering of the main research question and suggestions for future work.

2 Methods and Materials

In order to answer the main research question described in the previous chapter, it is necessary to first discuss the methods that were used. This includes discussing the methods used in [3] and [11]. Furthermore, we will look at the methods used in two papers regarding the appliances for athletes.

2.1 Machine Learning Approach

The paper *"Controlling Embedded Systems Remotely via Internet-of-Things Based on Emotional Recognition"* introduces a way to transfer human facial emotional vision to a wifi signal and thereby controlling an IoT application. At first hand, this might not seem relevant to the objective of improving the performance of athletes. However, using this approach it is possible to automatically detect symptoms of various emotions such as depression, anxiety and sadness. For that reason, this method could still be used to deduce an analysis of an athlete.

The methodology of the paper is divided into three phases: Training the model, making predictions and lastly using the Internet-of-Things for communication. Firstly, the training is done using images from the *GENKI-4* database. After reading the images, pre-processing operations are performed. Next, face detection is used to crop the images

such that only the face is used in the following steps. The algorithm for face detection used is *Viola-Jones* [10]. Then, histogram orientation gradient (HOG) is applied in order to perform feature extraction. HOG produces a vector for each sample, consisting of a number of features. The HOG descriptor is computed by dividing the images into cells, then for each cell a histogram is computed of gradient directions for the pixels in the cell. The magnitude of the gradient is calculated as follows:

$$G(i,j) = \sqrt{g_x(i,j)^2 + g_y(i,j)^2}$$

In this function g_x and g_y are the gradient images computed by taking the convolution of the image and the gradient estimation filters, which are defined as $hx = [-1,0,1]$ and $hy = [-1,0,1]$. The gradient orientation can then be obtained using the dominant gradient angle, which as defined as follows:

$$\theta(i,j) = \tan^{-1}\left[\frac{g_y(i,j)}{g_x(i,j)}\right]$$

Using discretization and normalization the descriptor is then formed.

Lastly the classification is done using support vector machine (SVM).

The next phase is using the model to make predictions. The same operation is performed as in the previous phase, however it is applied on a real-time video from which snapshot images are captured. The prediction is done by using the SVM based on the training SVM model. After deriving a result, a control signal is generated and sent through serial communication to the Arduino board. The last phase includes the communication which uses the Internet-of-Things platform. A receiver is programmed to control the specific function for the connected device. All communication is performed using the HTTP protocol.

2.2 Deep Learning Approach

The evolution of GRAPHICS PROCESSING UNITS (GPUs) has reached a stage where GPUs are sufficiently powerful that the use of deep learning has become a viable option. Research has shown that deep learning, with its use of CNNs, can return highly accurate results [3]. It is important to note that this assumes a training set where most, if not all, images are accurately labeled.

The approach proposed by Benamara et al. [3] applies deep learning to the feature extraction and classification stages. In addition, two more stages are added: *label smoothing optimization* and *ensemble learning / fine-tuning*.

2.2.1 Facial Detection and Pre-Processing

Two facial detection methodologies were chosen, with the final choice being based on the user's hardware configuration. As explained prior, this is because deep learning is computationally intensive and therefore needs a GPU. The two methodologies are:

- The VIOLA-JONES method, a non-deep learning based method.
- The YOU ONLY LOOK ONCE (YOLO) v2 method, which is a deep learning-based method. [3]

The pre-processing part of this phase involves cropping detected faces, converting the image into grayscale and resizing the resulting image. The input image is then normalized on an $[0,1]$ scale. [3]

2.2.2 Feature Extraction and Classification

A deep convolution neural network is used in the *feature extraction and classification* phase. Several models were created:

- **Model A —** Nine convolutional layers with a kernel size of 3×3, except for the last convolutional layer, which has a kernel size of 1×1. The model has a total size of 1.37 million parameters. Several other layers were also added, e.g. to reduce overfitting.
- **Model B —** Same as model A, but with a kernel size of 3×3 for the ninth convolutional layer as well. Another convolutional layer was added with a kernel size of 1×1. The model has a total of 1.71 million parameters.
- **Model C —** An extension of model B, but with the addition of another convolutional layer having a kernel size of 3×3, resulting in a total number of 2.17 million parameters.
- **Model D —** An extension of model C, with a total of 2.83 million parameters.

2.2.3 Label Smoothing Optimization

Mislabeled images in the training set decreases the accuracy of the system. Using the label smoothing optimization technique, these effects can be minimized.

By using noise, the confidence factor is reduced, thus decreasing the effects of mislabeled training set images.

The cross-entropy loss function λ' is therefore defined as follows:

$$\lambda' = -\sum_{y=1}^{K}\left[(1-\varepsilon)p(y|x) + \frac{\varepsilon}{K}\right]\log(q(y|x))(1),$$

where λ' denotes the loss, K the number of classes, ε the smoothing factor, $p(y|x)$ the true label probability distribution and $q(y|x)$ the probability that the image belongs to y for each case x. [3]

In the proposed approach, the smoothing factor was defined as $\varepsilon = 0.2$.

2.2.4 Ensemble Learning and Fine-Tuning

Benamara et al. propose several ensemble learning and fine-tuning techniques, such as K-fold Cross-Validation and Bootstrap Aggregation.

In K-FOLD CROSS-VALIDATION, the training data set is split k-fold. $k-2$ different subsets are used for the training phase, while the two remaining subsets are used for model selection and error estimation respectively [2, 3].

BOOTSTRAP AGGREGATION is also known as BAGGING and is a technique that can help reduce variance for algorithms with a high variance (e.g. classification and regression trees) [1], thus reducing overfitting. It is also used to improve the accuracy of models. Bootstrap Aggregation works by aggregating the predictions of multiple weaker, individual models to get to a final prediction [7].

Additionally, the initial weights of the model may be fine-tuned, possibly leading to increased performance. Lastly, the creation of an ensemble probability function was proposed, based on a number of models M:

$$q(y_i|x_i)_{ens} = \frac{1}{M}\sum_{m=1}^{M}\frac{e^{z_{y_i}}}{\sum_{j=1}^{K} e^{z_j}}$$

The above formula considers an average, however an alternative formula considering the maximum estimate was also proposed:

$$q(y_i|x_i)_{ens} = \max_{M}\frac{e^{z_{y_i}}}{\sum_{j=1}^{K} e^{z_j}}$$

In both formulae described above, z_{y_i} denotes the logit.

2.3 Impact on Athletes

There exist various studies regarding the impact of facial expressions on the performance of athletes. In this paper we will be focusing on two papers. Firstly, *Emotional expressions by sports teams: An analysis of World Cup soccer player portraits* [6], which focuses on emotional display of soccer players and its connections with their performance on the field. Secondly, we will be discussing *The effects of facial expression and relaxation cues on movement economy, psychological, and perceptual responses during running* [4]. This papers focuses on the relation between the emotional display of runners and

Fig. 2: Example of player portraits from Panini sticker albums. Extract from 2014 Panini album displaying defensive players Campagnaro (Argentina) and Boateng (Germany).

their performance. In this section we will be discussing the methods used in these papers.

For the analysis of world cup soccer players, photo portraits of the players were used, like in Figure 2. In total 4896 portraits were collected, from 76 different squads. Analysis of the portraits was done using a tool named *FaceReader 6*, which outputs the emotions anger, happiness, disgust, fear, sadness and surprise. The emotions used further in the research are anger and happiness. The portraits which performed best were the ones with good lightning and a frontal view of the face. Furthermore, a larger percentage of the pictures were correctly analyzed for Caucasian faces.

Then, after analyzing the different portraits, it has to be compared to the performance of the concerning teams. For this data was gathered from *The Rec.Sport.Soccer Statistics Foundation.* [8]. In total three measurements for performance were used: goal difference, number of goals scored and number of goals conceded. Additionally, special attention was given to the first stage of the world cup, because no teams have left the tournament yet at that point.

Secondly, we will take a look at the method for linking emotional expressions to the performance of runners. For the study, 24 endurance runners competing on club-level were used, who all were completely healthy. For each participant, two sessions were performed. Where each session was done under the same circumstances for each runner, with respect to hydration, time of session and food consumed. Session one was done on a treadmill and session two consisted of four blocks of 6 minutes running. Each block was done either while smiling, frowning, consciously relaxing or with normal attentional focus.

With regards to the data collection, for session two respiratory exchange variables (VO_2,VCO_2), respiratory frequency, tidal volume, minute ventilation (V_E), respiratory quotient and heart rate were measured continuously. Additionally, following each block completed, the participated rated their perceived effort with regards to to how hard, heavy or strenuous they perceived the session. The participants were also asked to rate their ability to maintain attentional cues during a block, as a manipulation check. Lastly, statistical analyses was performed. For this REPEATED MEASURES ANALYSIS OF VARIANCE (RM-ANOVA) was performed for each of the primary dependent variables, secondary respiratory variables (see Table 6) and the manipulation check.

3 RESULTS

This section discusses the results of the different approaches. These results are then related to the possible impact of recognizing the psychological state of athletes.

3.1 Machine Learning

The results of the machine learning approach is divided into two parts, the results of training and testing the feature extraction and classifier, and the part which includes controlling the remote embedded device. We will be focusing on the first part, since this is most relevant to our research.

We have included the results from experiments in the tables 1 and 2. During the experiments that Zedan et al. performed they modified the parameters of *HOG* and *SVM*, in order to find the best recognition rate. The image size used in the experiments is [100 x 100] pixels. Furthermore, the block size (bs) of *HOG* is fixed at two pixels for all the experiments.

Table 1 shows experiments conducted using the learning algorithm (LS). Here, the most accurate experiment is experiment number one, with an accuracy of 88.6%. The CONFUSION MATRIX on the right indicates the correctly recognized emotions. The confusion matrix has the following format:

TN	FN
TP	FP

where, given two categories A and B, an image i and a prediction p_i:

- **True Negative (TN):** If $i \in A$, then p_i correctly states that $i \notin B$
- **True Positive (TP):** If $i \in A$, then p_i correctly states that $i \in A$.
- **False Negative (FN):** If $i \in A$, then p_i (incorrectly) states that $i \in B$
- **False Positive (FP):** If $i \notin A$, then p_i (incorrectly) states that $i \in A$.

For experiment one, 809 non-smiling faces are correctly recognized, while 110 are not. Moreover, 963 smiling faces are correctly recognized and 118 are not.

Additionally, Table 2 shows the experiments performed with sequential minimal optimization (SMO), which is another learning algorithm. The best experiment again has an accuracy of 88.6%. The confusion matrix has the same correctly recognized faces as with the previous algorithm.

This means that the recognition rate is not affected by changing the algorithm used. The paper proposes to decrease the training and testing time by using a smaller image size of 80 by 80 pixels. Using this method the accuracy is increased to 88.9%.

3.2 Deep Learning

In order to assess the validity of their method, Benamara et al. carried out experiments consisting of three phases:

i. Training single models on a FER database (without label smoothing)

ii. Training single models using label smoothing ($\varepsilon = 0.2$)

iii. Fine-tuning of single-model weights, both with and without label smoothing [3]

For the experiments, the *FER 2013*, *SFEW 2.0* and *ExpW* databases were used for training, validation and test images. In this paper, however, we will specifically focus on results of the FER 2013 database for simplicity.

The models were tested both individually and as part of an ensemble (e.g. *AB*, *AC*, *ABC*, *ABD*, etc).

After performing the experiments, the results were then analyzed based on three aspects: (ensemble) performance, label smoothing optimization and computation time.

Exp. No.	HOG-CS	HOG-bs	Image size	Feature length/image	Training time	Testing time	Accuracy	ConMat)
1	11	2	100	864	4.6	1.1	88.6000	809 110 118 963
2	7	2	100	2808	7.3	4.6	87.9000	801 118 124 957
3	9	2	100	1440	7.31	3.7	88.4500	808 111 120 961
4	13	2	100	432	5.3	0.3	87.7	802 117 129 952
5	15	2	100	360	6.40	0.4	88.20	807 112 124 957

Table 1: HOG and SVM optimized parameters for training and testing with the LS learning algorithm

Exp. No.	HOG-CS	HOG-bs	Image size	Feature length/image	Training time	Testing time	Accuracy	ConMat)
1	7	2	100	2808	97.74	4.0	87.80	800 119 125 956
2	9	2	100	1440	92.01	2.5	88.45	808 111 120 961
3	11	2	100	864	3.32	0.5	88.60	809 110 118 963
4	13	2	100	432	3.2	0.5	87.70	801 118 128 953
5	15	2	100	360	2.95	0.3	88.10	806 113 125 956

Table 2: HOG and SVM optimized parameters for training and testing with the SMO

Model	No Label Smoothing	Label Smoothing
A	66.40%	67.15%
B	65.76%	66.95%
C	65.51%	68.60%
D	67.26%	67.85%

Table 3: Object recognition performance of each single CNN model, expressed in terms of validation accuracy. The FER 2013 database was used for the validation set. Source: [3]

3.2.1 (Ensemble) Performance and Label Smoothing

Table 3 lists the object recognition performance of each single CNN model, expressed in terms of accuracy. For each model the results are shown both with and without label smoothing optimization applied.

Table 4 displays the object recognition performance of various ensemble models. For each ensemble the results are shown both with and without label smoothing optimization applied.

Ensemble	No Label Smoothing	Label Smoothing
AB	68.96%	69.43%
AC	68.93%	70.27%
AD	69.77%	69.63%
BC	68.43%	69.57%
BD	67.46%	69.16%
CD	68.68%	69.60%
ABC	69.30%	70.88%
ABD	69.41%	69.74%
BCD	68.88%	69.91%
CAD	69.69%	70.19%
ABCD	70.00%	70.66%

Table 4: Object recognition performance of each single CNN model, expressed in terms of average validation accuracy. The FER 2013 database was used for the validation set. Note that this table does not include the maximum accuracy. Source: [3]

Compared to other reported performances regarding the FER 2013 database, the proposed method outperformed methods such as those proposed by Mollahosseini et al. in 2016 (66.40%) and Devries et al. in 2014 (67.21%), but with a reported performance of 72.72% it has lower performance than the methods proposed by Kim et al. in 2016 (73.73%) and Pramerdorfer et al. in 2016 (75.20%). [3]

It must be noted, however, that when using the SFEW 2.0 database a much lower performance of 51.04% is recorded. Among the compared methods, the best-performing method has an accuracy of 55.15% using a SFEW 2.0 dataset. [3]

3.2.2 Computation Time

Experiments were carried out in order to compare computation time across different configurations. The Viola Jones and YOLO v2 face detection methods were tested on multiple hardware configurations. In all cases the ABC ensemble was used.

Face Detection Method	Hardware	Total time in ms
Viola Jones	i7 CPU	141.97 ± 3.34
YOLO v2	i7 CPU	307.71 ± 3.67
Viola Jones	GeForce GTX 1080	17.23 ± 0.84
YOLO v2	GeForce GTX 1080	13.48 ± 1.48
Viola Jones	Tegra X1	137.22 ± 3.37
YOLO v2	Tegra X1	203.04 ± 1.84

Table 5: Comparison of computation times of different face detection methods on different hardware. Source: [3]

3.3 Impact on Athletes

In this chapter we will discuss the results of the two different methodologies for measuring the impact of facial recognition on athletes.

Firstly, for the world cup soccer players we focus on the results which described the relationship between emotion display and behavior. The paper presents econometric evidence that both the emotions anger and happiness can be linked to the performance of a team. Here, the display of both anger and happiness is a sign of a better team performance, due to more goals being scored than conceded. Specifically,

Measure	Smiling	Frowning	Relaxed	Control	p	Partial n^2
Primary variables						
VO2 (ml/min/kg)	32.90 (4.05)	33.84 (3.99)	33.63 (3.89)	33.65 (4.18)	0.001	0.20
Heart Rate (bpm)	146.86 (14.46)	148.65 (14.41)	146.96 (16.02)	147.30 (13.84)	0.231	0.06
Perceived Effort (AU)	11.25 (1.49)	12.29 (1.88)	11.38 (1.76)	11.63 (1.44)	0.004	0.17
Affective Valence (AU)	2.58 (1.77)	1.96 (1.83)	2.50 (1.50)	2.54 (1.25)	0.266	0.06
Activation (AU)	2.83 (0.96)	3.63 (1.13)	2.96 (1.12)	2.94 (1.20)	0.001	0.24
Manipulation Check (%)	82.08 (16.41)	85.42 (13.51)	87.08 (8.59)	81.25 (16.50)	0.312	0.05
Secondary Variables						
VCO2 (ml/min/kg)	31.16 (4.22)	32.07 (4.40)	31.58 (4.07)	31.73 (4.49)	0.025	0.14
Respiratory Frequency (bpm)	38.80 (7.39)	38.55 (9.40)	36.58 (7.57)	36.62 (8.36)	0.079	0.10
Tidal Volume (L)	1.75 (0.45)	1.83 (0.52)	1.84 (0.50)	1.86 (0.55)	0.083	0.10
Minute Ventilation (L/min)	65.64 (13.35)	67.16 (13.02)	64.95 (12.82)	65.02 (13.30)	0.047	0.11
Respiratory Quotient (AU)	0.95 (0.04)	0.95 (0.05)	0.94 (0.04)	0.94 (0.04)	0.298	0.05

Table 6: Outcomes for primary and secondary variables during each attentional focus condition.

happiness is linked to scoring more goals and anger to conceding fewer goals. The same holds for the teams position in the world cup, because they usually reach a better position in the tournament when displaying either anger or happiness.

Next, we take a look at the impact of facial recognition for runners. The outcomes can be seen in Table 6. The study showed that most (58.33%) participants were most economical when smiling while running. The perceived effort by runners was higher while frowning than for both smiling and relaxing. Lastly, there was no significant effect on the heart rate of runners, related to their emotions showed.

4 Discussion

In the past sections we have focused on two methods for facial recognition: A deep learning and machine learning approach. We have also looked at the results of these two methods in Section 3. Furthermore, we have looked at two applications of facial recognition within soccer and running.

In this section we will discuss the different methods and how they can influence the impact on athletes.

The machine learning approach proposed by Zedan et al. seems very promising. Over multiple experiments the accuracy was between 87.7% and 88.6%, and ultimately, when decreasing the size of the images used, the accuracy was increased to 88.9%.

In contrast to the machine learning approach, we consider a Deep Learning method by Benamara et al. The accuracy for this approach is significantly lower than the machine learning method. The accuracy ranges between 65% and 70%, by alternating between having label smoothing or not, and applying an ensemble model. By looking only at the accuracy of both methods, one might suggest using the machine learning approach, since that one seems to be much more precise. However, there is another factor that must be considered: the number of emotions recognized by each program. The emotional recognition which makes use of machine learning, is only able to recognize two types of emotions: either smiling or non-smiling. On the other hand, the deep learning approach can recognize up to seven types of emotions: happiness, surprise, neutral, disgust, sadness, fear and anger.

In order to make a decision about the best method for applying facial recognition to sports, we first need to take another look at what impact it could have on athletes. The results of the two methods showed that a correlation could be detected between an athlete's emotions and their performance. For soccer players, there was a clear link between the emotions anger/happiness and the goal difference of the competing teams. In addition to that, for runners a connection could be made to their running economy and perceived effort.

The facial expressions and emotions used in both papers included smiling, frowning, consciously relaxing, normal attentional focus for runners and anger, happiness, surprise, disgust, sadness and fright for the soccer players. Because of the many emotions used in both studies, the deep learning approach is most appropriate to apply, since it can be used to analyze a greater number of emotions.

5 Conclusion

In this paper we have looked at two different methods for facial recognition. In addition, we have looked at how they might be used to detect the psychological or emotional state of athletes.

Although the machine learning approach proposed by Zedan et al. has greater accuracy, the deep learning approach can be used if a more specific assessment of an athlete's emotional state is required. This is since it can be used to analyze a greater number of emotions.

By employing emotion recognition on athletes, new possibilities are opened up. If a soccer team where the players are happy is predicted to perform better, then it would make sense to use this information to improve team morale. In case one or more players are frequently exhibiting emotions that can be considered *negative*, it could be worth looking into the underlying causes. This, in turn, would improve general team morale and might lead to improved performance.

It can therefore be concluded that facial expression recognition has, indirectly, a positive impact on athletes' performance.

5.1 Future Work

For future work it would be useful to take a look at a greater number of approaches, in order to possibly find an even better solution.

Furthermore, it would be useful to see if the machine learning approach can be adapted to support a greater width of emotions and facial expressions.

Acknowledgements

The authors wish to thank Dr. G. Azzopardi for proposing the topic and providing a list of source papers.

References

[1] Bagging and Random Forest Ensemble Algorithms for Machine Learning, Dec 2020. Accessed March 16th, 2021.

[2] D. Anguita, L. Ghelardoni, A. Ghio, L. Oneto, and S. Ridella. The'k'in k-fold cross validation. In *ESANN*, pages 441–446, 2012.

[3] N. K. Benamara, M. Val-Calvo, J. R. Álvarez-Sánchez, A. Díaz-Morcillo, J. M. Ferrández-Vicente, E. Fernández-Jover, and T. B. Stambouli. Real-time facial expression recognition using smoothed deep neural network ensemble. *Integrated Computer-Aided Engineering*, (Preprint):1–15, 2020.

[4] N. E. Brick, M. J. McElhinney, and R. S. Metcalfe. The effects of facial expression and relaxation cues on movement economy, physiological, and perceptual responses during running. *Psychology of Sport and Exercise*, 34:20–28, 2018.

[5] freepik. Character showing emotions free vector. image: Freepik.com.

[6] A. Hopfensitz and C. Mantilla. Emotional expressions by sports teams: An analysis of world cup soccer player portraits. *Journal of Economic Psychology*, 75:102071, 2019.

[7] H. Kandan. Bagging the skill of Bagging(Bootstrap aggregating). *Medium*, Jan 2019. Accessed March 16th, 2021.

[8] rec.sport.soccer. The rec.sport.soccer statistics foundation. http://www.rsssf.com/.

[9] M. Valueva, N. Nagornov, P. Lyakhov, G. Valuev, and N. Chervyakov. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*, 177:232–243, 2020.

[10] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001.

[11] M. J. Zedan, A. I. Abduljabbar, F. L. Malallah, and M. G. Saeed. Controlling embedded systems remotely via internet-of-things based on emotional recognition. *Advances in Human-Computer Interaction*, 2020, 2020.

Consistency Trade-offs in Distributed Systems

R.J.M. van Beckhoven, R.M.Sommer

Abstract—Distributed systems have been become an extremely commercially interesting way of deploying software systems due to their intrinsic advantages. Distributing a system over multiple computers has advantages for reliability, high-availability, and performance. In this paper we address the implications of distributed systems and what trade-offs are made in recently developed systems. We focus on the implementation of the Amazon S3 and ZooKeeper distributed systems and relate these to the CAP-theorem. We find that both systems forfeit the consistency property, but do guarantee eventual consistency. Both systems embrace properties that maximize the data-centric consistency and provide consistency guarantees from a client-centric point of view.

Index Terms—Consistency, Distributed computing, CAP, S3, ZooKeeper

1 Introduction

Due to the ever-growing nature of the internet and the shift towards cloud computing, distributed system have gained in popularity rapidly. A distributed system is a software system, which consists of multiple components that are located on different networked computers [12]. Given that the resources are available, a distributed system has the advantage that it is relatively straightforward to achieve high availability, high performance and scalability.

However, problems can arise with designing models that organize and manage the data. For example, modern day social media applications generate an enormous amount of data. Users expect that the data is available at any given moment and that the data they receive is up-to-date. However, in a distributed network, this is hard to achieve. It is a challenge to have a database which is highly available, consistent, and partition tolerant. Historically, the ACID principle is used to ensure a reliable database, focusing on consistency. More recently, the BASE model has been introduced as an alternative. This model aims at providing more flexibility in data management, focusing on availability over consistency. What both these principles lack is any notion regarding partition tolerance. Inspired by these principles, the CAP theorem was introduced by Brewer in 2000 [3] and later proven by Gilbert and Lynch [7]. The theorem addresses the problem of a distributed system having the properties **C**onsistency, **A**vailability, and **P**artition tolerance. More recently, Brewer elaborated on a novel interpretation on the claims in the CAP theorem [4]. The idea is that a partition is always a possibility in a distributed system, which should be managed. Therefore optimizing the consistency-availability trade-off is the desirable choice to make.

In this paper we revisit the database principles ACID and BASE in Section 2. We relate these principles to the original CAP theorem and the novel interpretations of the CAP theorem. Afterwards, we examine this new interpretation of the CAP theorem by doing a case study of two distributed database implementations in section 3: Amazon S3 [2] and ZooKeeper [8]. We look at how they are implemented and how they tackle the availability-consistency compromise. Afterwards, we discuss our findings in section 4 and present our conclusions in Section 5. Finally, we provide potential future work in the field in section 6.

2 Background

Traditional approaches for database systems, such as RDBMSs, provide ACID properties to guarantee a certain reliability of a database to the user. With NoSQL databases, these ACID rules are hard to enforce and would resolve in an enormous impact on performance if they were enforced. Instead, these database systems opt for more lenient approaches towards reliability, which follow principles such as BASE and CAP.

2.1 ACID

A system that enforces ACID rules, has the following four guarantees:

- Atomicity
- Consistency
- Isolation
- Durability

Atomicity implies that all operations are atomic; each transaction is considered as a single unit and either the entire transaction is committed, or the transaction is discarded upon failure. This prevents partial updates of the system which could result in incorrect states of the system. Consistency ensures that the database has a valid state at any given point in time. No transaction can alter state in such a way that invariants are violated. Isolation ensures that, when concurrent transactions occur, the final state of the database system will be similar to the state if these transactions had occurred in sequence. Durability ensures that committed data will remain persisted, even when failures or outages occur.

2.2 BASE

For systems that do not require these strict guarantees on data reliability, ACID transactions impose limits on the scalability of the system. A more loose approach to consistency is introduced by the concept of BASE:

- Basically Available
- Soft state
- Eventually consistent

Basic availability is an approach in which the data appears to be available most of the time. This can be achieved by using a distributed approach to database management, where data is spread over multiple systems. Soft state discards the entire consistency guarantee of ACID; consistency is not guaranteed after writes and is not imposed on all replicas of the data. However, consistency is achieved by eventual consistency, which implies that the database management system will ensure that data is consistent at some point in the future.

- *R.J.M. van Beckhoven, MSc Student Computing Science at UG. E-mail: r.j.m.van.beckhoven@student.rug.nl*
- *R. M. Sommer, MSc Student Computing Science at UG. E-mail: r.m.sommer@student.rug.nl.*

2.3 CAP-theorem

The CAP-theorem [3] states that any distributed storage system can only employ two out of three properties:

- Consistency
- (High) Availability
- Partition tolerance

This theorem originated in the year 2000 and was mainly aimed at making system designers aware of the trade-offs imposed by distributed systems.

The definition of a partition is a combination of expected time-bounds and consistency; a partition occurs when consistency cannot be guaranteed after an expected time-bound. This time-bound is determined by system designers based on target response times. When a network partition occurs, subsystems of a distributed system are unable to communicate to each other. This results in failing to achieve this consistency. When the system enters a partition, it has to opt for either availability or consistency. This can be done by cancelling the operation or by proceeding with the operation respectively. Proceeding imposes a risk where data becomes inconsistent and cancelling decreases availability.

This theorem results in three different models which can be employed: Consistency and Partition tolerance (CP), Availability and Partition tolerance (AP) and Consistency and Availability (CA).

CP implies that, when a partition occurs, the system forfeits availability. The system ensures that the state of the system remains consistent, potentially by shutting down parts of the system.

AP implies that, when a partition occurs, the system forfeits consistency. Eventual consistency is a model that follows AP, since it is not consistent at any point in time, but it is highly available because of that.

CA implies that when a partition occurs, the system forfeits partition tolerance. In theory, this means that no action is taken by the system. However, in practice this often means that either availability or consistency is forfeited. For example, if the system has a protocol such as Paxos, which needs to ensure that all nodes have acknowledged the write, the system would potentially not reach consensus. This, in turn, results in the system being unavailable.

In distributed systems, the assumption is made that partitions can always occur, because network switches may eventually fail. This assumption also implies that one cannot simply forfeit partition tolerance in a distributed system. When no action is taken upon a partition, this eventually results in either a state that is inconsistent or a system that is temporarily unavailable, thus forfeiting either consistency or availability.

While the original CAP theorem suggests that system designers have to opt for two out of three, the better approach is not to focus on just two, but rather focus on optimizing consistency and availability [4]. With the assumption that partitions rarely occur, the effect of a partition can be minimized. However, this still means that the same trade-off has to be made by the system at the event of a partition.

At the event of a partition, the two sides of the partition potentially disagree about the state of the system. The system should handle this partition by entering a partition mode after which both sides agree about the state. There are several options to choose what to do during the partition mode. For example, choices regarding maintaining invariants can be made. Invariants can be maintained during the partition, potentially affecting the availability of the system. Another option would be to allow invariant violations during the partition mode and fix these during the recovery. During this recovery, the two states have to be merged into a state that is sound. This could be done in a similar fashion to how merging works in git. However, this also implies that merging should not cause conflicts. Achieving this can be done by putting constraints on the system during the partition, potentially restricting certain operations. Yet another option is to use operations that are strictly commutative. Doing so ensures that all operations can be merged safely after a partition. While this theory sounds interesting, in practice it is not easy to implement. In order to aid with this commutativity, Commutative Replicated Data Types (CRDTs) can be used [10]. After recovery, the state of the system is consistent and can therefore operate in a normal mode again.

2.4 Generic distributed database

A database can be distributed in two ways. The first is by fragmenting the data, such that the data itself is spread over multiple nodes. The second option is to replicate the full data over multiple nodes. We define a primitive distributed database as a system in which data is replicated over multiple nodes. Every node has a complete copy of the data, such that every node is able to respond correctly to read requests. Replication leads the database to be fault-tolerant and enhances the availability of the system. The write requests are managed by a broadcast protocol, implemented using a master-slave architecture. Here the master receives the request and ensure that each slave will update its data-model. This results in consistent data over all replicas. We assume a partition can occur within the system and the system continues to operate when the partition occurs. This results in the system having two options. The system can keep responding to read-requests and accepting that the data is not up-to-data, thus forfeiting consistency. Or the system can stop responding to read-requests and guarantee the integrity of the data, thus forfeiting availability. In this generic model, prioritizing different aspects can either lead to the system achieving high availability or high consistency.

3 Methods

In the following section, we will review two research papers on two different distributed systems, Amazon S3 [2] and Apache ZooKeeper [8]. We take a look at what the systems intend to achieve and how they have implemented this. We relate these implementations to the novel CAP theorem implementation.

3.1 Amazon S3

Amazon Simple Storage Service or S3 is a popular distributed database service which is used in many applications such as IoT, Data lakes and Machine Learning. Its internal structure is an object storage which offers durability, availability, performance and other quality attributes.

3.1.1 Implementation

S3 uses key-value buckets to store objects. These buckets are logical structures that can be created by the user. Objects in these buckets are identified by a key and a version ID. This key is a unique key, of which multiple versions are tracked within S3.

Up to December 2020, S3 used an eventual consistency model. This model imposes a window in which data is inconsistent after a write. By this design decision and following the CAP theorem models, S3 employs an AP model where it forfeits consistency. This model can be seen in Figure 1.

This model has been pivoted to a strong consistency model [1]. In the strong read-after write consistency model, S3 defines a set of actions which can guarantee strong consistency. For PUTs and DELETEs of objects, reading is guaranteed to provide the latest version of the object (See Figure 2). This still means that certain operations are not guaranteed to provide strong consistency. For example, concurrent writing causes non-deterministic behaviour and can therefore not guarantee this. Instead S3 determines internally which write takes precedence, based on the order of the writes. This does not necessarily imply that the client observes the same precedence.

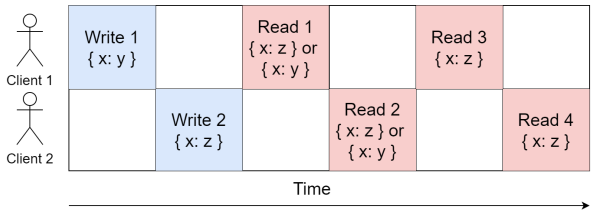


Fig. 1. Timeline of events during two writes (PUT) and 4 reads (GET), with an eventual consistent model of S3.

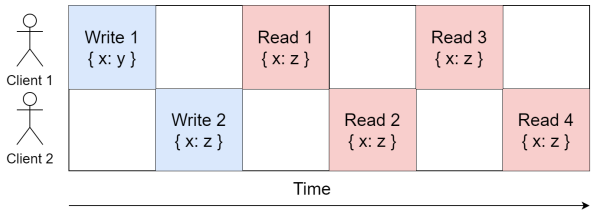


Fig. 2. Timeline of events during two writes (PUT) and 4 reads (GET), with a strong consistent model of S3.

3.1.2 Terminology

In the research paper [2], the following terms are used to describe the results:

Inconsistency window
The time it takes after a write, before the system is consistent again. This consistency implies that every node of this distributed system contains the same state.

Monotonic read consistency
Monotonic read consistency is a property which determines that no following reads will return a stale value after a read returns a new value.

3.1.3 Research methods

Bermbach and Tai [2] consider two approaches: a data-centric and a client-centric approach. The data-centric approach focuses on the internal state of the system. It defines consistency when all replicas of the system contain an identical state. The client-centric approach measures the consistency guarantees from a client perspective, based on stale data being returned. The data-centric approach provides more useful insights for developers, whereas the client-centric approach is a more useful approach for clients. The distributed system could have measures which hide the eventual consistency for clients.

For the evaluation of S3, a client-centric approach is taken. A set of EC2[1] instances is used to deploy a reader. Since data is spread over multiple replicas, a single reader is unlikely to cover all replicas, thus resulting in inaccurate measurements. To increase the accuracy of the measurements, 12 readers are deployed over 3 availability zones. Each reader then reads the S3 system with an interval of 10ms. With these components, the inconsistency window and monotonic read consistency can be determined.

3.1.4 Results

The results [2] show that there are actually two phases in which S3 operates: a phase in which the inconsistency window is stable and a phase in which the inconsistency window follows an increasing pattern, matching a sawtooth wave (referred to the paper by LOW and SAW respectively). During this LOW phase, a median window length of 28ms is measured. For the SAW phase, the window length ranges between 0 and 12s. While there are certain issues in this research, imposed mostly by clock syncing strategies using NTP, there are no clear indications that could explain the behaviour of the SAW phase. This behaviour is unprecedented in any other distributed storage systems and can only be explained by internal design choices from Amazon. Furthermore, the monotonic read consistency is violated in 12% of the time.

[1] aws.amazon.com/ec2

3.1.5 Discussion

Since the paper from 2011, the consistency model of S3 has been updated in December 2020. Instead of eventual consistency, it employs a strong read-after-write model. This ensures that after a write (either a PUT or DELETE), all consequent reads obtain the latest state. A reason for this architectural change is that clients that require strong consistency often resort to using a 3rd party implementation as a layer to guarantee this, resulting in a significant decrease in performance.
By employing a strong read-after-write model, S3 is guaranteed to have no more monotonic read violations. This should come with a trade-off on availability or performance. However Amazon is not transparent about how S3 is implemented, which results in only being able to guess at this point. No research has been performed on S3 since this pivot.
In terms of which CAP theorem model is employed, S3 employs a CP model nowadays. However, it is open for discussion whether this is actually a CA model, since there is no clear indication what happens during a partition.

3.2 ZooKeeper

ZooKeeper is a service with the primary goal of helping in the coordination of processes in a distributed system. Its design relies on ideas proposed in previously introduced coordination services, fault-tolerant systems, distributed algorithms, and file systems. Through its efficient design and simple interface, ZooKeeper provides a reliable distributed coordination service.

3.2.1 Terminology

In the research paper by Hunt et al. [8], the following terms are used to describe the system:

Wait-free data object
A wait-free data object is determined as an object that is implemented using non-blocking primitives, such that it can be accessed regardless of other processes using it.

A-Linearizability
Asynchronous Linearizability, or A-linearizability, indicates the precedence of (write) requests received by the system. An A-linearizable system guarantees the order of execution of the requests when they are processed asynchronously.

3.2.2 Implementation

The ZooKeeper service provides a wait-free coordination kernel, which can be accessed via a simple API. The service makes use of an ensemble of servers using replication. This setup enable ZooKeeper to achieve high availability and high performance, while relaxing the consistency guarantees.
ZooKeeper consists of three components in a pipelined architecture as depicted in Figure 3. We provide a short overview of the components within the system:

- Request Processor — The request processor is responsible for the handling of write-request on the distributed file system.
- Atomic Broadcast — The atomic broadcast protocol ensures that the state of all replicas of the database are updated in order. This guarantees that each replica will update its state according to an identical stream of write requests, hence local replicas do not diverge. However, since at any point in time some servers may have applied more state updates, it does not guarantee 100% consistency.
- Replicated Database — The file system is replicated over servers, each server handling read requests of its own collection of clients.

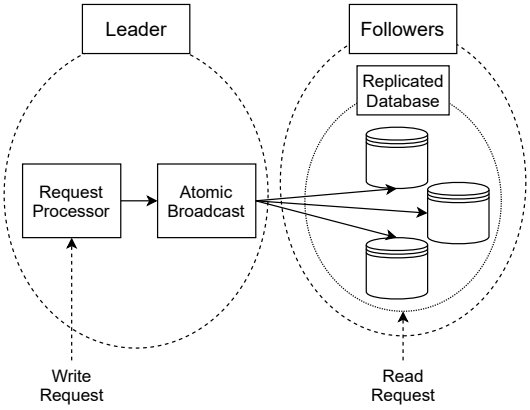


Fig. 3. Illustration of the components within the ZooKeeper service.

The database of ZooKeeper is implemented as an abstraction of a file system. A set of data nodes (znodes) are organized according to a hierarchical name space. The znodes are manipulated by clients via write-request through the ZooKeeper API. Znodes can be either regular or ephemeral. They are not designed for general data storage, but map to abstractions of the client application. These abstractions often contain metadata necessary for the coordination process.

Write operations
Whenever a server receives a request requiring coordination between servers this request gets forwarded to a single server, the leader. Subsequently, the ZooKeeper servers that have replicated database (followers) will be updated via the atomic broadcast protocol using two-phase commit. This guarantees A-linearizability. All requests that update the database are serializable and respect precedence.

Read operations
Read requests are processed locally in ZooKeeper ensuring good read-performance. The read operation is a simple in-memory operation on the replica, having no disk operations or agreement protocols. The drawback of the fast read is that it does not guarantee precedence order and may return outdated values. ZooKeeper implements a cache on the client side via a watch-mechanism. Whenever a ZooKeeper client sends a read request to the server it can set the watch-flag. The watch-flag is a one-time trigger ensuring the client will be notified whenever the resource is updated. This prevents continuous polling to the server and hence enhances fast reads.

Each read request is tagged with an id which indicates the state of the replica the client is connected to. The id ensures that whenever the client reads a value for the second time, the returned value will be as least as recent as the value returned in the first request.

For read requests, ZooKeeper provides the option of *sync*. This implements the functionality of waiting for all write-requests to complete before returning the response on the read request.

3.2.3 Research methods

During the evaluation of the ZooKeeper service, there are a number of things which are investigated. First and foremost is the throughput of the read- and write-requests. ZooKeeper intends to maximize availability and aims at keeping a high throughput, despite failing parts within the system. For the evaluation of ZooKeeper various parts of the system are simulated to fail. During this evaluation, a varying number of servers are examined in addition to varying read:write workload ratios. The throughput of the individual components is measured in addition to the complete system.
The latency of the ZooKeeper service is measured by executing a synchronous create request with 1K of data to the Service and afterwards deleting asynchronously. These create-requests are a form of a write-request. The latency of requests is expressed as the number of completed request per second. The similar Chubby [6] system is used as reference.

3.2.4 Results

For the throughput in the complete system, extreme read-write ratios are explored for a varying number of servers. Hunt et al. [8], show that the number of servers has a negative impact on the write performance, while having a positive effect on the read performance. This is because the read-workload can be distributed over the multiple servers. However, the write-requests go through the atomic broadcast component, which needs to coordinate the update of the data-models on all servers.

By forcing clients to send requests to the leader of the cluster, the read- and write-performance both decrease. For reads this is expected, as they do not take advantage of the distribution of the data-model. This distribution is possible, because of the relaxed consistency guarantees. For writes, the decrease in performance is possibly due to the extra CPU and network load.

Hunt et al. [8], examined the throughput of the ZooKeeper service while simulating various failures within the service. Failures of several followers and the leader are simulated. With failing followers, the service is able to sustain a high throughput. The throughput for a failing follower roughly decreases by the write-requests processed by the failing follower. When the follower recovers, the throughput is increased again. This happens slowly, as clients only switch to a new follower whenever the connection to the previous follower is broken. When the leader fails, the system elects a new leader in approximately 200ms. Its A-linearizable operations allow for quick reconfiguration of the system. During the election of a new leader, the system does not process requests. However, a throughput of zero is not observed, as the sampling period is in the order of seconds.

Latency
The latency of ZooKeeper is also measured by Hunt et al. [8] and is compared to a similar coordination system [6]. Distributing the system over multiple servers has a negative impact on the latency of requests, while increasing the number of followers in the system has a positive effect on the latency. The results show that the latency of the system is on average 1.2ms for 3 servers and 1.4ms for 9 servers. Even though the requests used in the evaluation of Chubby are smaller, the throughput of ZooKeeper is more than 3 times higher compared to the throughput of Chubby.

3.2.5 Discussion

ZooKeeper's distributed data-model design leads to the service having a high-availability. In the results, great throughput is sustained for read- and write-requests while simulating failures in the system. A positive influence on availability of having a distributed replicated data-model is shown.

ZooKeeper implements eventual consistency as it allows the replicas of the data-model to diverge. The writes go through an atomic broadcast protocol, which is responsible for keeping all data-models up-to-date. This gives no guarantee of all replicates being in the exact same state at any time.It is probable that servers do not process the request simultaneously. When nodes/servers fail they are efficiently restarted by the leader using snapshots of a previous state that have been saved on disk. After restoring the snapshot state-changing events that happened after the snapshot are replayed to guarantee all write-requests on the data-model will be executed on the replica.

In terms of which CAP theorem model is employed, both CP and AP can be argued for. Whenever a partition occurs within the system two group arise: the majority partition and the minority partition. ZooKeeper uses majority quorums [13], meaning the the

majority is required to elect a new leader. When the leader is in the majority partition, this partition will continue operating. Otherwise, the majority partition will elect a new leader and will proceed operating. In the minority partition the servers will shut down and revert to leader election. The system will fail to elect a leader and shutdown. Initially the system aims at providing up-to-date information and stops the disconnected server from responding to read requests. The server informs the clients of the unavailability by providing a disconnected error message to the client. ZooKeeper, however, provides the option to connect in read-only mode to the disconnected server.

Combining the partition behaviour to the eventual consistency model ZooKeeper utilizes, we observe that ZooKeeper can be used both as an AP model and as a CP model. An AP model using eventual consistency can be configured by clients using a read-only connection to disconnected servers. One could argue this is not an available system as update operations are cannot be issued. ZooKeeper implements eventual consistency, which counteracts the CP model. Even though consistent reads are not guaranteed, ZooKeeper implements methods that benefit consistency in the case of a partition.

4 Discussion

Amazon claims that by employing strong read-after write consistency, there is no impact on performance [1]. However, this seems unlikely, since there has to be a trade-off to guarantee consistency. In an ideal situation where no partitions occur, it makes sense that there is no trade-off between availability and consistency. In the case that a partition occurs, a decision has to be made between availability and consistency. Google's *Spanner* [5] imposes a similar strong consistent model, which also claims to be both consistent and highly available. While the system itself has major internal differences, there is an explanation why the assumption of having both consistency as availability can be made. Partitions occur extremely rarely, due to it running in a private global network. In the case of a partition, Paxos groups in combination with two-phase commit, strong consistency is still ensured. In the case that Paxos fails due to this partition, the system becomes unavailable. Amazon S3 has to make a similar decision during a partition, where it either forfeits consistency or availability. Since it employs a strong consistent model, it is most likely that it forfeits availability. However, it is unclear whether that is done by restricting operations or other means.

The applications of a distributed system have a major impact on design decisions regarding trade-offs between consistency and availability. A large variety of NoSQL databases exist, which employ a large scala of trade-offs. By researching similar systems to S3 or ZooKeeper, potential overlapping design decisions could be found which result from similar drivers.

The research field of distributed systems is an ever-evolving field, where practices can be outdated or obsolete in a matter of years. The research performed on Amazon S3 turned out to be outdated because of an internal architectural change in S3. Nevertheless, the paper proposed interesting methods and concepts which are generic enough to be used in other contexts. In addition to this research turning out to be outdated, the lack of transparency from Amazon did not make it easier to link the findings to architectural decisions. Due to this missing information, S3 essentially becomes a black-box system, which could affect the accuracy of the research.

5 Conclusion

In this paper we discussed how the BASE model is different from the traditional ACID model. We related these models to the original CAP theorem and have examined nuances of the CAP theorem. We have examined this new interpretation of the CAP theorem by doing a case study of two data models: Amazon S3 and ZooKeeper.

Both the Amazon S3 and the ZooKeeper service provide distributed databases with relaxed consistency guarantees. Both services use an eventual consistency model, partially forfeiting the consistency in the CAP theorem. Consistency is guaranteed after a period of no additional writes. In the Amazon S3 paper this is referred to as the *inconsistency window*. At the end of this window, all replicated instances of the database have processed the write-requests. The same happens within the ZooKeeper service. Write-request are processed via an atomic broadcast protocol leading to consistency whenever all write-requests going through the atomic broadcast protocol are processed.

In both systems we can see the trade-off at work. A replicated database where each replica is kept up-to-date via a write processing engine gives relaxed consistency guarantees, but does guarantee a high availability as the failure of one server results in clients being accommodated at different servers. We found that the Amazon S3 service employs an AP model where it forfeits consistency. However the new S3 model implements a strong consistency model, prioritizing the consistency over availability. As there are no mentions of what happens during a partition we cannot define with certainty whether the model is a CP model or a CA model. For the ZooKeeper service there are arguments for it to be an AP model or a CP model. In the case of a partition, requests are blocked by the disconnected servers, indicating that ZooKeeper prioritizes consistency over availability. However, as the model implements an eventual consistency model, no 100% consistency is guaranteed.

In the new interpretation of the CAP theorem, the vision was that one should not abandon the third property all together. Both systems are partition tolerant, but handle the availability-consistency trade-off differently. Amazon S3 and ZooKeeper both use an eventual consistency model. In this model S3 prioritizes availability, while ZooKeeper offers more consistency guarantees.

6 Future work

A potential extension to this research would be to explore what the differences are between the research on S3 in 2011 and the current implementation of S3. Because of aforementioned assertions made by Amazon regarding the consistency and availability of the system, it appears that the negative aspects of an eventual consistent model have been completely negated. This implies that there is no trade-off to begin with, which voids the whole CAP theorem.

Another extension to this research would be to compare more than two distributed systems. In contrast to S3, ZooKeeper has not switched models and is unchanged in its approach to the consistency-availability trade-off. Since its introduction, ZooKeeper has been adopted by a plethora of Apache applications, due to its proven stability [11]. Etcd is a similar system to ZooKeeper and it would be interesting to see whether the system developed a different consistency-availability trade-off. Another well-known key-value storage system is Redis. Redis has similar challenges regarding the properties of the CAP theorem, like any distributed system. Redis offers multiple distributions which have different perspectives on these properties. For example, it has an implementation called CRDB, which revolves around using CRDTs to improve conflict resolution [9]. This approach of using CRDTs is a novel way of resolving some of the challenges imposed by the CAP theorem.

References

[1] J. Barr. Amazon s3 update – strong read-after-write consistency. Available at `https://aws.amazon.com/blogs/aws/amazon-s3-update-strong-read-after-write-consistency/`, Last visited in March 2021.

[2] D. Bermbach and S. Tai. Eventual consistency: How soon is eventual? an evaluation of amazon s3's consistency behavior. In *Proceedings of the 6th Workshop on Middleware for Service Oriented Computing*, MW4SOC '11.

Association for Computing Machinery, New York, NY, USA, 2011. doi: 10.1145/2093185.2093186

[3] E. Brewer. Towards robust distributed systems. p. 7, 01 2000. doi: 10.1145/343477.343502

[4] E. Brewer. Cap twelve years later: How the "rules" have changed. *Computer*, 45(2):23–29, 2012. doi: 10.1109/MC.2012.37

[5] E. Brewer. Spanner, truetime and the cap theorem. Technical report, 2017. Available at `https://research.google.com/pubs/pub45855.html`, Last visisted in March 2021.

[6] M. Burrows. The chubby lock service for loosely-coupled distributed systems. In *7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2006.

[7] S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, June 2002. doi: 10.1145/564585.564601

[8] P. Hunt, M. Konar, Y. Grid, F. Junqueira, B. Reed, and Y. Research. Zookeeper: Wait-free coordination for internet-scale systems. *ATC. USENIX*, 8, 06 2010.

[9] R. Labs. Active-active geo-distribution (crdts-based) — redis labs. `https://redislabs.com/redis-enterprise/technology/active-active-geo-distribution/`. Last visisted in March 2021.

[10] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski. Conflict-free replicated data types. vol. 6976, pp. 386–400, 07 2011. doi: 10.1007/978-3-642-24550-3_29

[11] I. Sudasingha. A comparison between distributed coordination giants etcd3 and apache zookeeper. `https://imesha.me/apache-curator-vs-etcd3-9c1362600b26`, July 2017. Last visisted in March 2021.

[12] A. Tanenbaum and M. van Steen. *Distributed Systems: Principles and Paradigms*. Pearson Prentice Hall, 2007.

[13] ZooKeeper. Zookeeper internals documentation. `https://zookeeper.apache.org/doc/r3.2.2/zookeeperInternals.pdf`, 2008. Last visisted in March 2021.

A review of Distributed Machine Learning Algorithms

Bedir Chaushi (S4309588)

Abstract—Most machine learning algorithms currently in development, rely on different architectures and ecosystems. The need for performing scalable, reliable, robust computations has led several ML (Machine Learning) algorithms to switch to distributed machine learning architectures. Different software systems that use ML have unique, and different characteristics, such that: time complexity, heterogeneity of systems, and fault tolerance. Various machine learning systems adapt to these needs and perform several tasks. In this article we will discuss for federated Learning and parameter server architecture.
The parameter server architecture is introduced as a milestone for distributed ML. In this model, the parameter server distributes data to worker (slave) nodes and commands the learning process via a master node. Workers run computations locally and send back feedback to the server that aggregates. The Federated Learning (FL) approach gains insight on edge devices that each owns private data, without enforcing any movement of the data (unlike with the parameter server). The devices collaboratively train a model, which is hosted on a centralized server. The server decides which of the devices should participate in the learning round through random selection. Selected devices send their updates to the server after several learning iterations over their private data. Both of the mentioned architectures distribute the system among multiple nodes, although the biggest difference between them is that federated learning trains particular algorithm across multiple edge devices, where parameter server architectures gives its contribute by allocating data among shared data. Federated learning, unlike paramater server, keeps data disestablished, thus it has privacy concerning advantage in comparison with parameter server.
This article gives a general view of mentioned distributed machine learning systems and algorithms. It addresses how distributing machine learning algorithms solve problems derived from the current state-of-art machine learning algorithms and systems from both centralized and distributed approach.

Index Terms—Distributed Machine Learning, ML algorithms, federated learning, parameter server.

1 Introduction

Machine learning is the key for observing valuable data, identifying various patterns, making predictions and handling variety and huge amount of data. However, in order to achieve high performance and to increase the quality of aforementioned factors, high volume of training data and computational power is needed. Since large amount of processing this data has been difficult to handle from locally centralized machines, there is a need for distributing the machine learning workload across multiple machines and turning the centralized system into a distributed one. Thus, Distributed optimization and implication is becoming a requirement for solving large scale machine learning problems. These distributed machine learning frameworks and algorithms present new challenges as different systems hold range of requirements, in which distributed machine learning manner must adapt to it. Different frameworks, architectures, and algorithms, such as federated learning, parameter server approach, are used for achieving distributed manner of solving machine learning problems. These systems typically are designed to improve performance, increase accuracy, and scale to larger input data sizes. Increasing the input data size for many algorithms can significantly reduce the learning error and can often be more effective than using more complex methods [4].

Parameter server maintains the current model and regularly distributes it to the workers who in turn, they make all necessary calculations and send it back to the server. The server then applies all the updates to the central model. This is repeated until the model optimizes and get in desired conditions.

In federated learning, this framework is improved to minimize communication between the server and the workers. Federated learning leaves the training data distributed on the mobile devices and learns a shared model by aggregating locally computed updates, and compression techniques can be applied when uploading the updates to the server.

This paper is intended to give an idea to the reader about how distributed machine learning approach solves current state-of-art issues of centralized approach of implementing machine learning algorithms, pros and cons between different approaches (federated learning and parameter server), and their advantages.

In the beginning, the reader should expect a brief overview of machine learning and distributed machine learning. In the beginning of following section the current state-of-art problem is defined and after, how parameter server solves the problem. Similar structure applies to the following section where the paper gives insight for federated learning. After that, a brief comparison between two mentioned architectures is given, pros and cons between them and gossip learning is introduced as a solution to both parameter server and federated learning. In after-coming section, a comparative study is introduced; current state-of-art of distributing generative adversarial networks using parameter server and how distributing it with federated learning has advantages over previous one. At final section, a conclusion is provided and further research is suggested.

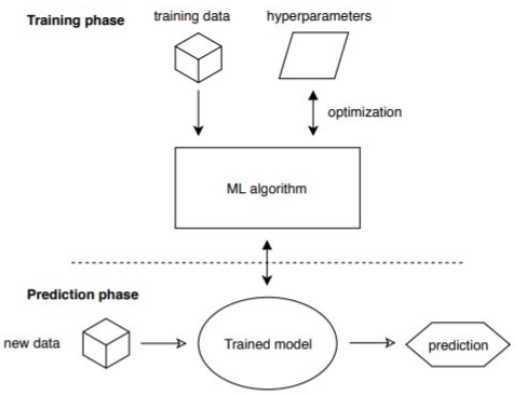


Fig. 1. General Overview of Machine Learning. During the training phase a ML model is optimized using training data and by tuning hyper parameters. Then the trained model is deployed to provide predictions for new data fed into the system [13]

• *Bedir Chaushi is with University of Groningen, E-mail: b.chaushi@student.rug.nl.*

2 Background

In order to provide a better understanding of the subject, we will briefly discuss the following.

2.1 Machine Learning

Machine Learning (ML) algorithms are increasingly being used to analyse datasets and build decision making systems for which an ordinary solution is nearly impossible due to the complexity of the problem. Examples include controlling self-driving cars [1] or predicting consumer behaviour [6] and various other topics.

These systems automatically learn models from training data, and typically consist of three components: feature extraction, the objective function, and learning. Feature extraction processes the raw training data, such as documents, images, and user query logs, to obtain feature vectors, where each feature captures an attribute of the training data.[3] In addition, In ML hyper-parameter serve an important role where its value is used to control the learning process, see fig. 1.

2.2 Distributed machine learning

The parameter server architecture [7] was a milestone for distributed ML, significantly increased speed the computation over a single centralized process. In this model, the parameter server allocates data (also called batches) to workers and coordinates the learning process. Workers run computations locally and send back their errors to the server that accumulates. In turn, workers pull the up–to–date model from the server and iterate until the convergence of the global model is reached. The Federated Learning (FL) [10] approach has a great impact on edge devices that each owns private data, without imposing any movement of the data (unlike with the parameter server). The devices collaboratively train a global model, which is hosted on a centralized server. The server decides which subset of the devices should participate in a learning round through random selection. Selected devices, in turn, send their updates to the server after a number of learning iterations over their private data.

3 Parameter server

The current state-of-art of almost every centralized machine learning problems rely on large amounts of data for training and then for implementation models from it. Terabytes or petabytes of data are trained every day to gain insights and leverage big companies. Trained data consists models of weights that will optimize for error in conclusion for most cases. The number of weights run is in order of billions to trillions. In such big models, both learning and implementation on a single machine is not possible. Since parameters need to be shared and then updated across multiple nodes using which these nodes perform and perfect their computations, these large numbers can be become bottleneck when it comes to sharing. Sharing is expensive in terms of bandwidth, synchronization for sequential ML algorithms, fault tolerance on commodity machines that can have high failure rates up to 10% [7]. Parameter sever proposes a new framework for addressing these challenges of the current state-of-art and building distributed machine learning algorithms.

3.1 Design Ideas

In order to give a solution to challenges mentioned above, Parameter Server proposes the following design requirements:

Efficient communication: The asynchronous communication model does not block computation (unless requested). It is optimized for machine learning tasks to reduce network traffic and overhead.

Flexible consistency models: Consistency helps with reducing the cost of synchronization. It also allows developers to choose between algorithmic convergence and system performance.

Elasticity for adding resources: Allows for adding more capacity without restarting the whole computation.

Efficient Fault tolerance: Given high rate of failures and large amounts of data, allow for quick recovery of tasks in a second or so - if the machine failures are not catastrophic.

Ease of use: Structure the API to support ML constructs such as sparse vectors, matrices, or tensors.[1] [7]

When solving distributed data analysis problems, the issue of reading and updating parameters shared between different worker nodes is a must. The parameter server framework provides an efficient mechanism for aggregating and synchronizing model parameters and statistics between workers.

Two key challenges occur in constructing a high-performance parameter server system:

Communication. The paradigm of updating parameters as key value stores is inefficient. Values are typically small (floats or integers), and the overhead of sending each update as a key value operation is high. As many learning algorithms represent parameters as structured mathematical objects, such as vectors, matrices, or tensors, we can improve the mentioned idea. At each iteration, typically a part of the object is updated. Workers usually send a segment of a vector, or an entire row of the matrix. This provides an opportunity to automatically batch both the communication of updates and their processing on the parameter server and allows the consistency tracking to be implemented efficiently.

Fault tolerance. as noted earlier, is critical at scale, and for efficient operation, it must not require a full restart of a long-running computation. Live replication of parameters between servers supports hot failover. Failover and self-repair in turn support dynamic scaling by treating machine removal or addition as failure or repair respectively.

The parameter server is designed to simplify developing distributed machine learning. The shared parameters are presented as (key, value) vectors to perform linear algebra operations. They are distributed across a group of server nodes. Any node can both push out its local parameters and pull parameters from remote nodes. By default, workloads, or tasks, are executed by worker nodes; however, they can also be assigned to server nodes via user defined functions. Tasks are asynchronous and run in parallel [13].

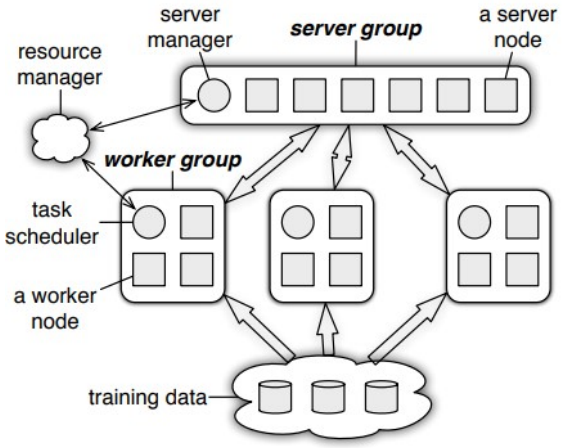


Fig. 2. Architecture of a parameter server communicating with several groups of workers[7]

Parameter Server consists of server groups to facilitate running of multiple algorithms in the system. Parameter server nodes are grouped into a server group and several worker groups as in Fig. 2 and 3. Each server node in the server group is responsible for a partition of the parameter/data. Server nodes communicate with each other to replicate and/or to migrate parameters for reliability and scaling. A server manager is responsible for maintaining the stable view of the server groups. It performs real time checks and assigns ownership of parameters to each server node. Each worker group runs an application. A worker typically stores locally a portion of the training data to compute local

[1] https://medium.com/coinmonks/parameter-server-for-distributed-machine-learning-fd79d99f84c3.

statistics such as gradients. Workers communicate only with the server nodes (not among themselves), updating and retrieving the shared parameters. There is a scheduler node for each worker group. It assigns tasks to workers and monitors their progress. If workers are added or removed, it reschedules unfinished tasks. The parameter server supports independent parameter namespaces. Parameter namespaces can be used for parallelizing work further among multiple worker groups [13]. In addition, same parameter namespace can be shared among multiple groups: a typical example being one group supporting the real-time assumptions, while other worker groups can support the development of the model and updating of shared parameters.

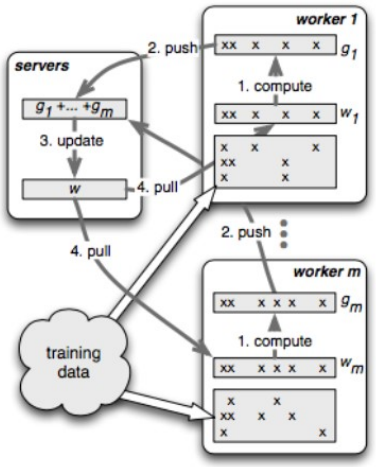


Fig. 3. Each worker only caches the working set of w rather than all parameters[7]

3.2 Key-value parameters

Most common parameter server systems use key-value pairs for communicating the shared parameters. An example of this would be feature-id and its weights. For LDA[2], the pair is a combination of the word ID and topic ID, and a count. The important insight is that values are mostly some linear algebra primitives such as vectors or matrices and it is useful to be able to optimize operations on these constructs. Typical operations are dot product, matrix multiplication, L-2 norms etc. So, keeping the key-value semantics and providing values as vectors, matrices is useful for optimizing most common ML operations.

Weights pulled from the server nodes and gradients are pushed to the server node. Supporting a range-based push and pull would optimize for the network bandwidth usage. Hence the system supports w.push(R, destination), w.pull(R, destination) for pulling the data. In both cases, values corresponding to keys in range R are pushed and pulled from the destination node. Setting R to a single key, gives the simple key-value read-write semantics. Since gradients g share the same keys as that of w, w.push(R, g, destination) can be used for pushing local gradients to the destination (fig 3).

3.3 Sparse Logistic Regression

As a use case to parameter server, this section, provides a machine learning algorithm suited in this architecture, Sparse Logistic Regression taken from [7]. Sparse logistic regression is one of the most popular algorithms for large scale risk minimization [2]. By using this algorithm in parameter server architecture, network traffic is reduced and by implementing the desired algorithm to this architecture, faster result is gained.

[2]https://sebastianraschka.com/Articles/2014pythonlda.html

3.3.1 Problem definition and data

There are collected an ad click prediction dataset with 170 billion examples and 65 billion unique features. This dataset is 636 TB uncompressed (141 TB compressed). The parameter server is run on 1000 machines, each with 16 physical cores, 192GB DRAM, and connected by 10 Gb Ethernet. 800 machines act as workers, and 200 are parameter servers [8].

In the example a distributed regression algorithm [8][9] is used. In this approach only a block of parameters is updated in an iteration. After that, the workers compute both gradients and the diagonal part of the second derivative on this block. Then, the parameter servers themselves must perform complex computation: the servers update the model by solving a proximal operator based on the aggregated local gradients. Fourth, there is used a bounded-delay model over iterations and use a "KKT" (Karush-Kuhn-Tucker)[3] filter to suppress transmission of parts of the generated gradient update that are small enough that their effect is likely to be negligible. Both Systems A and B consist of more than 10K lines of code. The parameter server only requires 300 lines of code for the same functionality as System B. The parameter server successfully moves most of the system complexity from the algorithmic implementation into a reusable generalized component

3.3.2 Results

Three systems are compared while using same objective value. A better system achieves a lower objective in less time. System B outperforms system A because it uses a better algorithm.

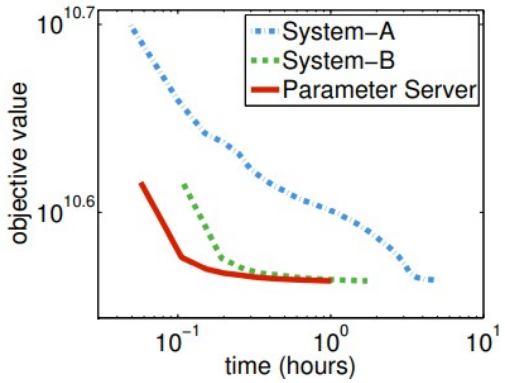


Fig. 4. Convergence of sparse logistic regression[8].

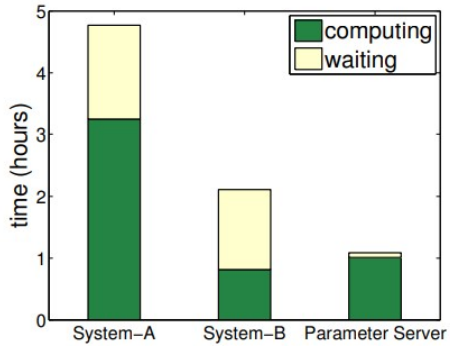


Fig. 5. Time per worker spent on computation and waiting during sparse logistic regression[9]

[3]Karush-Kuhn-Tucker conditions. https://www.cs.cmu.edu/ ggordon/10725-F12/slides/16-kkt.pdf

The parameter server outperforms System B while using the same algorithm. It does so because of the efficacy of reducing the network traffic and the relaxed consistency model. Workers can begin processing the next block without waiting for the previous one to finish, hiding the delay. Workers in System A are 32% idle, and in system B, they are 53% idle, while waiting for the barrier in each block. The parameter server reduces this cost to under 2%. This is not entirely free: the parameter server uses more CPU than System B for two reasons. System B optimizes its gradient calculations by careful data pre-processing. Next, asynchronous updates with the parameter server require more iterations to achieve the same objective value. Due to the significantly reduced communication cost, the parameter server reduces in half the total time. Also, there is significant growth in network reduction per component. 50% traffic of traffic is saved when senders and receivers to cache the keys. This is because both key (int64) and value (double) are of the same size, and the key set is not changed during optimization (see fig. 4 and 5). In addition, data compression is effective for compressing the values for both servers (20x) and workers when applying the KKT filter (6x).[7]

4 Federated Learning

The need of user end data, especially the one derived from devices like tablets and mobile phones, is inevitable [10]. Collecting such data at a central location has become serious issue due to data protection rule. For this reason, there is an increasing interest in methods to improve the current state-of-art, and build a system that leaves the raw data on the device and process it from the device without access to sensitive data.

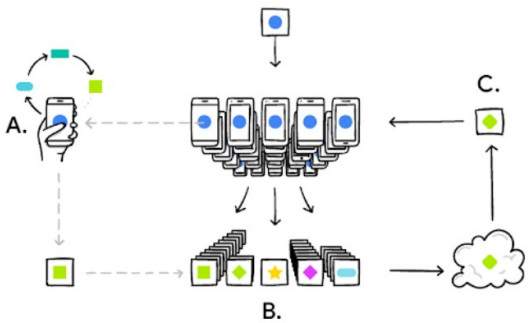


Fig. 6. Phone personalizes the model locally, based on your usage (A). Many users' updates are aggregated (B) to form a consensus change (C) to the shared model, after which the procedure is repeated. From google AI blog.

Federated Learning enables mobile phones to collaboratively learn a shared prediction model while keeping all the training data on device, eviting the ability to do machine learning from the need to store the data in the cloud. This goes beyond the use of local models that make predictions on mobile devices by bringing model training to the device as well. Federated Learning allows for smarter models, lower latency, and less power consumption, all while ensuring privacy. And this approach has another immediate benefit: in addition to providing an update to the shared model, the improved model on your phone can also be used immediately, powering experiences personalized by the way you use your phone.[4]

4.1 Benefits and optimization

Federated learning is key for optimizing and giving a solution to several problems:

Training on real-world data from mobile devices provides a distinct advantage over training on proxy data that is generally available in the data center.

This data is privacy sensitive or large in size (compared to the size of the model), so it is preferable not to log it to the data center purely for the purpose of model training (in service of the focused collection principle).

For supervised tasks, labels on the data can be inferred naturally from user interaction [10].

Federated learning has major privacy advantages compared to data center training data. The information transmitted for federated learning, even the minimal update is necessary to improve a particular model. The updates themselves can and should be temporary. They will never contain more information than the raw training data. Further, the source of the updates is not needed by the aggregation algorithm, so updates can be transmitted without identifying meta-data over a mix network. Federated optimization has several key properties that differentiate it from a typical distributed optimization problem and must be considered as challenges to achieve through implementing such an architecture:

Non-IID. The training data on a given client is typically based on the usage of the mobile device by a particular user, hence any user's local dataset will not be representative of the population distribution.

Unbalanced Some users will make much heavier use of the service or app than others, leading to varying amounts of local training data.

Massively distributed The number of clients participating in an optimization can much larger than the average number of examples per client.

Limited communication Mobile devices are frequently offline or on slow or expensive connections [10].

4.2 Federated Averaging algorithm (FedAvg)

The server aggregates the changes (i.e., weights) received from all the devices. Using a new algorithm called the federated averaging algorithm, the devices train the generic neural network model using the gradient descent algorithm[5], and the trained weights are sent back to the server. The server then takes the average of all such updates to return the final weights. The following pseudocode shows how the federated averaging algorithm works.

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

A typical round of learning consists of the following sequence. A random subset of members of the Federation (known as clients) is selected to receive the global model synchronously from the server. Each selected client computes an updated model using its local data. The model updates are sent from the selected clients to the server. The server aggregates these models (typically by averaging) to construct an improved global model. Of course, the subset selection step was necessitated by the context in which Google originally applied fed-

[4] https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

[5] https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

erated learning: on data collected through millions of handsets in its Android ecosystem.[6].

4.2.1 Experimental results

In the article [10] is demonstrated an experiment on a large-scale next-word prediction task to demonstrate the effectiveness of the approach on a real-world problem. The training dataset consists of 10 million public posts from a large social network, grouped on posts by author, for a total of over 500,000 clients. This dataset is a realistic alternative for the type of text entry data that would be present on a user's mobile device. Limited are each client dataset to at most 5000 words, and report accuracy (the fraction of the data where the highest predicted probability was on the correct next word, out of 10000 possibilities) on a test set of 1e5 posts from different (non-training) authors. These experiments required significant computational resources and. All runs trained on 200 clients per round; FedAvg used B=8 and E=1. Explored are variety of learning rates for FedAvg and the baseline Federative Stochastic gradient descent (FedSGD). FedSGD with n= 18.0 required 820 rounds to reach 10.5% accuracy, while FedAvg with n= 9.0 reached an accuracy of 10.5% in only 35 communication rounds (23× fewer then FedSGD).

5 Parameter Server vs Federated Learning

As it is shown so far, both parameter server architecture and federated learning have huge improvements by introducing a decentralized approach to solving complex issues and building large distributed systems.

The parameter server approach distributes workload among several nodes by synchronizing and controlling them via a master or server node. It does not involve a highly complex architecture and is easy to implement. Whilst this system has an advantage because it has a much simpler setup, its biggest problem arises here. What if data comes from end-users and thus privacy issues must be taken into consideration? As a comparison to parameter server, federated learning comes in handy to answer the aforementioned question.

Federated learning has such architecture where the worker nodes stay at the end device and perform calculations, gain insights into the data without exposing it to the master node, therefore it does not share sensitive data. The server in this situation is responsible to give training models to the workers and workers only give insight from data to the server without exposing the entire sensitive data. As it is explained, the biggest disadvantage of federated learning is that it is a much more complex system to be implemented in comparison with a parameter server. It also involves several computations for resolving which end-user to be taken into consideration. Another disadvantage is that this model needs more communications with the server and this results in network overload and bottleneck. This issue is not happening as often in parameter server as in federated learning.

5.1 Gossip Learning

Another approach where there is minimal infrastructure costs, where data privacy is in high performance and communication is minimal with another node, would be ideal on solving problems related to state-of-art of federated learning and parameter server. These problems are tried to solve using Gossip Learning.

As discussed, performing data mining over data collected by edge devices, most importantly, mobile phones, is of very high interest. A possible solution to this challenge is federated learning. In addition to federated learning, gossip learning has also been proposed to address the same challenge. Gossip learning supports decentralized approach where parameter server is not needed. Nodes exchange and aggregate models directly. The biggest advantage of gossip learning, that favors it against federated learning is that no infrastructure is needed, thus it is significantly cheaper, robust, and both data storing computation giving training model is performed on end device. A key question, however, is how the two approaches compare in terms of performance.

The results of several comparisons done outside the scope of this paper, shows that gossip learning is in general comparable to the centrally coordinated federated learning approach, and in many scenarios, gossip learning outperforms federated learning [5].

6 Comparative study: distributing GAN over FeGAN

So far in this paper, two distributed machine learning systems have been discussed and how they solve current state-of-art problems of centralized systems. In this section, a comparative study will be shown and how federated learning solves current state-of-art problems of parameter server in particular system, generative adversarial networks.

Generative Adversarial Networks (GANs) have had a huge success since they were introduced. GANs belong to the set of generative models. It means that they can produce new content[7]. GANs enable learning the statistical distribution of a target dataset and generating new samples from that dataset. This feature can be used in a wide range of applications such as generating pictures from text descriptions, producing videos from still images, or increasing at will an image resolution.

In simple terms, GANs consists of two main parts: a generator and discriminator. To illustrate, I will take a simple example: in a system that the aim is to generate sample images from a base image, discriminator will produce new sample point from random selected points of an image. After that, discriminator will opt to distinguish real values from derived values. This will loop several times and, in each iteration, sample points derived from generator will be close similar to the original ones as shown in fig. 7. This process will continue until discriminator will not differentiate between sample and derived points and sample images will be produced according to.

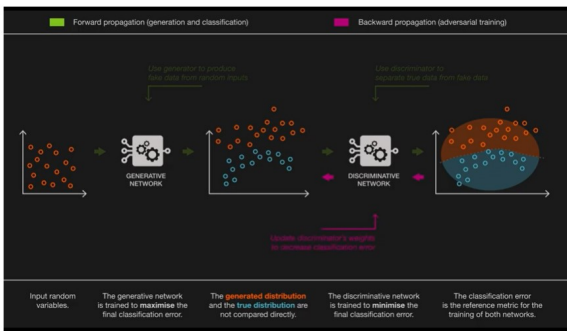


Fig. 7. Taken from [12]

6.1 Distributing GANs

MD-GAN is state-of-art of applying distributed system over GANs. MDGAN controls a single generator, at a central location, and distributes the discriminator across multiple devices. Such an architecture follows the parameter server model, where the server is the generator and the workers are discriminators.

Distributing the training significantly improves the system throughput, but in the other hand, due to the fact that there is only one centralized discriminator, the architecture does not scale, which means adding devices will not improve throughput.

As a solution to the current state-of-art, another architecture for distributed training is Federated Learning (FL), in which training happens on the edge devices that own private data, assisted by a central server. Combining GAN training with this architecture can generate impressive applications on edge devices including text-to image translation. This combination will be named as FeGan [3].

FeGAN reexamines the general Federated Learning (FL) paradigm which has a central server holding global model and a set of computing. Fig. 8 describes the FeGAN architecture. The training model is

[6] An Overview of Federated Learning.https://medium.datadriveninvestor.com/an-overview-of-federated-learning-8a1a62b0600d

[7] https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

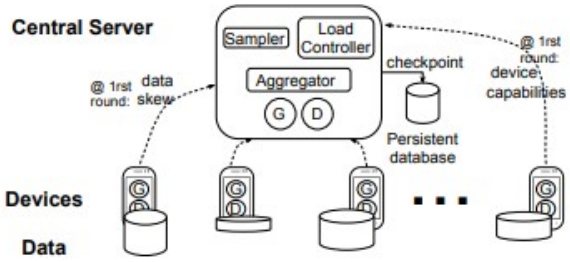


Fig. 8. FeGAN architecture taken from [3]

composed of two neural networks: the generator G and the discriminator D. The server orchestrates the communication load by selecting which devices should contribute to updating the model at a given round. Each device owns a GAN locally, composed of a local generator and a local discriminator. Data stored on each device remains local; only the output of the local computation is sent to the server. Given its local nature, data might be unbalanced and possibly not identically nor independently distributed (non-iid) across devices. FeGAN and MD-GAN are two distributive alternatives to GAN; in paper [3] they are discussed intensely. FeGAN's careful design allows for scaling the training of GANs. While MD-GAN provides better throughput at a small scale (up to 32 devices), FeGAN achieves improvement in throughput with the number of devices (experimented with up to 176 devices) with even a lower bandwidth consumption. FeGAN systems can avoid GAN specific issues that a MD like deployment can encounter. It utilizes the variety of information released by the devices at the beginning of the learning, with regards to their local data.

7 Conclusion and future work

In this paper we discussed for several advantages of distributed machine learning systems and algorithms. In a world that the importance of data is inexcusable, scaling, improving, and gaining insights of data is a must. Distributed ML is a key factor on better computation. Two key systems discussed such that parameter server and federated learning. There is no absolute winner in context of comparison, but these architectures come to ease in specific platforms and systems. Some systems may need abstraction to data and not imposing it. Federating learning is the best choice in this case. Other systems may need key value storing and a distribution between workers and master nodes. In this case parameter server in inevitable.

The contribution of this paper is to give the reader a better knowledge on how distributed machine learning solves or improves current state-of-art problems derived from centralized and distributed machine learning approaches. As future work, I would propose to implement another system, named "Lambda Architecture". This system focuses on scaling large incoming data and making computations, running machine learning algorithms without significant loss of time, by expressing three layers: speed layer, where streams of data are computed; batch layer, where batch works are implemented to compute complex computation over huge amount of data, and serving layer, where two aforementioned layers are merged and a final result is given.

Acknowledgements

I would like to thank the reviewer, Mr. Majid Lotfian Delouee for their valuable feedbacks.

References

[1] M. Bojarski, D. D. Testa, D. Dworakowski, Bernhar, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, , and K. Zieba. End to end learning for self-driving cars. In *CoRR abs/1604.07316 (2016). arXiv:1604.07316 http://arxiv.org/abs/1604.07316*, Nov. 2016.

[2] K. Canini and Sibyl. A system for large scale supervised machine learning. In *Technical Talk*, 2012.

[3] R. Guerraoui, A. Guirguis, A.-M. Kermarrec, and E. L. Merrer. Fegan: Scaling distributed gans. In *Proceedings of Middleware 2020. ACM*, 2020.

[4] A. Halevy, P. Norwig, and F. Pereira. The unreasonable effectiveness of data. In *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, volume 13, pages 1–7, Oct. 1979.

[5] S. Hegedűs, G. Danner, and M. Jelasity. Gossip learning as a decentralized alternative to federated learning. In *Proceedings of Distributed Applications and Interoperable Systems*, pages 74–90, 2019.

[6] A. E. Khandani, A. J. Kim, and A. W. Lo. Consumer credit-risk models via machine-learning algorithms. In *Journal of Banking Finance*, volume 34, pages 2767–2787, Nov. 2010.

[7] M. Li, Andersen, J. W. S. D. G. Park, A. J. Ahmed, A. Josifovski, V. Long, J. Shekita, E. J., and S. B.-Y. Scaling distributed machine learning with the parameter server. In *OSDI*, 2014.

[8] M. Li, D. G. Andersen, and A. J. Smola. Distributed delayed proximal gradient methods. In *NIPS Workshop on Optimization for Machine Learning*, 2013.

[9] M. Li, D. G. Andersen, and A. J. Smola. Communication efficient distributed machine learning with the parameter server. In *Neural Information Processing Systems*, 2014.

[10] H. B. McMahan, M. E., and R. D. H. S. Ar-cas. A communication-efficient learning of deep networks from decentralized data. In *Proceedings of AISTATS*, 2017.

[11] F. A. Narudin, A. F. andNor Badrul Anuar, and A. Gani. Evaluation of machine learning classifiers for mobile malware detection learning. In *Soft Comput*, pages 343–357, 206.

[12] J. Rocca. Understanding generative adversarial networks (gans). In *https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29*, 2019.

[13] J. Verbraeken, M. Wolting, J. Katzy, and J.Klopenburgs. A survey on distributed machine learning. 2020.

Comparison of Workflow Management Tools for Distributed Data Science Applications

Job Heersink and Sytse Oegema

Abstract—With the enormous increase of value and quantity of data, most non-distributed data analytics method will no longer be able to satisfy the needs of the average data scientist. Most of these non-distributed data analytics methods do not scale and will not be able to handle enormous amounts of data in a reasonable amount of time. Therefore, distributed methods for data science applications increase in popularity. Many workflow management tools have been created for the purpose of providing a distributed data science method which is easy to use. The choice between which workflow management tool to use can be quite hard and it is not always apparent what use case fits what tool best. Our goal with this paper is to make this choice easier. In this paper, we will evaluate Dagster, Apache Airflow and Luigi and determine their optimal use case in the field of data science.

Index Terms—data science pipeline, data analysis, big data, software containers, container orchestration

1 Introduction

It is no mystery that data and data analytics play an important role in our society. The quantity of data produced is increasing by the year and it does not seem to be dimming down any time soon. It was recently predicted that in 2020, each individual human being would generate around 1.7 megabytes of new information every second and the total amount of the accumulated digital universe of data would reach a stunning 44 zettabytes [1] .

This tendency can also be referred to as big data and it poses a real challenge within the field of data science. The application of machine learning on big data-sets can require an enormous amount of resources, which most computers are not able to provide alone. That is why there is a significant need for a distributed and scalable data analysis workflow.

One of the main challenges is that this workflow should be able to be distributed over multiple machines, possibly in different locations, to be able to provide the resources necessary for the machine learning algorithms. If necessary, more machines can be added without the need of changing the core of the code.

Another challenge is effective transition between different environments. In some cases, it might be necessary to move the application to another, possibly bigger, cluster to do its work on. Since machine learning technologies can suffer from technical debt and quick changes to its code and structure, it is very hard to predict how the machine learning implementation will perform across multiple machines. We will therefore need a consistent distributed machine learning workflow.

It might sometimes be necessary to analyse and process large amounts of data in real time. In some cases, it might even be disastrous if this continuous process is interrupted by a system fault. Take for example the continuous analysis of stock market data: If the stock prediction system is down, the company will not make any profit. That is why the distributed Machine Learning workflow needs to be fault tolerant as well.

- *Job Heersink is from the Rijks University of Groningen, E-mail: j.g.heersink@student.rug.nl.*
- *Sytse Oegema from the Rijks University of Groningen, E-mail: s.oegema@student.rug.nl.*

reviewed 17 March 2021; accepted 29 March 2021;
For information on obtaining reprints of this article, please send an e-mail to: j.g.heersink@student.rug.nl or an e-mail to: s.oegema@student.rug.nl .

A popular means of introducing scalability and consistency across multiple machines in software engineering is currently the use of containers. Such a container can be seen as a software encapsulation of an operating system process which allows a process to work with private resources including memory and CPU [2]. Data science pipelines can benefit from containers by splitting the pipeline in separate processing steps. Each processing step can be implemented as a container and be distributed over different machines thereby realizing scalability and redundancy. Container orchestration platforms that have been developed for this purpose can manage the different container of the data science pipeline. [2].

However, data scientists should not have to deal with the extra complexity of containers and container orchestration frameworks. Therefore, workflow management tools have been created as an alternative. These tools help data scientists to focus on the data science by abstracting away from the complexity of distributed computing[3]. They provide a platform to specify workflows that can be executed in a distributed fashion over multiple machines. A data science pipeline can be constructed on the platform of a workflow management tool and the scaling and redundancy is handled by the container orchestration framework.

In this paper, we compare three workflow management tools, created specifically for data science applications. The workflow management tools also use containers and container orchestration to achieve the distributed execution of workflows. We evaluate the workflow management tools on four criteria: complexity, clarity, implementation and scope. We compare these platforms with themselves and one platform that was evaluated by Andrea De Lucia and Evi Xhelo[4]. The rest of this paper is structured as follows: First, in section 2, we will discuss the related research to our work. Second, in section 3, we will discuss some background information on the data-science pipeline, containers, container orchestration platforms and the 3 workflow management tools we will be comparing. Third, in section 4 and section 5, we will evaluate these tools on the 4 criteria and compare them. Finally, in section 6, we will conclude our paper and discuss possible future work.

2 Related Work

A great number of studies focus on the general application of container orchestration. The article by Asif Khan: *Key Characteristics of a Container Orchestration Platform to Enable a Modern Application* [2] presents a set of capabilities that a container orchestration platform needs to satisfy in order to perform optimally. The article on *Container-based Cluster Orchestration Systems: A Taxonomy and Future Directions* by Maria Rodriguez et al. [5] presents similar characteristics and elements of importance to container orchestration

frameworks. On top of that, Rodriguez et al. compare a set of container orchestration frameworks based on the defined characteristics. Other research of Emiliano Casalicchio: *Container Orchestration: A Survey* [6] presents a survey of the state of the art in container orchestration technologies, which among other things indicates wide adaptation of container orchestration technologies not only in enterprise applications but also in other areas like data analytics.

The scalability of data science pipelines is a common research topic. The most common programming model for big data analyses is MapReduce which was originally developed by Google[7]. In the MapReduce model data is represented by key-value pairs. Map functions can modify or map the data to a new key-value data format and reduce functions can reduce the number of values per key thus creating a smaller data set[8]. Popular distributed processing frameworks such as Hadoop and Spark use the MapReduce programming model for distributed data analysis. Multiple studies research the possibilities of the MapReduce paradigm in relation to big data processing with the help of these frameworks[9, 10, 11]. However, the usage of containerization in data scinece pipelines is a less frequently researched topic.

In their paper: *Approaches for containerized scientific workflows in cloud environments with applications in life science* [12], Spjuth et al. present 7 scientific workflow frameworks that use container orchestration framework Kubernetes. Only 3 of their 7 frameworks are natively build on top of a container orchestration framework while the others now support a container orchestration framework. De Lucia and Xhelo provide a better comparison between 3 native container orchestration solutions in their *Data Science Pipeline Containerization* [4] paper. They compare Kubeflow, Cloudflow, and OpenWhisk on the criteria; complexity, clarity, implementation, and scope. OpenWhisk, a serverless platfrom that executes function on occurence of events, is not designed specifically for data science pipelines.

In this paper, we would like to extend their research to include 3 new technologies that are specifically designed for data pipeline workflows and compare it with the ones present in the paper by Andrea De Lucia and Evi Xhelo.

3 Background

Distributed data processing emerged over the last year due to the continuously growing demand for data processing and amount of data. This increased usage of distributed data processing raised the necessity of management and orchestration frameworks for data science pipelines. In this section we will provide some background information on data science pipelines, containers, and container orchestration platforms.

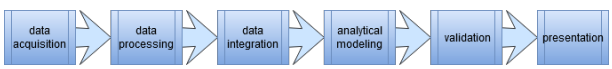


Fig. 1. data-driven analysis steps

3.1 Data Science Pipeline

A data science pipeline simply presents the data processing steps in a data analysis process. While the steps in a data analysis process vary per process, all data analysis process follow a similar pipeline that is graphically represented in figure 1. The steps in figure 1 can be read as follows:

1. Data Acquisition - the data is either collected from one or multiple sources, or continuously generating data.
2. Data Processing - the data quality is improved either by replacing or deleting missing and invalid values.
3. Data Integration - the data is combined and modified to obtain a data format that can be used in the analytical model.
4. Analytical Modeling - the data is processed by an analytical model that produces results. Analytical models can have various purposes such as filtering, predicting, or clustering.
5. Validation - the analytical result is validated based on model or process specific rules.
6. Presentation - the analytical results are presented in human interpretable format.

There are several problems to be tackled before obtaining valuable data science pipeline. The first problem that needs to be tackled is that data should be gathered for the analysis. Moreover, the right data needs to be collected and the data format needs to fit requirements of the specific pipeline. The most important requirement is that the data format should fit the analytical model. Valuable results are only obtained when an analytical model fits the situation. All in all, after a valuable data science pipeline is constructed, designers do not need to worry about compatibility issues when executing the pipeline in the production environment and scalability issues for handling big data.

3.2 Container

A container is a package of software that contains source code together with the dependencies to execute that source code. Containers do not replace virtual machines. Containers share the operating system kernel with the host machine and those are often hosted as a virtual machine[14]. Container images also are significantly smaller then virtual machine images due to their sharing kernel behavior[4]. Additionally, containers provide a private namespace or computing environment in the host kernel, which is particularly useful for handling software dependencies. Furthermore, containers can specify computational resource limits for the software in terms of CPU and memory usage. Multiple publicly open source available containerization software exists. Some examples are Docker[15], Linux Containers(LXC)[16], and containerd[17]. Docker is the most popular container solution available today[18].

Containers are ideally suited for microservices or microservice architectures, because of the small container image size. Microservice architectures aim to distribute a piece of software over modules of microsverices, where a microservice is a ”cohesive, independent process interacting via messages”[19]. The processing steps in a data science pipeline can very well be constructed as separate microservices, because each processing step performs an independent operation and is only connected to the other steps by the input and output of data. Processing steps can be developed as independent microservices that can be executed in their own container. A microservice data science pipeline then consists of a directed input output pipeline of containers.

3.3 Container Orchestration

A simple microservice based software application can easily be managed manually as long as the number of containers remains small. However, manually orchestrating greater numbers of containers becomes quite tedious quite fast. Container orchestration frameworks offer a more sustainable solution, especially for software applications that require scaling based on processing load. Container orchestration frameworks offer numerous features to automate container management and orchestration like; state management, container scheduling, fault tolerance, security, networking, service discovery, continuous deployment, and monitoring[2].

There a number of container orchestration frameworks available today which can either be used on-premise or in the cloud. Kubernetes[20], Apache Mesos[21], and Docker Swarm[22] are examples of open source container orchestration frameworks that can be run on-premise. Cloud providers like Google, Amazon, and Microsoft provide their own container orchestration frameworks. Even though, Kubernetes has originally been developed by Google and is still greatly influenced by Google, big cloud providers support Kubernetes as well[23].

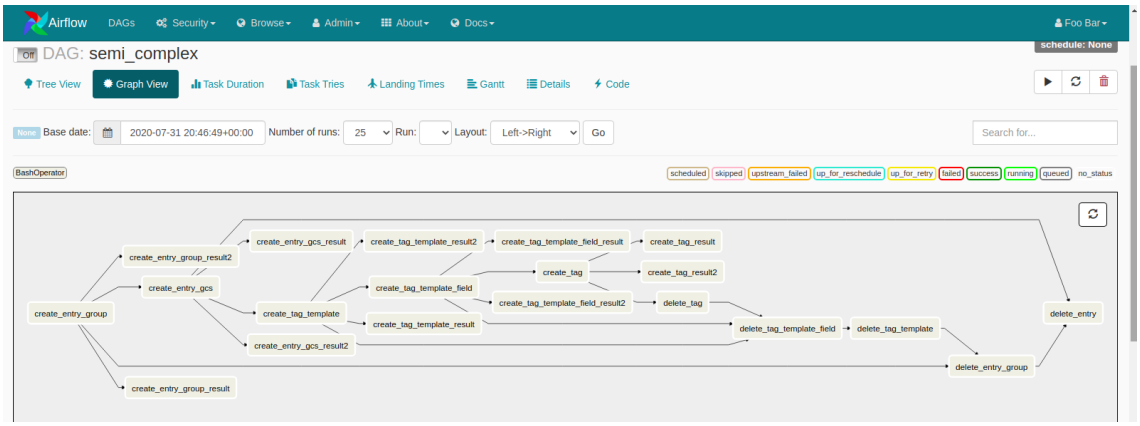


Fig. 2. An example of the UI of apache Airflows[13]

Container orchestration frameworks facilitate a production environment for container based microservice architectures. They provide simple functionality to horizontally scale applications over a large number of resources. Combining this powerful scaling with cloud resource enables software developers to create software that can handle big scale workloads. Data scientists can utilize these features to process large amount of data or big data in parallel. Each individual process step in a data science pipeline can be scaled horizontally if the pipeline is designed according to the microservice principles. Meaning that, a computationally more expensive process step can be scaled more than simple process steps.

A microservice architecture in combination with container orchestration frameworks empower data scientist to use horizontal scaling for each individual process step in the data science pipeline. In a data science pipeline, the analytical modeling process step is computationally more expensive than the data integration process step. More parallel containers can be used for the analytical modeling, for example 4 containers, than for the data integration where for example 2 containers can process the same amount of data. Figure 3, shows an example of how each process step in the pipeline can individually scale.

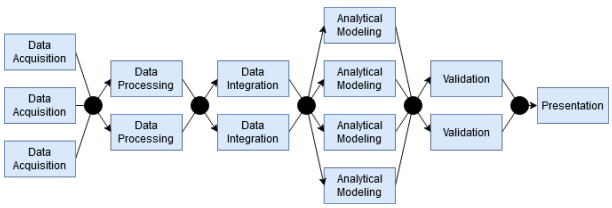


Fig. 3. Data science pipeline example that scales horizontally per processing step

3.4 Workflow Management Tools

workflow management tools are responsible for managing and maintaining fault-tolerant pipelines in a distributed environment. They can be utilized with container orchestration platforms and are able to run the pipeline on the actual environment of the orchestration platforms.

Here, we present some of the popular workflow management tools that can be useful for data science applications. We look at Apache Airflow, Dagster and Luigi, and briefly describe their architecture and implementation. We use similar requirements as Andrea De Lucia and Evi Xhelo used in their paper *Data Science Pipeline Containerization*[4]. However, we extend their 3 requirements with a fourth requirement. That gives us the following platform selection criteria:

1. open source software
2. compatibility with containers
3. compatibility with Kubernetes
4. specifically data science pipeline oriented

We add this final criteria to limit the scope of our research. The aim of this analysis is to compare frameworks that abstract away from the bare level container orchestration by providing data science pipeline specific tools. Andrea De Lucia and Evi Xhelo analyse 3 frameworks in their paper of which only Kubeflow[24] satisfies our 4 criteria. We will take this into account in our comparison of frameworks as well in section 5.

3.4.1 Apache Airflow

Apache Airflow[25] is a platform which allows for the manufacturing, scheduling and monitoring of workflows programmatically using a combination of their own user interface and Python[26]. Here, a workflow is a sequence of processing steps that processes a set of data. Scheduling in Apache Airflow is the planning, controlling and optimizing of the processing steps. Airflow allows the user to create a Directed Acyclic Graph (DAG), a directed graph with no directed cycles, representing the workflow with the help of Python scripts. This graph will show each task's relationships and dependencies. Additionally, graphs provide the linking of concurrent processing steps of the particular data science pipeline. An example of the directed acyclic graphs created by Apache Airflow can be seen in figure 2. Each node represents a specific processing step and each edge represents a dependency. For instance, before `create_entry_gcs` can be executed, `create_entry_group` needs to have finished first. Tasks without dependencies to one-another can execute in parallel or distributed over multiple machines.

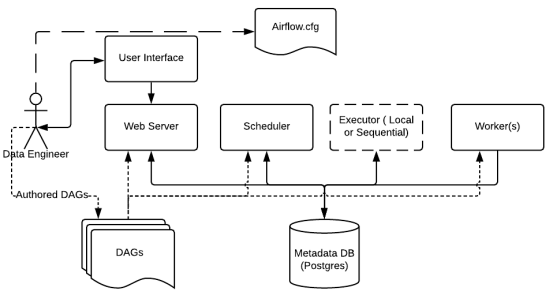


Fig. 4. Basic Airflow architecture

The basic architecture of Airflow, which can be seen in figure 4, consists of the following components: First, we have the **DAGs**, which contain the Python code and represents the data pipeline to be run by

Airflow. These files are stored in the location specified by the config file `Airflow.cfg`. The **user interface** facilitates the creation and execution of the DAGs and connects to the **web service** to communicate with the other components of the system. Then **the scheduler** monitors the DAGs and triggers the tasks whose dependencies have been met. It continuously checks the metadata database, which contains the status of all tasks, to see which task can be executed next. Finally, **the executors** are responsible for executing the different tasks. They interact with the scheduler to retrieve information on the resources that are needed to run the task. There are many available sorts of executors. Some may run locally, like the Sequential Executor and the Local Executor, but some can also be run distributed, like the Kubernetes Executor.

3.4.2 Dagster

Dagster[27] is a data orchestrator designed for machine learning, analytics and "Extract, Transform, Load"(ETL) [28]. Dagster is quite similar to Airflow, it also uses a Directed Acyclic Graph to represent the pipeline of an application and it uses Python to define the functionality of the pipeline. Dagster's terms for a workflow is a **pipeline** and their processing steps which contain the actual functionality with defined input and output are called **solids**. An important quality of Dagster that makes it different from other workflow systems is how it defines the dependencies between solids. Namely, dependencies are defined as ("solid A requires a particular output from solid B") and not just ("solid A runs after solid B"). An example of such a dependency can be seen in figure 5. This allows for the dependency on data rather than the termination of other solids.

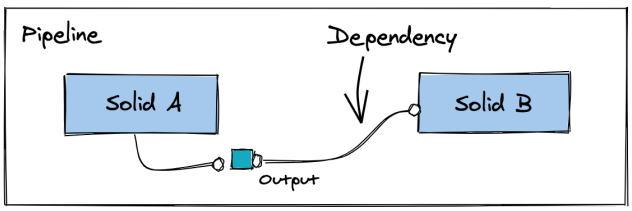


Fig. 5. Dagster pipeline example[27]

Dagsters architecture is relatively straightforward. A solid runs in isolation and continuously streams data-aware events to the Dagster infrastructure, which contains the core of Dagster, the database and the event log. This instance in turn communicates with the Dagster assets manager and Dagster's live monitoring to give the user insight into how the execution is going. An sketch of this architecture can be seen in figure 6.

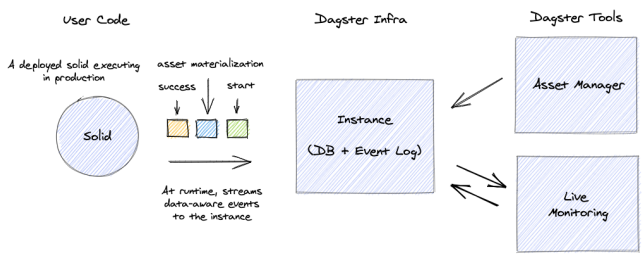


Fig. 6. Dagster architecture[29]

The system itself supports a large variety of deployments. ranging from local, Celery and Docker to large scale clusters like Kubernetes. It also allows for the deployment of the pipeline on Airflow by compiling the pipeline to a format that can be understood by a third-party scheduling system.

3.4.3 Luigi

Luigi [30] is a Python package that allows for the building of complex pipelines of batch jobs[31]. It handles dependency resolution, visualization, workflow management, handling failures and command line integration. It works much the same way as Apache Airflow and Dagster, and also utilizes a dependency graph to know what task to execute next. The main difference between Luigi and other systems is the size of Luigi itself. It is quite small and offers just a bit more than the basics to create a pipeline for data science applications. It does not have a fancy UI, it is just a web page with a simple node graph that shows the execution status.

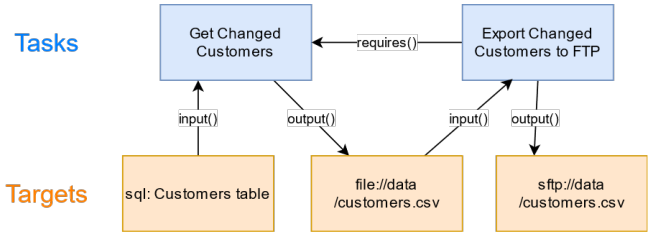


Fig. 7. Luigi architecture

The main architecture of Luigi consists of **Tasks** and **Targets**. A target is usually a file outputted by a task and a task is the executed Python code that consumes targets generated by other tasks. So, it is as simple as a task that generates a target, and when that target is available, another task start working with it. In figure 7, you can see an example of the workings of Luigi. Here, the targets can be anything, ranging from a .csv file, and SQL table or a file on a distributed file system.

Luigi does have the shortcoming of not being able to be applied on near real-time pipelines or continuously running processes, since the focus is on batch processing. It is also not as scalable as other alternatives. Although it can be used with Kubernetes, the creators did mention that Luigi is not meant to be scaled beyond more than tens of thousands of jobs [30].

4 Analysis

In this section, we evaluate the different workflow management tools. We use the same evaluation criteria as presented by Andrea De Lucia and Evi Xhelo in their paper *Data Science Pipeline Containerization*[4] to be able to compare our evaluated systems with theirs. They evaluated the systems using the following criteria:

- **Complexity**: Represents to what extend the system is easy to use from the start and practical in the field of data science.
  - *key factors*: Documentation, community and learning curve
- **Clarity**: Represents to what extend the application created with the platform is easy to understand, debug and reproduce.
  - *key factors*: UI and architecture
- **Implementation**: Represents to what extend development is made easier and how powerfull the tool can be.
  - *key factors*: type of programming languages. implemented technologies and scalability
- **Scope**: Represents to what extend the system supports different frameworks, programming languages and operating systems.
  - *key factors*: available plugins and support for external technologies.

We will summarize the evaluation of each workflow management tool in a table by assigning a value to each criteria. These values can be '- -', '-', '+' and '++', meaning terrible, poor, good and great respectively.

4.1 Apache Airflow

Here, we evaluate the Apache Airflow platform with respect to data science applications. We evaluate this system on complexity, clarity, implementation and scope.

- *Complexity*: Although Airflow is a very powerful tool, it has a very steep learning curve. In addition to that, it seems that the documentation is a bit lacking. For example, there is hardly any information on how to setup your Kubernetes environment to work with Airflow. However, the documentation of the scheduling of DAG runs for instance is very clear[32]. The documentation emphasizes that scheduled jobs are executed for the first time only after the scheduled interval has elapsed[13].

  Also, Airflow does have a very active community. If you are experiencing a specific problem, chances are that someone has posted a solution online that can help you out.

- *Clarity*: Although the documentation is not as extensive as one could have hoped, the UI is rather simple and understandable. It gives a good overview of the underlying pipeline, which makes debugging easier.

- *Implementation*: Airflow uses Python to define workflows[26]. This is lowers the learning curve seeing that Python is a regularly used language by data scientists.

- *Scope*: Airflow has a wide variety of execution modes that make use of other technologies like Celery, Kubernetes and Mesos. It also supports a number of plugins, which adds functionality to the UI, and operators, which introduces a wide range of connectors to external systems like databases, execution engines, and cloud providers.

4.2 Dagster

Here, we evaluate the Dagster platform with respect to data science applications. We evaluate this system on complexity, clarity, implementation and scope.

- *Complexity*: Dagster is a very powerful tool, but can be quite complex to work with. The software is quite new, the documentation[27] is not detailed and it is also not as popular as some other alternatives. This makes it more difficult to find solutions to problems you might encounter while working with this system.

- *Clarity*: Dagster does come with a UI[29] to give an overview of the execution of the program, however due to the powerful nature of Dagster the UI is not as simple to work with as other alternatives.

- *Implementation*: The fact that Dagster uses Python to define your workflow is very beneficial to data scientist, because this language is very popular in that field. In addition to that, Dagster creates dependencies between processing steps based on necessary data instead of process termination. Processing steps start processing data as soon as the first data arrives from the previous processing step.

- *Scope*: Dagster provides support for a large number of external systems. The main ones are of course Celery, Docker and Kubernetes, but Dagster also provides support for working with Airlfow, Amazon Web Services (AWS) and Google Computing Services.

4.3 Luigi

Here, we will evaluate the Luigi data orchestration library with respect to data science applications. We will evaluate this system on complexity, clarity, implementation and scope.

- *Complexity*: Luigi is simple and relatively easy to use. The documentation[30], although not very extensive, should tell you everything you need to know. Because Luigi is not used as much as alternatives, the community is not as active as other systems. This makes it hard to find solutions to problems you might encounter while working with this library.

- *Clarity*: Luigi does come with a UI, but it is nowhere near as sophisticated as others. As depicted in the documentation[30], the dependency graph itself is just a plain webpage with a colored nodes and the visualiser page can be a bit of a hassle to work with as well. Luigi can however, because of its simplicity, provide a decent CLI tool to work with.

- *Implementation*: Similar to Airflow, Luigi uses Python to define workflows. Thus, lowering the learning curve for data scientists that have used Python before. Although Luigi can be used in combination with Kubernetes, one of its main drawbacks is the scalability of Luigi. Namely: Luigi cannot be used in a distributed manner. Luigi works well with small batch jobs, but when the quantity of tasks reach a certain point, Luigi will begin to struggle. As mentioned in the documentation[30]: "While you can probably schedule a few thousand jobs, it's not meant to scale beyond tens of thousands".

- *Scope*: Luigi provides support for an average number of external systems. The main technologies in the field of distributed datascience, like Python, Docker and Kubernetes, are supported and Luigi seems to provide some libraries for Spark and Hadoop as well[30].

5 Comparison

The frameworks discussed in this paper: Airflow, Dagster, and Luigi do not provide an exhaustive list of available frameworks that satisfy our research criteria. Nevertheless, we find it important to present a clear and structured comparison of these frameworks. We select the best framework for each of the evaluation criteria:

- *Complexity*: Luigi offers a very minimal and straight to the point user interface. This makes Luigi simpler and easier to use than the other frameworks.

- *Clarity*: Dagster provides a very powerful user interface. It provides detailed overviews of the pipeline execution. On top of that, Dagster provides numerous example setups in their documentation.

- *Implementation*: Dagster combines code with drag and drop functionality in their user interface to create a data science pipeline. Their powerful user interface shows detailed overviews of the pipeline execution and clearly indicates dependencies between pipeline execution steps.

- *Scope*: Airflow is the most feature rich framework in this paper. It supports a lot of different technologies and systems and its pipeline can even be converted to other orchestration platforms. In addition to that, in Airflow more complex structures can be created due to the direct interaction with Kubernetes.

Luigi is the simplest of the 3 frameworks analysed in this paper, while Airflow and Dagster offer more complex options. Because Luigi is relatively simple, it is ideal for data scientists starting to look into containerization of their data pipeline or people that are new in this area. Luigi should however not be used in large applications, since Luigi provides very limited scalabe capabilities. In any case, Airflow or Dagster are more suitable for more experienced data scientists and more complex data science pipelines. People that want to consider all the operating settings of their pipeline might benefit from Airflow, where people that can work with generic settings might benefit from Dagster.

Since, the same evaluation criteria were used in this paper as in the paper of Andrea De Lucia and Evi Xhelo [4], we also compare our results with theirs. However, we cannot compare the results directly because we introduced a fourth criteria on the frameworks. In this paper, only frameworks that are specifically data science oriented are considered. Only one out of the three frameworks evaluated by De Lucia and Xhelo, Kubeflow, satisfies this criteria. Kubeflow similarly to Airflow directly orchestrates the data science pipeline on Kubernetes and abstracts away from the containerization details. Naturally, the architecture and impelementation of Kubeflow differs from Airflow, but in terms of functionality they seem comparable. Therefore, we tied the best score of Airflow with Kubeflow as displayed in table 1.

	Complexity	Clarity	Implementation	Scope
Airflow	-	++	+	++
Dagster	- -	+	++	+
Luigi	++	-	- -	+
Kubeflow	-	+	++	-

Table 1. Comparison of the platforms

6 Conclusion

In conclusion, we have shown and evaluated 3 frameworks on their complexity, clarity, implementation and scope. Although there is no definitive winner, we have pointed out some of the aspects of each platform that would be very beneficial in some use-cases. We have shown that, Airflow or Dagster are more suitable for large and complex applications where more experienced data scientists are needed. In addition to that, we have shown that Luigi is not designed for large scale applications, but because of its simplicity, is a good candidate for relatively inexperienced data scientists. We also shortly discussed kubeflow, one of the platforms evaluated by Andrea De Lucia and Evi Xhelo[4], and included that in the summary of the evaluation.

6.1 Future Work

Future work into containerization frameworks for data science pipelines can look into multiple directions. More containerization framework specifically oriented towards data science pipelines can be evaluated based on similar criteria as used in this paper. Also, a clearer distinction could be made between frameworks that directly use an container orchestration framework for their pipeline, and frameworks that export their pipeline to another pipeline framework. Finally, future research could executing the same pipeline on different frameworks to evaluate the performance of the workflow management tools.

Acknowledgements

The authors wish to thank Mostafa Hadadian for reviewing this paper.

References

[1] Femi Osinubi. *Data the new Smart*. PricewaterhouseCoopers Limited. 2018.

[2] A. Khan. “Key Characteristics of a Container Orchestration Platform to Enable a Modern Application”. In: *IEEE Cloud Computing* 4.5 (2017), pp. 42–48.

[3] Rafael Ferreira da Silva et al. “A characterization of workflow management systems for extreme-scale applications”. In: *Future Generation Computer Systems* 75 (2017), pp. 228–238.

[4] Andrea De Lucia and Evi Xhelo. “Data Science Pipeline Containerization”. In: p. 39.

[5] Maria A. Rodriguez and Rajkumar Buyya. *Container-based Cluster Orchestration Systems: A Taxonomy and Future Directions*. 2018.

[6] Emiliano Casalicchio. “Container Orchestration: A Survey”. In: *Systems Modeling: Methodologies and Tools*. Ed. by Antonio Puliafito and Kishor S. Trivedi. Cham: Springer International Publishing, 2019, pp. 221–235. ISBN: 978-3-319-92378-9.

[7] Kyong-Ha Lee et al. “Parallel data processing with MapReduce: a survey”. In: *AcM sIGMoD Record* 40.4 (2012), pp. 11–20.

[8] Jeffrey Dean and Sanjay Ghemawat. “MapReduce: simplified data processing on large clusters”. In: *Communications of the ACM* 51.1 (2008), pp. 107–113.

[9] Jyoti Nandimath et al. “Big data analysis using Apache Hadoop”. In: *2013 IEEE 14th International Conference on Information Reuse & Integration (IRI)*. IEEE. 2013, pp. 700–703.

[10] Harshawardhan S Bhosale and Devendra P Gadekar. “A review paper on big data and hadoop”. In: *International Journal of Scientific and Research Publications* 4.10 (2014), pp. 1–7.

[11] Matei Zaharia et al. “Fast and interactive analytics over Hadoop data with Spark”. In: *Usenix Login* 37.4 (2012), pp. 45–51.

[12] Ola Spjuth et al. “Approaches for containerized scientific workflows in cloud environments with applications in life science”. In: (2020).

[13] apache. *Apache airflow newcomer docs*. URL: https://airflow.apache.org/blog/apache-airflow-for-newcomers/ (visited on 02/18/2021).

[14] Carl Boettiger. “An introduction to Docker for reproducible research”. In: *ACM SIGOPS Operating Systems Review* 49.1 (2015), pp. 71–79.

[15] Adrian Mouat. “Using Docker: Developing and Deploying Software with Containers”. In: 2015.

[16] Shashank Mohan Jain. *Linux Containers and Virtualization: A Kernel Perspective*. Apress, October 2020.

[17] H. C. Lee G. N. Schenker H. Saito and K. C. Hsu. *Getting Started with Containerization*. Packt Publishing, March 2019.

[18] Claus Pahl. “Containerization and the paas cloud”. In: *IEEE Cloud Computing* 2.3 (2015), pp. 24–31.

[19] Nicola Dragoni et al. “Microservices: yesterday, today, and tomorrow”. In: *Present and ulterior software engineering* (2017), pp. 195–216.

[20] Joe Beda Brendan Burns and Kelsey Hightower. *Kubernetes: Up and Running: Dive Into the Future of Infrastructure*. O’reilly, September 2017.

[21] Dharmesh Kakadia. *Apache Mesos Essentials*. Packt Publishing, June 2015.

[22] Fabrizio Soppelsa and Chanwit Kaewkasi. *Native Docker Clustering with Swarm*. Packt Publishing, December 2016.

[23] David Bernstein. “Containers and cloud: From lxc to docker to kubernetes”. In: *IEEE Cloud Computing* 1.3 (2014), pp. 81–84.

[24] 2018-2021 The Kubeflow Authors. *Kubeflow*. URL: https://www.kubeflow.org/ (visited on 02/20/2021).

[25] apache. *Apache airflow*. URL: https://airflow.apache.org/ (visited on 02/18/2021).

[26] Bas P. Harenslak and Julian Rutger de Ruiter. *Data Pipelines with Apache Airflow*. Manning publications, 2021.

[27] dagster-io. *dagster*. URL: https://dagster.io/ (visited on 02/19/2021).

[28] Software Engineering Daily. *Dagster with Nick Schrock [transcript]*. 2019.

[29] Nick Schrock. *Dagster: The Data Orchestrator*. URL: https://medium.com/dagster-io/dagster-the-data-orchestrator-5fe5cadb0dfb (visited on 02/19/2021).

[30] The Luigi Authors Revision. *Luigi*. URL: https://luigi.readthedocs.io/en/stable/ (visited on 02/19/2021).

[31] Anuj Kumar. *Architecting Data-Intensive Applications*. Packt Publishing, July 2018.

[32] Apache Airflow. *Scheduler*. URL: https://airflow.apache.org/docs/apache-airflow/stable/scheduler.html.

Visual Object Detection

Pooja Gowda S4410963, Ajay Krishnan S4618165

Abstract — Visual object detection is a class of computer vision and image recognition. The urge to identify an object accurately with less computational time offers several methods. Hence, a class of neural networks that uses a region-based convolution neural network (R-CNN), Fast R-CNN, Faster R-CNN, and Single Shot Multibox Detector (SSD) are proposed. Every new proposal is an upgraded version from the existing method that provides its challenges and optimized solutions. Further on, You Only Look Once (YOLO) is proposed for accurate real-time detection and followed by the upgraded versions of YOLO to the state-of-art object detectors like YOLOv4 and PP-YOLO. This paper intends to discuss every method proposed in detail, and throwing some light on future enhancements.

Index Terms — Paddle Paddle-YOLO.

1 INTRODUCTION

The objective of this research divides into two branches. The first is to understand the existing and proposed methods approaches. Another being to know the domain that requires attention in each system and the future enhancements required. The first improvised version of the convolutional neural network is R-CNN [1]. The CNN features combine with region proposals to boost the stagnant performance from several years. This method requires training a limited amount of labelled data and localizing objects with a deep neural network. To achieve this, it works within a paradigm “recognition using regions" introduced by Gu et al. in [12]. The model here uses unsupervised pre-training due to the deficit amount of labelled data. This method proposes to show an efficient paradigm that learns higher capacity CNNs when the data is less. This method successfully manages to increase mean average precision (mAP) to 58.5% from 30.5% on PASCAL VOC 2007 [7]. The mAP score is the mean average precision values of all the classes and Intersection-over-Union (IoU) thresholds. Later, an improvised approach is proposed with the name Fast R-CNN [2]. As the name suggests, the intended method provides faster and better object detection. How is this achieved?

Fast R-CNN uses a single-stage algorithm to train. The classification of object proposals and refining of spatial locations occur at one step.

This model uses a combination of R-CNN and Spatial Pooling Pyramid network (SPPnet) [13]. Hence, helping in performing the detection in one single stage. This method achieves mAP 65.7% on VOC 2010 and 2012, mAP 66.9% on VOC 2007 [7]. Surprisingly, the Faster R-CNN [4] accomplishes to get mAP 73.2% on VOC 2007 and 70.4% on VOC 2012 [14]. Where did the previous method lack?

In Fast R-CNN, the congestion point was in a network that generates regional proposals. The time taken in this stage was similar to the classifying network. To work on this area Region Proposal Network (RPN) is used. In this approach, an image loads as input and generates a collection of rectangular object proposals as an output with an objectness score.

- *Ajay Krishnan is a student at Rijksuniversiteit Groningen,*
- *E-Mail: a.krishnan.2@student.rug.nl*
- *Pooja Gowda is a student at Rijksuniversiteit Groningen,*
- *E-Mail: p.gowda@student.rug.nl*

The mentioned methods have rapid improvements in terms of accuracy and speed. It is highly computationally exhaustive while working on embedded systems and high-end hardware. SSD does not continuously re-sample features for bounding box predictions. This method gains mAP 74.3% from 73.2% of Faster R-CNN on VOC 2007 [7]. The removal of the stage where the bounding box proposals and the following features are re-sampled creates a significant improvement.

All methods discussed above process an image regressively to detect the objects. In the case of YOLO [5], the system needs to look at the image just once and generates the bounding boxes necessary for object detection in real-time. The high confidence score of a bounding box denotes the presence of an object defined within. Even YOLO comes with few limitations and the primary one being is, the method can detect only a single object inside one grid cell. YOLOv2 [6] and YOLOv3 [9] are proposed to improve the accuracy and speed of the existing method. In the present stage, YOLOv4 [10] and PP-YOLO [11] are considered state-of-the-art object detection methods. YOLOv4 is provided with supplementary assistance by the Darknet. This method obtains 43.5 % of AP on the COCO dataset [10] with a speed of 65 FPS (frames per second) on Tesla V100(GPU). Lastly, PP-YOLO is proposed recently with a tremendous improvement in mAP of 45.2% from 43.5% of YOLOv4 on the COCO data set [11]. This is the quickest and most accurate method in the present.

The results exceed all the existing methods accuracy and speed. All these methods worked in different features while improving one constituent of a model at one proposal. Each improvised model worked on the previous method drawbacks and created an efficiently performing state-of-the-art.

2 VISUAL OBJECT DETECTION METHODS

Visual object detection is a methodology introduced to identify the objects within the image that can be stored or streamed in real-time. This process uses a class of neural networks called Convolutional Neural Network (CNN) to detect the objects. In “Return of the Devil in the Details: Delving Deep into Convolutional Nets” [8] by K. Chatfield et al. The three strategies used to understand the flow of the detection are discussed in detail. It concludes with a remark that the fine-tuning using deep representation and linear SVM of the input will improve the performance further. Hence, improvised versions of object detection methods were introduced.

2.1 R-CNN

Detecting an object method that was in use, the Convolutional Neural Network (CNN) failed to improve its overall performance in the past few years. Better performing models proposed included ensemble systems with multiple low-level features of an image having high-level context. Several methods proposals came to light to increase existing mean average precision. One of them was R-CNN (Regional Convolutional Neural Networks), a region with CNN features [1]. The recognition happens in several stages downstream. Hence, this suggests that there is a presence of hierarchical processes to compute features for classifying the objects. CNN was in high requirement in the 1990s, then the usage gradually decreased. Krizhevsky et al. reintroduced CNNs in 2012, and this time with improved accuracy on ImageNET Large Scale Visual Recognition Challenge (ILSVRC) [15].

To improve the performance, localizing the objects and training the model with higher capacity was required. A sliding window detector as an alternative has been in the presence for less than two decades or so. This method is constrained to objects such as faces, pedestrians. To sustain high spatial resolution, CNN has two pooling and convolutional layers. But here the CNN localizes the objects by operating within the paradigm instead. While testing, this method produces approx. 2000 category-independent region proposals for each input image, fetches a fixed-length feature vector produced by the proposal, and finally classifies every region proposal using a CNN using a linear Support Vector Machine (SVMs). The lacking of labelled data caused a challenging situation. The proposed solution is to use unsupervised pre-training and accompanied by supervised fine-tuning. The following sections consist of this information in detail.

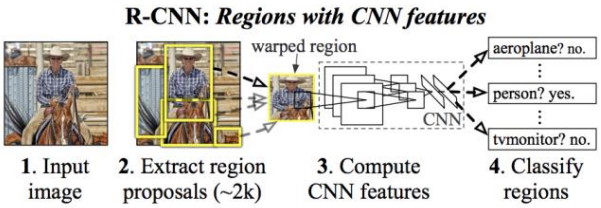


Figure 1: Warped training samples present in bounding box [1].

As per **Figure 1**, this whole process consists of three components. The first one is about the generation of category-independent region proposals. Then, a feature vector of a constant length is extracted per region. Lastly, this is a set of class-specific linear SVMs.

2.1.1 Designing the Components:

1) Region Proposals: The selective search method helped to generate region proposals. Hence, enabling a measured comparison with the previous detection results.

2) Feature Extraction: From every region proposal, around a 4096-dimensional feature vector is extracted. Further, The RGB and a mean subtracted image of dimension 227 x 227 is propagated forwardly through five convolutional layers and two consecutive layers. Initially, the image data in that region gets converted into a form that fits with the CNN having a pixel size of 227 x 227. Further, All the pixels are closely warped within a boundary box irrespective of their size and aspect ratio of the candidate region.

2.1.2 Detection during the Testing:

While testing the method, the selective search runs over the input image by extracting approximately 2000 region proposals. Each proposal warped and forward propagate it using CNN, resulting in reading the features from selective layers. Score each class concerning each extracted feature by SVM. Lastly, greedy non-maximum suppression is applied. Here, a selected region is removed when Intersection-over-Union (IoU) overlaps with an area consisting of a higher score with respect to the learned threshold.

2.1.3 Training:

In supervised pre-training, CNN is separately pre-trained on an extra data set (ILSVRC 2012) [16,17] with image-level annotations or bounding box labels.

In domain-specific fine-tuning, Stochastic gradient descent (SGD) is applied to train the CNN parameters using just the warped region proposals from VOC. All the region proposals are considered >= 0.5 IoU [16,17] overlap with a ground-truth box as positives for a selected box class keeping the rest are negative.

2.1.4 Object category classifiers:

If an image region consisting of an object is a positive example. While a background region that has no object at all is considered as a negative example. When the region of an image with objects partially present within the bounding box, it is difficult to label such images. The challenge faced here is resolved using an IoU overlap threshold, values under this threshold are set to negative.

2.2 FAST R-CNN

Fast Region-based Convolutional Network (Fast R-CNN) is a proposed object detection method [2]. This method is an upgraded form of R-CNN intending to have better accuracy and speed. The challenges faced are always at two stages of object detection: 1) A large number of proposals is processed 2) These candidates give approximate localization this should be refined further on. In this method, the refined spatial locations are classified along with object proposals in a single-stage algorithm.

This method results in training a deep detection network faster than SPPnet by obtaining high accuracy when compared on PASCAL VOC 2012 resulting in a 66% mAP value [2]. R-CNN and SPPnet both have their disadvantages e.g. training in R-CNN is multistage, expensive in terms of space and time, Object detection takes a longer time than expected. In SPPnet, the fine-tuning algorithm fails to update the convolutional coming before the spatial pyramid pooling.

In Fast R-CNN, the network takes a complete image as an input and a set of object proposals. Further, A Region of Interest pooling (RoI) layer extracts a constant length feature vector of a feature map per object proposal. Then, the feature is connected with a series of fully connected (fc) layers, this gets divided into two more branches, one estimates softmax probability with respect to K objects and background classes and the other produces four numerical real-values for every K number of object classes [2].

Fast R-CNN architecture consists of four sections as following: 1) ROI pooling layer 2) Initialization from pre-trained network 3) Fine-tuning for detection 4) Scale invariance.

As per **Figure 2**, every section runs sequentially. Firstly, in the ROI pooling layer, the features from a valid region are converted to a tiny feature map with a set spatial range of H x W (e.g., $7 x 7$), H and W are layer hyper-parameters which is free of any specific RoI. Then with three pre-trained ImageNet models, this method is executed. Here each model consists of five max-pooling layers, while around five to 13 convolutional layers.

While fine-tuning, this method uses stochastic gradient descent (SGD), where the batches are hierarchical in nature, it then uses a streamlined training process. In the first stage of fine-tuning, softmax

and bounding box regressors are optimized rather than training a softmax classifier.

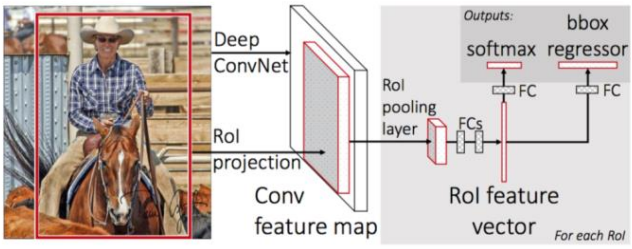


Figure 2: Fast R-CNN Architecture [2], .

Finally, in scale invariance object detection, two ways are traversed: 1) Brute force learning and 2) Using image pyramids. The first approach processes each image at a pre-defined pixel size during training and testing. In the second one, the image pyramid is utilized to nearly scale-normalize every object proposal received. Once the network is fine-tuned, detection takes place. Here, the network takes an image as input and a list of object proposals R to calculate the score. R is considered around 45k while testing. When the image pyramid is used, each RoI is closest to 224 sq pixels in area. The results obtained are compared on VOC 2007, 2010, and 2012. On all of them, Fast RCNN obtains high results 65.7% on VOC 2010 and 2012, while on VOC 2007 mAP increases to 68.1% [2].

2.3 FASTER R-CNN

As discussed above, recent advances were made to improve the overall object detection accuracy. Due to the Fast R-CNN method, the computational expenses drastically decreased, but the time consumed while generating region proposals was higher. Hence, the region proposals become the computational congestion in the object detection systems.

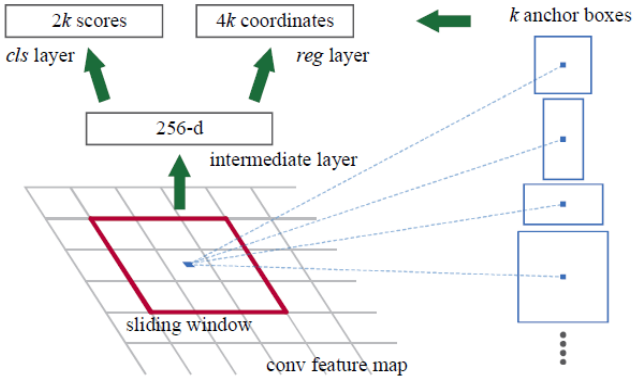


Figure 3: Region Proposal Network (RPN) [3].

The Faster R-CNN proposes to change the algorithm by computing the proposals with a deep net [3]. This approach results in a solution to the above bottleneck situation. Here, Region Proposal Networks (RPNs) are introduced to assign convolutional layers among object detection networks.

Hence, it reduces the expense in terms of time and space for computing proposals. The Region-based detectors use the convolutional feature maps to increase the computational speed. The RPNs proposed here is a fully-convolutional network and be trained end-to-end, especially the network that detects the region proposals. A Region Proposal Network fetches an image as input and generates a set of rectangular object proposals with their corresponding object score.

The minimal size of the network slides over the convolutional feature map output generated by the previously distributed convolutional layer. Further, a lower-dimensional vector is outlined per sliding window (256-d for ZF and 512-d for VGG) [3]. The vector resulted is fed into the branched fully-connected layers: a box-classification layer and a box regression layer.

In **Figure 3**, each sliding-window location *k* region proposals are predicted. The classification layer generates *2k* scores which evaluate the possibility of an object or not an object corresponding to each proposal. Here, the values of *k* proposals are taken relative to *k* reference boxes, called anchors. A significant approach called translation-invariant is used. Here, in terms of functions and anchors, the generation of proposals is relative to the anchors.

2.3.1 Loss function:

Each anchor is identified with a binary class. Every anchor is assigned with a positive or negative label based on the IoU value overlapped over any ground-truth box. The highest valued and the value higher than 0.7, the anchor is marked as positive. When the value is lower than 0.3, the anchor is considered negative. The rest is not included as a contribution for training purposes.
So, the loss function formulated as follows [3]:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{Ncls}\sum_i^i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{Nreg}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

Here,
i- the indexed value of an anchor.
p_i- predicted probability of an anchor.
p_i^* - 1, an anchor is positive.
p_i^* - 0, an anchor is negative.
t_i- vector describing the 4 parameterized coordinates of the bounding box predicted.
t_i^* - vector describing the 4 parametrized coordinates of the ground-truth box of a positive anchor.
L_{cls} = log loss over classes
(object/not an object)
$L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$, R is robust loss function [3].
$p_i^* L_{reg}$ - regression loss for a positive anchor ($p_i^*=1$).
N_{cls} and N_{reg} - the Normalization functions respectively.
λ - balancing weight.

Method	#proposals	data	mAP (%)	time (ms)
SS	2k	07	66.9	1830
SS	2k	07+12	70.0	1830
RPN + VGG, unshared	300	07	68.5	342
RPN+ VGG, shared	300	07	69.9	198
RPN+ VGG, shared	300	07+12	73.2	198

Table 1: Detection Results observed on VOC 2007 test set [3].

To optimize the training "image-centric" sampling strategy is adapted. Here, 256 random anchors per image are examined to estimate the loss function. A mini-batch with positive samples lesser than 128 is filled with negative ones. All current layers are set with a value of the standard deviation of 0.01 obtained by a zero-mean Gaussian distribution. The values initialized for the rest layers were determined by pre-training a model for the classification of ImageNet.

This method is evaluated on VOC 2007 and VOC 2012. The **Table 1** shows the results respectively.

2.4 SSD

Single Shot Multibox Detector (SSD) is a proposed framework that uses an individual deep neural network. Here, the bounding box adjusts automatically irrespective of the size and aspect ratio of an object defined. The results obtained are compared with results obtained on PASCAL VOC, ILSVRC, and COCO detection [4]The time taken to detect is estimated in seconds per frame (SPF), where the faster R-CNN runs at seven frames per second. The approach used here is on basis of a convolutional network that is feed-forward and generates a constant sized set of scores and bounding boxes for identifying the instances of object classes present within the boxes. Further, a non-maximum step is applied which produces the resultant detections. An auxiliary formation is combined with the network. This results in generating detections that consist the features that are multi-scaled, convolutional predictors, and with default box size or aspect ratio. Refer to the **Figure 4,** to understand the method in brief.

Training involves several categories matching strategy, knowing the training objective, Scale selection and an aspect ratio for default boxes, hard negative mining, and finally, data augmentation. After training, the results are compared against Fast R-CNN and Faster R-CNN on PASCAL VOC 2007.

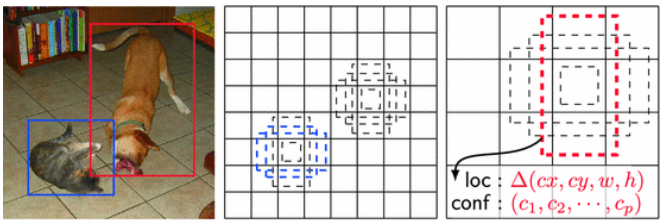


Figure 4: SSD framework. (a) for each object training an input image and ground-truth boxes is needed. (b) and (c) Evaluation of default boxes at each location that are of different scales 8 x 8 and 4 x 4 feature map [4].

The mAP obtained is 81.6% by SSD512 which is higher than fast R-CNN and faster R-CNN. For a more suitable understanding of SSD, a controlled evaluation is carried out to discover the individual component performance. Data augmentation, here Fast and Faster R-CNN employs horizontal flip and an image to train. Like YOLO, a comprehensive strategy of sampling is applied. Also, the Higher amount of default box shapes results in better performance. Even the faster atrous and different resolutions of multiple output layers add to the overall performance percentage. This method is evaluated on PASCAL VOC2012, COCO, and ILSRVC.

2.5 YOLO

Compared to previously discussed region proposal classification networks like Fast R-CNN which performs detection on various region proposals and thus ends up performing multiple predictions for various regions in an image, YOLO (You Only Look Once) architecture is more like F-CNN (Fully Convolutional Neural Network).

In this architecture, the input image is split in the (*mxm*) grid in which the individual grid produces 2 bounding boxes and class probabilities for those bounding boxes. YOLO only needs to look at the image once to detect all the objects, hence the name YOLO, is a faster model compared to other models discussed above. An individual neural network predicts class probabilities and bounding boxes straight from the input images in a single look at the image. End-to-end optimization can be done for the execution of object identification since the complete pipeline is a singular system. While frame detection is been performed as a regression problem, there is no need for a complex pipeline. The background is mistaken as patches in several images for objects in the Fast R-CNN method because it neglects to see the broader context. [5]

YOLO splits an input image into grids and if the grid cell contains the center of an object fall into a grid cell, then that grid cell will be accountable for identifying that object. Bounding boxes describes a rectangle that encloses these cells that have detected an object. YOLO then outputs a confidence score that tells how certain it is, that a bounding box encloses some object. The confidence scores reflect how confident is the model about a box containing an object. By doing this score, this can stop the model from identifying backgrounds, so if no object subsists in the cell, the confidence scores will be zero.

The bounding boxes predict if a significant object is present and also the class of the object, which was trained on the PASCAL VOC dataset. The confidence score for the bounding box along with the class predictions is combined into one final score that tells the probability that this bounding box contains a specific type of object. Since the final output of the image will have a lot of bounding boxes, the bounding boxes with a confidence score higher than a particular threshold will only be considered

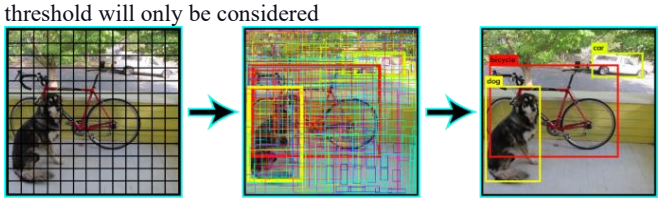


Figure 5: Identification of an object with bounding boxes using YOLO [5].

The image that is inputted is classified into an $S \times S$ grid. Individual grid cell predicts B bounding boxes and confidence scores for the boxes. Each grid cell also predicts C conditional class probabilities for the grid cells containing an object. This is described as a tensor prediction [5],

$$S \times S \times (B * 5 + C)$$

YOLO is a precise object detector and is quick, making it absolute to connect to a camera and help in real-time object detection. It takes minimum time to retrieve outputs from the camera device and identify the objects. This makes it helpful for computer vision applications for tracing and identifying objects as they travel around. [5]

The limitation of YOLO is that, since the object can be detected by classification and localization network, it means that any grid cell can detect only one object. If a grid cell includes more than one object the model will not be able to recognize all of them. Hence, close object detection is the area where YOLO faces an extreme amount of challenge. If the grid is a 7x7 and each grid can detect only one object, that is the maximum number of objects the model can detect is 7x7 = 49 objects. [5] YOLO also struggles to detect objects in different or unexpected aspect ratios or configurations as it predicts bounding boxes from the data. [5]

2.6 YOLOv2 / YOLO 9000

The real-time object detection YOLO introduced at the CVPR (Computer vision and pattern recognition, 2016) [6] shows that it is faster than other visual object detection methods for a variety of detection fields. YOLO 9000 is considered to be better, faster, and stronger and thus has numerous improvements over YOLOv1. YOLO9000 is a real-time framework for the discovery of more than

9000 object classifications by collectively optimizing detection and classification.

	Pascal 2007 mAP	Speed
DPM v5	33.7	.07 FPS | 14 s/img
R-CNN	66.0	.05 FPS | 20 s/img
Fast R-CNN	70.0	.5 FPS | 2 s/img
Faster R-CNN	73.2	7 FPS | 140 ms/img
YOLO	63.4	45 FPS| 22 ms/img

Table 2: YOLO object detection Speed Comparison [6]

On each of the convolutional layers, batch normalizations are applied in YOLOv2. By adding so, YOLO helps the mAP value to have a 2% improvement [6]. This leads to vital advancements while rejecting the need for other sorts of regularization.

In the case of the primary YOLO, the classifier network was trained at 224×224 and the resolution was then raised to 448 for object detection. As a result, when shifting to detection, a shift to learning object detection has to be done by the network and adapt to the current input resolution. While for YOLOv2, the model on the images is initially trained at 224×224, which are then fine tuned at the full 448×448 resolution on ImageNet before training for detection. A 4% improvement in mAP value is accomplished by this high-resolution classification network. [6]

In the case of Faster R-CNN, the bounding boxes are predicted using anchor boxes or handpicked priors. Predicting offsets instead of coordinates, make the problem simple, thus makes it simpler for the neural network to learn. Based on this, YOLOv2 uses these anchor boxes to predict bounding boxes by rejecting a fully connected layer. While using the anchor box, predicting class and objectness for each bounding box instead of a grid cell means one grid cell can contain multiple class confidence.

Multiscale training is implemented in YOLOv2. Here instead of fixing the input size, it is been trained at different resolutions to predict with different dimensions given as input. The network operates rapidly at smaller sizes, which in results offers an easy tradeoff between precision and pace for the YOLOv2 detection method [6].

2.7 YOLOv3

YOLO 9000 was the fastest, and also one of the most accurate algorithms, but a couple of years down the line, however, it no longer the most accurate with the algorithms like RetinaNet [19], and SSD exceeding it in terms of accuracy. It still, however, was one of the fastest. But that speed has been traded off for boosts inaccuracy in YOLOv3. YOLOv3 is an improvement over preceding YOLO detection networks. Compared to previous versions, it emphasizes multi-scale detection, a stronger feature extractor network, and some changes in the loss function.

The former YOLO versions have utilized Darknet-19 as a feature extractor that consists of 19 layers. YOLOv2 added 11 more layers to Darknet-19 [6] making it a total of 30-layer architecture. The algorithm still faced a challenge while detecting small objects due to the down sampling of the input image and losing fine-grained features.

YOLOv3 came up with a better architecture where the feature extractor used was a hybrid of YOLOv2, Darknet-53[20] (a network trained on the ImageNet), and on the residual networks (ResNet) that manages 53 convolution layers, hence the name Darknet-53. This network is developed with consecutive 3x3 and 1x1 convolution layers supported by a skip connection (organized by ResNet to help the activations scatter through deeper layers without gradient diminishing).

The 53 layers of the Darknet are then accumulated with 53 more layers for the development of the detection head, thus making YOLOv3 a total of 106 layers. This commences to a large architecture making it slow-paced compared to YOLOv2, although improving the accuracy at the same time. Compared to YOLO and YOLOv2, which predict the output at the end layer, YOLOv3 predicts boxes at 3 different scales. At every scale, YOLOv3 uses 3 anchor boxes and predicts 3 boxes for each grid cell. Each object is assigned to only one grid cell in one detection tensor [9].

2.8 State of the Art (YOLOv4 and PP-YOLO)

The below section addresses the present state-of-the-art object detecting methods, which are YOLOv4 and PP-YOLO.

2.8.1 YOLOv4

The 4th generation of YOLO was released in April 2020, by Alexey Bochkovskiy et. al, Chien-Yao Wang et. al, Hong-Yuan Mark Liao et. al, introduced via a paper titled "YOLOv4: Optimal Speed and Accuracy of Object Detection". YOLOv4 has additionally supported the Darknet and has obtained an AP value of 43.5 percent on the COCO dataset together with a real-time speed of 65 FPS [10]on the Tesla V100, which beat the foremost precise and quickest detectors in terms of both pace and precision. To make the designed detector perform on a single GPU, additional design improvements were made. One was that, the modern system of data augmentation where developed, which are Mosaic and Self-Adversarial Training (SAT). The following was choosing optimal hyper-parameters while involving genetic algorithms.

This version of YOLO is trained and performed on a standard GPU of 8-16GB Video RAM. YOLOv4 takes the influence of state of art BoF (bag of freebies) and many other BoS (bag of specials) [21]. The BoF improves the accuracy of the detector, without increasing the inference time. They only increase the training cost. The BoS raise the inference cost by a bit amount though they significantly enhance the precision of object detection.

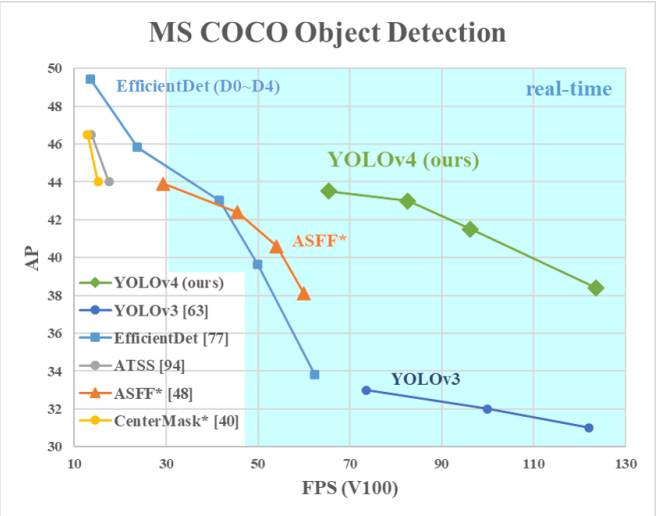


Figure 6: Comparison of FPS versus AP for the State-of-the-Art object detectors and proposed YOLOv4 [10].

2.8.2 PP-YOLO

Three months after the release of YOLOv4, PP-YOLO was introduced via a paper titled "PPYOLO: An Effective and Efficient Implementation of Object Detector", by Xiang Long et al [11]. PP-YOLO is based on an open-source deep-learning program which is known as Paddle-Paddle. This method proposes a model that starts from YOLOv3 object detector. Instead of using the original backbone which is the Darknet53 model, the proposed model uses ResNet50-vd-dcn [22].

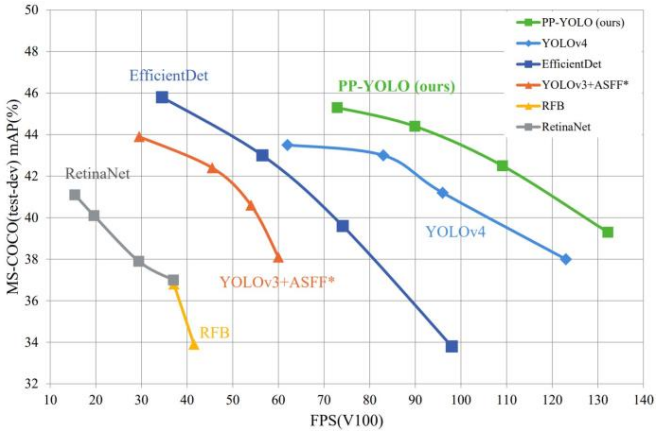


Figure 7: Comparison of FPS versus mAP for the State-of-the-Art object detectors and proposed PP-YOLO [11].

Various techniques are used in the paper to improve efficiency. Firstly, a large batch size is used that improves the stability of training and produces better results by the model. EMA (Exponential moving average) is also implemented, where the moving averages of the trained parameters during the interference produce better results. Drop block which is a form of structured dropout is also implemented in PP-YOLO, where the units in a continuous region are dropped together, which is applied only on the detection head because adding to the backbone architecture decreased the performance of the model. According to the paper, the PP-YOLO can achieve a mAP of 45.2% COCO dataset which exceeds the 43.5% of YOLOv4 [11].

3 CONCLUSION

This paper presents several proposed object detection methodologies and their unique approaches chosen for optimizing the accuracy and speed of object detection. While R-CNN consists of two major components, first to apply a highly compatible convolutional neural network to localize and classify the objects. Secondly, setting a sample to train large CNNs having limited training labeled data. The combinational classification tools from computer vision and deep learning gives the result. The Fast R-CNN method is a clean and faster update to R-CNN and SPPnet. This paper concludes that sparse object proposals seem to increase the detector quality.

The Faster R-CNN introduces Region Proposal Networks (RPNs) for a better performing region proposal computation. This is achieved by distributing the convolutional features with the detection network that is downstream. The proposed step reduces the computational time taken while generating a regional proposal. It improves the accuracy of the whole object detection method.

However, the Single-shot Multibox detector (SSD) method achieves to generate a combined network with multiple features mapped on the top of it by using a multi-scale convolutional bounding box. Here validation of training strategies and bounding boxes leads to enhanced execution. SSD model performs with a minimum amount of quantity higher predictions of the box than any other methods. It is shown that SSD successfully achieves high speed and accuracy than the previous object detection methods. It performs better than Fast R-CNN and Faster R-CNN on COCO and PASCAL VOC. The speed of the real-time SSD300 model is around 59 FPS. The You Only Look Once (YOLO) method detects objects in real-time with precision and in the quickest manner. YOLOv4 and PP-YOLO are the present state-of-the-art object detectors, which is the most accurate and fastest method proposed to date. With this, we conclude our research by noting that every method has limitations and improvements. Also, every improvised version has a unique way of dealing with the previous method limitation.

4 REFERENCES

[1] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation.

[2] Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[3] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

[4] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.

[5] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

[6] Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7263-7271).

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results, 2007.

[8] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In BMVC, 2014.

[9] YOLOv3: An Incremental Improvement Joseph Redmon Ali Farhadi 2018

[10] YOLOv4: Optimal Speed and Accuracy of Object Detection Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao 2020

[11] PP-YOLO: An Effective and Efficient Implementation of Object Detector Xiang Long, Kaipeng Deng, Guanzhong Wang, Yang Zhang, Qingqing Dang, Yuan Gao, Hui Shen, Jianguo Ren, Shumin Han, Errui Ding, Shilei Wen 2020

[12] C. Gu, J. J. Lim, P. Arbelaez, and J. Malik. Recognition using re- ´ gions. In CVPR, 2009.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In ECCV, 2014.

[14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. IJCV, 2010

[15] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In NIPS, 2012.

[16] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC2012). http://www.image-net.org/ challenges/LSVRC/2012/.

[17]] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In CVPR, 2009.

[18] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In ECCV, 2012

[19] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. ´Focal loss for dense object detection. arXiv preprint arXiv:1708.02002, 2017.

[20] J. Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016

[21] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, pages 2169–2178. IEEE, 2006.

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

Benefits of Provenance-Based Templates in Data Science and Data Visualization

Nitin Paul, Merijn Schröder

Abstract— Data is touted as the 'new oil' in the Information Age. Comprehensive and accurate collection, storage and analysis of the colossal amount of data generated by the global systems has began to fuel fields such as research, governance and economics. With utilization of insights from data playing such a crucial role in the modern world, data provenance has begun to emerge as both a critical requirement for validity of insights, such as in research and policy making, and a topic of continued research and development. Multiple studies focus on the generation of provenance in particular. In this paper we discuss `PROV-TEMPLATE`, one of the existing methods for provenance tracking using provenance-based templates, which provide a standardized, inter-operable format for collecting provenance information. The advantages and limitations of using templates are covered, both in data science in general and in data visualization in particular. We also discuss how provenance data collected in visualization workflow can be used to suggest related visualizations and even partially automate the process of making changes to existing visualizations.

Index Terms— Data provenance, provenance-based template, data visualization

1 Introduction

Data provenance is information tied to a record outlining its origin and relevance, along with methodologies and tools used in its acquisition, processing and storage. The quality of provenance data is increasingly becoming a necessary factor in determining the usability and validity of available data. This holds especially true in the area of research, where provenance of research data is critical for reproducibility and legitimacy of scientific results. Availability of provenance records becomes a necessary and often a legal requirement in the domains of research such as climate and Earth sciences where the insights gained from the analysis of research data are intended to aid policy making and governance [11]. Data provenance also plays an important role outside academia, for example in a corporate environment where provenance data and system status information is used for preemptive network security measures and causality analysis after an attack on the network [21]. A Decision Support System (DSS) is also an example in which data provenance can be valuable [5]. Despite the rapid increase in accuracy and reliability of DSSs, the wider acceptance has been much slower. The main reason for that is the perceived lack of transparency caused by the generation of recommendations by a black box. Potentially, data provenance can clarify the reasons why recommendations are made by the system which, in turn, can speed up the adoption of DDSs.

In view of this rising importance of having an effective provenance strategy across myriad domains of modern society, a significant amount of progress has been made in designing standards and tools for recording provenance. However, the heterogeneous nature of infrastructure and methodologies employed in collection, analysis and storage of data has hampered widespread adoption of provenance tools. This poses a challenge for effective provenance tracking, since end-to-end integration of provenance in the data analysis workflow is necessary to generate usable provenance tracking data, and supporting domain specific tools and workflows is difficult. Scientific workflow management systems like Kepler [1] have integrated provenance features as core functionality into their product, but the onus is now on the end-user to work within the confines of this system, which is not always feasible or practical. The focus therefore has been to devise general solutions that either can be integrated into existing infrastructure or used to generate retrospective provenance data using existing logs or output. Undertakings like the ENVRIplus project were tasked with development of generic tools and services to help build inter-operable Research Infrastructure (RI) for climate and Earth science research in Europe [11].

The most widely-used and referred standard for provenance is W3C's `PROV` recommendation [13], which evolved from the Open Provenance Model (OPM) [17]. The fundamental units of the `PROV` ontology are known as Starting Point terms consisting of 3 primary classes:

1. `prov:Entity` which represents resources
2. `prov:Activity` representing an action performed over an Entity
3. `prov:Agent` which represents persons or machines responsible for some Action over an Entity

Figure 1 illustrates a typical network of the three primary classes of objects in a `PROV` document and the correlation between them, indicated by the labelled arrows connecting the entities to each other and themselves. The most important relationships are: *used* (an activity used an artefact), *wasAssociatedWith* (an agent is linked to an activity), *wasGeneratedBy* (an activity generated an entity), *wasDerivedFrom* (an entity was derived from another entity), *wasAttributedTo* (an entity was attributed to an agent), *actedOnBehalfOf* (an agent acted on behalf of another agent) and *wasInformedBy* (an activity used an entity generated by another activity). The `PROV` standard is designed to describe retrospective provenance (`r-PROV`), which is analogous to timestamped logs of the steps executed to reach the current state of the data. This makes `PROV` relatively easy and less disruptive to integrate with existing Research Infrastructure for storing, manipulating and visualizing data. The barrier to the adoption of `PROV` is further lowered by an ecosystem of tools such as online tools that support the `PROV` standard[1], a public repository for provenance data known as ProvStore [7], packages such as ProvToolbox[2] for Java that enables users to manipulate provenance descriptions and convert `PROV` documents to related standards such as `RDF`, `PROV-XML`, `PROV-N`, and `PROV-JSON`. Similar functionalities are also provided by the `PROV` Python Package[3]

Availability of robust standards such as the `PROV` standard and the tools built around it should help make the adoption of provenance tracking more prevalent across domains, however many substantial

- *Nitin Paul, Student number: S4420098, E-mail: n.paul@student.rug.nl.*
- *Merijn Schröder, Student number: S3328481, E-mail: m.l.schroder@student.rug.nl.*

[1] https://openprovenance.org/, https://provenance.ecs.soton.ac.uk/
[2] https://github.com/lucmoreau/ProvToolbox
[3] https://github.com/trungdong/prov

problems need to be solved before it becomes an ubiquitous component of data analysis workflows. Two prominent cases where collection of provenance data has been difficult to integrate are systems involving large-scale sensor networks that are highly automated, often running on proprietary software environments which makes it difficult to modify the underlying source code or architecture to integrate provenance data collection, and smaller-scale semi-automated systems that incorporate human sampling of data and observation along with machine operations. The latter often have heterogeneous workflows that combine both machine and human operations on data, such as collecting door-to-door citizen data by municipalities employing volunteers or workers who might not have the required domain knowledge to reliably collect provenance data. Tools such as handheld applications developed by Urbanopoly Project [2] can help streamline manual provenance tracking. Adopting the `PROV` standard for both highly-automated and semi-automated workflows is a reasonable and minimally disruptive step to incorporate provenance tracking in the analysis process. Specially annotated scripts can be used to generate both retrospective provenance, by using tools such as NoWorkflow system [19], and prospective provenance, using tools such as the YesWorkflow system [12], which can be easily integrated to mainstream tools such as Jupyter Notebook. Retrospective provenance can also be generated by employing the 'Passive Monitoring' methodology, which involves making use of the existing outputs to generate provenance, without making modifications to the existing setup.

One of the approaches for generating provenance in a retrospective manner is by making use of provenance-based templates, as described by Moreau et al. in their paper titled *A Templating System to Generate Provenance* [14], which is our primary focus in this paper. We investigate the benefits and limitations of this approach for generating provenance data and use of templates in data visualization.

We summarize the topic of provenance-based templates by searching for related literature in Section 2. After clarifying the underlying concepts, we investigate the use-cases where provenance-based templates are useful and can provide adequate value. We proceed by investigating the application of templates in data visualization in Section 3. Finally, in Sections 4 and 5 we summarise our findings and answer our research questions.

2 Provenance-Based Template

A template is defined as a pattern which can be used to produce results which are similar. A template is defined as anything that serves as a pattern on which the properties of a product can be based. For instance, this document can be reduced to a template for scientific papers by removing the text and only keeping the information about the positions and formatting of different elements, such as the title and the list of authors. The main advantages of such a template is the consistency over different products and the assurance of validity. Another paper written based on the same template will have the same formatting as this paper has. Besides, if the information of the template is valid, using the template will result in reusing the same valid information, which makes the new product less prone to errors.

This idea of a template can be applied to the generation of provenance. As mentioned in Section 1, there are some challenges with collecting provenance data for applications. We briefly touched upon some efforts which focus on finding the best approaches for facing these challenges. Using templates is one of these approaches. An implementation of a templating system to generate provenance is introduced by Moreau et al. in [14]. Data provenance is generated in a retrospective manner, rather than when an application is run or when its source code is compiled. First, a template provenance file is created by a designer. This template contains the shape of the provenance data to be generated, but without the actual provenance data. Instead, at the places where the values have to be filled in, placeholders are added. These placeholders act as variables - a mechanism injects the actual values into the placeholders by assigning the values to the corresponding variables. This approach results in a provenance document in one of the standardized representations. The architecture of this approach is shown in Figure 2, adopted from W3C Recommendation [14].

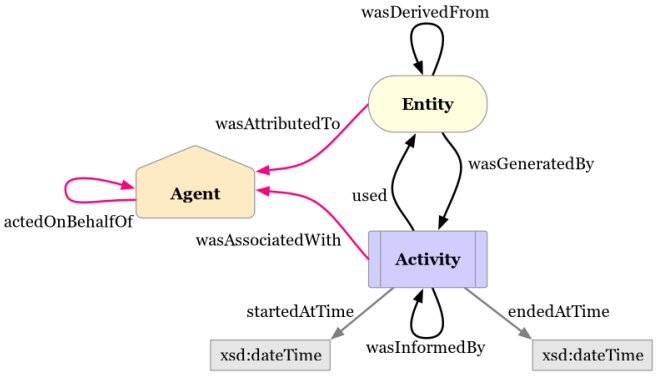


Fig. 1. PROV-O starting point, adopted from W3C Recommendation

We will proceed to discuss the benefits of using provenance-based templates for collecting provenance data, including an overview of its advantages such as the inherent separation of responsibilities, ease of maintenance, ability to integrate automated checks into the process of data provenance collection and the ease of consumption of standardized provenance data, followed by the limitations.

2.1 Benefits

Moreau et al. have developed applications [14] that generate provenance data directly. This means that the generation of provenance data was implemented in the same code base as the actual features of the application, which shifts the responsibility of provenance data generation to the software developers. The authors did this comparison between the more traditional approach and the templating approach in order to find the differences in the workflow of implementing it. In this section we go over four of these benefits: the separation of responsibilities introduced by the templating approach, less and easier maintenance because of the use of template, the possibility for checks during run-time, and a more efficient consumption of provenance generated by the same application.

2.1.1 Separation of Responsibilities

One workflow which is used is the Provenance Challenge workflow [15]. This workflow is a series of procedures which are performed by a system. The key point of this approach is that these procedures have to be implemented in the same code base as the application of which provenance data has to be generated. As a result, the software developers need to have a thorough understanding of the generation of data provenance in order to implement it correctly. Besides that, the code for generating provenance and the code of the application itself are not properly separated. One could argue that this leads to the violation of the principle of separating responsibilities in applications.

`PROV-TEMPLATE` eliminates this issue since the generation of provenance is not implemented in the application itself, but rather in a retrospective manner. Because of this, software developers need to have very little knowledge about the way data provenance is handled and there is a clear separation of responsibilities.

2.1.2 Maintenance

The Python library ProvPy [4] was tested by the authors as well. With this approach, the number of lines written to implement the generation of provenance is between the 2.5% and 20.0% of the lines of code written for the Provenance Challenge workflow. This reduces the time and effort required for the maintenance of provenance data generation code alongside the development of application features. However, a drawback to this approach is that when specifications are revised, non-trivial work has to be done by the developer. The authors have shown that a small change, e.g. changing a type, affected 60 lines of code in their application. While using the templating approach, no change

[4] https://pypi.org/project/provpy/

had to be made to the actual application at all. One of the benefits of the templating approach is the reduction in time spent on maintenance compared to using the `ProvPy` library. Using templates, minor revisions to the shape of the provenance data can be made without introducing the need for changes in the application, rather only in the template.

A number of basic modifications to a template can be made without changing anything in the application itself. Examples of these changes are: changing a constant, adding or dropping an attribute, and adding or dropping a node. Larger modifications to templates do however introduce the need for changes in the application, like renaming a template, adding or dropping a template, or merging or splitting templates. Also when adding and dropping variables in a template some modifications have to be done within the application itself.

A subset of possible modifications to a provenance-based template is listed in Table 1 including whether the bindings remain correct (✓), the bindings become potentially incomplete (I), the bindings become superfluous (S), or the change in the template can be detected and solved at compile-time (C), or at run-time (R).

Another benefit of the templating approach is that a library of templates for the complete application is maintained in one location. This makes the maintenance less cumbersome, as demonstrated by Lenz et al [10].

2.1.3 Checks

Additionally, using templates allows for the possibility to introduce safety checks during run-time. These checks can for instance be used to make sure that the used bindings in an application are compatible with the corresponding template.

Not only the communication between the application and the template should be valid. The template itself should be valid as well. Essentially, a template is a provenance document, without any meaningful values filled in. There already exist several provenance validators, such as [16], which is introduced by the same authors who also introduced the templating approach.

Manual checking remains an important aspect as well. The use of templates makes this easier because of the relatively small number and size of templates which have to be checked. For instance, detecting the presence of a loop or detecting the absence of attributes are done most easily manually. When provenance data is generated by an application without using templates, these problems are much harder to spot.

2.1.4 Provenance Consumption

An application which consumes the provenance data it has generated, traditionally has to perform graph queries and post-processing on the provenance data. This is needed because of the lack of information about the structure of the provenance data in the application itself. This information becomes available with the introduction of templates. The bindings, which are used by the application to submit the right information to the template, can also be used to consume the provenance data directly.

2.2 Limitations

The use of templates for generating provenance also introduces challenges and has some limits. We outline the major challenges.

2.2.1 Granularity

By definition, templates are general. Therefore, the challenge faced by the template designer is to make the template applicable on all possible types of data and operations, and not leave out any relevant information. For instance, provenance can be generated for a REST application which processes invoices from multiple suppliers. It is likely that suppliers will have different invoice structures, and therefore the template for generating provenance has to accommodate all different types. Ideally, the template should work when a new invoice is added.

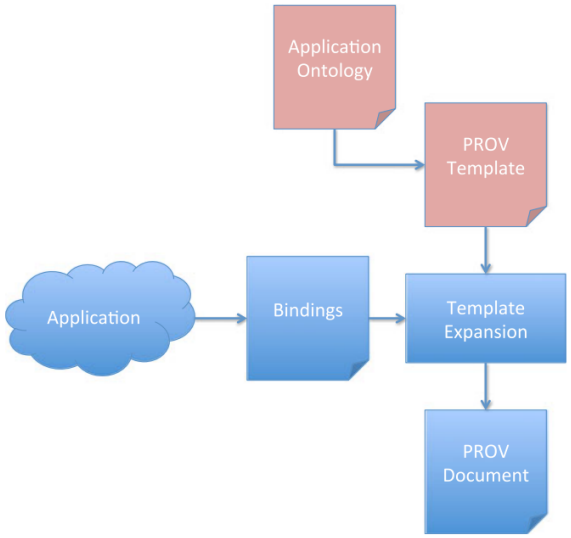


Fig. 2. The architecture of the approach of generating provenance using templates, adopted from [14]

2.2.2 Overlapping Responsibilities

As mentioned is Section 2.1, there is a separation of responsibilities when working with templates. The developer has to develop the application and has little knowledge about data provenance. A `PROV` expert designs the template for the application and does not have an in-depth knowledge about the logic within the application itself. However, despite of this separation of responsibilities and knowledge, there is still some overlap. For the `PROV` expert to design a template, he has to know exactly what set of bindings is submitted by the application so that he can include these in the template.

The design of the template is the first development phase. In the second phase the integration of the application and the template is completed. This requires both knowledge about the application, in order to develop it, and knowledge about data provenance and the designed template, to make sure the integration is done properly. To successfully complete the integration, pair-programming is regarded as the best solution [14].

Pair-programming has its benefits and limits [6]. In this case, however, the limits seem to outweigh the benefits. The major benefits of pair-programming are the potential different views on a problem and the sharing of knowledge. However, when using pair-programming for integrating the application and the provenance template, the (required) knowledge and field of expertise differs a lot between the programmer and the `PROV` expert. Essentially, they solve different problems. This eliminates the benefit of having different views on a problem. Furthermore, because of the difference in field of expertise, the benefit of conveying knowledge is limited.

The main challenges of pair-programming are not posed by the task at hand, but rather the ability of the programmers working on the task to work together cohesively. The programmers have to form a good pair, both socially and with regards to their workflow. If these requirements are not met, it is likely integrating will either fail or will be done in a poor manner.

Additionally, for both developing the application and designing the template, knowledge about the domain is a necessity [5], which adds a third dimension to the difficulty faced in integrating provenance data generation to the software development workflow.

3 Provenance-Based Templates in Data Visualization

Data visualization is an essential component of data analysis that enables engineers, researchers, lawmakers, managers and consumers to harness key insights from data. As a result, visualization research has grown into a well-established research area with well-defined agendas.

	Abstract Bindings	Concrete Bindings	Tabular Data	Bindings Fragment	Template Compilation
Rename template	I/S	I/S	I/S	I/S	C/R
Add template	I	I	I	I	I
Drop template	S	S	S	S	C
Split template	I	I	I	I	C
Merge template	S	S	S	S	C
Modify template					
Change constant	✓	✓	✓	✓	✓
Add attribute	✓	✓	✓	✓	✓
Drop attribute	✓	✓	✓	✓	✓
Add relation	✓	✓	✓	✓	✓
Drop relation	✓	✓	✓	✓	✓
Add node	✓	✓	✓	✓	✓
Drop node	✓	✓	✓	✓	✓
Add variable	I	I	I	I	R
Drop variable	S	S	S	S	C

Table 1. Consequences of modifications in a template, adopted from [14]

However, building visualization pipelines is a major bottleneck in data exploration, especially with complex data [18]. Multiple collections of related visualization pipelines are often created while exploring expansive or complex data sets, in order to fine tune the various parameters or implement different visualization algorithms using a trial-and-error process [23] which often results in a majority of visualization not being used in the final analysis. Although the final visualizations may be different, the underlying pipelines often have common parameters such as a shared data source. Tracking modifications to data pipelines and related parameters facilitates generation of provenance information [4], which can be used to switch between different iterations of a visualization and the corresponding pipelines to simplify the laborious task of analyzing complex data. The provenance data can also be used to simplify and partially automate the process of generating related visualizations, eliminating the need for reinventing the pipeline again, and making it possible to expand upon or adapt existing work. This approach of using provenance data to assist in the process of generating visualizations will enable any user, even without adequate domain knowledge related to the data being explored, to generate visualizations of comparable quality to a user with extensive domain knowledge.

The proposed framework consists of two key components: an interface for querying data flows and a visualizations by analogy mechanism to semi-automatically generate visualizations [20]. The query interface makes use of the same interface which was used to create the pipelines, making it intuitive to use. The interface supports both simple keyword-based queries (for example, looking up pipelines created by a specified user) as well as complex and highly specific structure-based queries (for example, finding visualizations that implement a smoothing function before computing an isosurface for irregular grid data, along with the related pipelines and sub-pipelines). A reference implementation is illustrated in Figure 3. The visualizations by analogy component allows reusing existing pipelines to generate related visualizations without requiring users to manually modify the data flow specifications. This is implemented by calculating the difference between two comparable pipelines and applying it to a third pipeline.

The framework proposed in [20] can work with established visualization systems such as AVS Explorer [22] and IBM's Data Explorer[5]. The framework can also work in tandem with existing provenance-enabled scientific workflow management tools that record provenance data for both data products as well as metadata for the workflow used for data processing [3]. Encouraging research such as a novel mechanism for tracking parameters in visual data mining proposed by Kreuseler et al. [9] and a formal calculus for parameter changes proposed by Jankun-Kelly et al. [8] pave the way for adoption of continued refinement of strategies proposed in [20].

[5]https://www.research.ibm.com/dx

3.1 Pipeline Operations

A *visualization system* is a system that lets users display a graphical representation of data, based on a set of specified rules. These programmatic rules for a given visualization constitute its *pipeline*. The pipeline is made of *modules*, that define an operation or functionality, and *connections* that define the data flow between modules. The state of a module is represented by *module parameters.*. The set of all visualization pipelines is denoted by V.

3.1.1 Pipeline Differences

In any visualization workflow involving moderately complex data, often multiple visualization pipelines are created iteratively and refined to generate the required visualization. To understand the workflow, it is imperative to understand the difference between two given pipelines. To illustrate the operation, a function δ is defined, which is function that takes two pipelines p_a and p_b and returns a function that transforms p_a to p_b. This can be expressed as:

$$\delta_{ab} = \Delta(p_a, p_b) \tag{1}$$

We can say that δ_{ab} is the sequence of transformation required to derive p_b from p_a. The *domain context* D of δ_{ab} is defined as the set of all module parameters that must exist in p_a for δ_{ab} to be applicable. The *range context* R of δ_{ab} is defined as the set of all transformations that must be applied to p_a to transform it into p_b. δ can fail if one module in $D(\delta)$ does not exist. Any pipeline p_a can be transformed by finding $\delta(p_a)$. Similarly, if we know that a pipeline p_b was derived from p_a, we can derive p_a from p_b by finding δ_{ab}^{-1}

3.1.2 Matching Pipelines

Another important operation relevant to working with pipelines is finding similar pipelines. This operation can either yield a binary result (two pipelines either match or not) or a mapping of similarities based on various metrics. Such a mapping function can be defined as $map : V \times V \rightarrow (D \rightarrow D)$, which takes two pipelines p_a and p_b as inputs and returns a mapping from the domain context of p_a to the domain context of p_b. To construct the mapping, the problem is formulated as a weighted graph matching problem. Let $G_a = (V_a, E_a)$ be the graph corresponding to p_a where V_a represents the modules in p_a and E_a is the set of connections in p_a. To calculate the similarity between the vertices of the graph, which represent the modules, we define a scoring function s such that $s : V_a \times V_b \rightarrow [0.0, 1.0]$. The value of s will be 1.0 if two modules are exactly the same. The matching operation can be represented as:

$$\sum_{(v_a, v_b) \in M} s(v_a, v_b) \tag{2}$$

where v_a and v_b are modules of p_a and p_b respectively, and M is the set of all modules in p_a and p_b. A matching is *good* when (2) is maximized.

The operations described are theoretically hard to compute and therefore the framework makes use of heuristics. The information stored in δ is used to reduce the search space and increase the effectiveness of these heuristics.

3.2 Query-By-Example

In a visualization workflow involving multiple data sets, different visualization algorithms and corresponding pipeline parameters, it becomes impractical to search through the large sets of pipelines to locate

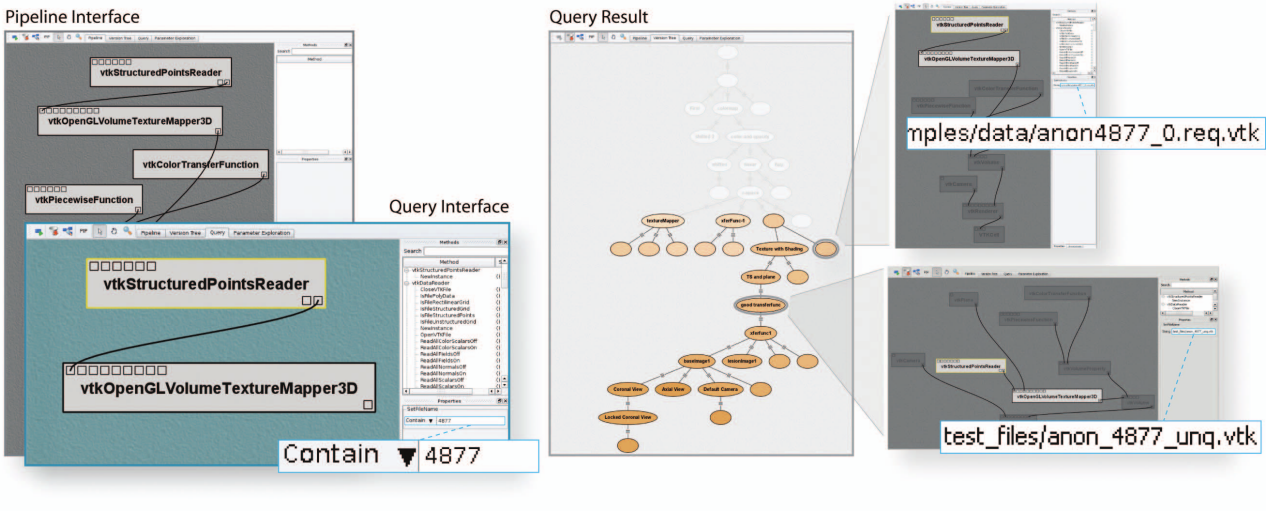


Fig. 3. Query by Example interface implemented in VisTrails, adopted from [20]. The user specifies a sub-pipeline and a search query (the string "4877" in this example) and the system returns all pipelines with matching structure containing the supplied query string

a specific pipeline by manually examining each one of them. *Query-by-example* is envisioned as an effective mechanism for navigating and searching a large set of pipelines by visually building a pipeline fragment containing the required parameters. The interface to build a query is the same as the one used to build the pipeline, making it easy for the user to work with the system, instead of having to learn a query language.

The pipeline difference function δ can be used to optimize the search function by reducing the search space. For example, for a query pipeline p_q and two candidate pipelines p_a and p_b, if p_a matches the search query and we know δ_{ab}, we can check if domain context $D(\delta_{ab})$ matches any module from p_q or that any module in p_a will be modified in order to transform it to p_b, then we can say that p_b is not a match for p_q, since some of the modules in p_b will now not match with p_q. We can thus iteratively reduce the search space by calculating the δ for any matching pipelines.

3.3 Visualization by Analogy

Two pairs of ordered values are considered analogous if the relationship between the first pair is the same as the relationship between the second pair. This analogous reasoning can be used to update multiple visualizations simultaneously by inferring the necessary changes between two pipelines as an analogy and applying it across all matching pipelines. As an example, if an identical change must be made to a set of related visualization pipelines without manually making the edits (which might be repetitive and error-prone), the change can be made only once on a pipeline and captured as an analogy. This analogy can now be applied on a set of selected pipelines using Equation (2) for a uniform transformation.

Analogies can also be captured for a specific pipeline update in a workflow system and used like a macro operation. For example, given a pipeline that reads protein data from a local file and creates a visualization, an update analogy can be captured to integrate functionality to read data from an online source instead. This analogy can now be thought as a portable set of rules to modify the input module in a visualization pipeline to enable it to read data from an online source, and can now be used across projects and users. This provides a standard way to make changes to a visualization pipelines without knowing the underlying domain knowledge.

4 Discussion

In this paper, we focused primarily on the benefits and limits of using provenance-based templates to generate provenance data, and how provenance data can be leveraged to help simplify and partially automate the data visualization workflow. The next step is to make the provenance data available to the users in an accessible form, ideally embedded into the interface of the application they are using. For example, data provenance is an important part of a decision support system [5]. Based on this information, the users of the system decide whether they trust and will act upon the suggestions provided by the system. It is also crucial to present the provenance-data in an intuitive form since the users of these systems may not possess expertise in the field of data provenance to utilize the information in a raw or unstructured form. Future work might be necessary to find out how to achieve these goals.

5 Conclusion

The last couple of years a number of approaches were proposed to make the generation of data provenance easier and less error prone. The use of templates is one of these approaches. We took an in-depth look at the idea of using provenance-based templates for provenance data generation, and the benefits and limitations of this approach. We have found that using templates results in the separation of responsibilities. This holds for both software and the people developing the application. For the latter, the main distinction made is between the software developer and the PROV expert. The integration of templates with the application requires both persons from the different fields to work together, however, which may lead to additional challenges.

In general, templates reduces the need for maintenance and make maintaining the application easier and more efficient. Using templates also enable application developers to add run-time checks. In some circumstances using templates can make consuming the provenance data generated by the same application more easy.

A challenge we found regarding templates is the trade-off between making the template general and not leaving out any relevant information.

We also discussed techniques to leverage provenance data to help users navigate large sets of visualizations, effectively query visualization pipelines using powerful visual interfaces and use analogies to simultaneously manipulate multiple pipelines. A major challenge we discussed is the considerable amount of work required to make application development and template design work together cohesively. Another significant challenge is to keep the templates general and applicable across myriad domains and use-cases. Domain specific knowledge remains key to improving widespread adoption of provenance standards and integration of tools and methodologies into workflows. For example, using analogies to replicate changes across a set

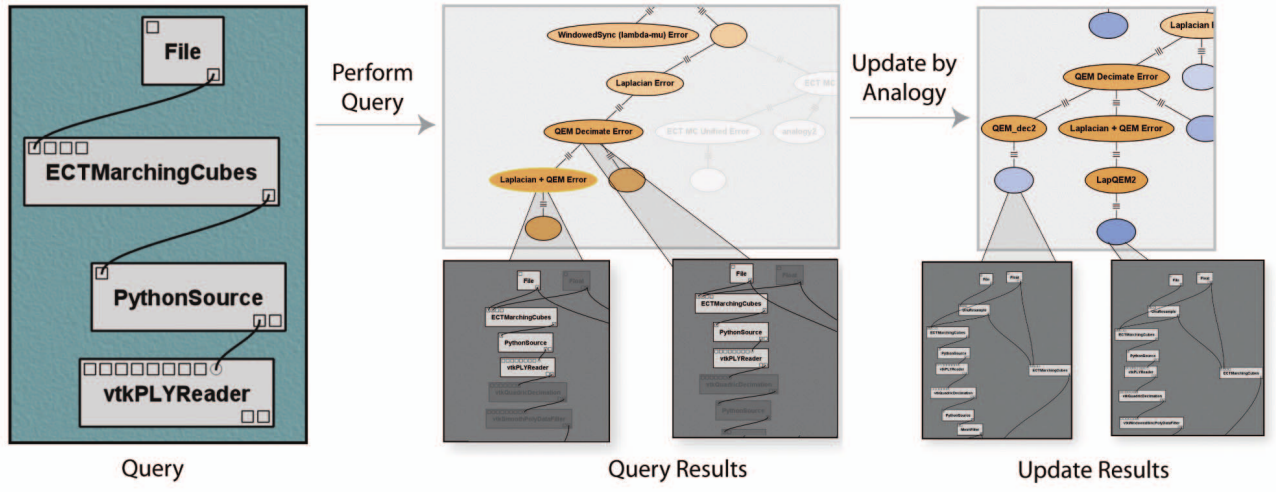


Fig. 4. Analogy based update implemented in VisTrails, adopted from [20]. The user searches for a query pipeline and updates multiple matching pipelines using analogy by adding modules.

of pipelines operates by finding the difference between two pipelines represented as graphs, and finding the distance between corresponding nodes. These distances and weights may vary across domains and the algorithm must be adjusted accommodate for such specific cases. Further work on data provenance tools and methodologies must incorporate a wide array of domains, or designed to be extensible, in order to be widely adopted.

Acknowledgements

We wish to thank Lorenzo Amabili helping us getting started in a new research field by pointing us in the right direction and providing valuable feedback. We also wish to thank M. Biehl for sharing his own experiences in writing papers with us. Finally, we would like to thank Jeroen de Baat for reading our draft paper thoroughly and providing us with rich feedback.

References

[1] I. Altintas, O. Barney, and E. Jaeger-Frank. Provenance collection support in the kepler scientific workflow system. In L. Moreau and I. Foster, editors, *Provenance and Annotation of Data*, pages 118–132, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[2] P. Buneman, A. Chapman, J. Cheney, and S. Vansummeren. A provenance model for manually curated data. In L. Moreau and I. Foster, editors, *Provenance and Annotation of Data*, pages 162–170, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[3] S. Callahan, J. Freire, E. Santos, C. Scheidegger, C. Silva, and V. Huy. Managing the evolution of dataflows with vistrails. 01 2006.

[4] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. Vistrails: Visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, page 745–747, New York, NY, USA, 2006. Association for Computing Machinery.

[5] V. Curcin, A. Fairweather, R. Danger, and D. Corrigan. Templates as a method for implementing data provenance in decision support sytems. *Journal of Biomedical Informatics*, 16:1–21, June 2016.

[6] B. Hanks, S. Fitzgerald, R. McCauley, L. Murphy, and C. Zander. Pair programming in education: a literature review. *Computer Science Education*, 21:135–173, June 2011.

[7] T. D. Huynh and L. Moreau. Provstore: A public provenance repository. In B. Ludäscher and B. Plale, editors, *Provenance and Annotation of Data and Processes*, pages 275–277, Cham, 2015. Springer International Publishing.

[8] T. J. Jankun-kelly, K. liu Ma, S. Member, M. Gertz, and I. C. Society. A model and framework for visualization exploration. *IEEE Transactions on Visualization and Computer Graphics*, 2007.

[9] M. Kreuseler, T. Nocke, and H. Schumann. A history mechanism for visual data mining. In *Proceedings of the IEEE Symposium on Information Visualization*, INFOVIS '04, page 49–56, USA, 2004. IEEE Computer Society.

[10] M. Lenz, H. A. Schmid, and P. F. Wolf. Software reuse through building blocks. *IEEE Software*, 4(4):34–42, 1987.

[11] B. Magagna, D. Goldfarb, P. Martin, M. Atkinson, S. Koulouzis, and Z. Zhao. Data provenance. *Towards Interoperable Research Infrastructures for Environmental and Earth Sciences*, July 2020.

[12] T. M. McPhillips, T. Song, T. Kolisnik, S. Aulenbach, K. Belhajjame, K. Bocinsky, Y. Cao, F. Chirigati, S. C. Dey, J. Freire, D. N. Huntzinger, C. Jones, D. Koop, P. Missier, M. Schildhauer, C. R. Schwalm, Y. Wei, J. Cheney, M. Bieda, and B. Ludäscher. Yesworkflow: A user-oriented, language-independent tool for recovering workflow information from scripts. *CoRR*, abs/1502.02403, 2015.

[13] P. Missier, K. Belhajjame, and J. Cheney. The w3c prov family of specifications for modelling provenance metadata. 03 2013.

[14] L. Moreau, B. V. Baltajery, T. D. Huynh, D. Michaelides, and H. Packer. A templating system to generate provenance. *IEEE Transactions on Software Engineering*, 44(2):103–121, February 2018.

[15] L. Moreau et al. The first provenance challenge. *Concurrency and Computation: Practice and Experience*, 20(5):409–418, April 2008.

[16] L. Moreau, T. D. Huynh, and D. Michaelides. An online validator for provenance: Algorithmic design, testing, and api. *17th International Conference, FASE 2014, Hels as Part of the European Joint Conferences on Theory and Practice of Software*, pages 291–305, April 2014.

[17] L. Moreau, N. Kwasnikowska, and J. Van den Bussche. The foundations of the open provenance model. 04 2009.

[18] T. Munzner, C. Johnson, R. Moorhead, H. Pfister, P. Rheingans, and T. S. Yoo. Nih-nsf visualization research challenges report summary. *IEEE Computer Graphics and Applications*, 26(2):20–24, 2006.

[19] L. Murta, V. Braganholo, F. Chirigati, D. Koop, and J. Freire. noworkflow: Capturing and analyzing provenance of scripts. In B. Ludäscher and B. Plale, editors, *Provenance and Annotation of Data and Processes*, pages 71–83, Cham, 2015. Springer International Publishing.

[20] C. E. Scheidegger, H. T. Vo, D. Koop, J. Freire, and C. T. Silva. Querying and creating visualizations by analogy. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1560–1567, November/December 2007.

[21] Y. Tang, D. Li, Z. Li, M. Zhang, K. Jee, X. Xiao, Z. Wu, J. Rhee, F. Xu, and Q. Li. Nodemerge: Template-based efficient data reduction for big-data causality analysis. *ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*, October 2018.

[22] C. Upson, T. Jr, D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. van Dam. The application visualization system: A computational environment for scientific visualization. *Computer Graphics and Applications, IEEE*, 9:30 – 42, 08 1989.

[23] J. J. van Wijk. Views on visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):421–432, 2006.

An Overview of Communication Protocol Specifications

Bauke Risselada and Floris Westerman

Abstract— With many computers connected to the internet, the demand for safe, reliable and secure protocols is ever increasing. To be able to validate such properties, several formalisms were developed in which the protocol in question can be expressed. In this paper, three of them are compared to each other: Multiparty Session Types (MPST), High-level Message Sequence Charts (HMSC), and the Blindingly Simple Protocol Language (BSPL). To serve as a base for this comparison, a toy protocol is designed and expressed in each of these formalisms.
The rationales between these formalisms vary greatly, and there is no 'universal fit' for any protocol. Instead, our comparison shows that the formalisms complement each other and each focus on their own niche: formal analysis for MPST, visualisation and implementation guidance for HMSC, and business processes for BSPL. Therefore, the choice of a suitable formalism is dependent on the design and purpose of the protocol under consideration.

Index Terms—Formalism, Protocol, Multiparty Session Types, Message Sequence Charts, Blindingly Simple Protocol Language.

1 Introduction

Over the past decade, increasingly many systems have been connected to the internet. With the advent of the Internet of Things, this trend will only accelerate. All these systems heavily rely on many layers of communication protocols: fundamental ones such as those in the seven layers of the OSI model [16], as well as many application-specific ones. Furthermore, in recent years there have been increasingly many data breaches, hacks, and system infiltrations. Therefore, there is an ever-increasing need for a way to ensure the safety, reliability, and correctness of communication protocols and their implementations.

While not trivial, it is possible to ensure these properties of protocols and their implementations. Central in such analysis is a consistent, precise, and unambiguous representation of a protocol: a so-called *formalism*. Various formalisms have differing properties; while some allow for formal correctness proofs, others focus on simplifying and assisting implementation. Therefore, the choice of formalism for a protocol highly influences its characteristics.

In recent years, multiple such formalisms have been developed. In this paper, we will discuss three of these in particular: *Multiparty Session Types* [4, 15], *(High-level) Message Sequence Charts* [2, 6, 7], and the *Blindingly Simple Protocol Language* [12]. In sections 3 through 5 we will first introduce these formalisms by means of expressing a single 'toy' protocol in each of them, after which we will compare and discuss their use cases and applicability in different situations in sections 6 and 7. Finally, we will list some potential alternatives to the formalisms described here in section 8.

2 Publication Protocol

The 'toy' protocol we will consider is designed in such a way as to highlight the strengths and weaknesses of each approach. It concerns a simplified view of the process of publishing a paper in academia. We identify three roles: writer, publisher, and reviewer. The process works as follows:

1. The writer will send their draft to a publisher.
2. The publisher will select a reviewer for the draft, and forward the draft to the reviewer.
3. The reviewer will engage in an open review process with the writer. The reviewer can choose to either accept or reject the paper, or request revisions. After revision, the reviewer can again choose from the same three options, any number of times.
4. If the paper has been accepted, the reviewer will inform the publisher.
5. The publisher will determine a publication date of the final paper.

This protocol is fairly straightforward, but serves as a great example for our purposes. A representation as a Multiparty Session Type can be found in Listing 1, the graphs of the Message Sequence Charts are shown in Figure 2, and the representation in the Blindingly Simple Protocol Language is found in Listing 2.

- *Bauke Risselada. E-mail: b.p.risselada@student.rug.nl.*
- *Floris Westerman. E-mail: f.p.westerman@student.rug.nl.*

3 Multiparty Session Types

Multiparty Session Types (MPST) [4, 15] are a very formal and precise approach to expressing a protocol we will discuss in this paper. They are a generalisation of (binary) session types [14]: while binary session types capture communications between two parties, MPST generalise this to allow for any number of parties. We will discuss a slightly simplified version of MPST that is sufficient for demonstration purposes, as defined in [15].

The concept central to session types is *types*, as the name already implies. A type captures a sequence of messages, in a very precise mathematical form. In MPST, we distinguish between *global* types and *local* types. Global types describe an entire protocol with all parties, whereas local types describe the protocol only from the perspective of a single party. Where global types specify messages between parties and its order, a local type is only concerned with one other party and whether a message is incoming or outgoing.

The description of such a type is closely related to the concept of process calculi [1], their notations share a strong resemblance as well. Essentially, the type puts constraints on the behaviour of a program, while a calculus encodes this behaviour. An example of a calculus relevant in the context of protocols would be the Calculus of Communicating Systems (CCS) [8].

3.1 Global Types

Before formalising the notion of a global type, it is instructive to first look at an example. Consider an elementary handshake protocol between parties `A` and `B`. In this protocol, `A` will send a message *init*(`void`) to `B`, after which `B` will reply with an acknowledgement *ack*(`void`) or deny the connection with *deny*(`void`). The global type for this protocol is written as follows:

$$G = \texttt{A}\rightarrow\texttt{B}:\mathit{init}(\texttt{void})\cdot\texttt{B}\rightarrow\texttt{A}:\{\,\mathit{ack}(\texttt{void})\cdot\texttt{end},\mathit{deny}(\texttt{void})\cdot\texttt{end}\,\},$$

where G is the global type, the $\cdot$ symbol represents sequencing, and `end` denotes termination of the protocol. The brackets {} denote a choice between messages, in this case between *ack*(`void`) and *deny*(`void`). Repetition can be expressed as well, by means of a 'label-goto' system: we mark the start of the loop with $\mu\mathbf{X}$, and at some later point we invoke $\mathbf{X}$ to jump back.

The full specification of MPST [4] supports a number of features we do not include here for brevity and clarity. These include concurrent execution of processes, as well as the possibility to send types themselves, in addition to sending values.

The precise formal specification is in accordance with [15], with some minor changes to clarify the typography. As already indicated above, every message consists of a *label* l_i and a *value*. These values have some type S (boolean, number, etc.), which we will call a *sort* as to not confuse them with the global and local types of a protocol.

Definition 1 Sorts *are defined by the following simple grammar*:

$S ::= \quad \texttt{nat} \mid \texttt{int} \mid \texttt{bool} \mid \texttt{text} \mid \texttt{void} \mid \texttt{date}.$ □

Definition 2 Global types *are defined by the following grammar*:

$G ::= \quad \texttt{end} \mid \mu\mathbf{X} \cdot G \mid \mathbf{X} \mid \texttt{p}\to\texttt{q}:\{\, l_i(\texttt{S}_\texttt{i}) \cdot G_i \,\}_{i\in I},$

where $\texttt{p} \neq \texttt{q}$, $I \neq \emptyset$, *and* $l_i \neq l_j$ *for all distinct* $i, j \in I$. *Furthermore,* $\mathbf{X}$ *cannot occur without a corresponding* $\mu\mathbf{X}$. □

The sorts `text`, `void`, and `date` are non-standard and are purely added for ease of use when representing the publication protocol later on. These do not influence the applicability of the theory, as `void` is equivalent to a `bool` that is always `true`, `date` can be encoded as an integer, and `text` can be encoded as a binary string in `nat`. The last term in the global type represents the choice as introduced before, denoted as a set of alternatives I. When there is only a single option, we use the simplified notation $\texttt{p}\to\texttt{q}:l(\texttt{S}) \cdot G$.

Note that the choice described above only allows for choosing between different message contents; the message will always have the same sender and recipient. It is not possible to choose between sending a message from `p` to `q` *or* `r`, for example. This prevents any confusion at the other parties: for `q` it is not possible to determine whether the message is simply late, or will not arrive at all. Furthermore, the labels of each option must be different, as otherwise it would not be possible to distinguish between the messages.

3.2 Local Types

Local types, as described before, are the representation of the protocol (captured as G) from the perspective of one of the parties. In the case of our handshake, we have two parties: `A` and `B`. From the perspective of `A`, we first send a message and then receive one of two options:

$$T_\texttt{A} = \texttt{B}!\mathit{init}(\texttt{void}) \cdot \big(\texttt{B}?\mathit{ack}(\texttt{void}) \cdot \texttt{end} \,\&\, \texttt{B}?\mathit{deny}(\texttt{void}) \cdot \texttt{end}\big),$$

where $T_\texttt{A}$ is the local type, and $\cdot$ and `end` are the same as for global types. A sent message is denoted by !, while a received message is denoted by ?. To further emphasise this difference we will write all sent messages in blue and all received messages in red.

Repetition in local types is represented exactly the same as for global types, but representing choice requires us to distinguish between internal and external choice. Consider the global type message $\texttt{p}\to\texttt{q}:\{\, a(\texttt{void}),\ b(\texttt{void}) \,\}$. Here, `p` is the party that has to choose between messages $a(\texttt{void})$ and $b(\texttt{void})$. For this party, this is an *internal* choice, represented by the operator $\oplus$. On the other hand, `q` will simply receive one of both messages and will deduce which choice was made. Therefore, this is an *external* choice, represented by the operator &.

Definition 3 Local types *are defined by the following grammar*:

$$T ::= \texttt{end} \mid \&_{i\in I}\, \texttt{p}?l_i(\texttt{S}_\texttt{i}) \cdot T_i \mid \oplus_{i\in I}\, \texttt{q}!l_i(\texttt{S}_\texttt{i}) \cdot T_i \mid \mu\mathbf{X} \cdot T \mid \mathbf{X},$$

where $I \neq \emptyset$ *and* $l_i \neq l_j$ *for all distinct* $i, j \in I$. *Furthermore,* $\mathbf{X}$ *cannot occur without a corresponding* $\mu\mathbf{X}$. □

When omitting the party names `p`, `q`, the grammar for a local type reduces to that of a binary session type [14]. We now introduce a way to construct local types from a given global type, which we will call the *projection* of a global type *onto* one of its participants. This translation might seem intuitive and easy, but has some subtleties related to choices in the global type that do not involve the participant under consideration.

In the easiest case, all continuations of the protocol after such a choice are the same. Then, we can simply ignore this particular message as it does not influence the remainder of the protocol. In more complicated cases, we require some sort of 'compatibility' between the continuations such that from the perspective of the outsider participant, the choice can be represented as a (sequence of) external choices. We capture this behaviour using the so-called *merging operator* [15], which we will not discuss in formal detail.

Definition 4 *The* set of participants *of a global type* G, *written* $\texttt{pt}\{G\}$ *is defined by the following inductive relation*:

$$\texttt{pt}\{\texttt{end}\} = \texttt{pt}\{\mathbf{X}\} = \emptyset$$
$$\texttt{pt}\{\mu\mathbf{X} \cdot G\} = \texttt{pt}\{G\}$$
$$\texttt{pt}\{\texttt{p}\to\texttt{q}:\{\, l_i(\texttt{S}_\texttt{i}) \cdot G_i \,\}_{i\in I}\} = \{\texttt{p}, \texttt{q}\} \cup \texttt{pt}\{G_i\} \ \textit{for}\ i \in I.$$ □

Definition 5 *The* projection *of a global type* G *onto a participant* `r`, *written* $G \upharpoonright \texttt{r}$, *is defined by the following inductive relation*:

$$\texttt{end} \upharpoonright \texttt{r} = \texttt{end}$$
$$\mathbf{X} \upharpoonright \texttt{r} = \mathbf{X}$$
$$(\mu\mathbf{X} \cdot G) \upharpoonright \texttt{r} = \begin{cases} \mu\mathbf{X} \cdot (G \upharpoonright \texttt{r}) & \textit{for}\ \texttt{r} \in \texttt{pt}\{G\} \\ \texttt{end} & \textit{for}\ \texttt{r} \notin \texttt{pt}\{G\} \end{cases}$$
$$\texttt{p}\to\texttt{r}:\{\, l_i(\texttt{S}_\texttt{i}) \cdot G_i \,\}_{i\in I} \upharpoonright \texttt{r} = \&_{i\in I}\big[\texttt{p}?l_i(\texttt{S}_\texttt{i}) \cdot (G_i \upharpoonright \texttt{r})\big]$$
$$\texttt{r}\to\texttt{q}:\{\, l_i(\texttt{S}_\texttt{i}) \cdot G_i \,\}_{i\in I} \upharpoonright \texttt{r} = \oplus_{i\in I}\big[\texttt{q}!l_i(\texttt{S}_\texttt{i}) \cdot (G_i \upharpoonright \texttt{r})\big]$$
$$\texttt{p}\to\texttt{q}:\{\, l_i(\texttt{S}_\texttt{i}) \cdot G_i \,\}_{i\in I} \upharpoonright \texttt{r} = \bigsqcap_{i\in I}(G_i \upharpoonright \texttt{r}) \quad \textit{for}\ \texttt{p} \neq \texttt{r} \neq \texttt{q},$$

where $\bigsqcap$ *is the merging operator. The projection is undefined in other cases that do not match this induction.* □

It turns out that the inverse operation, combining several local types into a global type, is not always possible. In fact, it can be proven that when a given set of local types can be expressed as a global type, any correct implementation of the protocol is guaranteed to terminate, and will never 'hang' or 'get stuck' [15]. This property is a consequence of the design of global and local types, encoded in their defining grammars.

3.3 Typing the Publication Protocol

With an initial understanding of MPST and its global and local types, we can use it to express the publication protocol as outlined in section 2. The full specification can be found in Listing 1. We first define the global type of the entire protocol, G. We have $\texttt{pt}\{G\} = \{\texttt{Writer}, \texttt{Publisher}, \texttt{Reviewer}\}$, as expected. After that, we see the projections of G onto each participant.

With the indentation used, we can clearly see the flow of the protocol: we start with a submission to the publisher, after which the publisher sends the draft to the reviewer to start the review process. The review process is started with a loop definition $\mu\mathbf{X}$. Each loop iteration, the reviewer will have three choices of messages to the writer: $\mathit{accept}(\texttt{void})$, $\mathit{reject}(\texttt{text})$, or $\mathit{review}(\texttt{void})$. After a review message, the writer will send a revision and the loop will start over. After rejection, the reviewer will inform the publisher. After acceptation, the reviewer will inform the publisher as well, which will then determine and communicate a publication date.

In this setup we can immediately see a weakness of MPST: it is not possible to define optional or conditional messages. For example, it would have been more concise to define only the choices $\mathit{review}(\texttt{text})$ and $\mathit{result}(\texttt{bool})$, such that the publisher would only send a publication date if the result is `true`. This is not possible in MPST; the only choice possible is between message labels, as long as they are between the same two parties.

In the projections of G, we can see how this behaviour manifests itself in internal and external choices. Only the reviewer has a local choice in this scenario; both the writer and the publisher only have an external choice. In the projection of the publisher, we can see the merging operator $\bigsqcap$ in action: the message $\texttt{Reviewer}\to\texttt{Writer}:\{\ldots\}$ in G does not involve the publisher. The merging operator will look at all possible continuations of this message, and combine them as an external choice:

- In the $\mathit{accept}(\texttt{void})$ branch, the publisher will receive a message $\mathit{accept}(\texttt{void})$ and should send a message $\mathit{pubDate}(\texttt{date})$. The projection of this branch is $\texttt{Reviewer}?\mathit{accept}(\texttt{void}) \cdot \texttt{Writer}!\mathit{pubDate}(\texttt{date}) \cdot \texttt{end}$.

$$
\begin{aligned}
G = {} & \texttt{Writer}\rightarrow\texttt{Publisher}:\mathit{submission}(\texttt{text})\cdot \\
& \texttt{Publisher}\rightarrow\texttt{Reviewer}:\mathit{draft}(\texttt{text})\cdot \\
& \mu\mathbf{X}\cdot\texttt{Reviewer}\rightarrow\texttt{Writer}:\{ \\
& \quad \mathit{accept}(\texttt{void})\cdot \\
& \quad\quad \texttt{Reviewer}\rightarrow\texttt{Publisher}:\mathit{accept}(\texttt{void})\cdot \\
& \quad\quad \texttt{Publisher}\rightarrow\texttt{Writer}:\mathit{pubDate}(\texttt{date})\cdot\texttt{end}, \\
& \quad \mathit{reject}(\texttt{void})\cdot \\
& \quad\quad \texttt{Reviewer}\rightarrow\texttt{Publisher}:\mathit{reject}(\texttt{void})\cdot\texttt{end}, \\
& \quad \mathit{review}(\texttt{text})\cdot \\
& \quad\quad \texttt{Writer}\rightarrow\texttt{Reviewer}:\mathit{revision}(\texttt{text})\cdot\mathbf{X} \\
& \}
\end{aligned}
$$

$$
\begin{aligned}
G \upharpoonright \texttt{Writer} = {} & \texttt{Publisher}!\mathit{submission}(\texttt{text})\cdot\mu\mathbf{X}\cdot(\\
& \quad \texttt{Reviewer}?\mathit{accept}(\texttt{void})\cdot\texttt{Publisher}?\mathit{pubDate}(\texttt{date})\cdot\texttt{end} \\
& \quad \&\ \texttt{Reviewer}?\mathit{reject}(\texttt{void})\cdot\texttt{end} \\
& \quad \&\ \texttt{Reviewer}?\mathit{review}(\texttt{text})\cdot\texttt{Reviewer}!\mathit{revision}(\texttt{text})\cdot\mathbf{X} \\
&)
\end{aligned}
$$

$$
\begin{aligned}
G \upharpoonright \texttt{Reviewer} = {} & \texttt{Publisher}?\mathit{draft}(\texttt{text})\cdot\mu\mathbf{X}\cdot(\\
& \quad \texttt{Writer}!\mathit{accept}(\texttt{void})\cdot\texttt{Publisher}!\mathit{accept}(\texttt{void})\cdot\texttt{end} \\
& \quad \oplus\ \texttt{Writer}!\mathit{reject}(\texttt{void})\cdot\texttt{Publisher}!\mathit{reject}(\texttt{void})\cdot\texttt{end} \\
& \quad \oplus\ \texttt{Writer}!\mathit{review}(\texttt{text})\cdot\texttt{Writer}?\mathit{revision}(\texttt{text})\cdot\mathbf{X} \\
&)
\end{aligned}
$$

$$
\begin{aligned}
G \upharpoonright \texttt{Publisher} = {} & \texttt{Writer}?\mathit{submission}(\texttt{text})\cdot\texttt{Reviewer}!\mathit{draft}(\mathit{text})\cdot(\\
& \quad \texttt{Reviewer}?\mathit{accept}(\texttt{void})\cdot\texttt{Writer}!\mathit{pubDate}(\texttt{date})\cdot\texttt{end} \\
& \quad \&\ \texttt{Reviewer}?\mathit{reject}(\texttt{void})\cdot\texttt{end} \\
&)
\end{aligned}
$$

Listing 1. The publication protocol expressed as a Multiparty Session Type. This listing includes both the global type and its projections on all participants. Processes implementing the local types have not been specified.

- In the *reject*(void) branch, the publisher will receive a message *reject*(void). The projection of this branch is Reviewer?*reject*(void).
- In the *review*(void) branch, the publisher is no longer involved. The projection is end.

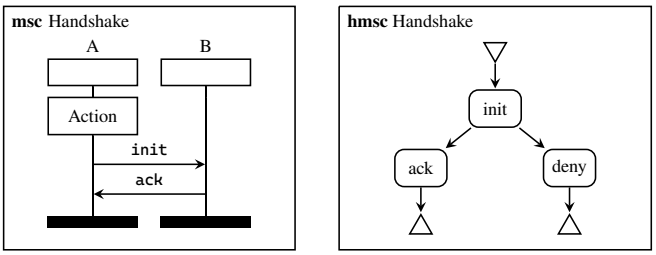
4 (High-level) Message Sequence Charts

(High-level) Message Sequence Charts (MSCs) [2, 6, 7] are a graphical approach to capturing protocols, and can be seen as a mere formalisation of the drawings many of us already make when illustrating a protocol or sequence of interactions. MPST can be directly mapped to (H)MSCs: they have similar expressive power. MSCs have been around since 1992 [2], but are limited in their expressive power: they can not capture loops or conditionals. Therefore, we will mainly discuss the extension that is the High-level Message Sequence Chart as introduced in 1996 [6], that allows for the composition of multiple MSCs [7].

4.1 Basic Message Sequence Charts

For clarity, we will call conventional MSCs *basic*. A basic MSC is a graph representing a linear communication timeline between a number of parties. It has three main components: instances, actions, and messages. We start with a number of *instances* (parties) that are represented by vertical lines. These lines represent the passage of time downwards. Furthermore, we define *actions* to be processes local to an instance. Lastly, we have *messages* between two instances with a label.

The MSC for the handshake protocol introduced before can be seen in Figure 1a. We have two parties, A and B. First, A will perform an action 'Action', after which it sends an `init` message to B. Then, B sends `ack` back. Modelling a choice, for example to return either `ack` or `deny`, is not immediately possible, but commonly requires two



a. b.

Figure 1. Example diagrams of both an (a) MSC and an (b) HMSC for the handshake.

separate diagrams illustrating both possible scenarios. Another way this might be done is through drawing 'regions' (boxes) around some interactions and marking them as alternatives. This same limitation holds for modelling repetitions.

A big advantage of MSCs is that the time dimension has been made explicit. This allows us to concretise delays and timeouts, and terminate or introduce instances throughout the lifetime of the chart. Of course, timing aspects are often implementation details rather than design choices, but they do provide more insight to the end user or software developer regarding the desired implementation.

4.2 High-level Additions

A high-level MSC makes up for the shortcomings of basic ones. They allow us to combine multiple basic MSCs into a more complex framework. The base elements in an HMSC are start and end symbols, ▽ and △, respectively, and *references*, essentially invocations of another (H)MSC. This allows for more advanced compositions, as HMSCs do allow repetition and branching.

The HMSC for the handshake protocol can be seen in Figure 1b. The respective MSCs for the 'init', 'ack', and 'deny' references are omitted, as they are only single messages. We see that the handshake protocol starts with an 'init' reference, which would be an MSC with two parties, A and B, exchanging the single 'init' message. After this, either 'ack' or 'deny' is invoked, but the HMSC makes no statements on how this is determined (it might even be up to chance); the HMSC only states what the possibilities are. After either of these two messages, the protocol terminates.

Additionally, HMSCs allow us to construct loops, by simply connecting an arrow from one reference to a previous one. To simplify notation, a *connector* is introduced, denoted by ◯. This is a node in the graph that represents a connection from each incoming edge to each outgoing edge.

4.3 Charting the Publication Protocol

As for MPST, we will express our publication protocol from section 2 into a set of (H)MSCs. The complete set of 2 HMSCs and 5 basic MSCs can be found in Figure 2. In this setup, we have split off the review interactions into its own HMSC, to keep the overview clear. The entry point of the protocol is '**hmsc** Publication'.

The 'root' MSC is very straightforward: we start with paper submission, and then we perform paper review. The submission is a short MSC, where the writer will first perform a local action 'Writing', and then submit the draft to the publisher. The subsequent review process is slightly more complicated. We start with the 'Draft' MSC, that simply represents the publisher sending the draft to the reviewer. Afterwards, we connect with a connector, that allows us to go to either 'Reject', 'Accept', or 'Revise'. These three are all simple MSCs representing the messages as outlined before. After 'Revise', we connect back to the connector, which allows us iterate.

4.4 Mapping an MPST

As indicated before, it is possible to directly map an MPST to a set of (H)MSCs. A straightforward mapping from MPST to MSC would be

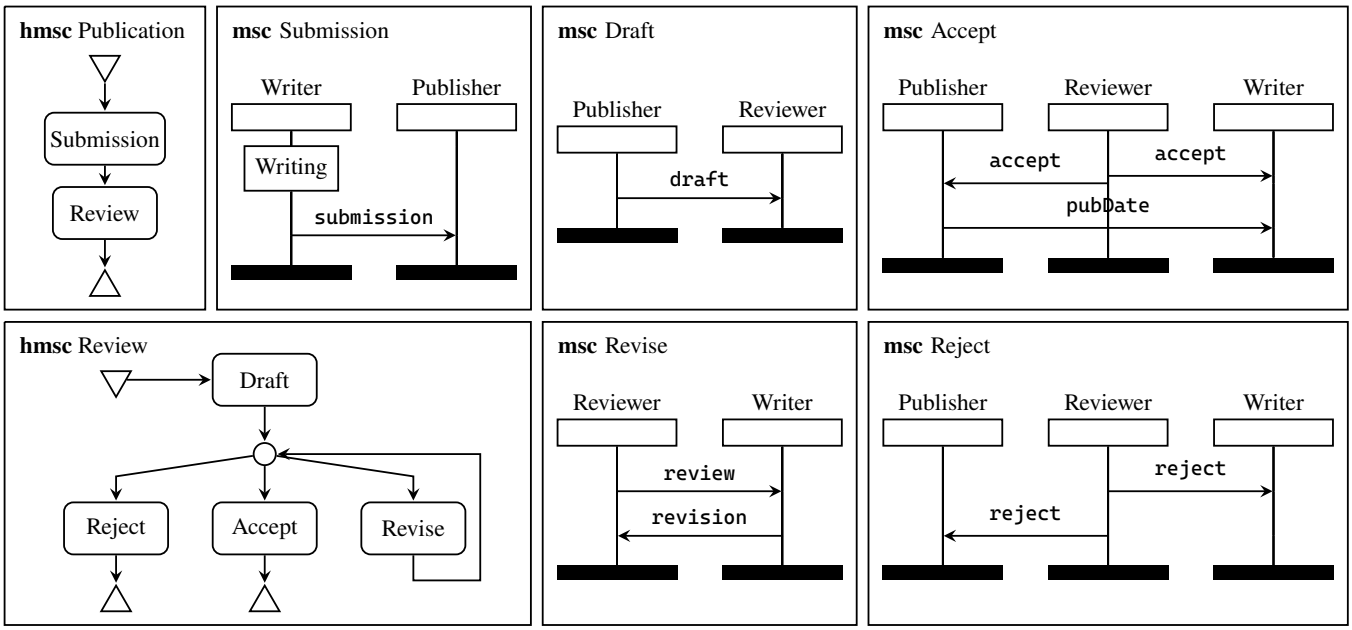


Figure 2. The publication protocol expressed in (High-level) Message Sequence Charts. The entry point of the protocol is '**hmsc** Publication'.

the following procedure. We start by creating a single HMSC with a starting node $\triangledown$, considering this our 'current' node. Then, we iterate recursively over G:

- We map **end** to $\triangle$ and finish the current branch.
- We map $\mu\mathbf{X}$ to a connector node in the HMSC, and consider that our current node.
- We map $\mathbf{X}$ by an arrow from the current node back to the respective connector node, finishing the current branch.
- We map $\mathtt{p}\rightarrow\mathtt{q}:l(\mathtt{S})$ to a new MSC reference, and consider that our current node. The MSC we refer to only contains this single message.
- We map $\mathtt{p}\rightarrow\mathtt{q}:\{\ l_i(\mathtt{S_i})\ \cdot\ G_i\ \}_{i\in I}$ to n different MSC references (I having n elements), such that for each option in this choice we obtain a branch in the HMSC. For each i, we continue the algorithm for G_i.

At the end, we go over our HMSC and merge MSC references where possible and appropriate. We could extract a sub-HMSC if that makes sense, dependent on the semantics of our protocol.

Unfortunately, the reverse operation is not as easy. Since MSCs put very little to no constraints on the actual messages one puts in, it is easy to design a protocol that would never work. While MPST can provide a guarantee that a protocol will terminate, an MSC cannot.

5 Blindingly Simple Protocol Language

The Blindingly Simple Protocol Language (BSPL) [12] is a more recent formalism, focused on capturing meaning, rather than details of the operations of the protocol. While in a way MPST and MSC characterise a protocol by the structure and sequence of its messages, BSPL leaves structure and sequence implicit: the protocol is characterised by its messages, their meaning, and the emergent behaviour produced by the protocol.

Essentially, this is similar to the difference between an imperative (MPST/MSC) and a declarative (BSPL) computer program. In an imperative program, one states what actions need to be taken, in what order, and in what way. In a declarative program, one only specifies the unknowns to be computed, together with all relevant information such as function definitions. The computer will then find out how to actually compute the desired result.

What makes this approach simple and powerful, is that only two main constructs are required: the definition of a protocol with messages, and the composition of such protocols. The order of messages is purely derived from the specifications in these protocols. BSPL upholds several distinguishing principles [12]:

- Each protocol has two or more *roles* and one or more *parameters*. A role represents one party in the protocol, and a parameter represents a piece of information that is established and agreed upon by all parties during execution.
- There are no hidden flows of information, and the order of operations can be deduced from the specified information flows.
- No notion of state is necessary, as all relevant information is made explicit in the protocol; a party's business logic is irrelevant.
- Each protocol instance is unique, as some or all of the parameters can together define a key, uniquely representing a single instance of the protocol.

5.1 Defining a BSPL Protocol

We demonstrate the construction of a BSPL protocol, again using the handshake. We will introduce the basic concepts: the protocol itself, its *public interface*, and some basic messages.

The handshake expressed in BSPL.

```
1 Handshake {
2     role A, B
3     parameter out ID key, out result
4
5     A -> B: init[out ID]
6
7     B -> A: ack[in ID, out result]
8     B -> A: deny[in ID, out result]
9 }
```

The protocol starts off with Handshake as the *protocol name*. Within the *public interface* that follows, two `role`s are declared: `A` and `B`. Below that, the `parameter`s are given, of which at least one is a `key`. These `key` parameters together define the unique instance of the protocol.

Parameters can be preceded by either an `in`, `out`, or `nil` prefix. `in` means that the parameter must be instantiated (by another protocol) **before** the protocol can be executed, whereas `out` means that the parameter will be instantiated **during** the execution. The use of `nil` is crucial in some constructions, as it allows for a parameter to either be `in` or `out`. Lastly, a parameter can be suffixed `nilable`, meaning that it does not have to be instantiated during execution.

The roles and parameters are followed by one or more *references*, which can be either a message or an invocation of another protocol. At least one of the parameters of such a reference should be denoted as `key` in the public interface of the source protocol, to allow linking the nested protocol instance to the source protocol instance.

In this example, we have three messages. The first message, `init[]`, is sent from role `A` to role `B`. It produces an `ID` parameter to uniquely identify the handshake. In the next step, `B` can choose to either `ack[]` or `deny[]` the handshake. As `out` parameters can only be instantiated once, exactly one of these messages can be sent within an instance of the protocol: they are mutually exclusive. Once all `out` parameters listed in the public interface are instantiated, a protocol instance can terminate. In this case, this is when the `out result` parameter has been instantiated using either `ack[]` or `deny[]`.

Parameters that are not listed in the public interface, but are still used in message definitions or protocol references, are called *local parameters*. These parameters do not have to be instantiated during execution of a protocol, only the parameters in the public interface impact termination. Consequently, even if the intended final message in a protocol might normally not carry any data, but would merely signal a result (such as the `ack[]` or `deny[]` in the handshake), an arbitrary `out` parameter is still required to ensure these messages will be executed. In this case, the `out result` is only present to prevent early termination, but does not carry any information.

A message might require an `in` parameter that is yielded as an `out` parameter by another message. This is what enforces the order of operations within the protocol, e.g. the `ack[]` or `deny[]` messages cannot be sent before the `init[]` message is sent. This also allows for composition of protocols: a protocol that lists a parameter as `in` in its public interface can be included in another protocol that has a message or other protocol instantiating this parameter.

5.2 Declaring the Publication Protocol

Listing 2 shows the BSPL representation of the publication protocol from section 2. We first define the Review protocol, which is strictly between the roles Reviewer and Writer. This process can play out in two ways. The first way is that the Reviewer produces a `review[]`, telling the Writer to adjust their draft and sending along comments. The Writer responds with a `revision[]` message which yields a `newDraft`. While in our description we could loop back to the `review[]` step, this setup in BSPL only allows for a single review round: after a review, the parameter `comments` has been instantiated.

The reviewer then sends the Writer a `result[]` message, yielding a boolean `out accepted` parameter indicating whether the draft has been accepted or not. This message can just as well be sent immediately, without the `review[]` and `revision[]` interactions, as designed. This illustrates how choice can be implemented. Note that the `revision[]` message is dependent on the `out comments` from `review[]`. However, it would be possible for the reviewer to send a `result[]` message after sending a `review[]`, without waiting for the `revision[]`. In BSPL, it is not possible to prevent this.

This Review protocol is used in the composed protocol Publication, which plays out between a Reviewer, Writer, and Publisher. The Writer sends a `submission[]` message with corresponding `subNo` and `draft` to the Publisher, which the Publisher in turn sends to the Reviewer along with a `reviewId` to allow for Review protocol instantiation.

At this point, the Review protocol plays out, in order to obtain the `accepted` parameter. This parameter is only known to the Writer and Reviewer, so we need an additional message to the Publisher. This message cannot send `in accepted`, as then the message would not be required. Therefore, we introduce the parameter `out result`, containing the same information. In fact, we choose to introduce two separate messages, one for `accepted[]` and one for `rejected[]`. The latter contains an additional `out terminated`, allowing the protocol to terminate, while the first does not and thus requires the `pubDate[]` message to be sent before termination.

```
Review {
    # The review process only concerns interaction between the reviewer and the writer
    role Reviewer, Writer

    # The review process takes a reviewId identifying the review session, and emits a boolean result
    parameter in reviewId key, out result

    # The repetitive behaviour of this process cannot be represented in BSPL. With this setup, only a single review round can be done.
    Reviewer -> Writer: review[in reviewId, out comments]
    Writer -> Reviewer: revision[in reviewId, in comments, out newDraft]

    # The reviewer can choose to either send a review or a result to the writer
    Reviewer -> Writer: result[in reviewId, out result]
}

Publication {
    role Reviewer, Writer, Publisher
    parameter out subNo key, out reviewId key, out result, out terminated, out pubDate nilable

    # The publication process depends on the review process, which can only be invoked once the review message is sent
    Review(Reviewer, Writer, in reviewId, out result)

    # The writer produces a submission number by submitting their draft
    Writer -> Publisher: submission[out subNo, out draft]

    Publisher -> Reviewer: review[out reviewId, in draft]
    Reviewer -> Publisher: accepted[in reviewId, in result]
    Reviewer -> Publisher: rejected[in reviewId, in result, out terminated]
    Publisher -> Writer: pubDate[in subNo, out terminated, out pubDate]
}
```

Listing 2. The publication protocol expressed in the Blindingly Simple Protocol Language, where the review process has been extracted as a smaller, independent protocol.

5.3 Working with BSPL Protocols

As already alluded to before, certain features or properties of BSPL can at first be difficult to comprehend or interpret properly. Most of these confusions seem to arise from the use of the `in` and `out` keywords, the meaning of which might not be immediately obvious in this context. Upon reading the statement `A -> B: message[in ID, out info]`, it would not be out of the ordinary to think of `message[]` as some kind of interaction between `A` and `B`, where `A` puts in an `ID`, and somehow the `info` arises out of that interaction, similar to a function call in a programming language.

However, that is of course not the case. The `out` keyword only describes that `A` should come up with an appropriate value for this parameter. For an external observer, the actual message exchanged between both parties would be identical to one with `in info`. Essentially, the distinction

between `in` and `out` is only a vehicle to declare an implicit order dependence between messages, and does not say anything about the semantics of the messages.

This same detail manifests itself in the use of `out` parameters purely to prevent early termination. When designing in BSPL, one will quickly introduce many 'control' parameters that do not contribute to the flow of information or the meaning of a protocol, but only serve to ensure proper functionality. In the publication protocol, the parameters `reviewId` and `terminated` serve such a role. This role can also be taken by otherwise meaningful parameters, being included in other messages just to control the flow, without contributing to the semantics of the messages. This takes place in the Review protocol, where `comments` is sent back to the Reviewer after revision, in order to ensure the order of these messages.

6 Comparative Analysis

Finally, we can compare the three formalisms we discussed above. We will compare them on various points, highlighting the differences and reiterating findings from expressing the publication protocol.

6.1 Imperative vs Declarative

The main identifying property of BSPL is its declarative approach, whereas MPST and MSC use an imperative approach. In an imperative approach, the protocol is defined by the structure of its messages. Such an approach is excellent for a precise specification and rigorous analysis of a protocol; it allows us to validate the correctness of implementations of the protocol. This is crucial for foundational protocols such as TCP, HTTPS, and IPv4.

However, such mathematical precision is not always required or even desired. When designing more abstract protocols, such as the interaction between a bank and a customer, we move away from a simple message exchange, and rather talk about information that needs to be exchanged, and what information should end up where. We would define 'messages' such as `Sender→Receiver: ship[ ... ]` [12], that encapsulate an entire complex chain of interactions that can take place over multiple days. We do not require that such a message succeeds; it is a business operation and we would call the shipping company when a package does not arrive.

For such a situation, a declarative approach might be better. In this case, the distinction between protocol messages as pure pieces of information, and the message composition as the behaviour of the protocol, is less clear. The information to be sent also says something about how and when it should be sent. While this can pose problems for protocols such as TCP and HTTPS, it can be a good fit for business processes, where the focus will likely be on the information flows in the bigger picture as opposed to the technical (implementation) details of a protocol. The uniqueness of each protocol instance by means of the key adds to this; a BSPL protocol models a (unique) business process instead of a precise sequence of messages.

A side-effect of the declarative approach is the use of a global state for all parameters. This makes it very easy for the problem of *non-local choice* [5] to emerge: it is perfectly legal in BSPL to let party `B` send a message with a parameter that party `A` came up with, without sending it to `B`. This is especially relevant in protocols with more than two roles.

6.2 Concurrency

Another important differentiating factor is concurrency. Especially for larger protocols, it might be that multiple parties work together in parallel, and messages might be exchanged simultaneously, before 'synchronising' again. Such behaviour is easy to encode in MSCs, and full MPST supports this as well. However, the formal analysis of such protocols is tedious [4, 7].

In principle, BSPL allows for concurrency as well, but only implicitly. For example, two independent messages might be exchanged at the same time, or an entire sequence of messages might be performed in parallel. However, it is not possible to explicitly specify such behaviour; it is up to the implementation to achieve any concurrency.

6.3 Visualisation

Clearly, MSC is the most visual formalism we discussed, as it is the only one. The explicit inclusion of the concept of time makes MSC very expressive, allowing a wide range of protocols and interactions to be modelled. A graphical depiction is arguably clearer than a textual representation such as BSPL and MPST. However, MSCs can quickly become difficult to work with, as can already be seen in the implementation of the publication protocol in Figure 2. For more complicated protocols, the number of short MSCs will increase rapidly, cluttering the overview gained by using MSCs to begin with.

Since MPST can easily be mapped to a set of MSCs, we consider it to allow for decent visualisation as well. BSPL, on the other hand, cannot easily be visualised. Due to the fact that the order of operations is made implicit, it is hard to draw a diagram such as an MSC. This is an inherent difficulty with declarative approaches that is not unique to BSPL.

6.4 Expressive Power

A last point of differentiation between the formalisms is their expressive power. The expressive power of a formalism determines what class of protocols can be expressed with them, and is a deciding factor in choosing a suitable formalism.

MPST is the least 'free' formalism: all interactions have to adhere to strict rules and constraints. On the other hand, this does offer us great options for analysis. Local types can be used to prove the correctness of implementations, and can even help guarantee termination of our protocol. Furthermore, MPST is the only formalism that does not have the problem of non-local choice [5], as it prevents this by design of its grammar - as long as the local projections for all parties are defined.

MSCs have a great expressive power, as there are almost no restrictions on the charts. The explicit consideration of time allows expressing implementation details of a protocol, such as timeouts, exponential back-offs, and delays. Furthermore, they allow us to represent local computation in 'Actions', no other formalism allows us to do so explicitly. However, this lack of restrictions also allows us to make nonsensical protocols. MSCs do not have inherent correctness properties.

Lastly, BSPL, too, has a large expressive power, but in a different domain. It is fairly difficult to say a lot about individual messages between parties, but of the three options, only BSPL allows us to express the 'meaning' of a protocol, encoding not only single messages but potentially entire business processes.

7 Conclusions

After all the consideration and comparison above, we can say that there is no clear 'best' option when designing a protocol. We would argue that one should choose between either BSPL, or a combination of MPST and MSCs. The best choice depends on the purpose and scope of the protocol under consideration.

We think that MPST and MSCs complement each other perfectly; with MPST serving as a tool to formally analyse a protocol and say things about its inherent structure, while MSC can serve as a method to make the protocol more accessible and encode implementation details and guidance straight in the protocol itself, further simplifying the implementation and reducing the number of implicit assumptions.

On the other hand, BSPL offers unique possibilities with its declarative approach. It allows the designer to think on a higher level of abstraction, focusing on the information flow and the added value of a protocol as a whole. The declarative notation is clear and straightforward; no complex notation is necessary. Especially for relatively straightforward communications between just two parties, eliminating the issue of non-local choice, BSPL is well-suited.

However, BSPL does have its limitations: it does not allow elementary constructs such as a loop, and confuses between protocol behaviour and message data. It tries to lift to a higher level of abstraction, but does

not reach it by still being centred around discrete messages and one-off interactions. This leaves us between a rock and a hard place: we still have to deal with the order of messages, but we cannot intuitively do so because we are at a too high abstraction level.

8 Alternatives

For BSPL, it seems like it just misses the right balance between abstraction and concretisation. Perhaps a better approach would be even more abstract, almost closer to business descriptions. One in which we could define processes that play out over multiple parties, such that we only need to be concerned with what information we need, and what the dependencies between these pieces of information are. However, such description would not benefit from a very formal format such as BSPL.

Another approach would be to try and overcome the shortcomings in BSPL: to allow control over the order of operations and discrete messages, while also re-instantiating the distinction between information and flow control. This is exactly what Bliss [13] aims for: it extends BSPL, while introducing a clear distinction between five types of parameters: **1.** key parameters, **2.** payload parameters, **3.** completion parameters, **4.** integrity parameters, and **5.** control parameters.

For MPST, a wide range of alternative type theories and formalisms are available [1]. These all have different 'specialties', dependent upon the requirements for the protocol. For example, MPST can be extended with so-called trace-based semantics, that offer even greater correctness guarantees to a protocol [9].

Lastly, for MSC, a well-known alternative is UML [11]. UML is a large specification, covering many aspects of software design besides just protocols. Therefore, UML is an attractive alternative: it allows one to incorporate the protocol in a larger system design. Furthermore, the differences between UML and MSC are small and mostly about syntax [3]. It is not surprising then, that work has been done to pursue a unification of the two [10].

References

[1] J. Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335(2-3):131–146, May 2005. doi: 10.1016/j.tcs.2004.07.036

[2] Ø. Haugen. Using MSC-92 effectively. In R. Bræk and A. Sarma, eds., *SDL '95 with MSC in Case*, pp. 37–49. Elsevier, Amsterdam, Jan. 1995. doi: 10.1016/B978-0-444-82269-7.50008-3

[3] Ø. Haugen. Comparing UML 2.0 Interactions and MSC-2000. In D. Amyot and A. W. Williams, eds., *System Analysis and Modeling*, Lecture Notes in Computer Science, pp. 65–79. Springer, Berlin, Heidelberg, 2005. doi: 10.1007/978-3-540-31810-1_5

[4] K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. In *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '08, pp. 273–284. Association for Computing Machinery, New York, NY, USA, Jan. 2008. doi: 10.1145/1328438.1328472

[5] P. B. Ladkin and S. Leue. Interpreting Message Flow Graphs. *Formal Aspects of Computing*, 7(5):473–509, Sept. 1995. doi: 10.1007/BF01211629

[6] S. Mauw. Message Sequence Chart (MSC). *Draft Recommandation Z. 120*, p. 74, 1996.

[7] S. Mauw and M. A. Reniers. High-level Message Sequence Charts. In A. Cavalli and A. Sarma, eds., *SDL '97: Time for Testing*, pp. 291–306. Elsevier Science B.V., Amsterdam, Jan. 1997. doi: 10.1016/B978-044482816-3/50020-4

[8] R. Milner. *A Calculus of Communicating Systems*. Lecture Notes in Computer Science. Springer-Verlag, Berlin Heidelberg, 1980. doi: 10.1007/3-540-10235-3

[9] L. Padovani, M. Dezani-Ciancaglini, and G. Castagna. On Global Types and Multi-Party Session. *Logical Methods in Computer Science*, Volume 8, Issue 1, Mar. 2012. doi: 10.2168/LMCS-8(1:24)2012

[10] E. Rudolph, J. Grabowski, and P. Graubmann. Towards a Harmonization of UML-Sequence Diagrams and MSC. In R. Dssouli, G. v. Bochmann, and Y. Lahav, eds., *SDL '99*, pp. 193–208. Elsevier Science B.V., Amsterdam, Jan. 1999. doi: 10.1016/B978-044450228-5/50014-X

[11] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language Reference Manual, The (2nd Edition)*. Pearson Higher Education, 2004.

[12] M. P. Singh. Information-driven interaction-oriented programming: BSPL, the blindingly simple protocol language. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '11, pp. 491–498. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, May 2011.

[13] M. P. Singh. Bliss: Specifying Declarative Service Protocols. In *2014 IEEE International Conference on Services Computing*, pp. 235–242, June 2014. doi: 10.1109/SCC.2014.39

[14] K. Takeuchi, K. Honda, and M. Kubo. An interaction-based language and its typing system. In C. Halatsis, D. Maritsas, G. Philokyprou, and S. Theodoridis, eds., *PARLE'94 Parallel Architectures and Languages Europe*, Lecture Notes in Computer Science, pp. 398–413. Springer, Berlin, Heidelberg, 1994. doi: 10.1007/3-540-58184-7_118

[15] N. Yoshida and L. Gheri. A Very Gentle Introduction to Multiparty Session Types. In D. V. Hung and M. D´Souza, eds., *Distributed Computing and Internet Technology*, vol. 11969, pp. 73–93. Springer International Publishing, Cham, 2020. doi: 10.1007/978-3-030-36987-3_5

[16] H. Zimmermann. OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications*, 28(4):425–432, Apr. 1980. doi: 10.1109/TCOM.1980.1094702

university of groningen / faculty of science and engineering / computing science