ANNALES MATHEMATICAE ET INFORMATICAE

VOLUME 58. (2023)

INSTITUTE OF MATHEMATICS AND INFORMATICS
ESZTERHÁZY KÁROLY CATHOLIC UNIVERSITY
HUNGARY, EGER

Selected papers of the 12th International Conference on Applied Informatics

HU ISSN 1787-6117 (Online)

A kiadásért felelős az
Eszterházy Károly Katolikus Egyetem rektora
Megjelent a Líceum Kiadó gondozásában
Kiadóvezető: Dr. Nagy Andor
Műszaki szerkesztő: Dr. Tómács Tibor
Megjelent: 2023. november

Contents

Annales Mathematicae et Informaticae
58 (2023) pp. 1–8
DOI: 10.33039/ami.2023.08.013
URL: https://ami.uni-eszterhazy.hu

eHealth and Smart Solutions framework for health monitoring in the course of the pandemic

János Dávid Balogh[a], **Attila Adamkó**[b]

[a]Doctoral School of Informatics University of Debrecen, Debrecen, Hungary
balogh.janos.david.official@gmail.com

[b]Department of Information Technology University of Debrecen, Debrecen, Hungary
adamko.attila@inf.unideb.hu

Abstract. Our main objective focuses on the different design principles for applications that can receive and store various data from smart devices, e.g., smart bands, smart watches, and smart homes, and could gain more information about its users to some extent to their health. We would like to collect this information, store it, then create understandable diagrams and show them to the user in a modern, responsive environment. We placed great emphasis on the underlying architecture, which holds the necessary features (scalability and extensibility). We have followed the design process's standards, recommendations, and protocols. The result is an application framework that fulfills the duty of an XXI. century smart solutions application that could monitor people's health and even help them live healthier lives by avoiding chronic disorders, e.g., obesity and/or diabetes. We focused on predicting and classifying COVID-19 disease according to the collected information and research.

Keywords: smart solutions, smart devices, healthcare, architecture, data gathering, data analysis, machine learning, covid, data model

AMS Subject Classification: AMS Subject Classifications

1. Data collection

Problem 1.1 (Data collection)**.** *The first problem is finding a solution to gather health data from various IoT devices and send them to the main application. Every device has its own structure, so we had to find a way to handle that.*

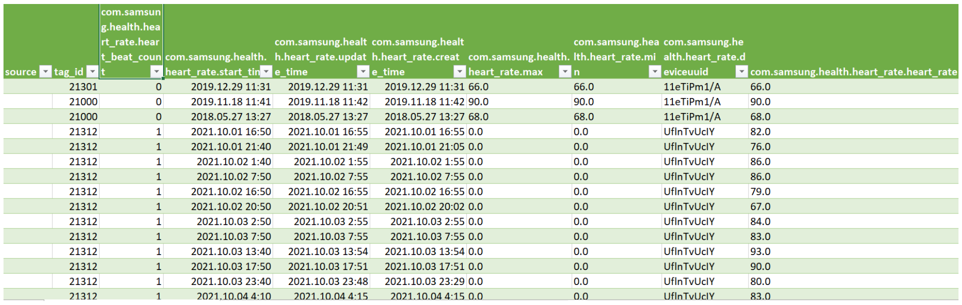
Submitted: July 30, 2023
Accepted: August 7, 2023
Published online: August 30, 2023

1.1. Smartwatch with SDK

Many smartwatches have more precise sensors to measure heart rate, distance, stress level, and much more health information. These watches come out with operating systems from the factory, and the manufacturers provide SDK. We could develop an application that collects the above-mentioned health information over time into one file and send it to our architecture via a REST API endpoint.

Also, these devices have default smartphone applications where we can export data easily without our own application, and the users can share manually with us.

Example 1.2. The Figure 1 shows the data we collected from our test device, a Samsung Galaxy Watch 2. This spreadsheet focuses on heartbeat rate and displays minimum and maximum heartbeat and the current measurements. Besides the measurements, it informs us of the measurement date, device, and more useful health information.



source	tag_id	com.samsung.health.heart_rate.heart_beat_count	com.samsung.health.heart_rate.start_tin	com.samsung.health.heart_rate.update_time	com.samsung.health.heart_rate.create_time	com.samsung.health.heart_rate.max	com.samsung.health.heart_rate.min	com.samsung.health.heart_rate.deviceuuid	com.samsung.health.heart_rate.heart_rate
	21301	0	2019.12.29 11:31	2019.12.29 11:31	2019.12.29 11:31	66.0	66.0	11eTiPm1/A	66.0
	21000	0	2019.11.18 11:41	2019.11.18 11:42	2019.11.18 11:42	90.0	90.0	11eTiPm1/A	90.0
	21000	0	2018.05.27 13:27	2018.05.27 13:27	2018.05.27 13:27	68.0	68.0	11eTiPm1/A	68.0
	21312	1	2021.10.01 16:50	2021.10.01 16:55	2021.10.01 16:55	0.0	0.0	UflnTvUclY	82.0
	21312	1	2021.10.01 21:40	2021.10.01 21:49	2021.10.01 21:05	0.0	0.0	UflnTvUclY	76.0
	21312	1	2021.10.02 1:40	2021.10.02 1:55	2021.10.02 1:55	0.0	0.0	UflnTvUclY	86.0
	21312	1	2021.10.02 7:50	2021.10.02 7:55	2021.10.02 7:55	0.0	0.0	UflnTvUclY	86.0
	21312	1	2021.10.02 16:50	2021.10.02 16:55	2021.10.02 16:55	0.0	0.0	UflnTvUclY	79.0
	21312	1	2021.10.02 20:50	2021.10.02 20:51	2021.10.02 20:02	0.0	0.0	UflnTvUclY	67.0
	21312	1	2021.10.03 2:50	2021.10.03 2:55	2021.10.03 2:55	0.0	0.0	UflnTvUclY	84.0
	21312	1	2021.10.03 7:50	2021.10.03 7:55	2021.10.03 7:55	0.0	0.0	UflnTvUclY	83.0
	21312	1	2021.10.03 13:40	2021.10.03 13:54	2021.10.03 13:54	0.0	0.0	UflnTvUclY	93.0
	21312	1	2021.10.03 17:50	2021.10.03 17:51	2021.10.03 17:51	0.0	0.0	UflnTvUclY	90.0
	21312	1	2021.10.03 23:40	2021.10.03 23:48	2021.10.03 23:29	0.0	0.0	UflnTvUclY	80.0
	21312	1	2021.10.04 4:10	2021.10.04 4:15	2021.10.04 4:15	0.0	0.0	UflnTvUclY	83.0

Figure 1. Health information from smartwatch.

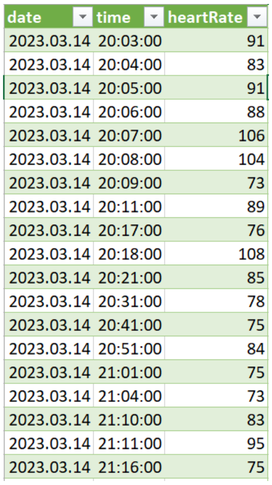
1.2. Smart bands

We talked about the smartwatch solution, but these devices still are more expensive than an average person could afford. That is why we shifted our focus to smart bands, which are more affordable for larger crowds.

These devices are also equipped with sensors, maybe not the same precision level as the smartwatch sensors, but they still could get information about the health information we want to have.

The greatest disadvantage of smart bands is the lack of an open operating system and SDK. We had to find a solution to collect the data from the devices. Modern Xiaomi and Huawei devices offer a REST API endpoint where we authorize ourselves and get the requested information. We used that endpoint in an application where the user just adds their API key and authorization information so we could get our hands on the data and send it to our architecture.

Example 1.3. As we can see in the Figure 2, the smart bands send less information about the heartbeat rate compared to the smartwatch, but we still get what is necessary, the date and the measurements, which are crucial to our research.



date	time	heartRate
2023.03.14	20:03:00	91
2023.03.14	20:04:00	83
2023.03.14	20:05:00	91
2023.03.14	20:06:00	88
2023.03.14	20:07:00	106
2023.03.14	20:08:00	104
2023.03.14	20:09:00	73
2023.03.14	20:11:00	89
2023.03.14	20:17:00	76
2023.03.14	20:18:00	108
2023.03.14	20:21:00	85
2023.03.14	20:31:00	78
2023.03.14	20:41:00	75
2023.03.14	20:51:00	84
2023.03.14	21:01:00	75
2023.03.14	21:04:00	73
2023.03.14	21:10:00	83
2023.03.14	21:11:00	95
2023.03.14	21:16:00	75

Figure 2. Health information from smartband.

1.3. Renaming

As we work with many types of devices, we are bumping into one great obstacle. Different devices have various data storing structures containing the same type of data, but the structure is different. If we would like to work with them effectively, we should get them brought into a common structure. The above-mentioned figures show what the problem is, the smartwatch heartbeat rate measurement has a long specific name, but the smart band heartbeat rate is a simple one. We dedicated a service which is between the receiving endpoint and the storing to handle these various formats and produce them into our structure, but we do not lose important information.

2. Database and storing

2.1. Database

We were taking account of the difference of smart devices and sensors, they gather and deliver more or less the same data structure, but we wanted to make sure that the varying data structure will not cause problems in storing these data. That is the reason why we decided to use NoSQL database instead of the standard regular SQL databases. The choice of ours was the MongoDB. It offers a flexible document data model database which stores data in JSON format. Ad-hoc queries and secondary indexing are supported which makes truly powerful ways to access our data. The database does not contain any business logic, on how to process the data. It has only one job and it fulfills that perfectly. In our application we implemented a service which handles the repository. The repository has the connection with the database and is responsible for the CRUD operations and various queries This book [4] guided us to implement a data model which will contain our sensitive data and is memory effective.

2.2. Sorting out false measurements

When we are wearing the smart devices during our life the devices automatically measure our health and sometimes they could get a false information when the sensors are not touching well the user or any kind of problem. Iif this happens they could record abnormally high heartbeat rae or zero value and we cannot work with these false information, we need to find a way to get rid of these data.

We used the Simple Moving Average (SMA) to find out false measurements. SMA attaches a value to every i-th element using the average of suum of all of i elements. When this value is too high or low we label it as false measurements and do not work with anymore.

Example 2.1. As the Figure 3 shows it means a lot when we are using SMA. The red line represents the original measurements with false values and it clearly points out what happens when it contains false information. The big spike is when the heartbeat rate is too high and the downhill when it failed to measure any heartbate. The blue line is after SMA and it is a more consistent representation of the measurements so we could work with valid and supposedly true values after storing in the database.

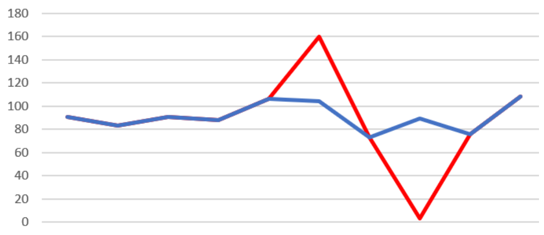


Figure 3. Diagram showing the sorting out.

3. Architecture

3.1. Goals

As we stated before one of our goals is a flexible data model which could be a base of a general data managing platform. For this purpose, we relied on NoSQL database MongoDB that is a document data model storing data in BSON objects. A BSON object is a binary JSON object, this format helps in data storage and network transfer. We could store sensor data with different structure easily and even process them. In the backend we used the bson package and the Document data type to work with the database.

Our other goal was to create an architecture which satisfies our needs. We wanted to design a layered architecture that is scalable, modular, expandable.

Scalable, we would like to work with larger data inputs and more users so we must be ready to receive more requests. To do that we must scale our architecture without a problem, or we would suffer heavy data loss.

Modular, anytime we can decide which module is needed or not. We could wire or unwire any module with ease.

Expandable, if any new requirements come in the picture, we could meet them by adding new features and modules to the architecture.

For fulfilling these properties, we decided to implement a microservice architecture. In our project we expanded it to see how it will work out. We studied [1] to get more understand in modern software architecture engineering and [6] to learn developing in Spring Boot which will support our goals.

3.2. Microservices

The microservices architecture is a collection of services that are highly maintainable and testable and independently deployable. Spring offers an opportunity if we would like to wire many services together. All our services share the same database, but they have different endpoints with REST API interface and business logic.

We chose this architecture because it is easy to expand later when we are adding more and more services to the application.

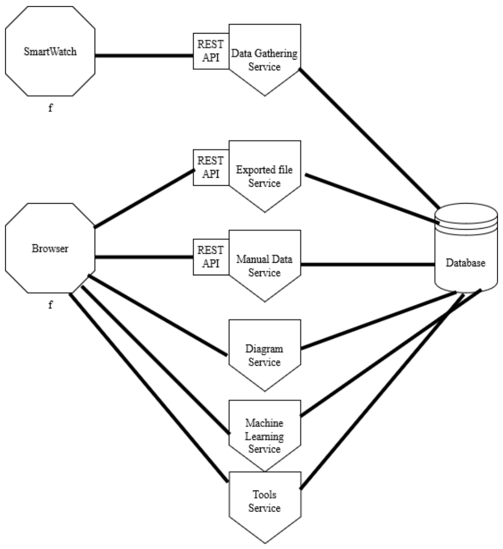


Figure 4. The architecture.

As the Figure 4 shows, we have two external entry point: the smart watch and a browser. The smart watch sends the data through the REST API interface and its service to the database

The browser can send data as exported file or/and manually. We provided different services to them, and they handle the rest to the database. We query the database in the creating diagrams, tools and machine learning services and their results are shown in the browser.

3.3. Creating graphs

After we got data from the users and they are stored in our database, finally we could work with them. First, we need to check if they are normalized. It means none of them has a different format. If they do, we have to normalize them. The main goal of the application is to provide value added services, give feedback and motivation about the lifestyle change. For this reason, our illustrations are easily readable. To create diagrams, we used the R script language. R is famous for its data processing and plotting abilities, it is widely used in analytic, statistic works, and Spring can invoke third party programs. Our script needs a dataset to work with it. For this purpose, we wrote a query which collects the user's relevant data from the database. The script waits for a csv file, so a simple CSV writer class writes the result into a csv file. The newly created csv file is passed into the script as argument and runs. Creates diagrams such as calorie burn by date, heart rate rating, graph of steps, calorie intake and burn by date and many more. These diagrams are exported as images, and they are shown to the user by the frontend part of the application.

3.4. Machine learning

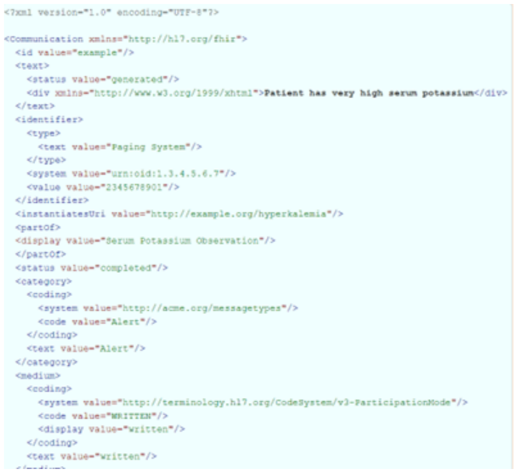
The greatest feature should be our machine learning classification which will be using our collected data to classifiy the user or the patient if they are infected with any disease. Our personal approach was from our experience with the COVID-19 which caused lots of loss and trouble in our world. [5] helped ouur research to get started to develop a model in machine learning in Python. The architecture will pass the input values to the script and call it and will be waiting for it's output. We used the most popular algorithms to train our model and test it with the data that we had and it looks like it was satisfying. According to this research [2] we saw that if a person with COVID-19 produces more heartbeat even if the person is sitting or in resting position. Because we collected heartbeat rate and more health information we could classify if the user is dealing with COVID-19. This works as a binary classification where we import the user's data which contains heartbeat rate and activity position and we calculate average heartbeat by day when they are resting or sitting. The average heartbeats by days will be used as input data and if they show increasing values day by day and they are out of normal person heartbeat rate in resting position interval we could suppose they are infected with the disease, we label them with true value. If the average values within the interval and shows no increasing values they are healthy and we label them with false value.

In the future we would like to work with infected people to wear smart devices to collect more precise information about their heartbeat and symptons to improve our machine learning model to predict their condition.

3.5. HL7

Because our project works with healthcare, we found it worthwhile to use the Health Level Seven standards. The Health Level 7 standards are a set of international standards for transfer of clinical and administrative data between software application used by various healthcare providers.

The name comes from using the application layer which is layer 7 in the OSI model. The HL7 Standards are produced by Health Level Seven international and used by bodies e.g., American National Standards institute and International Organization for Standards. [3] showed a way to implement and use the HL7 standards to achieve communication between our architecture and healthcare providers.



```xml
<?xml version="1.0" encoding="UTF-8"?>

<Communication xmlns="http://hl7.org/fhir">
  <id value="example"/>
  <text>
    <status value="generated"/>
    <div xmlns="http://www.w3.org/1999/xhtml">Patient has very high serum potassium</div>
  </text>
  <identifier>
    <type>
      <text value="Paging System"/>
    </type>
    <system value="urn:oid:1.3.4.5.6.7"/>
    <value value="2345678901"/>
  </identifier>
  <instantiatesUri value="http://example.org/hyperkalemia"/>
  <partOf>
  <display value="Serum Potassium Observation"/>
  </partOf>
  <status value="completed"/>
  <category>
    <coding>
      <system value="http://acme.org/messagetypes"/>
      <code value="Alert"/>
    </coding>
    <text value="Alert"/>
  </category>
  <medium>
    <coding>
      <system value="http://terminology.hl7.org/CodeSystem/v3-ParticipationMode"/>
      <code value="WRITTEN"/>
      <display value="written"/>
    </coding>
    <text value="written"/>
```

Figure 5. Example XML for HL7.

We dedicated for a HL7 a new service which satisfies the standards for communication between healthcare software applications. It is heavily relied on XML documents because HL7 newer versions store the necessary data in it. It includes the patient's ID and status like body weight or other measurements such as heart rate.

We want to develop a service in our general architecture where medical approaches could benefit from it, helping the digitalization of healthcare. We are still working on it to be more precise with the data output from the smart watches, but we have already found a way to process data from them and convert them into XML to use HL7 Standards.

We are trying to build a relationship with the Clinic of the University of Debrecen to see how will work our solution in live environment where doctors take part in our research. They will get 24/7 healthcare information about the patient and if any trouble could occour they could take steps to prevent disease like the COVID or give aid if necessary.

4. Conclusion

While developing this application we gained more knowledge and insight into modern application architecture, responsible websites and even healthcare a bit. We would like to invest more in this project because we are dedicated in our goals to improve people's health using the instruments of technology.

We are planning to invite more people to the project using different smart devices to collect more data. If we have more data, we can continue the machine learning service because it requires more precise information about diet and service and their results.

References

[1] N. Ford, M. Richards: *Fundamentals of Software Architecture: An Engineering Approach*, O'Reilly Media, 2020.

[2] S. Kunal, M. K. Shetty, B. Shah, M. Girish, A. Bansal, V. Batra, S. Mukhopadhyay, J. Yusuf, A. Gupta, M. Gupta: *Heart Rate Variability in Post-COVID-19 Recovered Subjects Using Machine Learning*, Circulation (2021), doi: 10.1161/circ.144.suppl_1.14096.

[3] K. S. Mann, E. G. kaur: *Generation of CDA/XML Schema from DICOM Images using HL7 Standards*, IAEME Publications (2013).

[4] L. Silverston: *The Data Model Resource Book, Vol. 1: A Library of Universal Data Models for All Enterprises Revised Edition*, Wiley, 2001.

[5] P. Sodhi, N. Awashi, V. Sharma: *Introduction to Machine Learning and Its Basic Application in Python*, Proceedings of 10th International Conference on Digital Strategies for Organizational Success (2019).

[6] G. L. Turnquist: *Learning Spring Boot 2.0*, Packt Publishing Limited, 2017.

Annales Mathematicae et Informaticae
58 (2023) pp. 9–19
DOI: 10.33039/ami.2023.08.010
URL: https://ami.uni-eszterhazy.hu

Static analysis for safe software upgrade*

Dániel Ferenczi, Melinda Tóth

ELTE, Eötvös Loránd University, Budapest, Hungary
{danielf,toth_m}@inf.elte.hu

Abstract. Having applications accessible without downtime is no longer an exclusive requirement of mission-critical applications or traditional domains like communications. Running applications also require changes in the source code and upgrading live systems. Different approaches exist depending on the used technology. Systems implemented in Erlang can take the advantage of the underlying BEAM virtual machine and can be upgraded easily. However, source code has to be developed carefully once an upgrade is needed to not introduce run-time errors during the upgrade. We are developing a method to statically check the source code of Erlang applications for constructs that may lead to upgrading issues.

Keywords: Erlang, static analysis, software upgrade, hot code load

AMS Subject Classification: 68M15 Reliability, testing and fault tolerance of networks and computer systems

1. Introduction

With the ever-increasing use of e-commerce, even the owner of a simple webshop expects her site to operate without incidents all year round. Indeed, users expect high availability in general, and services for banking, commerce, news, and entertainment are expected to operate throughout the year with minimal disruptions. Running applications also require changes, however, the need to fix security issues or add new features and changes may happen at any time. An outage while a change is applied thus results in customer dissatisfaction, or even broken SLAs, and in the end, lost revenue. This presents a demand for seamless, "zero downtime" upgrades. The facilities for such upgrades depend on the stack chosen for

*Application Domain Specific Highly Reliable IT Solutions project has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme TKP2020-NKA-06 (National Challenges Subprogramme) funding scheme.

Submitted: July 28, 2023
Accepted: August 7, 2023
Published online: August 23, 2023

the development and operation of the application. These choices also determine what is possible during such an upgrade. Some tools [13] will launch new containers running a new version of the application in question, while slowly removing the previous release and possibly leading to a lost state [1]. Other tools allow for a more fine-grained approach, upgrading only the changed modules, and making state preservation possible [13, 16]. Errors in the use of these tools will however lead to unexpected behaviour or even downtime. This leads to the need to statically analyse the code responsible for the upgrade.

In this work, we demonstrate such a static analysis made for the language Erlang [5]. Erlang is a general-purpose, dynamically typed, functional programming language. It is designed to build distributed, concurrent, fault-tolerant computer systems. Originally designed for writing software in the telecommunications domain, the language and BEAM, the virtual machine it runs on, now see use as general tools for building fault-tolerant, concurrent, reliable software. Erlang was designed with high availability in mind. If we look at Joe Armstrong's, one of Erlang's designer's thesis [2], he notes that the option to "dynamically upgrade code" should be a feature of the language itself. This contrasts with other tools, that do not provide such feature and need additional tools to support code upgrades without downtime.

The goal of our work is to support safe software upgrades for the Erlang programming language by static program analysis. In this paper, we propose a method to identify servers which might crash or produce faulty behaviour after a live upgrade. We base our work on the static source code analyser and transformation tool, RefactorErl [4].

The rest of the paper is structured as follows. In Section 2 we illustrate the problem of upgrading through an example. In Section 3 we introduce RefactorErl and the algorithm used for pattern detection. Section 4 presents a solution for the problem in the context of *gen_server* behaviours. Sections 5 and 6 show possible ways to continue our research based on our existing foundation and related work respectively. We conclude our work in Section 7.

2. Problem statement

Erlang is distributed alongside the BEAM virtual machine, which compiles and runs Erlang code. Support for doing zero-downtime upgrades (in Erlang terms, "hot code loads") is built into the BEAM environment. It allows for state-preserving code changes on a module basis, in contrast to tools, like Kubernetes [9] that typically route connections to containers running new software versions. BEAM can keep two versions of the same module simultaneously. Code should thus be written with care, as expressions can be written in a way, that make code in the new version point to code in the previous version. Due to the limit in the number of versions BEAM can store, this will eventually lead to calls to versions no longer present in the virtual machine.

Avoiding these pitfalls requires care from the developer. For code meant to be updated usually the intent is for such expressions to get updated as well during an upgrade and point to code present in the new version of the module.

The code snippet in Figure 1 presents a typical example of problematic code that cannot be upgraded. The module `srv` defines a server that can be started with the `srv:start/0` function call. The function spawns/creates a new process and registers it with the `server` name. In the new process the `init/1` function is evaluated that initialises and starts the server's tail-recursive loop function. This is a standard way to develop a server in Erlang. The `loop/1` process stores an integer value and an updater function in its state. The process handles `{num, ClientPid}` and `upgrade` messages. In the former case, it answers to the requester with the updated counter value (line 19). In the latter one, it simply applies a qualified recursive call (line 16) to handle the code change and upgrades itself.

Having function references in the loop's state is dangerous however, as these references might become outdated during module upgrades. A careful developer will ensure that these references are fully qualified - which results in function calls

```
 1 -module(srv).
 2 -export([start/0,init/1,loop/1,
 3   adder/1,getNum/0]).
 4
 5 start() ->
 6   Pid = spawn(srv, init, [0]),
 7   register(server, Pid).
 8
 9 init(InitNum) ->
10   Adder = fun adder/1,
11   srv:loop({InitNum, Adder}).
12
13 loop({InitNum, Adder}) ->
14   receive
15     upgrade ->
16       srv:loop({InitNum, Adder});
17     {num, ClientPid} ->
18       NewNum = Adder(InitNum),
19       ClientPid ! {num, NewNum},
20       loop({NewNum, Adder})
21   end.
22
23 adder(N) ->
24   N + 42.
25
26 getNum() ->
27   ClientPid = self(),
28   server ! {num, ClientPid},
29   receive
30     {num, Num} -> Num
31   end.
```

```
 1 -module(srv).
 2 -export([start/0,init/1,loop/1,
 3   adder/1,getNum/0]).
 4
 5 start() ->
 6   Pid = spawn(srv, init, [0]),
 7   register(server, Pid).
 8
 9 init(InitNum) ->
10   Adder = fun srv:adder/1,
11   srv:loop({InitNum, Adder}).
12
13 loop({InitNum, Adder}) ->
14   receive
15     upgrade ->
16       srv:loop({InitNum, Adder});
17     {num, ClientPid} ->
18       NewNum = Adder(InitNum),
19       ClientPid ! {num, NewNum},
20       loop({NewNum, Adder})
21   end.
22
23 adder(N) ->
24   N + 42.
25
26 getNum() ->
27   ClientPid = self(),
28   server ! {num, ClientPid},
29   receive
30     {num, Num} -> Num
31   end.
```

Figure 1. Example Erlang server function. Note the differences in line 10 on how the `adder` function is referred.

using the implementations in the latest version of the module. On the left-hand side snippet, the reference at line 10 will keep its original value of the `Adder` function through module upgrades, when execution reaches line 18. This will result in an error as at this point the `Adder` symbol will eventually reference a version of the function no longer present in the virtual machine.

The right-hand snippet presents a fix to this, by using a fully qualified reference to the function in the fun expression. In Erlang using a fully qualified function inside a fun expression has the `fun module:function/arity` syntax. Note the difference in line 10 of the example. This, when called will reference the latest version of the adder function.

Apart from the pattern in the example, other expressions will also make upgrades unsafe in a similar fashion: a reference to an initial version of a function added to the server loop's state during the loop's initialisation. This reference eventually may become obsolete as the applications are upgraded.

Overall we have identified the patterns listed in Figure 2 as being unsafe.

- `fun(Arg) -> Body end.`
- `fun function/Arity`
- `fun(Arg) -> module:function(Arg) end.`

Figure 2. List of unsafe patterns.

In Erlang terms, we can say, that explicit fun expressions and non-fully qualified implicit fun expressions are unsafe as they cannot be upgraded. These patterns can also be present directly in state of the server, e.g. `loop{InitNum, fun adder/1}`, which also lead to undesired behaviour. Our method detects this, direct use as well. Finally, it is important to note, that this risk is present all throughout the code, e.g. clauses in receive blocks may also change the state, and can add references that are unsafe for code upgrades. Our present work offers a detection method for such patterns in RefactorErl. Other unsafe patterns also exist and are the scope of our future work, details of which are described in Section 5.

3. Detection methodology

The mentioned upgrade-related issues can be detected before the execution of the code, and during the development phase using static analysis techniques. In our particular case, we want to analyse the contents of the state passed to server loops. If the state contains function calls, they should be fully qualified, otherwise, code upgrades during the operation of the program may lead to errors. For this analysis, we are using RefactorErl [4] as a framework.

3.1. RefactorErl

RefactorErl is an open-source static analysis framework for Erlang. RefactorErl allows for analysing source code represented and stored in its Semantic Program

Graph (SPG) [8]. The SPG is a three-layered rooted graph, containing lexical, syntactic and semantic information about the analysed source code. Once the source code is stored, the tool offers different options for analysis, through its querying interfaces. The semantic query language provides an expressive language for the programmer to analyse her code [15], but RefactorErl also offers the option to run queries through a graph representation of the program. The more accessible semantic queries are also implemented using these graph queries. In our work, we have developed our algorithm in the graph query language and then exposed it as a semantic query term for easier use.

RefactorErl also provides other functions: code transformations, static analysis of security-related issues [3] and features for code comprehension but for the present work, we will focus on targeted static analysis.

3.2. Detection algorithm

To find the code fragments that are not safe during an upgrade (such as the one presented on Figure 1), we need to identify the arguments of the server function (the state of the server) and point the local function references in the state. These patterns might not be visible at the point of the server call. The state argument could easily consist of variables having their values returned by other functions. Figure 3 shows an example for this: in line 3 we do not know the exact values of the state but the server loop is still initialised with an unsafe reference from line 6. In such cases considering solely syntactic information will not help us find the possible value of the state.

```
1 init(InitNum) ->
2   Adder = srv:getAdder(),
3   srv:loop({InitNum, Adder}).
4
5 getAdder() ->
6   A = fun(N) -> N + 42 end,
7   A.
```

Figure 3. Example of a function returning a reference to a fun expression.

To overcome this obstacle we rely on the intraprocedural dataflow analysis [14]. Using the first order dataflow relation we are able to calculate possible values of an expression. RefactorErl builds the Dataflow Graph during the initial analysis, and stores the direct dataflow edges in the Semantic Program Graph. The first order dataflow reaching relation is implemented on the top of this graph. It allows us to track information between functions, and provides a way to determine the possible values of the state for our analysis.

When using our detection function, the developer is expected to provide a function as an input for our query to check its arguments across every instance where it is called. The algorithm will return the list of expressions where unsafe (as in Figure 2) arguments are defined.

Algorithm 1 Finding unsafe expressions in state.

```
1: function FIND(F)
2:     function_applications ← find_applications(F)
3:     for all $application \in function\_applications$ do
4:         argument_list ← get_arguments(application)
5:         expression_list ← get_subexpressions(argument_list)
6:         originating_exprs ← find_orig_exprs(expression_list)
7:         unsafe_exprs ← filter_to_unsafe(originating_exprs)
8:     end for
9: end function
```

Our detection algorithm finds these expressions as it is defined in Algorithm 1. At first, we gather all instances of application of our function. Our goal is to determine the 'safety' of the arguments for each application. Thus, for each application we gather the expressions used as arguments and find the originating expressions of these with dataflow reaching. Finally, we determine whether individual originating expressions are safe, and return them if they are not.

4. Analysing *gen_server* applications

Erlang distributions come bundled with a library called Open Telecom Platform. This library provides a construct called behaviour, which allows for abstracting away the complex, generic details of a pattern so that the developer only has to take care of implementing the specific parts of her application. It allows for code to follow common patterns which helps during the development life-cycle. OTP comes bundled with some built-in behaviours which can be suited for distributed applications: servers, state machines, event managers, and supervisors [6, 11]:

- gen_server: for implementing server applications
- gen_statem: for implementing state machines
- gen_event: for managing events and triggered actions
- supervisor: for adding fault tolerance

For the purposes of our research, it is worth looking at the *gen_server* behaviour, as server applications usually are implemented using it, rather than in a way akin to the example shown before. A developer implementing her server as a gen_server behaviour will have to write an Erlang module called in Erlang terms a *callback module*. This module needs to implement the behaviour's callback functions. The details of the implementation will of course depend on the problem the developer is working on. The structure and purpose of a behaviours callback functions is fixed however. For example, a server can be started with the `gen_server:start_link/4` function call that triggers a call to the callback module's `init/1` function that returns the initialised state of the server. Sending synchronous messages can be done through `gen_server:call/2` function calls that trigger a call to a `handle_call/3` callback function that returns the reply to the message and the modified state.

As *gen_server* behaviours split the implementation code to separate functions that deal with initialisation and have their own code running between interface and callback functions our solution will not work out of the box. An easy solution would be to simply add the entirety of the OTP library to the RefactorErl database. This would allow our dataflow analysis to traverse all relevant code, however, this comes with the cost of having to analyse the OTP library itself unnecessarily, as it will not contain code adding unsafe functions to the server state.

4.1. Example

A cleaner, more efficient solution would be to look at the behaviour functions and their callbacks and determine the points where the state is set, use these points as a basis for our analysis. This is possible, as behaviours enforce a structure, the points where the state is set are defined. Figure 4 shows an abridged *gen_server* implementation of the example from Figure 1.

```
 1 -module(srvg).
 2 -export([start_link/0, get_number/1]).
 3 -export([init/1, handle_call/1]).
 4 -behaviour(gen_server).
 5 start_link() ->
 6     gen_server:start_link({local, srvg}, srvg, [], []).
 7
 8 get_number() ->
 9    gen_server:call(srvg, get_num).
10
11 init([]) ->
12     Adder = fun srvg:adder/1,
13     InitNum = 0,
14     {ok, {Adder, InitNum}}.
15
16 handle_call(get_num, _From, {Adder, Num}) ->
17     NewNum = Adder(Num),
18     {reply, NewNum, {Adder, NewNum}};
19 ...
```

Figure 4. Example of a behaviour. Note the definition of a "safe" Adder in line 12.

This server can be started by calling the `srvg:start_link/0` function that calls the `gen_server:start_link/4` function. The first argument is the type and name of the server. The second argument of the call (`srvg`) defines the callback module. The started server evaluates the `init` function from the `srvg` callback module. In our example, the `init` function at line 11 initialises the server with the state in line 14. The `get_number/0` function at line 8 forwards the `get_num` request to the server and it waits for the result. The behaviour eventually processes the request with the `handle_call/3` function in line 16, setting the new server state to the `{Adder, NewNum}` tuple in line 18, and sends the `NewNum` value back to the caller process. This value will be the return value of the `gen_server:call/2` function call in line 9.

4.2. Modified algorithm

Behaviours can set the state for example in the return expression of initialisation or callback functions. In general, the values worth inspecting are those returned by the callback functions required by the *gen_server* behaviour. In practice we would have to change the input part defined in line 2 of Algorithm 1: server state is no longer defined in the applications of a server loop, but in the return value of the behaviour's callback functions. Thus we expect the user to provide a behaviour implementation, where we look for unsafe patterns in the return value of the callback functions that are manipulating the state, such as `handle_call/3, handle_cast/2, handle_info/2, code_change/2` function definitions. The new state is usually the last element of the returned tuple and we can use dataflow reaching to calculate the possible values. Once we have these expressions, we can filter out the non qualified function references.

The modified algorithm is preseneted in Algorithm 2. These modifications will allow our method to be used for analysing unsafe fun expressions in *gen_server* implementations.

Algorithm 2 Finding unsafe expressions in *gen_server* states.

```
1: function FIND(M)
2:     callbacks ← find_callback_functions(M)
3:     for all callback ∈ callbacks do
4:         state ← get_server_state(callback)
5:         expression_list ← get_subexpressions(state)
6:         originating_exprs ← find_orig_exprs(expression_list)
7:         unsafe_exprs ← filter_to_unsafe(originating_exprs)
8:     end for
9: end function
```

5. Further work

In addition to unsafe fun expressions, other patterns might also introduce risk to upgrades and are worth inspecting. Upgrades involving a change in the state structure are a clear example, where we could analyse if function applications still use the old state structure, or if there are references to elements removed from the state. This can be also examined in *gen_server* implementations, where state transformations for upgrades are handled by the `code_change` functions.

Analysing how modules depend on each other is another example. In order for upgrades to be safe in such a setup, upgrades have to respect the order of dependencies. Additionally, we could verify the existence of fully qualified self-references in server loops, which are required for upgrades in the first place.

Upon identification of further patterns, these additional checkers can be implemented using RefactorErl.

6. Related work

Apart from safe upgrades, code can be analysed for other properties that can present issues during operation. RefactorErl itself can be used to check the code for common vulnerabilities [3]. Ensuring safe upgrades is however a general problem, present across technologies.

Past research [1] has demonstrated challenges and solutions to upgrading distributed, multi-version systems and reasoning about their correctness, although their approach is not suited for preserving connection state.

HotSwap [16] presents a solution for software defined networks. Apart from upgrading without disruptions, the authors also ensure that rules, blacklists stay in effect by replaying events on the new version.

In the domain of modern container orchestration, Kustomize [10] includes tools for ensuring correct configurations for the popular orchestration system, Kubernetes [7]. These system's configuration is typically done in languages that lack type safety, and present a risk for updates when changing several configuration files.

Work has also been published as well on supporting zero-downtime releases from a kernel level [12] on different protocols to allow fast and frequent updates.

It would be worth investigating how general these solutions are, their caveats, and whether they can be applied to software stacks running Erlang.

7. Conclusion

In this work, we have looked at the importance of zero-downtime releases and summarised different approaches for achieving them. We presented how it is achieved in Erlang applications, and showed a potential cause for upgrade failures. To identify such problems before production we have extended the RefactorErl tool with a checker for unsafe use of local fun expressions in server loops in Erlang. We have also demonstrated how unsafe implementations can put upgrades at risk even in *gen_server* implementations. We also have shown a way to efficiently search for unsafe patterns when using behaviours.

Apart from local fun expressions in the state, other problems in the code might also impede safe upgrades. Investigating further unsafe patterns and methods for zero-downtime upgrades could be the topic of further studies, along with the analysis of other software stacks.

References

[1] S. Ajmani, B. Liskov, L. Shrira: *Modular software upgrades for distributed systems*, in: ECOOP 2006–Object-Oriented Programming: 20th European Conference, Nantes, France, July 3-7, 2006. Proceedings 20, Springer, 2006, pp. 452–476.

[2] J. Armstrong: *Making reliable distributed systems in the presence of software errors*, PhD thesis, 2003.

[3] B. Baranyai, I. Bozó, M. Tóth: *Supporting Secure Coding with RefactorErl*, Submitted to the ANNALES Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Computatorica (2020).

[4] I. Bozó, D. Horpácsi, Z. Horváth, R. Kitlei, J. Köszegi, T. M., M. Tóth: *RefactorErl - Source Code Analysis and Refactoring in Erlang*, in: Proceedings of the 12th Symposium on Programming Languages and Software Tools, ISBN 978-9949-23-178-2, Tallin, Estonia, Oct. 2011, pp. 138–148.

[5] F. Cesarini, S. Thompson: *Erlang Programming: A Concurrent Approach to Software Development*, O'Reilly Media, 2009, isbn: 9780596555856.

[6] F. Cesarini, S. Vinoski: *Designing for scalability with Erlang/OTP: implement robust, fault-tolerant systems*, " O'Reilly Media, Inc.", 2016.

[7] B. Copy, M. Bräger, A. P. Koufidis, E. Piselli, I. P. Barreiro: *Integrating IoT Devices Into the CERN Control and Monitoring Platform*, in: Proc. ICALEPCS'19 (New York, NY, USA), International Conference on Accelerator and Large Experimental Physics Control Systems 17, JACoW Publishing, Geneva, Switzerland, Aug. 2020, pp. 1385–1388, isbn: 978-3-95450-209-7, doi: 10.18429/JACoW-ICALEPCS2019-WEPHA125.

[8] Z. Horváth, L. Lövei, T. Kozsik, R. Kitlei, A. N. Víg, T. Nagy, M. Tóth, R. Király: *Modeling semantic knowledge in Erlang for refactoring*, in: Knowledge Engineering: Principles and Techniques, Proceedings of the International Conference on Knowledge Engineering, Principles and Techniques, KEPT 2009, vol. 54(2009) Sp. Issue, Studia Universitatis Babeş-Bolyai, Series Informatica, Cluj-Napoca, Romania, July 2009, pp. 7–16.

[9] *Kubernetes Documentation*, Accessed: 2023-01-12, url: https://kubernetes.io/docs/home/.

[10] *Kustomize Documentation*, Accessed: 2023-01-12, url: https://kubectl.docs.kubernetes.io/references/kustomize/.

[11] M. Logan, E. Merritt, R. Carlsson: *Erlang and OTP in Action*, 1st, USA: Manning Publications Co., 2010, isbn: 1933988789.

[12] U. Naseer, L. Niccolini, U. Pant, A. Frindell, R. Dasineni, T. A. Benson: *Zero Downtime Release: Disruption-Free Load Balancing of a Multi-Billion User Website*, 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, Virtual Event, USA: Association for Computing Machinery, 2020, pp. 529–541, isbn: 9781450379557, doi: 10.1145/3387514.3405885.

[13] I. Neamtiu, T. Dumitraş: *Cloud software upgrades: Challenges and opportunities*, in: 2011 International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems, IEEE, 2011, pp. 1–10.

[14] M. Tóth, I. Bozó: *Static Analysis of Complex Software Systems Implemented in Erlang*, Central European Functional Programming Summer School – Fourth Summer School, CEFP 2011, Revisited Selected Lectures, Lecture Notes in Computer Science (LNCS), Vol. 7241, pp. 451-514, Springer-Verlag, ISSN: 0302-9743, 2012.

[15] M. Tóth, I. Bozó, J. Kőszegi, Z. Horváth: *Static Analysis Based Support for Program Comprehension in Erlang,* In Acta Electrotechnica et Informatica, Volume 11, Number 03, October 2011. Publisher: Versita, Warsaw, ISSN 1335-8243 (print), pages 3-10.

[16] L. Vanbever, J. Reich, T. Benson, N. Foster, J. Rexford: *HotSwap: Correct and Efficient Controller Upgrades for Software-Defined Networks*, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13, Hong Kong, China: Association for Computing Machinery, 2013, pp. 133–138, isbn: 978-1-45032-178-5, doi: 10.1145/2491185.2491194.

Annales Mathematicae et Informaticae
58 (2023) pp. 20–29
DOI: 10.33039/ami.2023.08.002
URL: https://ami.uni-eszterhazy.hu

Evaluation of scalability in the Fission serverless framework

Balázs Fonyódi, Norbert Pataki, Ádám Révész

Department of Programming Languages and Compilers,
Faculty of Informatics,
Eötvös Loránd University,
Budapest, Hungary
fonyodi1balazs@gmail.com
patakino@elte.hu
reveszadam@gmail.com

Abstract. The efficient code execution often requires concurrency, so many programming languages, libraries and framework aim at parallelism. Based on the granularity and abstraction level, many approaches of concurrency are available. However, concurrency carries difficulties but modern ways try to make it more convenient.

A rather new solution is cloud computing which enhances the concurrency in a way that standalone virtual machines utilize the shared hardware. Containerization takes advantage of lightweight virtual machines called containers because they use a shared kernel of the operating system. Conteiner orchestration (e.g. Kubernetes) enables containerization among multiple hosts. Serverless programming supports container orchestration for individual function so every trigerred function may run in a different container which is inside a cluster of hosts.

In this paper, we briefly present the modern cloud computing ways of concurrency. This subtle distributed approach requires a comprehensive evaluation. We take advantage of the open source Fission serverless framework and implement some distributed algorithms in this realm using the Python programming language. For a deeper comprehension, we measure and evaluate the scalability of Fission framework and the entire system. We execute the distributed algorithms with different sizes of input and we fine-tune the configuration of the Fission framework.

Keywords: Function-as-a-Service, serverless, Fission, distributed algorithms

AMS Subject Classification: 68W15 Distributed algorithms

Submitted: July 28, 2023
Accepted: August 7, 2023
Published online: August 8, 2023

1. Introduction

Parallelism and concurrency play an important role in high performance computing. Based on the granularity, one can choose multithreaded, multicore or manycore solution for a more efficient application [13]. Grid computing and distributed algorithms are also available for a long time. On the other hand, these approaches inflict many challenges (for instance, race conditions, deadlock, resource guarantees, etc.) [6]. Programming languages, libraries and frameworks have been proposed, but the developers are still eager for a convenient, elegant, safe approach which supports the efficient concurrent execution of the code.

In recent years, cloud native computing became one of the most dominant paradigms for building applications. This was sped up with Google releasing Kubernetes in 2014, and the Cloud native trend does not seem to slow down any time soon. In this rapidly evolving landscape, developers continuously seeking new ways to optimize their infrastructure while reducing the complexity. One of the newer forms of developing in this environment is called *Function-as-a-Service* (FaaS). It is a so called serverless computing model where developers only write and run individual functions in the cloud. One of the advantages is that to run these event driven functions, it is not necessary to manage and understand the underlying infrastructure. Another great thing about FaaS is the scalability and reliability. Most of the FaaS platforms, such as AWS Lambda, Azure Functions or even open source providers such as Fission provide automatic scaling capabilities. They can dynamically allocate and deallocate resources as needed by the number of incoming requests. These functions run in an isolated environment, meaning that each function runs in its own container or pod. This means that the functions cannot interfere with each other and they are not impacted by other processes and so the risk of failures due to conflicts are reduced. This approach results in a very sophisticated concurrency model.

The concurrency has some important questions that belong to the performance. An intriguing one is how efficient to launch a new computation. What is the cost of triggering a new subcomputation? What is the cost of the communication between the subcomputations? A distributed algorithm should perform better, but the algorithm improves the runtime only if these mentioned costs are cheap enough [14].

In this paper, we take advantage of an open source FaaS platform called Fission. We evaluate the concurrency aspect of this FaaS platform with the implementation and the execution of recursive, distributed algorithms. We focus on the scalability aspect of the performance. The cost of a triggered new subcomputation is based on boot of a Docker container and communication over HTTP, thus it is valuable to check.

The rest of this paper is organized as follows. We introduce the cloud-based approaches from Docker to FaaS in Section 2. We present the environment of the evalution in Section 3. We present the implementation details of the distributed algorithms in Section 4. We discuss the result of the evaluation in Section 5. Finally, this paper is concluded in Section 6.

2. Approaches of the cloud computing

Containerization has become an emerging approach in modern software engineering since it enables the shipping of the software artifacts and products with all required dependencies in a platform-independent way [2]. Containerization eliminates the virtualization costs of not used OS services and the kernel itself per container. Moreover, containerization supports isolation effectively since the containers are seem to be separate operating systems but they use a shared kernel [10].

The containers are lightweight and they enable the fast and simple deployment and configurations. However, this approach is limited only one host. Kubernetes is a container orchestration system which manages Docker containers over multiple Docker hosts [8].

Function as a Service (FaaS) is a category of cloud computing services that provides a platform allowing programmers to develop, maintain, operate, scale and manage application functionalities without the complexity of building and maintaining the infrastructure typically associated with developing and deploying an application. This new abstraction approach eliminates further configuration and deployment cost. Building an application following this model is one way of achieving a “serverless” architecture [3]. This serverless programming approach provides the deployment of standalone function without launching any virtual machine or container [12].

Serverless programming is a rather new approach, however, there are real-world applications, for instance, Coca-Cola, Santander Bank and Expedia take advantage of this new paradigm [4].

Many frameworks are available for serverless programming, OpenFaaS, Kubeless and Fission to name a few open source tools [7]. Earlier, we defined our functional approach for the Kubeless realm [11]. However, it is still an open source artifact, VMWare has decided to stop driving and updating Kubeless [15]. Moreover, according to many aspects, Fission was evaluated as the most efficient serverless framework [9]. Furthermore, it has a wide language support and provides autoscaling which will be useful for measuring the speed of algorithms with different CPU settings.

3. Environment

3.1. Kubernetes

The environment of this research is created in a Kubernetes cluster. Kubernetes is the de facto standard in container orchestration systems. This paper might not be about Kubernetes, but there are some important terms that should be shortly introduced. Pods are the lowest level abstraction in a Kubernetes cluster. In this environment, a Docker container basically equals to a pod. It describes one or more containers in a shared network.

Our research focuses on the scalability of functions. These functions are running in pods and for scaling, we need more of these. Kubernetes has a solution for that. A ReplicaSet is responsible for managing a set of identical Pods. When one creates a ReplicaSet, the number of replicas is specified, along with a Pod template. The template describes the specification of each Pod that the ReplicaSet should create and manage. A ReplicaSet ensures that a specified number of identical Pods are running at any given time by creating or deleting Pods as needed.

3.2. Fission

Fission is a Kubernetes native serverless framework. Fission can be deployed to any Kubernetes cluster whether it is on a private cloud or a private computer. Developers can write short lived functions in multiple programming languages, such as Go, Python, Java or NodeJS. These functions can be triggered with HTTP requests or other event triggers. Functions can be easily deployed using one specific command and the fast cold-start time ensures that the pods get ready quickly. Another feature of Fission is automatic scaling by CPU usage. This means that the system can create or destroy instances when needed, so it does not use unnecessary resources. Figure 1 is a flowchart that presents how Fission works inside a Kubernetes cluster.

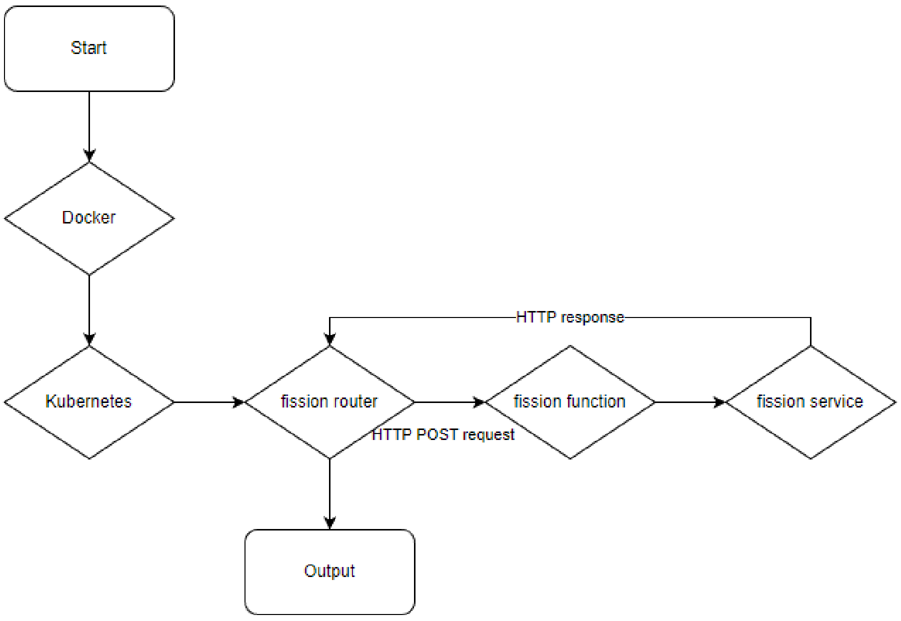


Figure 1. The workflow in our environment.

The router can send a HTTP request to a specific function which forwards to the Fission service. The service sends the HTTP response back to the router and the router creates the output.

4. Implementation and code overview

4.1. Karatsuba function

Karatsuba's algorithm is a fast multiplication algorithm which multiplies two numbers while reducing the number of recursive calls that is needed for the default multiplication [5]. This is achieved by splitting the two numbers into smaller ones, thus reducing them into subproblems in a divide and conquer manner and solving them recursively. The algorithm has a time complexity of $n^{\log_2 3}$ instead of n^2 for the traditional multiplication algorithm. The libraries needed for this version of code are Flask and Requests. The request subclass form flask enables the function to obtain the JSON data containing the two numbers, while the requests library allows the script to send HTTP requests to another instance of the same script to compute subtasks recursively. The function is split into three subfunctions for better readability. The main function handles the processing for the multiplication. This function obtains the two numbers with the `get_json()` from the request object, extracts the two numbers and passes both to the `karatsuba` function. The result from that is returned as a string. The `make_request` function takes the JSON data containing the two numbers and sends them to a URL as a post request. The `karatsuba` function is the part that handles the multiplication, it receives the two numbers and a string that represents the current recursion level. At first, the function checks whether the two numbers are single digit numbers. If they are, their product is returned, otherwise it computes the lengths of the two numbers, finds the maximum length, and splits each number into two parts of roughly equal length. The reason for this is to have three subproblems:

- Compute the product of the two upper halves of the numbers (`ac`)
- Compute the product of the two lower halves of the numbers (`bd`)
- Compute the product of the sum of the two halves of each number minus `ac` and `bd` (`ac_plus_bd`).

The function then returns the sum of the aforementioned subproblems, shifted accordingly to the required number of digits. The function has three recursive calls, but instead of handling these locally, each recursive call makes a HTTP request.

4.2. Merge sort algorithm

Merge sort algorithm developed by John von Neumann is a classic algorithm for sorting. The performance of this algorithm is typically evaluated in different concurrent situations.

This code also starts by importing the required Flask and requests modules. When the client sends a POST request to the '/merge' route, the Flask app receives it and triggers the main function. The application returns the sorted array as a JSON response. The sorting is done by sending POST requests to the same Flask app to sort each half of the array in a recursive way, and then merging the sorted halves using a standard merging algorithm.

The Python source code of the merge sort algorithm:

```
from flask import Flask, request, jsonify
import requests

app = Flask(__name__)

@app.route('/merge', methods=['POST'])
def main():
    data = request.get_json()
    array = data['array']
    call_id = data['call_id']
    if len(array) > 1:
        mid = len(array)
        left = array[:mid]
        right = array[mid:]
        sorted_left = merge_sort_helper(left, f"{call_id}_left")
        sorted_right = merge_sort_helper(right, f"{call_id}_right")
        return merge(sorted_left, sorted_right)
    else:
        return jsonify(array)

def merge(sorted_left, sorted_right):
    i = j = 0
    merged_array = []
    while i < len(sorted_left) and j < len(sorted_right):
        if sorted_left[i] <= sorted_right[j]:
            merged_array.append(sorted_left[i])
            i += 1
        else:
            merged_array.append(sorted_right[j])
            j += 1
    while i < len(sorted_left):
        merged_array.append(sorted_left[i])
        i += 1
    while j < len(sorted_right):
        merged_array.append(sorted_right[j])
        j += 1
    return jsonify(merged_array)

def merge_sort_helper(array, call_id):
    json_data = {'array': array, "call_id": call_id}
    try:
        headers = {'Content-type': 'application/json'}
        response = requests.post('http://router.fission.svc/merge',
                                 json=json_data,
                                 headers=headers)
        return response.json()
    except requests.exceptions.RequestException as e:
        print(e)

if __name__ == '__main__':
    app.run(debug=True)
```

5. Evaluation

The test environment was running on a home setup, using Windows with WSL, Docker, Kubernetes and Fission. As of now, we only measured the runtime of each function with different sized inputs with the `time` command. The first number that is going to be printed out is the moment the user hits the Enter key until the moment the function is completed.

5.1. Scalability

We have already discussed the concept of ReplicaSets. Fission by default creates three pods for one function but first we scaled it only to one. The functions were called different sized inputs, Karatsuba's algorithm with a digit number of 10, 32, 64, 128 and 256 length numbers and the merge sort with an array size of 10, 64, 128, 256 and 512.

The code was implemented so that every recursive call starts a new instance, this way, the resources are divided and in theory for very large numbers the runtime should be faster. However, starting these pods have a cost, referred to as a cold start which is about 100ms [1]. When a function is triggered, Fission starts the predefined pods, so that time was not accounted for in the measurements. When a recursive call happened, a new pod was started. Karatsuba's algorithm has three recursive calls, so even for small numbers, at least 4 or 5 pods started running, which equals to about 0.4-0.5 seconds that was wasted. The bigger the number, the more recursive calls happened which slowed down the algorithm significantly. With more Replicas, Fission should share the resources. But because of the nature of the code, the execution still starts more and more instances, as the size of inputs increases.

5.2. Measurements and results

Evaluating the performance of merge sort, our theory that dividing functions this way might not be the best approach on a small scale system, seems to prove us right. Since merge sort only has two recursive calls, the number of instances being started lowers significantly, thus the runtime of the function does not grow as much as with Karatsuba's algorithm.

Figure 2 and Figure 3 show the runtime of the analized functions and their runtime with different ReplicaSet configurations and with different input sizes.

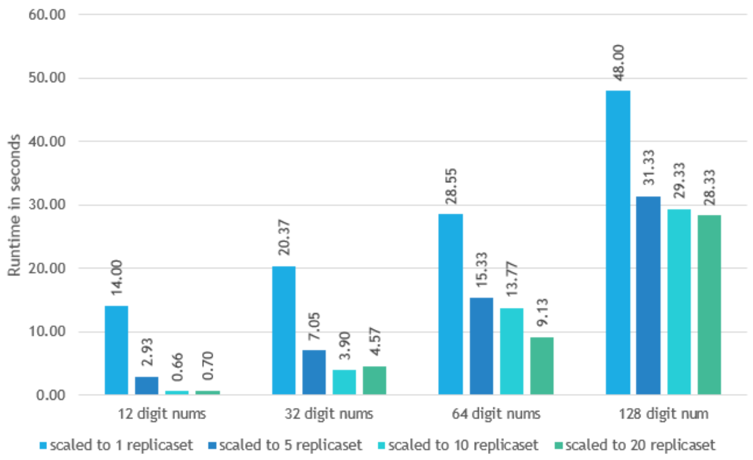


Figure 2. The runtime of Karatsuba's algorithm on multiple instances with different amount of data.

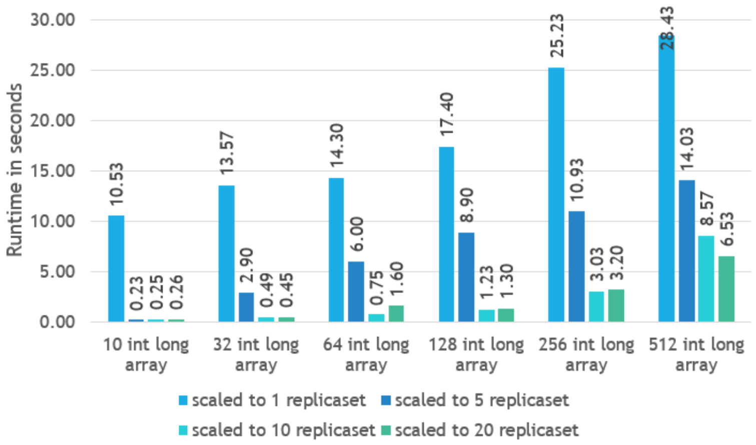


Figure 3. The runtime of merge sort on multiple instances with different amount of data.

An important thing to note is the starting number of the ReplicaSets. One would think that more ReplicaSets equal faster runtime, since Fission can divide the function into more resources. This is true until a certain amount of Replicas. However, from the Figure 2 and Figure 3, it is clearly visible that after a certain number, the runtime does not decrease or might even increase a little bit. This can occur because it actually takes time to divide the function to these resources. Interesting to note, that sometimes the more is less. Starting more replicas does not solve the runtime issue, it can even slow down the function execution a bit.

6. Conclusion

Cloud native computing provides high-level abstractions for concurrency that is essential for an improved performance. These abstractions assist the developers in a convenient way. FaaS services allow to deploy, maintain and operate separate functions in a cloud using containerization and orchestration.

We utilize the Fission serverless programming framework and started to evaluate how this granularity of concurrency improves the runtime. We implemented two classical algorithms (merge sort, Karatsuba's algorithm) in a recursive manner using the Python programming language. We measured the runtime with different sizes of inputs and with different configurations of Fission. However, the comparison and evaluation are not comprehensive, so our future work focuses on a more detailed analysis. Right now, we found that our cases have a rather high cost of the new subcomputation's start and HTTP communication.

References

[1] D. Balla, M. Maliosz, C. Simon: *Open Source FaaS Performance Aspects*, in: 2020 43rd International Conference on Telecommunications and Signal Processing (TSP), 2020, pp. 358–364, doi: 10.1109/TSP49548.2020.9163456.

[2] D. Bernstein: *Containers and Cloud: From LXC to Docker to Kubernetes*, IEEE Cloud Computing 1.3 (2014), pp. 81–84, doi: 10.1109/MCC.2014.51.

[3] P. Castro, V. Ishakian, V. Muthusamy, A. Slominski: *Serverless Programming (Function as a Service)*, in: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Los Alamitos, CA, USA: IEEE Computer Society, June 2017, pp. 2658–2659, doi: 10.1109/ICDCS.2017.305.

[4] P. Castro, V. Ishakian, V. Muthusamy, A. Slominski: *The Rise of Serverless Computing*, Commun. ACM 62.12 (Nov. 2019), pp. 44–54, issn: 0001-0782, doi: 10.1145/3368454.

[5] X. Fang, L. Li: *On Karatsuba Multiplication Algorithm*, in: The First International Symposium on Data, Privacy, and E-Commerce (ISDPE 2007), 2007, pp. 274–276, doi: 10.1109/ISDPE.2007.11.

[6] W.-m. Hwu, K. Keutzer, T. G. Mattson: *The Concurrency Challenge*, IEEE Design & Test of Computers 25.4 (2008), pp. 312–320, doi: 10.1109/MDT.2008.110.

[7] K. Kritikos, P. Skrzypek: *A Review of Serverless Frameworks*, in: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), 2018, pp. 161–168, doi: 10.1109/UCC-Companion.2018.00051.

[8] V. Medel, O. Rana, J. Á. Bañares, U. Arronategui: *Modelling Performance & Resource Management in Kubernetes*, in: Proceedings of the 9th International Conference on Utility and Cloud Computing, UCC '16, Shanghai, China: Association for Computing Machinery, 2016, pp. 257–262, isbn: 9781450346160, doi: 10.1145/2996890.3007869.

[9] S. K. Mohanty, G. Premsankar, M. di Francesco: *An Evaluation of Open Source Serverless Computing Frameworks*, in: 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2018, pp. 115–120, doi: 10.1109/CloudCom2018.2018.00033.

[10] Á. Révész, N. Pataki: *Containerized A/B Testing*, in: Proceedings of the Sixth Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, ed. by Z. Budimac, Belgrade, Serbia: CEUR-WS.org, 2017, 14:1–14:8, url: http://ceur-ws.org/Vol-1938/paper-rev.pdf.

[11] Á. Révész, N. Pataki: *LambdaKube - A Functional Programming Approach in a Distributed Realm*, in: 2021 4th International Conference on Geoinformatics and Data Analysis, ICGDA 2021, Marseille, France: Association for Computing Machinery, 2021, pp. 67–72, isbn: 9781450389341, doi: 10.1145/3465222.3465233.

[12] Á. Révész, N. Pataki: *Stack Traces in Function as a Service Framework*, in: Proceedings of the 11th International Conference on Applied Informatics (ICAI) (Eger, Hungary, Jan. 29–31, 2020), ed. by I. Fazekas, G. Kovásznai, T. Tómács, CEUR Workshop Proceedings 2650, Aachen, 2020, pp. 280–287, url: http://ceur-ws.org/Vol-2650/#paper29.

[13] A. C. Sodan, J. Machina, A. Deshmeh, K. Macnaughton, B. Esbaugh: *Parallelism via Multithreaded and Multicore CPUs*, Computer 43.3 (2010), pp. 24–32, doi: 10.1109/MC.2010.75.

[14] M. Tóth, I. Bozó, T. Kozsik: *Pattern Candidate Discovery and Parallelization Techniques*, in: Proceedings of the 29th Symposium on the Implementation and Application of Functional Programming Languages, IFL '17, Bristol, United Kingdom: Association for Computing Machinery, 2017, isbn: 9781450363433, doi: 10.1145/3205368.3205369.

[15] Q. L. Trieu, B. Javadi, J. Basilakis, A. N. Toosi: *Performance Evaluation of Serverless Edge Computing for Machine Learning Applications*, 2022, doi: 10.48550/ARXIV.2210.10331, url: https://arxiv.org/abs/2210.10331.

Annales Mathematicae et Informaticae
58 (2023) pp. 30–40
DOI: 10.33039/ami.2023.08.003
URL: https://ami.uni-eszterhazy.hu

Visualization of Read-Copy-Update synchronization contexts in C code

Endre Fülöp, Attila Gyén, Norbert Pataki

Department of Programming Languages and Compilers
Eötvös Loránd University
Budapest, Hungary
gamesh411@gmail.com
gyenattila@gmail.com
patakino@elte.hu

Abstract. The Read-Copy-Update (RCU) mechanism is a way of synchronizing concurrent access to variables with the goal of prioritizing read performance over strict consistency guarantees. The main idea behind this mechanism is that RCU avoids the use of lock primitives while multiple threads try to read and update elements concurrently. In this case, elements are linked together through pointers in a shared data structure. RCU is used in the Linux kernel, but there are user-space libraries which implement the technique as well. One of the user-space solutions is liburcu that is a C language library. Earlier, we defined our code comprehension framework for easing the development of RCU solutions. In this paper, we present our visualization techniques for the Microsoft's Monaco Editor.

AMS Subject Classification: 68W10 Parallel algorithms

1. Introduction

Read-copy-update (RCU) mechanism is used for synchronizing memory access in a way that guarantees deterministic read-access even during concurrent writes to the same memory region [4]. Unsynchronized access from multiple threads can lead to the evaluation of completely unexpected values, which in turn almost negates the programmers ability to reason about possible outcomes [11].

There are multiple families of solutions to this problem. One traditional solution is locking, where multiple threads are sequentially ordered at runtime, thus accesses to a memory region are mutually exclusive among threads. This can, however, lead

Submitted: July 30, 2023
Accepted: August 7, 2023
Published online: August 8, 2023

to performance degradations, live- and deadlock problems. Locking solutions use synchronization primitives like mutexes and various kinds of locks [5]. Another possible solution is lock-free programming, where synchronization is solved without explicit exclusion, eliminating most locking issues [9]. Many lock-free solutions use memory barriers and atomic variables [15]. RCU is a solution of higher abstraction level than those mentioned before. RCU can be implemented in the kernel- or in the user-space. Linux kernel uses data structures with RCU implementation since 2002 [13].

RCU can also be implemented in the user-space, one such library is *liburcu* written in C [3]. In order to provide synchronization using the liburcu library, the user must intersperse the application code with calls to library functions. In effect the side-effects of these invocations produce a context along the execution paths where accesses to a memory region are guaranteed to have desired properties. To help the comprehension of the synchronization provided by the library, we have devised a visualization technique. The goal of the proposed technique is to provide the users of the library a visual and interactive way of exploring the code, thus facilitating the correct and intended usage of the library. There is no silver bullet in software engineering [2]. However, visualization is an important aspect [6]. Visualization improves the comprehension in many ways [18]. Code comprehension often requires visualization on the top of the source code [10]. However, subtle details are in-use for more sophisticated approaches [16]. Our previous work presented our framework for the code comprehension of RCU contexts [7]. In the previous paper, we focus on framework, more precisely, the static analysis and Monaco Editor-related techniques. Unfortunately, the actual visualization has not been presented properly. In this paper, the major contribution belongs to the visualization of the contexts.

This rest of this paper is organized as follows. In Section 2, we provide a brief overview on related work. We present the Userspace RCU implementation and our static analysis methods in Section 3. Section 4 provides a brief explanation how the backend analysis techniques are defined in our earlier work. We present the approach of visualization in Section 5, and finally, this paper concludes in Section 6.

2. Related work

Visualizing concurrency aspects of programs can have the goal of assessing performance aspects of a particular solution [17]. One category of tools used to measure performance is sampling- and instrumenting profilers which are for both single- and multithreaded programs. These profilers produce aggregated performance statistics and/or traces of events which can be used for detailed performance analysis [1]. These statistics are consequently converted into visual representations like bar-charts and flamegraphs to provide an overview and highlight the proportions of each program parts contribution to a given metric. Compared to these visualizations, we propose a technique based on static analysis instead of dynamic profiling to reason about the structure of the RCU implementation. Another important aspect is that the analysis done by the RCU visualization technique is more qualitative in nature.

3. Userspace RCU implementation

3.1. RCU overview

RCU is implemented in program code as a set of API calls (free function calls in case of the *liburcu* C library), which implements concurrent publication of modifications on shared data, subscription for insertion into shared data structures, waiting for readers to complete their executions and finally to maintain different versions of the same data [4]. This API is geared towards read-heavy uses, where updates of values and structured data are relatively less frequent, and where consistency guarantees are not critical. Memory usage is another concern, as multiple version of the same data can lead to overuse.

Concurrent access to variables is done by associating regions of code with parts of programs executions, which read shared values (readers) [12]. These sections are called read critical sections. Read critical sections interact with synchronization points, which are usually used as part of the update part of the program executions (updaters). Readers subscribe to a specific version of the data they are reading, which is the one available at the beginning of the critical section. The end of a critical section is explicit in the code, which is needed for the updaters to detect if there is no more reading activity for a specific piece of data. Read critical sections does not enforce ordering inside a single section, nor do multiple sections between each other.

3.2. RCU contexts in liburcu

Userspace-implemented RCU library librcu is a compile- and link-time solution for using RCU primitives in arbitrary C software without depending on kernel features of the operating system (OS) [12]. The library supports multiple implementations of the RCU semantics, the API consists of free functions with prefixes corresponding to the name of the technique (urcu) and the implementation technique (i.e. mb for memory barrier, qsrb for quiescent state-based reclamation or signal for using posix signals). By default, API calls are implemented as external linkage functions, and the generated IR code therefore contains explicit references to the mentioned free functions. Using optimizations which cause the functions to be inlined will render the solution described here unusable. Inlining small functions can be the result of link-time optimization or by defining the `URCU_INLINE_SMALL_FUNCTIONS` preprocessor symbol before including the library headers.

Another limitation is that debug information must be generated alongside with the IR code. An example of an API function which is used for opening a read-side critical section by using memory barriers as implementation is `urcu_mb_read_lock()`.

There are two API functionalities, which must be used in pairs. For registering threads, one would used the `urcu_<flavor>_register_thread()` and `urcu_<flavor>_unregister_thread()`. These are used in a non-nested way (calling register while already registered is an error), but the other pair of API functions signifying the read-side critical sections can be nested indefinitely. These are the `urcu_<flavor>_read_lock()` and `urcu_<flavor>_read_unlock()` functions. The solution presented here is tailored towards the nestable usage, and can be extended to consider the non-nestable case. There are API calls which can only be safely used inside the context of registered threads (the majority of the librcu API) and there are API calls, which have special meaning when inside a read-side critical section (like `defer_rcu()` or `synchronize_rcu()`). The intended usage of the solution presented here is to provide information about the potential execution paths that are potentially enclosed in the mentioned API calls. It would help the software's discoverability, changeability, and maintainability to know which part of the code potentially contributes to the synchronization structure.

4. Code comprehension framework

Our earlied work proposed a code comprehension framework for the RCU synchronization contexts [7]. We have developed a static analysis solution based on the Clang compiler. Our static analysis tool takes advantage of the LLVM IR (Intermediate Representation) which is generated from the source code.

For context detection, the iterative algorithm of forward dataflow analyses uses reverse postorder traversal of the control-flow graph (CFG) elements in case of forward analysis in order for performance reasons. This results in a scalable method for gaining an overview about the synchronization aspects of the software. The modular nature of the approach lends itself to distributed use.

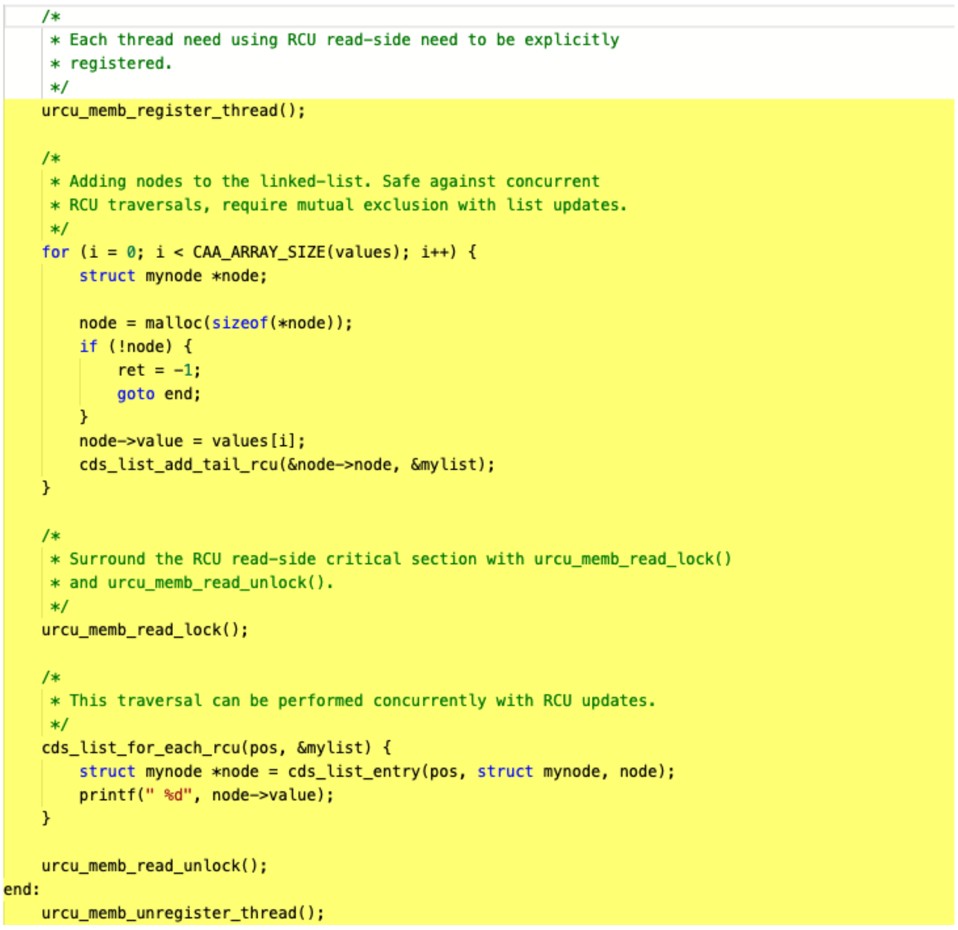
The transfer function saves the interesting locations (the instructions that can be used to get the locations), by appending them to the basic block level global fact, but only if this global fact is does not already contain them. In addition, if a context ending API call is detected, the exit state of the instruction set to the current global state of the basic block. The reverse postorder visitation guarantees, if a context starting instruction then happens to precede a context ending one, there is path in the CFG from the starter to the ending one. The set-like nature of the list in turn allows for the halting of the fixed-point algorithm in finite steps, as there are a finite amount of interesting locations inside a program.

The meet function is responsible for merging the exit states of multiple incoming dataflow facts. This is defined as the concatenation of the dataflow fact lists in a manner, that guarantees uniqueness of elements inside the resulting list, and the preservation of relative ordering among the interesting locations.

5. Visualization of the contexts

Monaco Editor is maintained by Microsoft and available worldwide for free [14]. It has a playground with full of interactive examples and provides wide access to the editor and it supports feature like colorize the editor line-by-line, add different error and warning messages or add a hover message when the cursor is hovered over the text. Doing all this with JavaScript programming language for the dynamic parts, CSS for styling and HTML to build the raw frame [8]. It gives full access to the Document Object Model (DOM) supplemented by its own special elements. However, it sets up some limitations.

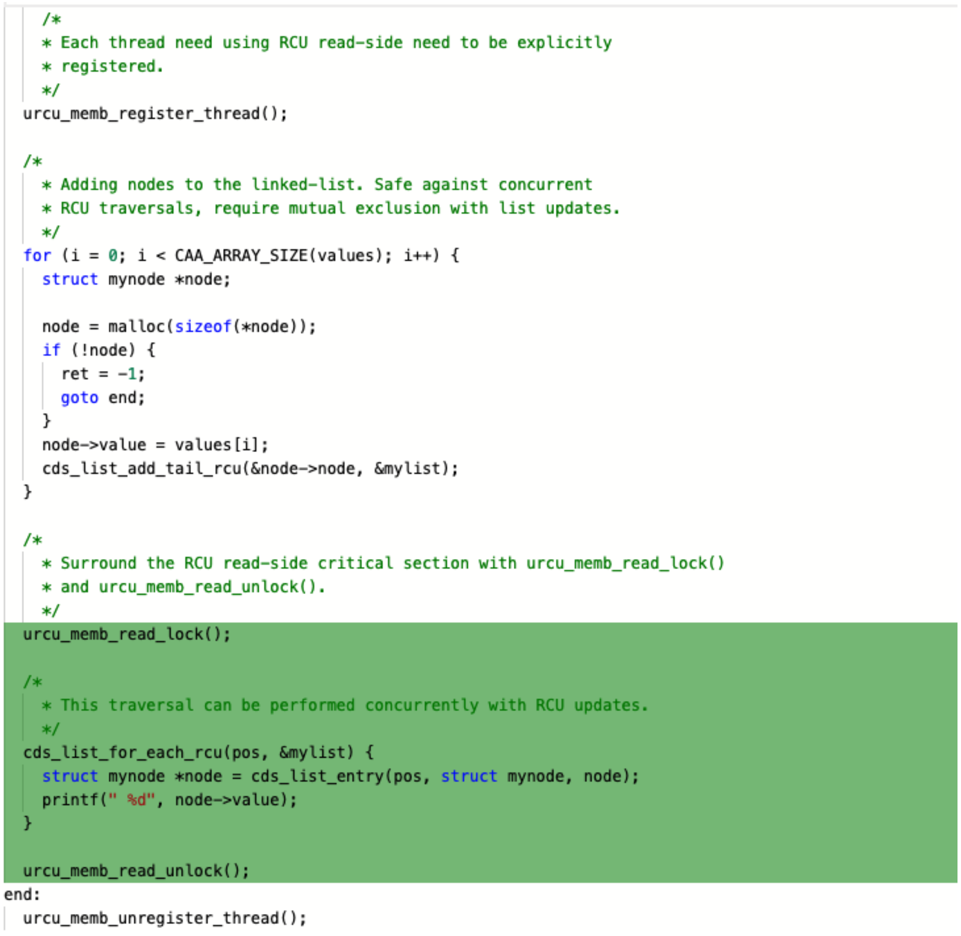
The figures below show the four aspects of Monaco Editor that we consider to be the most important. We would like to note that this is our implemented version of the code parser and the Monaco Editor. The C++ code is approximately the same in all four figures, with minimal changes in place, which were necessary in order to be able to present the different possible appearance methods.



```
    /*
     * Each thread need using RCU read-side need to be explicitly
     * registered.
     */
    urcu_memb_register_thread();

    /*
     * Adding nodes to the linked-list. Safe against concurrent
     * RCU traversals, require mutual exclusion with list updates.
     */
    for (i = 0; i < CAA_ARRAY_SIZE(values); i++) {
        struct mynode *node;

        node = malloc(sizeof(*node));
        if (!node) {
            ret = -1;
            goto end;
        }
        node->value = values[i];
        cds_list_add_tail_rcu(&node->node, &mylist);
    }

    /*
     * Surround the RCU read-side critical section with urcu_memb_read_lock()
     * and urcu_memb_read_unlock().
     */
    urcu_memb_read_lock();

    /*
     * This traversal can be performed concurrently with RCU updates.
     */
    cds_list_for_each_rcu(pos, &mylist) {
        struct mynode *node = cds_list_entry(pos, struct mynode, node);
        printf(" %d", node->value);
    }

    urcu_memb_read_unlock();
end:
    urcu_memb_unregister_thread();
```

Figure 1. Visualization of an RCU thread registration.

In order to make it easier to distinguish different visualization parts, we used separate colors to display the individual methods and code parts. Figure 1 shows a thread registration process. The editor highlights precisely the part of the code where an `urcu_memb_register_thread()` registration takes place, the end of which is indicated by `urcu_memb_unregister_thread()`. The editor highlights this part of the code sections in yellowish color.

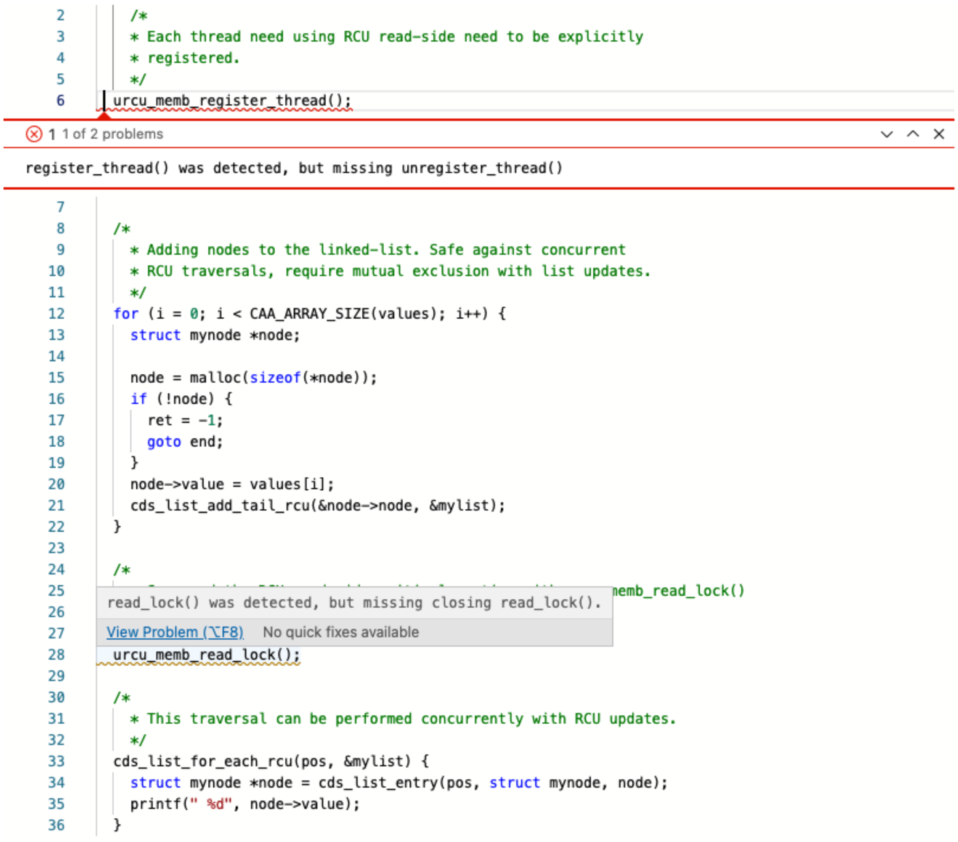


```
  /*
   * Each thread need using RCU read-side need to be explicitly
   * registered.
   */
  urcu_memb_register_thread();

  /*
   * Adding nodes to the linked-list. Safe against concurrent
   * RCU traversals, require mutual exclusion with list updates.
   */
  for (i = 0; i < CAA_ARRAY_SIZE(values); i++) {
    struct mynode *node;

    node = malloc(sizeof(*node));
    if (!node) {
      ret = -1;
      goto end;
    }
    node->value = values[i];
    cds_list_add_tail_rcu(&node->node, &mylist);
  }

  /*
   * Surround the RCU read-side critical section with urcu_memb_read_lock()
   * and urcu_memb_read_unlock().
   */
  urcu_memb_read_lock();

  /*
   * This traversal can be performed concurrently with RCU updates.
   */
  cds_list_for_each_rcu(pos, &mylist) {
    struct mynode *node = cds_list_entry(pos, struct mynode, node);
    printf(" %d", node->value);
  }

  urcu_memb_read_unlock();
end:
  urcu_memb_unregister_thread();
```

Figure 2. Visualization of an RCU lock snippet.

In Figure 2, we highlight another part of the previous code snippet where a read lock was created. Its registration starts at the `urcu_memb_read_lock()` line and ends with the `urcu_memb_read_unlock()` line. It is important to note that we can set the highlighting of these blocks ourselves, which should be in focus, as shown in Figure 1 and Figure 2 separately. We also have the option to display them at the same time. In this case, the different layers in the editor will be aligned.

The algorithm detects deficiencies that can cause problems at the code level, such as the unregistration of registered threads or locks. The editor also draws the user's attention to such cases, as shown in the Figure 3. It also reveals which part of the code is missing, and if there are several errors in the code, it shows how many errors are in the editor in total. We can use the up and down arrows on the right side of the error bar to jump back and forth between errors. With this feature, real-time errors can be displayed to users, thereby avoiding the occurrence of runtime problems.



```
  /*
  * Each thread need using RCU read-side need to be explicitly
  * registered.
  */
urcu_memb_register_thread();
```

```
register_thread() was detected, but missing unregister_thread()
```

```

/*
  * Adding nodes to the linked-list. Safe against concurrent
  * RCU traversals, require mutual exclusion with list updates.
  */
for (i = 0; i < CAA_ARRAY_SIZE(values); i++) {
  struct mynode *node;

  node = malloc(sizeof(*node));
  if (!node) {
    ret = -1;
    goto end;
  }
  node->value = values[i];
  cds_list_add_tail_rcu(&node->node, &mylist);
}

/*
                                                   memb_read_lock()
```

```
read_lock() was detected, but missing closing read_lock().
```

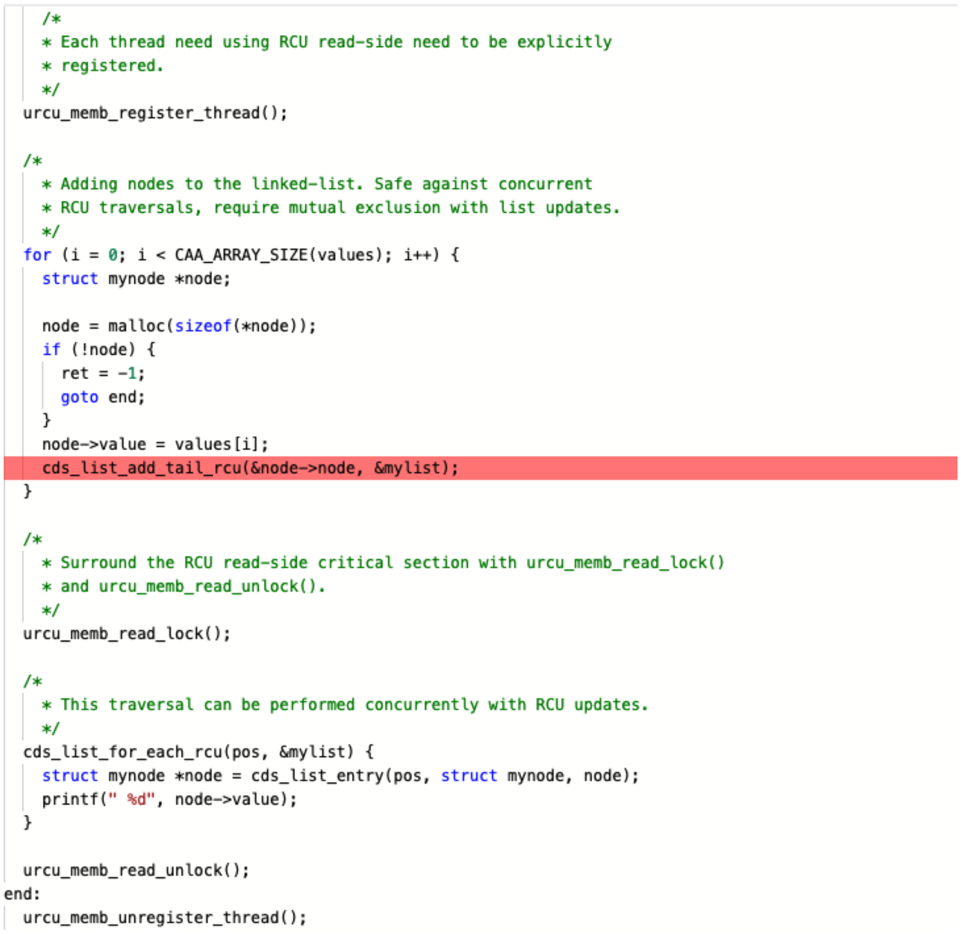
```
urcu_memb_read_lock();

/*
  * This traversal can be performed concurrently with RCU updates.
  */
cds_list_for_each_rcu(pos, &mylist) {
  struct mynode *node = cds_list_entry(pos, struct mynode, node);
  printf(" %d", node->value);
}
```

Figure 3. Visualization of an alert.

In Figure 4, one can see the highlighting of a code fragment that uses a RCU function that uses a shared variable outside the locking code snippet at runtime, potentially causing an error that could arise due to shared memory. By highlighting this, the user can better check whether the given piece of code has been provided with the appropriate error handling or threading methods, which can be used to avoid runtime problems due to shared memory space.

Figure 5 presents the comprehensive visualization of an RCU-based code snippet in the Monaco Editor. This approach makes many aspects of the RCU usage more comprehensible. Our solution makes the debugging procedure, and bug fixes easier.

In addition to these, the Monaco Editor visualization implementation we created is able to highlight potential runtime problems, such as over-indexing on the array or highlighting different ranges and displaying hover messages.

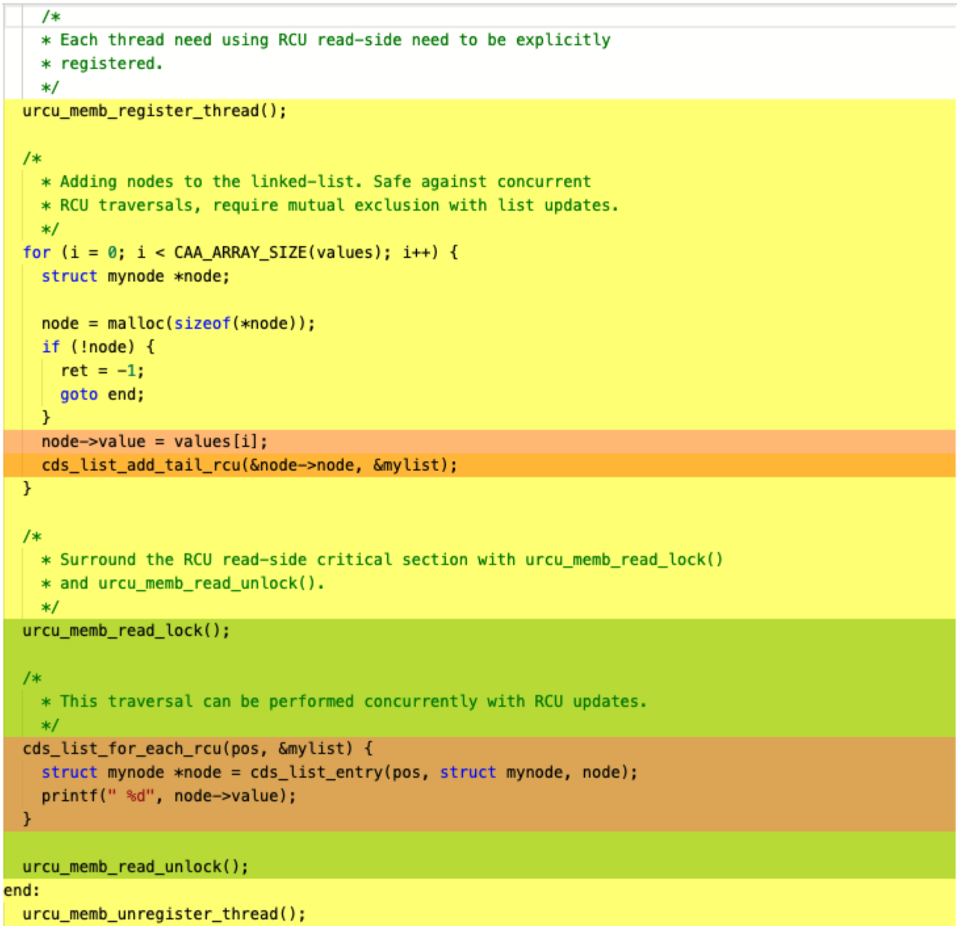


```
  /*
   * Each thread need using RCU read-side need to be explicitly
   * registered.
   */
  urcu_memb_register_thread();

  /*
   * Adding nodes to the linked-list. Safe against concurrent
   * RCU traversals, require mutual exclusion with list updates.
   */
  for (i = 0; i < CAA_ARRAY_SIZE(values); i++) {
    struct mynode *node;

    node = malloc(sizeof(*node));
    if (!node) {
      ret = -1;
      goto end;
    }
    node->value = values[i];
    cds_list_add_tail_rcu(&node->node, &mylist);
  }

  /*
   * Surround the RCU read-side critical section with urcu_memb_read_lock()
   * and urcu_memb_read_unlock().
   */
  urcu_memb_read_lock();

  /*
   * This traversal can be performed concurrently with RCU updates.
   */
  cds_list_for_each_rcu(pos, &mylist) {
    struct mynode *node = cds_list_entry(pos, struct mynode, node);
    printf(" %d", node->value);
  }

  urcu_memb_read_unlock();
end:
  urcu_memb_unregister_thread();
```

Figure 4. Visualization of shared data's usage outside the locking snippet.

6. Conclusion

Despite RCU is a very powerful mechanism and in a sense simplifies thread handling in order for someone to understand what is going on in the background, a deeper understanding of the topic is required. The visualization tool does not answer



```
  /*
  * Each thread need using RCU read-side need to be explicitly
  * registered.
  */
urcu_memb_register_thread();

/*
  * Adding nodes to the linked-list. Safe against concurrent
  * RCU traversals, require mutual exclusion with list updates.
  */
for (i = 0; i < CAA_ARRAY_SIZE(values); i++) {
  struct mynode *node;

  node = malloc(sizeof(*node));
  if (!node) {
    ret = -1;
    goto end;
  }
  node->value = values[i];
  cds_list_add_tail_rcu(&node->node, &mylist);
}

/*
  * Surround the RCU read-side critical section with urcu_memb_read_lock()
  * and urcu_memb_read_unlock().
  */
urcu_memb_read_lock();

/*
  * This traversal can be performed concurrently with RCU updates.
  */
cds_list_for_each_rcu(pos, &mylist) {
  struct mynode *node = cds_list_entry(pos, struct mynode, node);
  printf(" %d", node->value);
}

urcu_memb_read_unlock();
end:
urcu_memb_unregister_thread();
```

Figure 5. Comprehensive visualization in the Monaco Editor.

all questions, but it helps to comprehend the background processes better. Our previous work includes a code comprehension framework for RCU. This paper presents the visualization approach based on the framework. The visualization is implemented in the Microsoft's Monaco Editor that is a modern, customizable solution for high-level code comprehension.

References

[1] R. Bell, A. D. Malony, S. Shende: *ParaProf: A Portable, Extensible, and Scalable Tool for Parallel Performance Profile Analysis*, in: Euro-Par 2003 Parallel Processing, ed. by H. Kosch, L. Böszörményi, H. Hellwagner, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 17–26, isbn: 978-3-540-45209-6.

[2] M. Danisovszky, T. Nagy, K. Répás, G. Kusper: *Western Canon of Software Engineering: The Abstract Principles*, in: 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), 2019, pp. 153–156, doi: 10.1109/CogInfoCom47531.2019.9089999.

[3] M. Desnoyers, P. E. McKenney: *Userspace RCU*, https://liburcu.org/.

[4] M. Desnoyers, P. E. McKenney, A. S. Stern, M. R. Dagenais, J. Walpole: *User-Level Implementations of Read-Copy Update*, IEEE Transactions on Parallel and Distributed Systems 23.2 (2012), pp. 375–382, doi: 10.1109/TPDS.2011.159.

[5] M. Drocco, V. G. Castellana, M. Minutoli: *Practical Distributed Programming in C++*, in: Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing, HPDC '20, Stockholm, Sweden: Association for Computing Machinery, 2020, pp. 35–39, isbn: 9781450370523, doi: 10.1145/3369583.3392680.

[6] E. Fülöp, A. Gyén, N. Pataki: *A Framework for C++ Exception Handling Assistance*, in: Proceedings of the Ninth Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, ed. by Z. Budimac, CEUR Workshop Proceedings 3237, 2022, 4:1–4:13, url: http://ceur-ws.org/Vol-3237/paper-ful.pdf.

[7] E. Fülöp, A. Gyén, N. Pataki: *Code Comprehension for Read-Copy-Update Synchronization Contexts in C Code*, in: Geoinformatics and Data Analysis, ed. by S. Bourennane, P. Kubicek, Cham: Springer International Publishing, 2022, pp. 187–200, isbn: 978-3-031-08017-3, doi: 10.1007/978-3-031-08017-3_17.

[8] E. Fülöp, A. Gyén, N. Pataki: *Monaco Support for an Improved Exception Specification in C++*, IPSI Transactions on Internet Research 19.1 (Jan. 2023), pp. 24–31, doi: 10.58245/ipsi.tir.2301.05, url: http://ipsitransactions.org/journals/papers/tir/2023jan/p5.pdf.

[9] T. E. Hart, P. E. McKenney, A. D. Brown, J. Walpole: *Performance of memory reclamation for lockless synchronization*, Journal of Parallel and Distributed Computing 67.12 (2007), Best Paper Awards: 20th International Parallel and Distributed Processing Symposium (IPDPS 2006), pp. 1270–1285, issn: 0743-7315, doi: 10.1016/j.jpdc.2007.04.010, url: https://www.sciencedirect.com/science/article/pii/S074373150700069X.

[10] B. Kot, B. Wuensche, J. Grundy, J. Hosking: *Information Visualisation Utilising 3D Computer Game Engines Case Study: A Source Code Comprehension Tool*, in: Proceedings of the 6th ACM SIGCHI New Zealand Chapter's International Conference on Computer-Human Interaction: Making CHI Natural, CHINZ '05, Auckland, New Zealand: Association for Computing Machinery, 2005, pp. 53–60, isbn: 1595930361, doi: 10.1145/1073943.1073954.

[11] G. Márton, I. Szekeres, Z. Porkoláb: *Towards a High-level C++ Abstraction To Utilize The Read-Copy-Update Pattern*, Acta Electrotechnica et Informatica 18.3 (2018), pp. 18–26, doi: 0.15546/aeei-2018-0021.

[12] P. E. McKenney: *Is Parallel Programming Hard, And, If So, What Can You Do About It? (Release v2021.12.22a)*, 2021, arXiv: 1701.00854 [cs.DC], url: https://arxiv.org/abs/1701.00854.

[13] P. E. McKenney, J. Walpole: *What is RCU, fundamentally?*, 2007, url: https://lwn.net/Articles/262464/.

[14] Microsoft: *Monaco Editor*, https://microsoft.github.io/monaco-editor/.

[15] G. Nagy, Z. Porkoláb: *Read-Copy-Update as a Possible Locking Strategy in Scala*, in: Proceedings of the Seventh Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, ed. by Z. Budimac, CEUR Workshop Proceedings 2217, 2018, 12:1–12:8, url: http://ceur-ws.org/Vol-2217/paper-nag.pdf.

[16] Z. Porkoláb, T. Brunner: *Advanced Code Comprehension using Version Control Information*, IPSI Transactions on Internet Research 16.2 (July 2020), pp. 47–54.

[17] Z. Porkoláb, T. Brunner: *The CodeCompass Comprehension Framework*, in: Proceedings of the 26th Conference on Program Comprehension, ICPC '18, Gothenburg, Sweden: Association for Computing Machinery, 2018, pp. 393–396, isbn: 9781450357142, doi: 10.1145/3196321.3196352.

[18] W. Steingartner, M. Haratim, J. Dostál: *Software visualization of natural semantics of imperative languages - a teaching tool*, in: 2019 IEEE 15th International Scientific Conference on Informatics, 2019, pp. 000509–000514, doi: 10.1109/Informatics47936.2019.9119290.

Annales Mathematicae et Informaticae
58 (2023) pp. 41–55
DOI: 10.33039/ami.2023.08.001
URL: https://ami.uni-eszterhazy.hu

On the patterns of the nonstationary datagram based fast communication processes[*]

Zoltan Gal, Gyorgy Terdik

Faculty of Informatics,
University of Debrecen,
Hungary
gal.zoltan@inf.unideb.hu
terdik.gyorgy@inf.unideb.hu

Abstract. Nowadays expectations against modern communication services involve not just Quality of Service (QoS) enhancement for real-time applications but also increased transmission rate between the storing and processing of Big Data nodes. Transmission Control Protocol (TCP) has strict flow control of the data stream providing automatic adaptation to the path load of the process-to-process communication. User Datagram Protocol (UDP) based solutions are proposed to settle the communication efficiency. In this paper, we analyse the effect of three independent communication parameters on the efficiency of looped UDP communication: the size of the Maximum Transfer Unit (MTU), the bandwidth of the end-to-end session, and the segment size of the UDP protocol data unit. The usage of nonstationary multi-resolution methods helps to identify three characteristic patterns offering identification of the objective qualitative features of the looped datagram communication services.

Keywords: datagram, high-speed network, nonstationarity, Fourier transform, nonlinearity, Wavelet transform, Empirical Mode Decomposition, Variational Mode Decomposition

AMS Subject Classification: 62J02, 62P30, 68M11, 68M12, 93B17

[*]This work has been supported by QoS-HPC-IoT Laboratory and project TKP2021-NKTA of the University of Debrecen, Hungary. Project no. TKP2021-NKTA-34 has been implemented with the support provided by the National Research, Development, and Innovation Fund of Hungary, financed under the TKP2021-NKTA funding scheme.

Submitted: July 27, 2023
Accepted: August 7, 2023
Published online: August 15, 2023

1. Introduction

Intensive flow control of the TCP transport layer mechanism provides process-to-process service with low-efficiency usage of the communication paths resources like links, buffers and processors of the intermediary nodes [9]. The lack of justification for this strict connection-oriented mechanism implies technological reconsideration in practice. The strict limit of the TCP segment header of 40 bytes, the extreme complexity of multipath TCP evolution, the slow upgrade method of the new stacks, and the missing TCP header encryption make this forty years old mechanism not optimal for the next period. These issues opened development steps toward the usage of simpler and faster UDP extended with reliable services [7]. UDP is a connectionless transport layer mechanism and functions like any other datagram type service on the other positions of the communication stack: IEEE 802.3 on the datalink layer, Internet Protocol (IP) on the network layer and Simple Network Management Protocol (SNMP) on the application layer. All of these mechanisms function without prior setup or later acknowledgment phase of the corresponding protocol data unit (PDU) transmission. The absence of the PDU acknowledgment makes faster the transmission but leaves the layer service without dataflow control possibility. Communication mechanisms based on any of the datagram services require extra intelligence to provide reliability and security. The complexity and resource usage intensity of these extra solutions determines the efficiency and behavior of the applications.

The performance research works of the UDP show the necessity of a new transport layer mechanism combining the advantages of the TCP and UDP network services. The potential solution is QUIC with future development possibilities allowed by the Request for Comments (RFCs) standards created by the Internet technology responsible organization, Internet Engineering Task Force (IETF).

Methods of classical statistical time series analysis can evaluate the properties of data communication services by considering stationary features of the data. Time series which is captured from fast communication sessions shows nonstationarity of the real data transfers frequently. This property is caused by the network conditions changing rapidly and unpredictably over time. This is due to a variety of factors such as network congestion, varying traffic loads, and changes in the topology of the network. In high-speed networks the amount of traffic can vary significantly from moment to moment, leading to changes in network congestion and packet loss rates. Similarly, the network topology can change due to the addition or removal of network nodes or links, leading to changes in the routing paths and delays. Moreover, high-speed networks often use sophisticated network protocols and mechanisms such as congestion control and flow control to manage the flow of data and ensure efficient services. These mechanisms can also introduce variability and nonstationarity into the network conditions, as they can adjust the behaviour of the network dynamically based on the current traffic and congestion levels.

High-speed networks are nonlinear because their behaviour is not proportional to the input signal or traffic load. Instead, the response of the network can be highly nonlinear, meaning that small changes in the input signal can result in large and unpredictable changes in the behaviour of the network. One of the primary causes of non-linearity in high-speed networks is congestion. When a network becomes congested, packets can be dropped or delayed, leading to nonlinear changes in network behaviour. A small increase in the traffic load can lead to a disproportionate increase in packet loss rates or delays, resulting in a nonlinear response. Another factor that can contribute to nonlinearity in high-speed networks is the use of complex network protocols and mechanisms such as congestion control and flow control. These mechanisms can introduce non-linearities into the network behaviour by adjusting the flow of traffic based on the current network conditions.

Overall, the nonstationary and nonlinear nature of high-speed networks poses significant challenges for network design and management. That is why the use of advanced techniques and algorithms to ensure reliable and efficient communication over time is dispatched requirement today. Advanced modelling and simulation techniques are needed to predict the behaviour of the network accurately and optimize its performance. It also requires the development of novel algorithms and protocols that can manage the nonlinear response to changes in traffic load and congestion.

Highlights of this paper are the following:

- General, nonlinear and nonstationary properties of the packet-switched datagram mechanisms belonging to different logical layers (L2-datalink: Ethernet; L3-network: IP; L4-transport: UDP) are proved.
- Overview is given of the nonlinear and nonstationary methods used to analyse time series of the network traffic.
- Properties of the looped communication mechanism based on User Datagram Protocol are presented using captured data series from real network traffic.
- Clusterization of loop-based UDP traffics is made in function of the maximum transfer unit, the bandwidth of the end-to-end session and the segment size.

The structure of the rest of the paper is described as follows: related works on the efficiency of transport layer mechanisms are listed in section two. Section three gives an overview of the applied methodologies that have been considered and proven to be useful for us in this context of analysing nonstationary and nonlinear time series. The results and interpretation of the analysis are listed in chapter four. Finally, we conclude and give the possible continuation of the problems related to this research work.

2. Related work of the UDP performance evaluation

There have been developed several studies on the performance of the UDP in various scenarios. Testbed is applied to compare the performance of TCP and

UDP in terms of throughput, delay, and packet loss for different network conditions [16]. It was found that UDP outperforms TCP in terms of throughput for small data transfers, while TCP is more efficient for large data transfers. However, TCP experiences significant delay and packet loss under heavy network load, while UDP suffers from higher packet loss in all network conditions. It discusses the impact of Quality of Service (QoS) mechanisms on the performance of TCP and UDP, as well. It is found that QoS mechanisms can improve the performance of both TCP and UDP, but their effectiveness depends on the specific application and network conditions.

Another comparison of the performance of TCP and UDP protocols in various simulation scenarios was executed with ns-2 simulator [4]. The study evaluates the performance of TCP and UDP in terms of throughput, delay, and packet loss under different network conditions, including different traffic loads and network topologies. It was stated that the choice of the congestion control algorithm and the buffer size can significantly affect the performance of TCP and UDP and that different algorithms and buffer sizes may be more suitable for different network scenarios.

An experimental study of the throughput performance for UDP and VoIP traffic in IEEE 802.11 wireless networks is given in [10]. The study aims to investigate the impact of network parameters such as distance, packet size, and channel conditions on the throughput performance of UDP and VoIP traffic. The paper conducts a series of experiments using a testbed consisting of a set of wireless access points and clients. They measure the throughput performance of UDP and VoIP traffic under different network conditions, including varying the distance between the access point and the client, the packet size, and the channel conditions. The results show that the throughput performance of UDP and VoIP traffic is significantly affected by the network parameters. The throughput performance of UDP traffic decreases as the distance between the access point and the client increases, while the throughput performance of VoIP traffic remains relatively stable.

A research work proposes a new protocol called Performance Adaptive UDP (PA-UDP) for high-speed bulk data transfer over dedicated links [6]. The study aims to overcome the limitations of traditional UDP, which suffers from high packet loss and delay under heavy network load. PA-UDP is designed to adapt its performance to the network conditions by dynamically adjusting the packet size and transmission rate based on feedback from the receiver. The protocol uses a feedback mechanism that allows the receiver to inform the sender about the status of the network and adjust the packet size and transmission rate accordingly. The results of the work show that PA-UDP outperforms both TCP and UDP in terms of throughput and delay under heavy network load while maintaining low packet loss.

The research paper [13] proposes a new UDP-based protocol called UDT (UDP-based Data Transfer) for high-speed data transfer over wide area networks (WANs). This study aims to overcome the limitations of traditional TCP, which is not well-suited for high-speed data transfer over WANs due to its congestion control mechanism and its reliance on a reliable transport layer. UDT is designed to provide reliable and high-speed data transfer over WANs by using a congestion control algorithm that is optimized for high-speed networks, and by integrating several features such as error detection and recovery, flow control, and adaptive data rate control. The protocol also uses a feedback mechanism that allows the receiver to inform the sender about the status of the network and adjust the transmission rate accordingly.

A new protocol called Reliable Blast UDP (RBUDP) for high-speed bulk data transfer over wide area networks (WANs) is proposed [14]. The study aims to address the limitations of traditional UDP, which suffers from high packet loss and delay under heavy network load, and traditional TCP, which is not well-suited for high-speed data transfer over WANs due to its congestion control mechanism. RBUDP is designed to provide reliable and predictable high-speed bulk data transfer over WANs. It uses a combination of several techniques, including a reliability mechanism that ensures the reliable delivery of data, and an adaptive congestion control mechanism that adjusts the transmission rate based on the feedback from the receiver. The packet-blasting technique allows the sender to send multiple packets without waiting for an acknowledgment.

A paper that evaluates the security and performance of the QUIC (Quick UDP Internet Connections) protocol is [19]. The study aims to provide a comprehensive analysis of the protocol security features and performance characteristics to better understand the benefits and limitations of using QUIC. It provides an overview of the QUIC protocol and its key features, including the use of encryption, multiplexing, and congestion control. Then the security of QUIC is evaluated by analyzing its resistance to various types of attacks, including network-level attacks, cryptographic attacks, and protocol-level attacks. It compares the security of QUIC to that of other transport layer protocols such as TCP and Transport Layer Security (TLS). The performance of QUIC is evaluated in terms of throughput, latency, and fairness. They compare the performance of QUIC to that of TCP and other transport layer protocols using a testbed and several real-world scenarios. They analyze the impact of various network conditions, including network congestion and loss, on the performance of QUIC. The results show that QUIC provides better security than TCP and TLS, as it is less vulnerable to attacks such as network-level attacks and cryptographic attacks. The authors also found that QUIC performs better than TCP, especially in scenarios with high packet loss and network congestion. However, they observe that QUIC can be unfair in some scenarios, as it may prioritize traffic from certain connections over others.

Novelty in this paper is the usage of nonlinear and nonstationary empirical evaluation methods to evaluate looped data transfer traffic running on a datagram communication stack. Interpretation of the number of zero-crossings of the decomposed signal using the Fourier transform is another result of this research work.

3. Applied methodology

Since the UDP-based communication mechanisms are nonstationary and nonlinear processes we give an overview of the most important related statistical analysis methods of such time series. We will include Discrete Fourier Transform (DFT), Short Time Fourier Transform (STFT), Discrete Wavelet Transform (DWT), Empirical Mode Decomposition (EMD), and Variational Mode Decomposition (VMD) methods. In each case, we consider a time series $x(t) \in \mathbb{R}$ and $x[k], k = [0, 1, \dots, N-1]$ denotes its observations in N equidistant discrete time points. The common approach of these methods is to decompose the inter-arrival time (IAT) series of the UDP-based traffic denoted by $x(t)$ into a sum of orthogonal or approximately orthogonal modes and residual $\mathrm{res}(t)$. The exact or approximate orthogonality of the modes depends on the method applied and in the case of exact orthogonality, the residual is null. The number of modes k also depends on the method applied and the proper value of it can determine exactly with closed formula or approximate with an algorithm. The decomposition formula is given in the following equation:

$$x(t) = \sum_{i=1}^{k} \mathrm{mode}_i(t) + \mathrm{res}(t). \tag{3.1}$$

These modes belong to frequency bands being disjunct or nearly disjunct depending on the decomposition method applied [2, 18]. The main properties of the discussed decomposition methods are given in Table 1.

Table 1. Main properties of the decomposition methods.

Property	**DFT**	**STFT**	**DWT**	**EMD**	**VMD**
Time domain aspect	No	Yes	Yes	Yes	Yes
Frequency domain aspect	Yes	Yes	Yes	Yes	Yes
Filtering aspect	Global	Linear	Dyadic	Dyadic	Linear

It should be mentioned that each method listed except DFT has both time and frequency domain aspects. Despite this fact, the importance of the DFT is indisputable in the characterization of stationary modes. The discrete Fourier transform of signal $x(t)$, is:

$$\mathbb{F}\{x[n]\} = X[k] = \sum_{n=0}^{N-1} x[n] e^{-i2\pi kn/N}, \quad k = 0, 1, \dots, N-1,$$

see [3]. The inverse discrete Fourier transform of the sequence $X[k]$ is:

$$\mathbb{F}^{-1}\{X[k]\} = x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{i2\pi kn/N}, \quad n = 0, 1, \dots, N-1.$$

The module square $|X[k]|^2$ of $X[k]$ is called periodogram which leads to the power spectral density (spectrum) estimate of $x(t)$. Terms $W_N = e^{i2\pi/N} \in \mathbb{C}$ are the complex N^{th} roots of unity, where $i = \sqrt{-1}$. FFT is the fast algorithm for computing the DFT taking 2^r-points ($r \in \mathbb{N}$) for sequences with length $N = 2^r$. DFT has many important features that we mention here just the Convolution Theorem for two sequences $x_1[n]$ and $x_2[n]$:

$$\mathbb{F}\{x_1[n] * x_2[n]\} = \mathbb{F}\{x_1[n]\} \cdot \mathbb{F}\{x_2[n]\}.$$

The STFT is a signal processing technique that analyzes the frequency of signal content over time [12]. It does this by breaking the signal into small, overlapping segments and performing a Fourier transform on each segment. It uses a window function w[n] that is brief in duration. It has value one in the interval $[0, N_w - 1]$ and zero elsewhere. The SFTF of the signal $x(t)$ is given by the following formula:

$$X[n,k] = \sum_{m=n-(N_w-1)}^{n} w[n-m]x[n]e^{-i\frac{2\pi}{N}km}.$$

STFT examines the frequency content of a signal over time, which can be useful for analyzing changes in the frequency content of the signal.

The DWT is a type of transform that involves breaking a signal down into different frequency bands or scales using wavelet functions making it possible to analyze the signal in both time and frequency domains simultaneously [11]. DWT is based on the mother wavelet $\Psi(t) \in L^2(\mathbb{F})$, which fulfills following condition:

$$\int_0^\infty |\mathbb{F}\{\Psi(k)\}|^2 \frac{\mathrm{d}k}{k} < \infty,$$

where $\mathbb{F}\{\cdot\}$ represents the Fourier transform. Another requirement against the mother wavelet is to have vanishing moments for $M \geq 1$, i.e.:

$$\int_{-\infty}^{\infty} t^m \Psi(t)\,\mathrm{d}t = 0, \quad m = 0, 1, 2, \dots, M.$$

Discrete wavelets are created from the mother wavelet by the following formula:

$$\psi_{j,k}(t) = \frac{1}{\sqrt{2^j}} \Psi\left(\frac{t}{2^j} - k\right),$$

where j and k are called scaling factor and displacement in time, respectively. DWT of the signal $x(t)$ is given by the following formula:

$$DWT\{x(t)\}[j,k] = \int_{-\infty}^{\infty} x(t)\psi_{j,k}(t)\,\mathrm{d}t.$$

The orthogonal wavelet basis functions are typically designed to be localized in both time and frequency, which makes them well-suited for analyzing signals at multiple resolutions. This method is useful for detecting and analyzing localized features of different sizes, shapes, spikes or transitions.

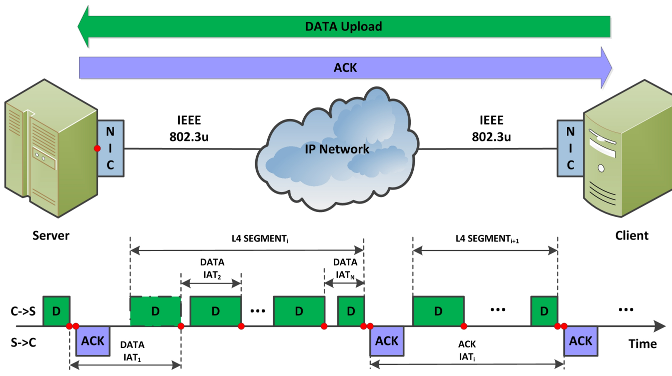


Figure 1. Measurement scenario of the looped UDP traffic.

EMD is a signal processing technique used to extract the underlying oscillatory components of a complex signal [15]. It decomposes a signal $x(t)$ into a set of intrinsic mode functions (IMFs) that represent the different oscillatory modes of the signal, from high-frequency components to low-frequency components. The EMD algorithm identifies the local maxima and minima of the signal and then fits an envelope to the signal by connecting these extrema using cubic splines. The difference between the original signal and the envelope is called a "residue" or "detail" signal, which represents the high-frequency components of the signal. The process is repeated on the residue signal, generating a new IMF and a new residue signal. This process continues until the residue signal can no longer be decomposed into further IMFs. The resulting IMFs are ordered by their frequency content, with the highest frequency components appearing in the first IMFs and the lowest frequency components appearing in the last IMFs. The Hilbert transform $\mathbb{H}\{x(t)\}$ of the signal $x(t)$ is given by the following formula:

$$\mathbb{H}\{x(t)\} = y(t) = \frac{1}{\pi} P \int_{-\infty}^{\infty} \frac{x(\tau)}{t-\tau}\,\mathrm{d}\tau,$$

where P is the Cauchy principal value. Hilbert transform is finite for each real function of class L^p. Based on $x(t)$ and $y(t)$ an analytic signal, $z(t) \in \mathbb{C}$ is created as:

$$z(t) = x(t) + iy(t) = a(t)e^{i\theta(t)},$$

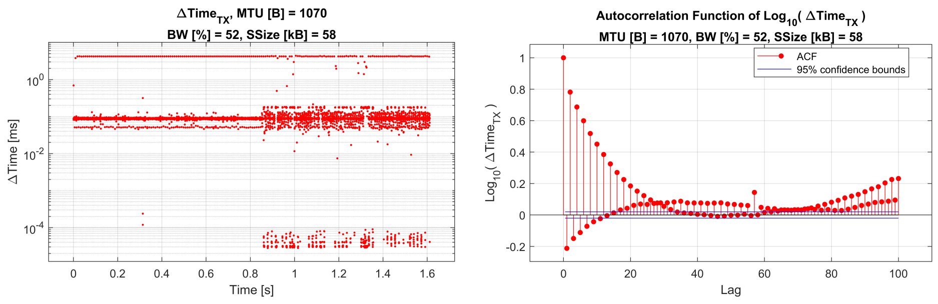
where $a(t)$ and $\theta(t)$ are the amplitude and the phase, respectively defined by:

$$a(t) = [x^2(t) + y^2(t)]^{1/2}, \quad \theta(t) = \arctan\left(\frac{y(t)}{x(t)}\right).$$

The instantaneous frequency $\omega(t)$ of the signal $x(t)$ is given by the formula:

$$\omega(t) = \frac{\mathrm{d}\theta(t)}{\mathrm{d}t}.$$

The resulting decomposition of signal $x(t)$ has the expression that conforms to equation (3.1). The number of modes k usually is less than twenty and these modes are just approximately orthogonal. The last residual is a monotone function in time. EMD is a data-driven method and does not rely on any predefined basis functions, making it suitable for analyzing nonstationary and nonlinear signals, such as those found in networking, biomedical and financial data, etc.



(a) **(b)**

Figure 2. a) Data Interarrival time series $x(t)$; b) Autocorrelation function of $x(t)$. File transfer case: (MTU, BW, SSize) = (1070 B, 52 %, 58 kB).

Variational Mode Decomposition (VMD) is another signal processing technique that, like EMD, it is used to extract the underlying oscillatory components of a complex signal [5]. The signal $x(t)$ is decomposed into a set of stationary functions by solving an optimization problem that seeks to find a set of complex-valued modes that satisfy a sparsity constraint given by:

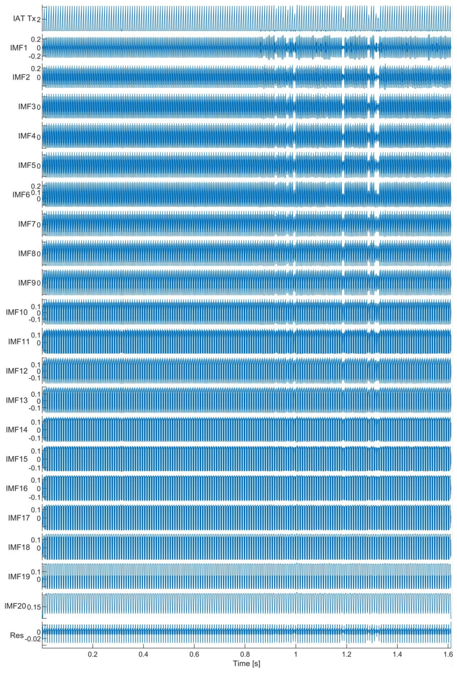
$$\min_{\{\mathrm{mode}_k\},\{\omega_k\}} \left\{ \sum_k \left\| \partial_t [(\delta(t) + \frac{i}{\pi t}) * \mathrm{mode}_k(t)] e^{-i\omega_k t} \right\|_2^2 \right\},$$

where ω_k are instantaneous frequencies, ∂_t is the partial derivative with respect to time, $\delta(t)$ is the Dirac delta function and $\|\cdot\|_2^2$ represents the square of Frobenius norm 2. The optimization problem is solved using an iterative algorithm that alternates between updating the mode functions and updating the sparsity constraint.

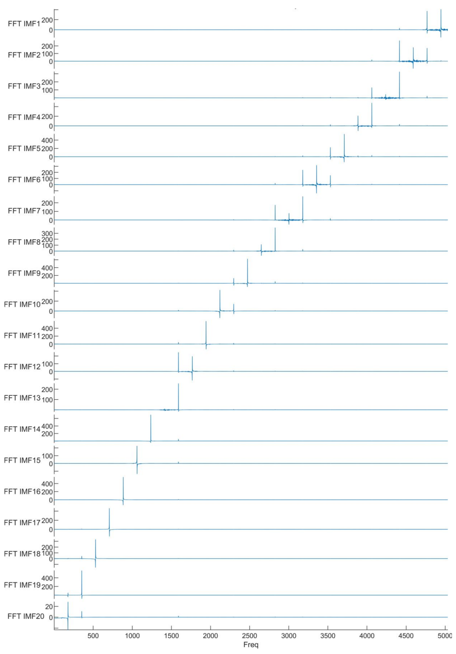
The resulting VMD modes are ordered by their frequency content, with the highest frequency components appearing in the first modes and the lowest frequency components appearing in the last modes. In this case, the modes conform to equation (3.1) are orthogonal. VMD has been shown to be effective in extracting the intrinsic modes of nonstationary and nonlinear signals, such as those found in audio, image, and biomedical data. VMD also has some advantages over EMD, such as better noise suppression and faster convergence.

4. Measurement scenario and basic features of the looped UDP data traffic

A UDP-based communication session with a loop was used to upload 1080 times a fixed-size data file to a test server with different combinations of the independent parameter triplets: Maximum Transfer Unit (MTU) of the interface card, Bandwidth at the application layer (Bw) and UDP segment size (SSize). The rule of the UDP loop is the following: at the reception of the data segment by the server one acknowledgment is sent back to the client. This control information sent to the client contains two main elements: i) binary status information about the successful reception or failure of the IP packets belonging to the UDP segment; ii) timestamp of the server. The status information is used for the retransmission of the wrong IP packets. The timestamp keeps track of the application-level bandwidth usage.



(a)



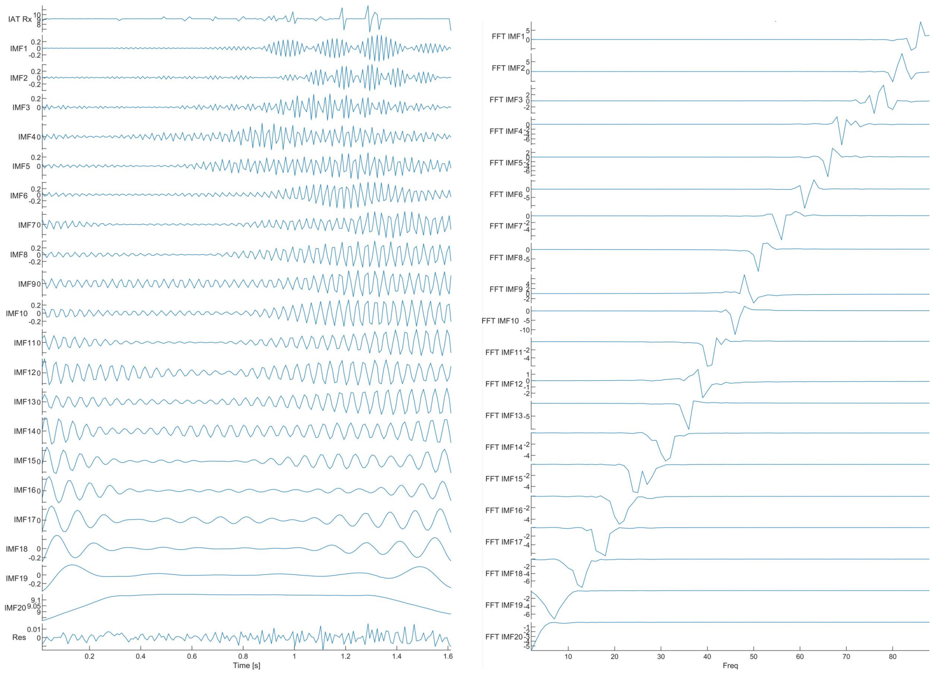
(b)

Figure 3. a) VMD (in [ms] scale) of data IAT series; b) Spectrum (frequency in [Hz]) of VMD of data IAT series. File transfer case: (MTU, BW, SSize) = (1070 B, 52 %, 58 kB).

The size of the file was 10 MB and the values of the network parameters were: $\mathrm{MTU}[i] = 510 + i \cdot 80$ [B], $i = 1, 2, \ldots, 12$, $\mathrm{Bw}[j] = 37 + j \cdot 6$ [%], $j = 1, 2, \ldots, 9$ and $\mathrm{SSize} = -2 + k \cdot 6$ [kB], $k = 1, 2, \ldots, 10$. IAT time series of the fast communication services (see figure 2a) have nonstationary characters (see figure 2b). Decomposed modes of the IAT serve to determine characteristic time patterns in multi-resolution frequency scales.

To characterize IAT processes in time-frequency domains we used the Hilbert transform of the modes. To extract the DC component applied the following property of the Hilbert transform: the $\mathbb{H}\{x(t)\}$ is the phase shift by $\pi/2$ of the original signal $x(t)$. The remaining AC components serve to determine the spectrum of the signals and extract patterns. We processed the EMD and VMD of the data and acknowledged IAT time series of the traffics. The number of modes is limited, $k \leq 20$ and based on it mode components were determined (see VMD figures 3a and 4a).

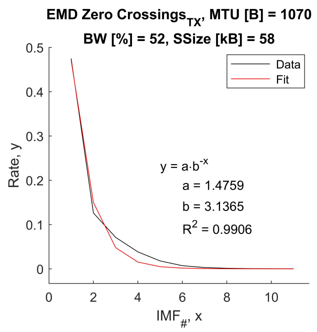
It is an important feature of the VMD that the Fourier transform of the modes have disjunct intensities in the function of the frequency and this dependence is close to being linear (see VMD figures 3b and 4b). The EMD of the data and acknowledgment traffic have different behaviour from the VMD. The EMD modes have nonlinear dependence on time and frequency. They are mentioned by the majority of scientific papers to be dyadic filters [1, 8, 17, 20].



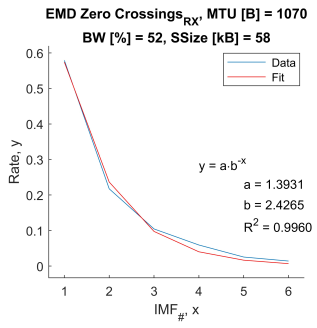
(a) **(b)**

Figure 4. a) VMD (in [ms] scale) of acknowledgment IAT series; b) Spectrum (frequency in [Hz]) of VMD of acknowledgment IAT series. File transfer case: (MTU, BW, SSize) = (1070 B, 52 %, 58 kB).

Because the EMD and VMD modes are symmetrical, independent and stationary time functions the zero-crossing rates represent a proper measure of the frequencies. For a clear view of this dependence see figures 5a and 5b.



(a)



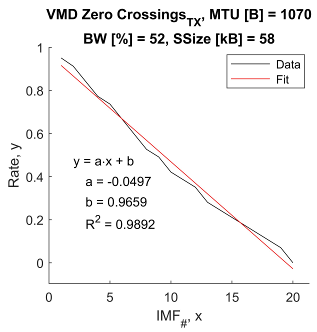
(b)

Figure 5. a) Zero-crossings of EMD of data IAT series; b) Zero-crossings of EMD of observed IAT series. File transfer case: (MTU, BW, SSize) = (1070 B, 52 %, 58 kB).

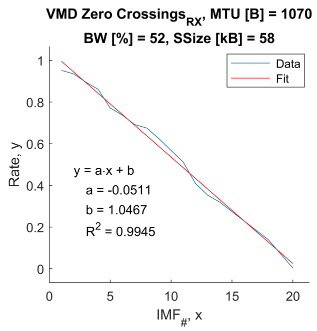
It was found that for all cases the zero-crossing rate of EMD and VMD modes for both data and acknowledgment traffic have their own exponential and linear relations, respectively:

$$\mathrm{EMDRate}_{Tx}(k) = a_{Tx}\cdot e^{-k\cdot b_{Tx}}, \quad \mathrm{EMDRate}_{Rx}(k) = a_{Rx}\cdot e^{-k\cdot b_{Rx}},$$
$$\mathrm{VMDRate}_{Tx}(k) = a_{Tx}\cdot k + b_{Tx}, \quad \mathrm{VMDRate}_{Rx}(k) = a_{Rx}\cdot k + b_{Rx},$$

where fit parameters a_{Tx}, b_{Tx}, a_{Rx} and b_{Rx} depend on the traffic case determined by the triplet (MTU, BW, SSize).



(a)

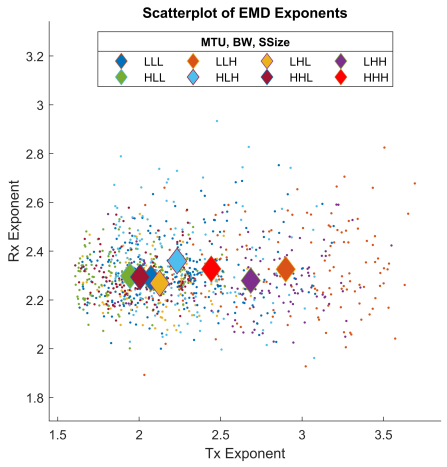


(b)

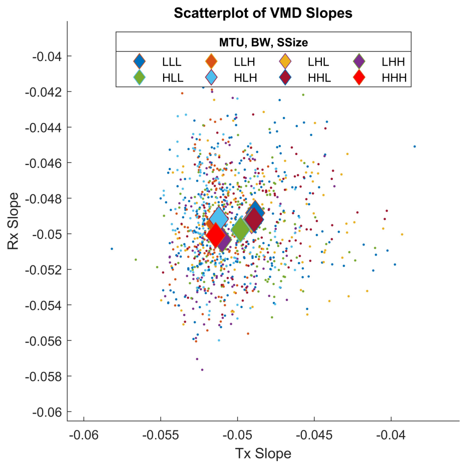
Figure 6. a) Zero-crossings of VMD of data IAT series; b) Zero-crossings of VMD of acknowledgment IAT series. File transfer case: (MTU, BW, SSize) = (1070 B, 52 %, 58 kB).

Each range of the MTU, BW and SSize was divided into lower (L) and higher (H) half, resulting in eight 3D subspaces of the independent parameter combinations: LLL, LLH, LHL, LHH, HLL, HLH, HHL, HHH (see figures 7a and 7b). Should mention here a very important aspect: the functions of zero-crossing rates of the EMD representing the frequencies of the modes are exponential functions of base different than 2 (see figure 7a). Most bases of exponent of transmit data are in the range $(1.7, 3.5)$ and bases values of acknowledgments are in the range $(2, 2.8)$ with the mean 2.3. This fact modifies the slightly the dyadic filter property of the EMD method. The corresponding slope of the linear fitting parameters of VMD is converged around $a^* = 0.05 = 1/20 = 1/k_{\max}$ which is caused by the maximum value of VMD modes $k \leq 20$ (see figure 7b).

Based on the position of mass points of each parameter subspace three clusters as group of traffic patterns can be identified: $[C_1, C_2, C_3] = [xxL, LxH, HxH]$, where character x has the meaning of neuter effect of the corresponding parameter (i.e. $xxL = LLL \vee LHL \vee HLL \vee HHL$, where $\vee$ is the logical operator OR).



(a)



(b)

Figure 7. a) Scatter plot of data and acknowledgment EMD exponents; b) Scatter plot of data and acknowledgment VMD slopes.

Having this clusterization property of the zero-crossing rate fitting parameters we can affirm that in the case of looped UDP services, we have three traffic patterns groups in the function of the maximum transfer unit, the bandwidth of the traffic and UDP segment size:

C_1) When the segment size, SSize is low the value of the other two parameters has no significant effect on the traffic. In this situation, the intensity of IP packet fragmentation is low at the data sender. The character of the traffic is stochastic with the bandwidth close to the limit set by the Bw parameter.

C_2) When the maximum transfer unit, MTU is low and the UDP segment size, SSize is high the bandwidth parameter, Bw has no strong effect on the traffic.

Intensive fragmentation is executed at the IP layer because large data segments are sent through short Ethernet frames. The character of the UDP traffic is dominated by fragmentation. The ratio of data frames over acknowledgment frames is the largest in these cases.

C_3) When the maximum transfer unit, MTU, and UDP segment size, SSize are high, the bandwidth parameter, Bw has no significant effect on the traffic behaviour. The intensity of the fragmentation at the IP layer is moderate, and the fluctuation of the bandwidth below the maximum is mostly reduced.

5. Summary of the results

A data file of 10 MB size was uploaded 1080 times with different traffic parameter triplets: maximum segment size, application bandwidth, and segment size of looped UDP traffic. An overview of the nonlinear and nonstationary analysis methods based on the decomposition of the interarrival times of the data upload and acknowledgment download traffics was given in the paper. It was found that the modes determined by Empirical Mode Decomposition and Variational Mode Decomposition belong to different frequency bands making it possible to characterize these stationary modes by the ratio of zero-crossing. The zero-crossing rates have an exponential and linear dependence on the number of modes of the Empirical Mode Decomposition and Variational Mode Decomposition, respectively. The Empirical Mode Decomposition is in the general b-base filter with $bget2$. The upload data traffic of the looped UDP has three groups of traffic patterns in the function of traffic parameter triplets (MTU, Bw, SSize), while the download acknowledgment traffic is a result of b-base filter with $b^* = 2.3$ instead of a dyadic filter ($b = 2.0$) published in a lot of scientific papers until now. More analyses are required to determine the dependence of the value b of the b-basis filter in the case of Empirical Mode Decomposition. Similarly, the dependence of the slope a in Variational Mode Analysis should continue to interpret.

References

[1] M. B. Abd-el-Malek, S. S. Hanna: *Using filter bank property to simplify the calculations of Empirical Mode Decomposition*, Communications in Nonlinear Science and Numerical Simulation 62 (2018), pp. 429–444, issn: 1007-5704, doi: 10.1016/j.cnsns.2018.02.035.

[2] R. Bazi, T. Benkedjouh, H. Habbouche, S. Rechak, N. Zerhouni: *A hybrid CNN-BiLSTM approach-based variational mode decomposition for tool wear monitoring*, The International Journal of Advanced Manufacturing Technology 5 (2022), pp. 3803–3817, doi: 10.1007/s00170-021-08448-7.

[3] P. Bloomfield: *Fourier analysis of time series: an introduction*, John Wiley & Sons, 2004.

[4] F. T. AL-Dhief, N. Sabri, N. A. Latiff, M. Abbas, A. Albader, M. A. Mohammed, R. N. AL-Haddad, Y. D. Salman, M. Khanapi, et al.: *Performance comparison between TCP and UDP protocols in different simulation scenarios*, International Journal of Engineering & Technology 7.4.36 (2018), pp. 172–176.

[5] K. Dragomiretskiy, D. Zosso: *Variational Mode Decomposition*, IEEE Transactions on Signal Processing 62.3 (2014), pp. 531–544, doi: 10.1109/TSP.2013.2288675.

[6] B. Eckart, X. He, Q. Wu: *Performance adaptive UDP for high-speed bulk data transfer over dedicated links*, in: 2008 IEEE International Symposium on Parallel and Distributed Processing, IEEE, 2008, pp. 1–10.

[7] *Fast transport layer protocol: QUIC*, Official web site of IETF QUIC Working Group, url: https://quicwg.org/.

[8] P. Flandrin, G. Rilling, P. Goncalves: *Empirical mode decomposition as a filter bank*, IEEE Signal Processing Letters 11.2 (2004), pp. 112–114, doi: 10.1109/LSP.2003.821662.

[9] Z. Gál, G. Kocsis, T. Tajti, R. Tornai: *Performance evaluation of massively parallel and high speed connectionless vs. connection oriented communication sessions*, Advances in Engineering Software 157-158 (2021), p. 103010, issn: 0965-9978, doi: 10.1016/j.advengsoft.2021.103010.

[10] S. Garg, M. Kappes: *An experimental study of throughput for UDP and VoIP traffic in IEEE 802.11 b networks*, in: 2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003. Vol. 3, IEEE, 2003, pp. 1748–1753.

[11] J. Gilles: *Empirical Wavelet Transform*, IEEE Transactions on Signal Processing 61.16 (2013), pp. 3999–4010, doi: 10.1109/TSP.2013.2265222.

[12] D. Griffin, J. Lim: *Signal estimation from modified short-time Fourier transform*, IEEE Transactions on Acoustics, Speech, and Signal Processing 32.2 (1984), pp. 236–243, doi: 10.1109/TASSP.1984.1164317.

[13] Y. Gu, R. L. Grossman: *UDT: UDP-based data transfer for high-speed wide area networks*, Computer Networks 51.7 (2007), pp. 1777–1799.

[14] E. He, J. Leigh, O. Yu, T. A. DeFanti: *Reliable blast UDP: Predictable high performance bulk data transfer*, in: Proceedings. IEEE International Conference on Cluster Computing, IEEE, 2002, pp. 317–324.

[15] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N.-C. Yen, C. C. Tung, H. H. Liu: *The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis*, Proceedings of the Royal Society of London A: mathematical, physical and engineering sciences 454.1971 (1998), pp. 903–995.

[16] H. K. Rath, A. Karandikar: *Performance analysis of TCP and UDP-based applications in a IEEE 802.16 deployed network*, in: 2011 The 14th International Symposium on Wireless Personal Multimedia Communications (WPMC), 2011, pp. 1–5.

[17] G. Rilling, P. Flandrin: *One or Two Frequencies? The Empirical Mode Decomposition Answers*, IEEE Transactions on Signal Processing 56.1 (2008), pp. 85–95, doi: 10.1109/TSP.2007.906771.

[18] R. C. Sharpley, V. Vatchev: *Analysis of the Intrinsic Mode Functions*, Constructive Approximation 24 (2006), pp. 17–47, doi: 10.1007/s00365-005-0603-z.

[19] M. Soni, B. S. Rajput: *Security and performance evaluations of QUIC protocol*, in: Data Science and Intelligent Applications: Proceedings of ICDSIA 2020, Springer, 2021, pp. 457–462.

[20] Y. Yang, J. Deng, D. Kang: *An improved empirical mode decomposition by using dyadic masking signals*, Signal, Image and Video Processing 9.6 (2015), pp. 1259–1263, issn: 1863-1711, doi: 10.1007/s11760-013-0566-7.

Annales Mathematicae et Informaticae
58 (2023) pp. 56–68
DOI: 10.33039/ami.2023.08.005
URL: https://ami.uni-eszterhazy.hu

The Fritz-John Condition System in Interval Branch and Bound method

Mihály Gencsi, Boglárka G.-Tóth

University of Szeged
{gencsi,boglarka}@inf.u-szeged.hu

Abstract. The Interval Branch and Bound (IBB) method is a good choice when a rigorous solution is required. This method handles computational errors in the calculations. Few IBB implementations use the Fritz-John (FJ) optimality condition to eliminate non-optimal boxes in a constrained nonlinear programming problem. Applying the FJ optimality condition implies solving an interval-valued system of equations. In the best case, the solution is an empty set if the interval box does not contain an optimizer point. Solving this system of equations is complicated or unsuccessful in many cases. This problem can be caused by the interval box being too wide, the defined system of equations containing unnecessary constraints, or the solver being unsuccessful. These unsuccessful attempts have a negative outcome and only increase the computation time. In this study, we propose some modifications to reduce the running time and computational requirements of the Interval Branch and Bound method.

Keywords: Global Optimization, Interval Arithmetic, Fritz-John condition, Branch and Bound method, Optimality condition

AMS Subject Classification: 90C26, 65G30, 90C30

1. Introduction

There are many applications in which we are looking for a rigorous solution to a mathematical problem. For example, in physics or chemistry, we look for the stability point of a substance. Sometimes we obtain a stability point, but in this environment, the material is very unstable; it can fall apart even with a small change.

In this study, we focus on solving the constrained nonlinear programming problems with inequality and general bound constraints. We deal with the following

Submitted: July 30, 2023
Accepted: August 7, 2023
Published online: August 8, 2023

n-dimensional nonlinear problem,

$$\begin{aligned} \underset{x \in \boldsymbol{y} \subseteq \mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m, \end{aligned} \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, m$ are continuously differentiable nonlinear functions, and the interval box $\boldsymbol{y} = [\underline{y}, \overline{y}]$ denotes a general bound constraint. We search for the global optimum using a guaranteed method, the Interval Branch and Bound (IBB) method. Solving a constrained nonlinear programming problem is, in general, very difficult. Sometimes we can only solve a low-dimensional instance or a smaller subproblem. In the IBB method, we replace the problem with smaller subproblems. We try to discard a subproblem by calculating upper and lower bounds and checking the feasibility or optimality. One of the best ways to rigorously compute the bounds for the subproblem is using interval arithmetic (IA). In IA, the rounding error or imprecision of the parameters is automatically taken into account by replacing the numbers with intervals. Today, several implementations of the IBB can be found in the literature. However, many of them do not use Fritz-John (FJ) or Karush-Kuhn-Tucker (KKT) optimality conditions to discard non-optimal subproblems. These mean solving an interval-valued system of equations. In this study, we use the improved version of IBB, which can solve both the unconstrained and the constrained cases in a reasonable time. We also study the solvability of the interval FJ optimality conditions, which are more general than the interval KKT optimality conditions.

In the following section, we introduce the basic terms and concepts of IA and the solution method for the interval-valued system of equations. We also describe the prototype of the IBB method. In Section 3, we study the FJ Condition System (FJ-CS) and extend it to intervals by defining the Normalized Interval Fritz-John Condition System (NIFJ-CS). In Section 4, we consider four methods to solve NIFJ-CS, analyze them, and present some additional improvements to reduce the running time of the methods. We compare the methods described using computational experiments in Section 5. Finally, in Section 6, we summarize this study and make suggestions for future directions.

2. Interval Branch and Bound method

In this section, we introduce the basic concepts of Interval Arithmetic (IA). We demonstrate the methods for solving interval-valued systems of equations, which we use to solve the NIFJ-CS. We also briefly introduce the prototype of the Interval Branch and Bound (IBB) method.

2.1. Interval arithmetic

Interval arithmetic is the basis of the Interval Branch and Bound method, in which numbers are replaced by a range of numbers called an *interval*. In this way, rounding and measurement errors are avoided by enclosing the number in intervals. Following the basic notation used in the literature [8], the *intervals* are denoted by $\boldsymbol{x} = [\underline{x}, \overline{x}]$, where $\underline{x}$ and $\overline{x}$ describe the lower and upper bounds of the interval, respectively. Therefore, we can define n-dimensional interval vectors as $\boldsymbol{x} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) \in \mathbb{I}^n = \mathbb{I} \times \cdots \times \mathbb{I}$, which can be called *intervalbox* or *box*, where $\mathbb{I}$ is the set of intervals. In addition, we define some properties of the intervals. For example, the *midpoint* of an interval $\boldsymbol{x}$ is denoted by $\operatorname{mid}(\boldsymbol{x}) = \frac{1}{2}(\overline{x} + \underline{x})$ or the *width* of the interval $\boldsymbol{x}$ by $\operatorname{wid}(\boldsymbol{x}) = \overline{x} - \underline{x}$. We can extend this to boxes as well, as follows. The midpoint of a n-dimensional box $\boldsymbol{x} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)^T$ is given by $\operatorname{mid}(\boldsymbol{x}) = (\operatorname{mid}(\boldsymbol{x}_1), \dots, \operatorname{mid}(\boldsymbol{x}_n))^T$ and the width of the box by $\operatorname{wid}(\boldsymbol{x}) = \max\{\operatorname{wid}(\boldsymbol{x}_i) : i = 1, \dots, n\}$.

Operations such as addition, multiplication, subtraction, and division can be extended to intervals. The *interval arithmetic operations* are defined by $\boldsymbol{x} \odot \boldsymbol{y} = \{x \odot y : x \in \boldsymbol{x}, y \in \boldsymbol{y}\}$ for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{I}$, where $\odot \in \{+, -, \cdot, /\}$ and $\boldsymbol{x}/\boldsymbol{y}$ is defined only if $0 \notin \boldsymbol{y}$.

Furthermore, $\mathbf{f} : \mathbb{I}^n \to \mathbb{I}$ is an *inclusion function* for $f : \mathbb{R}^n \to \mathbb{R}$ if it satisfies $\{f(x) : x \in \boldsymbol{x}\} \subseteq \mathbf{f}(\boldsymbol{x})$ for all interval boxes $\boldsymbol{x} \subset \mathbb{I}^n$ within the domain of f. In many cases, the inclusion function is wider than the image of the function because it is overestimated. Note that if $\mathbf{f}$ is an inclusion function, we can obtain the lower and upper bounds on f taking $\underline{\mathbf{f}}(\boldsymbol{x})$ and $\overline{\mathbf{f}}(\boldsymbol{x})$, respectively.

Elementary functions (such as sin, cos, exp, etc.) are easily extended to intervals. The simplest inclusion function is called *natural interval extension*, which means that we replace the numbers x by the box $\boldsymbol{x}$ at each occurrence in the function f and compute the inclusion function in interval terms. One of the most commonly used inclusion functions is the *centered form*, which is a better approximation of the inclusion function than the natural interval extension but takes more time to compute. We compute the centred form using equation $\mathbf{f}_c(\boldsymbol{x}) = \mathbf{f}(c) + \nabla\mathbf{f}(\boldsymbol{x}) \cdot (\boldsymbol{x} - c)$, where $c \in \boldsymbol{x}$ is usually the centre of the box and $\nabla\mathbf{f}(\boldsymbol{x})$ is the inclusion of the gradient of f over $\boldsymbol{x}$. We use Automatic Differentiation (AD) to compute the inclusion of gradients [12]. In this method, only the derivative rules are needed for the calculation, and we calculate the function value and the derivative at the same time. The interested readers can find more information about Interval Arithmetic methods in [3, 5].

2.2. Solvers for interval-valued system of equations

By solving an interval-valued system of equations, we want to find an enclosure of all possible solutions within a starting box. However, if the enclosures of the coefficients are too wide, we cannot remove any part of the box. Sometimes we can improve the solvability of the methods by using preconditioners. This simply means that we transform the system of equations to be more suitable for the solver by using a transformation matrix. We use the midpoint preconditioner, which can be found in [7], along with other preconditioners. To solve the NIFJ-CS we use an iterative method, the Interval Gauss-Seidel method (IGS), and a direct method, the Interval LU decomposition (ILUD). These two methods are straightforward extensions of methods to solve the real system of equations. The interested reader can find more information about these two methods and their implementations in [1, 13].

2.3. The prototype of the Interval Branch and Bound method

Branch and Bound is a framework for solving optimization problems in which the main problem is divided into subproblems and an attempt is made to discard the subproblems that do not contain the optimal point using some discarding rules. IBB is based on Interval Arithmetic and Branch and Bound concepts. All steps are extended to intervals. These steps must be specified for a particular implementation, and their choice can have a large impact on the efficiency of the method. Since we will only focus on one discarding test (the Fritz-John optimality test), the remaining steps are done in the usual way.

In an IBB method [6, 10], there are five main steps: selection, bounding, discarding, division, and termination. The prototype algorithm is shown in Algorithm 1.

Algorithm 1 Prototype Interval Branch and Bound method

```
$L_{work} \leftarrow \{\boldsymbol{x}\}; L_{result} \leftarrow \{\}$;
while $L_{work} \neq \emptyset$ do
    Select a box $\boldsymbol{x}$ from $L_{work}$                         ▷ Selection Rule
    Compute bounds for $\mathbf{f}(\boldsymbol{x}), \mathbf{g}_i(\boldsymbol{x}), \forall i \in M$        ▷ Bounding Rule
    if $\boldsymbol{x}$ cannot be discarded then                      ▷ Discarding Tests
        Divide $\boldsymbol{x}$ into subboxes $\boldsymbol{x}_1, \dots, \boldsymbol{x}_r$             ▷ Division Rule
        for i = 1 to r do
            if TerminationCriterion($\boldsymbol{x}_i, \varepsilon$) then      ▷ Termination Rule
                Store $\boldsymbol{x}_i$ in $L_{result}$
            else
                Store $\boldsymbol{x}_i$ in $L_{work}$
            end if
        end for
    end if
end while
return $L_{result}$
```

First, we initialize the working list (L_{work}) with the bound constraint and the result list (L_{result}) with an empty set. We stop the method when the working list is an empty set. In each iteration, we select a box $\boldsymbol{x}_{selected}$ from the working list with some selection rules. The selection rules can be LIFO, FIFO, or the lowest lower bound. In our implementation, we use the lowest lower bound selection rule. In the next step, we bound the objective of the box $\boldsymbol{x}_{selected}$ by computing the natural interval extension or the centered form. Sometimes, in the unconstrained case, we can discard the selected box, because it is monotone, concave, or does not contain an optimal point (Interval Newton test), by using higher-order information, e.g. gradient. When we solve a constrained nonlinear problem, we can add two more tests. The first is the feasibility test, where we investigate whether the selected box is a subset of the feasible area by computing the bounds on the constraints. In this test, there are three possible cases: the undetermined case when some computed bounds contain zero ($\exists i \in 1,\dots,m$: $0 \in \mathbf{g}_i(\boldsymbol{x}_{selected})$), the infeasible case when one of the lower bounds of the computed bounds is greater than zero ($\exists i \in 1,\dots,m$: $\underline{\mathbf{g}_i}(\boldsymbol{x}_{selected}) > 0$), and the strictly feasible case, when all the upper bounds of the constraints are less than zero ($\overline{\mathbf{g}_i}(\boldsymbol{x}_{selected}) < 0, \forall i = 1,\dots,m$). To examine the optimality of the box, we can use the FJ or KKT optimality tests. If the selected box is not discarded and satisfies the termination rule, we can move it to the result list. The termination rule usually examines the width of the interval or the width of the inclusion function. Otherwise, we divide the selected box $\boldsymbol{x}_{selected}$ into subboxes $\boldsymbol{x}_1,\dots,\boldsymbol{x}_r$ using the division rule. The division rule can be bisection, trisection, multisection, etc. In this work, we use bisection as a division rule, dividing the box into two subboxes along the widest dimension.

In the next section, we will discuss the interval version of the FJ optimality conditions in more detail. We will examine and compare four possible solution methods.

3. The Interval Fritz-John Condition System

The Fritz John conditions are necessary conditions for a solution to be optimal in nonlinear programming. For problem (1.1), the FJ optimality conditions [9] for a given point x are the equations

$$\mu_0 \nabla f(x) + \sum_{i\in M_b} \mu_i \nabla p_i(x) + \sum_{j\in M_c} \mu_j \nabla g_j(x) = 0 \tag{3.1}$$

$$\mu_i p_i(x) = 0, \quad i \in M_b \tag{3.2}$$

$$\mu_j g_j(x) = 0, \quad j \in M_c \tag{3.3}$$

$$\mu_i \geq 0, \quad i \in M_b \cup M_c \cup \{0\}, \tag{3.4}$$

where μ_i are the Lagrange multipliers, M_b and M_c is the set of the bound constraints and the general constraints, respectively. Thus, μ_0 is the Lagrange multiplier of the objective function. If the system can be solved, we confirm that x can be the optimal solution. Note that we can easily formulate a bound constraint for x_i as $p_{i_u}(x) = x_i - \overline{y_i}$ and $p_{i_l}(x) = \underline{y_i} - x_i$, where $\overline{y_i}$ and $\underline{y_i}$ are the upper and lower bounds of the general bound constraint, respectively.

The straightforward extension of the Fritz John optimality conditions (3.1)–(3.4) for a given box $\boldsymbol{x}$ are the interval-valued system of equations

$$\boldsymbol{\mu}_0 \nabla \mathbf{f}(\boldsymbol{x}) + \sum_{i\in M_b} \boldsymbol{\mu}_i \nabla \mathbf{p}_i(\boldsymbol{x}) + \sum_{j\in M_c} \boldsymbol{\mu}_j \nabla \mathbf{g}_j(\boldsymbol{x}) = 0 \tag{3.5}$$

$$\boldsymbol{\mu}_i \mathbf{p}_i(\boldsymbol{x}) = 0, \quad i \in M_b \tag{3.6}$$

$$\boldsymbol{\mu}_j \mathbf{g}_j(\boldsymbol{x}) = 0, \quad j \in M_c \tag{3.7}$$

$$\boldsymbol{\mu}_i \geq 0, \quad i \in M_b \cup M_c \cup \{0\}, \tag{3.8}$$

where $\mathbf{f}(\boldsymbol{x})$, $\mathbf{p}_i(\boldsymbol{x})$, $\mathbf{g}_j(\boldsymbol{x})$ are the inclusion functions, $\nabla\mathbf{f}(\boldsymbol{x})$, $\nabla\mathbf{p}_i(\boldsymbol{x})$, $\nabla\mathbf{g}_j(\boldsymbol{x})$ are the inclusions of the gradients of $f(x), p_i(x), g_j(x)$, respectively. Note that we can reduce the number of equations in the system by considering only the active constraints. We consider a constraint active if the inclusion of the constraint, $\mathbf{p}_i(\boldsymbol{x})$, $\mathbf{g}_j(\boldsymbol{x})$, contains zero. Let B and C be the set of active bound constraints and active constraints, respectively. We can formalize the equations (3.5)–(3.8) for active constraints by replacing M_b with B and M_c with C.

3.1. The normalization of the Lagrange multipliers

The Interval FJ-CS usually does not include a normalization condition. One possible way to normalize Lagrange multipliers, following [2], is to use the equation $\mu_0 + \sum_{i \in B \cup C} \mu_i = 1$. We can easily formulate it as an interval-valued function,

$$\mathbf{r}(\boldsymbol{\mu}) = \boldsymbol{\mu}_0 + \sum_{i \in B \cup C} \boldsymbol{\mu}_i - 1 = 0. \tag{3.9}$$

Moreover, adding this condition to the FJ-CS does not remove any solution, and we can replace the interval $\boldsymbol{\mu}_i \geq 0$ with $[0, 1]$ for all Lagrange multipliers, which improves the success rate of the IGS.

3.2. The Normalized Interval Fritz-John Condition System

As before, B and C are the set of active bound constraints and active constraints, respectively. Let $N = 1 + n + |B| + |C|$ be the dimension of the system, where n is the dimension of the problem. Set the Lagrange multipliers, $\boldsymbol{\mu}_i$ $i \in B \cup C \cup \{0\}$, to the interval $[0, 1]$. When formalizing the system of equations, we consider the normalization function $\mathbf{r}(\boldsymbol{\mu})$ extended to the intervals defined in (3.9). Denote all the variables by $\boldsymbol{t} = [\boldsymbol{x}, \boldsymbol{\mu}]^T$, which is N-dimensional. Thus, for the box $\boldsymbol{t}$, we formalize the Normalized Interval Fritz-John Condition System (NIFJ-CS) as

$$\boldsymbol{\phi}(\boldsymbol{t}) = \begin{bmatrix} \mathbf{r}(\boldsymbol{\mu}) \\ \boldsymbol{\mu}_0 \cdot \nabla\mathbf{f}(\boldsymbol{x}) + \sum_{i \in B} \boldsymbol{\mu}_i \cdot \nabla\mathbf{p}_i(\boldsymbol{x}) + \sum_{j \in C} \boldsymbol{\mu}_j \cdot \nabla\mathbf{g}_j(\boldsymbol{x}) \\ \boldsymbol{\mu}_i \cdot \mathbf{p}_i(\boldsymbol{x}) \qquad i \in B \\ \boldsymbol{\mu}_j \cdot \mathbf{g}_j(\boldsymbol{x}) \qquad j \in C \end{bmatrix} = 0. \tag{3.10}$$

Note that this NIFJ-CS is equivalent to the (3.9), (3.5)–(3.7) interval-valued system of equations.

4. Solving the Normalized Interval Fritz-John Condition System

When we solve an interval-valued optimality condition system, we can discard the box if the solution is an empty set. However, if the enclosures of the gradients are too wide, we cannot remove any part of the box. One possible way to solve the system of equations $\boldsymbol{\phi}(\boldsymbol{t}) = 0$ is to apply the Newton method. Applying the Newton method as in [2], we obtain the system of equations

$$\mathbf{J}(\boldsymbol{t}) \cdot (\boldsymbol{t} - t_0) = -\boldsymbol{\phi}(t_0), \tag{4.1}$$

where $\mathbf{J}(\boldsymbol{t})$ is the Jacobian matrix for $\boldsymbol{\phi}(\boldsymbol{t})$, i.e. $\mathbf{J}_{ij}(\boldsymbol{t}) = \frac{\boldsymbol{\partial}}{\boldsymbol{\partial t}_j}\boldsymbol{\phi}_i(\boldsymbol{t})$ $i, j = 1, \ldots, N$, and t_0 is an interior point of the box $\boldsymbol{t}$. Note that for t_0 we use the midpoint of the interval $\boldsymbol{t}$. In addition, we also use the midpoint preconditioner matrix, with which we transform (4.1) to the system of equations

$$P \cdot \mathbf{J}(\boldsymbol{t}) \cdot (\boldsymbol{t} - t_{mid}) = -P \cdot \boldsymbol{\phi}(t_{mid}), \tag{4.2}$$

where $P = \mathrm{mid}(\mathbf{J}(\boldsymbol{t}))^{-1}$. We solve the system of equations only if one bound constraint is active in each dimension. Otherwise, we skip solving NIFJ-CS, reducing the probability of an unsuccessful Newton method, as the system otherwise becomes too large. To solve (4.2), we use IGS, because this method is more efficient than the ILUD method. However, for IGS we need an initial box $\boldsymbol{t}$, but $\boldsymbol{x}$ is given, and $\boldsymbol{\mu}_i$ can be set to $[0, 1]$ for all Lagrange multipliers. Note that since an unsuccessful IGS step significantly increases the runtime of the IBB method, we apply the Newton method only once.

4.1. Estimating the Lagrange multipliers

In many cases, IGS cannot reduce or discard the box $\boldsymbol{x}$ because the initial bound of $\boldsymbol{t}$ is too large, that is, it contains many possible solutions. One way to reduce this problem is to estimate the Lagrange multipliers before applying the IGS method, and initialize $\boldsymbol{\mu}$ with the estimated bounds. The Lagrange multipliers can be approximated by solving the interval equations

$$\begin{gathered}\mathbf{r}(\boldsymbol{\mu}) = 1 \\ \boldsymbol{\mu}_0 \cdot \nabla\mathbf{f}(\boldsymbol{x}) + \sum_{i\in B}\boldsymbol{\mu}_i \cdot \nabla\mathbf{p}_i(\boldsymbol{x}) + \sum_{j\in C}\boldsymbol{\mu}_j \cdot \nabla\mathbf{g}_j(\boldsymbol{x}) = 0\end{gathered} \tag{4.3}$$

Note that (4.3) are the first two equations of (3.10). To solve these equations, we use the ILUD method mentioned in Section 2.2.

4.2. Methods

To solve the NIFJ-CS (4.2), we investigate four methods based on the Newton method, where we modify one or more steps for each variant.

4.2.1. The naive NIFJ-CS method

In the naive NIFJ-CS method, we formalize the Newton step directly as a system of equations (4.1) and solve with IGS. We initialize the first n component of $\boldsymbol{t}$ with the examined box $\boldsymbol{x}$ and the remaining components with the interval $[0,1]$ (initial bounds for the Lagrange multipliers). In the best case, we obtain an empty set; that is, there is no solution in the examined box $\boldsymbol{x}$. Thus, we can discard the box $\boldsymbol{x}$. Sometimes we obtain as a solution a tighter box within $\boldsymbol{x}$. In this case, we reduce the box $\boldsymbol{x}$ to it. In many cases, however, it is not possible to reduce the size of the box because it is overestimated. In this case, we obtain many boxes as solutions, as IGS splits each component where the diagonal coefficient interval contains zero. In general, we only divide the components $\boldsymbol{\mu}$ into subboxes and leave the box $\boldsymbol{x}$ unchanged. If the box $\boldsymbol{x}$ is unchanged, the Newton step was unsuccessful. If we obtain subboxes divided in the part of the box $\boldsymbol{x}$, we exchange $\boldsymbol{x}$ to these subboxes. Note that we do not store the estimated Lagrange multipliers because we do not want to increase the required memory for the IBB. We want to point out that in most cases the box $\boldsymbol{x}$ cannot be reduced or discarded.

4.2.2. Lagrange estimator method

In the Lagrange estimator method, we solve (4.3) using the ILUD method. In this case, we do not need initial bounds on the Lagrange multipliers. We solve it if the system is independent, but not underdetermined. We want to use these bounds to decide whether or not to discard the examined box. We can use the ILUD method only if the system is squared. In an overdetermined case, we solve the system using only the first $1+|B|+|C|$ equations. First, we check if there are upper bounds less than 0. If so, the box is discarded because some estimated Lagrange multipliers are negative. If we have more equations than variables, after the first check, we solve the remaining equations with the obtained $\boldsymbol{\mu}$. If the Lagrange multipliers do not satisfy all the equations, we discard the examined box.

4.2.3. Lagrange estimator + NIFJ-CS method

In this method, we first estimate the Lagrange multipliers using the method in Section 4.2.2. We might discard the box by the Lagrange estimator method. If we cannot discard the box, the estimated bounds on the Lagrange multipliers are truncated with $[0,1]$. We solve the system of equations with the calculated multipliers as described in Section 4.2.1. As a result, we can discard the box $\boldsymbol{x}$ or reduce it by the Newton method.

4.2.4. Taylor expansion of the NIFJ-CS

This method is very similar to the naive NIFJ-CS method in Section 4.2.1. However, in this case, we replace some intervals in the Jacobian matrix with their midpoints. We use the Jacobian of $\boldsymbol{\phi}(\boldsymbol{x})$ as $\mathbf{J}_{ij}(\boldsymbol{t}) = \frac{\boldsymbol{\partial}}{\boldsymbol{\partial t}_j}\boldsymbol{\phi}_i(\boldsymbol{t}_1,\ldots,\boldsymbol{t}_j,t_{j+1},\ldots,t_N)$ $i,j=1,\ldots,N$, where the $N-j$ components are real numbers. This reduction is valid, since the Taylor expansion of $\boldsymbol{\phi}(\boldsymbol{t})$ is done for each variable one by one. It might produce a tighter inclusion; however, the computational cost of the Jacobian is much higher. The interested reader can find more information on the background of this method in [3].

4.3. Additional improvements

As a further development, we examine the success of the methods based on the order of the equations and the variables. Ordering the variables does not have a huge effect on the solvability of the NIFJ-CS. It only increases the required computation time. Sorting the equations in descending order by

$$o_i = \frac{\overline{\mathbf{g}_i(\boldsymbol{x})}}{\overline{\mathbf{g}_i(\boldsymbol{x})} - \underline{\mathbf{g}_i(\boldsymbol{x})}},$$

we can improve the success rate of the methods described in Sections 4.2.1 and 4.2.3 without significantly increasing the computation time.

5. Computational experiments

In this section, we compare the four methods together with the IBB method without NIFJ-CS. We implemented the IBB method with the following discarding tests: Newton, midpoint, cutoff, concavity, monotonicity, feasibility, and FJ test.

In detail, we use the bisection method as the division step. The termination criterion for any box $\boldsymbol{x}$ is $\text{wid}(\boldsymbol{x}) < \varepsilon$ or $\text{wid}(\mathbf{f}(\boldsymbol{x})) < \varepsilon$, where $\varepsilon = 10^{-6}$. We sort the working list in ascending order by the lower bound and delete any box whose lower bound is greater than the current upper bound (cutoff test). We select the first element from the working list. We stop the algorithm when the working list is empty or when we reach the maximum running time (7 200 seconds).

We implement the IBB method in Matlab R2020a version 8 [4] and use Intlab 11 [11]. From Intlab, we used only the IA, AD, and ILUD methods. We implement other tests, methods, and functions. The complete project can be found on GitLab[1]. The abbreviations of the five methods can be seen in Table 1.

Table 1. Abbreviations of the methods.

IBBWO	IBB without FJ optimality conditions
NFJ	IBB with the naive NIFJ-CS method (see Section 4.2.1)
Lag	IBB with the Lagrange estimator method (see Section 4.2.2)
Lag + NFJ	IBB with the Lagrange estimator + NIFJ-CS (see Section 4.2.3)
Tay + NFJ	IBB with Taylor expansion of the NIFJ-CS (see Section 4.2.4)

[1]The IBB with Optimality condition - Intlab: https://gitlab.com/gencsimiska27/the-ibb-with-optimality-condition-intlab

Table 2. Computation time.

Test name	IBBWO	NFJ	Lag	Lag + NFJ	Tay + NFJ
circle	21.4	30.6	13.7	**12.7**	61.1
ex14_1_1	755.5	4 321.1	774.2	**750.5**	5 565.4
ex14_1_4	266.2	764.3	218.9	**201.5**	836.7
ex14_1_8	**41.9**	84.6	51.1	48.8	181.7
ex2_1_2	496.3	597.9	**55**	58.3	4 331.3
ex2_1_4	🕓	🕓	🕓	🕓	🕓
ex2_1_6	**346.9**	1 762.7	361.8	355.7	2 463.7
ex3_1_2	680.8	1 543.9	87.4	**80.5**	2 240
ex3_1_3	🕓	🕓	225.4	**201.8**	🕓
ex3_1_4	620.3	563.4	181.8	**142.5**	374
ex4_1_9	🕓	6 843.4	5 032.5	**4 501.1**	5 836.1
ex7_2_3	🕓	🕓	🕓	🕓	🕓
ex7_2_4	🕓	🕓	🕓	🕓	🕓
ex7_3_2	62	133.0	26.3	**14.3**	296.2
Gomez Levy	**3.2**	3.8	3.9	4.5	8.8
Mishra's Bird	0.7	0.9	0.6	**0.6**	0.7
Rbrock (disk)	**0.8**	1.5	0.9	0.9	1.7
Rbrock (cube)	**0.7**	1.4	0.9	1	2.4
Simionescu	64.4	10.2	**4.2**	5	132.8
Hansen Test	**0.5**	1.7	0.8	0.9	4.3
Avg. Comp. Time	1 968.2	2 273.4	1 432	**1 399.1**	2 556.9

We used two benchmarks to compare the methods. The first benchmark is the GLOBALLib[2], from which we chose 14 constrained nonlinear programming test cases having only inequality constraints. The second benchmark is called Test Functions for Constrained Optimization from the Wikipedia website[3], which contains five two-dimensional test cases. In some cases, general bound constraints were not given. Thus, we used a large interval, $[-10\,000, 10\,000]$, which encloses the optimum points in these cases. All runs were performed on an AMD Ryzen 7 3800X 8-Core Processor with 32 GB RAM. In addition, we use an additional test case from [2], named the Hansen test.

The running time for each test case and method can be seen in Table 2. The symbol 🕓 means that we cannot reach the required accuracy in 7 200 seconds, but we still have possible solutions in the result list. We can see that we can reduce the

[2]GLOBALLib: `http://www.gamsworld.org/global/globallib.htm`

[3]Test Functions for Constrained Optimization: `https://en.wikipedia.org/wiki/Test_functions_for_optimization`

required computation time for the IBB methods by using the optimality conditions in most cases. Only a few small problems cannot be improved with them. The NFJ method takes equal to or more time to compute than the other methods. It is because we try to solve an interval-valued system of equations with overestimated variables (initial bounds for Lagrange multipliers, overestimated enclosures of the gradients) and in many cases we are not able to reduce or discard the studied box. The calculation time of the Tay + NFJ method is sometimes long but almost always better than NFJ due to the sharper gradient enclosures. However, it takes more time to calculate the gradient with different input boxes.

In average time, the best method is the Lag + NFJ method, which takes 1 399.1 seconds on average for 20 test cases, but the Lag method comes very close. The small difference is due to the fact that the NIFJ-CS method is solved with estimated bounds on the Lagrange multipliers.

Table 3. Relative deviations of the Weighted Function Evaluations from the best results (in bold).

Test name	IBBWO	NFJ	Lag	Lag + NFJ	Tay + NFJ
circle	142%	384%	**26 464**	327%	2 185%
ex14_1_1	**1 057 048**	784%	100%	101%	981%
ex14_1_4	**165 154**	618%	132%	194%	1 036%
ex14_1_8	**37 518**	693%	121%	180%	761%
ex2_1_2	1 807%	4 282%	**19 719**	118%	10 127%
ex2_1_4	**2 519 349**	150%	124%	122%	239%
ex2_1_6	**116 566**	2 038%	100%	101%	2 809%
ex3_1_2	728%	3 993%	**69 215**	185%	7 905%
ex3_1_3	525%	2885%	**164 486**	155%	3 299%
ex3_1_4	482%	473%	**153 782**	141%	260%
ex4_1_9	171%	219%	**4 719 013**	126%	114%
ex7_2_3	**3 835 776**	536%	122%	130%	695%
ex7_2_4	**1 311 026**	185%	144%	153%	651%
ex7_3_2	209%	1 613%	**51 862**	273%	3 552%
Gomez Levy	**1 416**	374%	110%	116%	216%
Mishra's Bird	179%	138%	102%	103%	120%
Rbrock (disk)	**613**	497%	117%	122%	210%
Rbrock (cube)	**888**	336%	129%	158%	360%
Simionescu	1 270%	140%	**2 826**	107%	2274%
Hansen Test	**1 276**	161%	113%	141%	660%
Avg. WFE	121%	361%	**817 505**	113%	477%

In Table 3, we compare the weighted function evaluations (WFE), computed as WFE $=$ Feval $+ n \cdot$ Geval $+ \frac{n^2}{2} \cdot$ Heval, where n is the dimension of the problem, Feval, Geval, Heval are the number of function, gradient and Hessian evaluations, respectively. We report the best results in bold, and the relative deviation for the rest. One can see that, on average, the Lag method requires the least number of function evaluations. The evaluation for IBBWO, Lag, and Lag + NFJ is very close, and these three methods solve the problems with the least function evaluations. The other two methods require more evaluations, because many times solving NIFJ-CS is unsuccessful.

The relative deviations of the averages for the methods can be seen in Table 4. We compare the five methods in five aspects: computation time, weighted number of function evaluations (WFE), number of result boxes, number of iterations, and number of NIFJ-CS tests, respectively. Furthermore, we study the success rate of IBB methods that include NIFJ-CS. We can see that the Lag + NFJ method is the best in terms of computation time (1 399.1 s), number of results boxes (4), and FJ test (15 551). The success rate of this method is 49%, which is very high compared to the rest. IBBWO is worst in all aspects compared to the best results. The NFJ method is sometimes worse than the IBBWO method, which is caused by the high number of unsuccessful NIFJ-CS. The Lag method requires the least function evaluation because we do not solve the NIFJ-CS. The Tay + NFJ method requires the least number of iterations, but the number of function evaluations is significantly higher, increasing its computation time.

Table 4. The relative average deviations.

Method	IBBWO	NFJ	Lag	Lag + NFJ	Tay + NFJ
Computation time	141%	162%	102%	**1 399.1**	183%
WFE	121%	361%	**817 505**	113%	477%
No. result boxes	**4**	**4**	**4**	**4**	**4**
No. iterations	238%	320%	243%	239%	**21 340**
No. FJ Test	-	157%	104%	**15 551**	143%
FJ Success Percentage	-	7%	38%	**49%**	25%

6. Summary

We studied the applicability of optimality conditions in Interval Branch and Bound method to constrained problems. We introduced the NIFJ-CS, which is based on the Fritz-John optimality conditions. We found that the naive NIFJ-CS is difficult to solve. This is due to the overestimation of the Lagrange multipliers and the enclosure of the gradients. We studied four versions (NFJ, Lag, Lag + NFJ, Tay + NFJ) for solving the NIFJ-CS, and compared their efficiency together with the IBB method without optimality conditions in 20 test cases from the literature. We found that the best method for solving NIFJ-CS is the Lag + NFJ method. The average time required for this method is 1 399.1 seconds, and the success rate is 49%.

In the future, we want to improve the success rate of the Lag + NFJ method by introducing a preliminary test before trying to solve NIFJ-CS. The preliminary test should discard the box, which certainly does not contain an optimal point, within a short computation time. In addition, we plan to use constraint propagation to reduce the box as much as possible. We also want to extend the methods to problems with equality constraints.

References

[1] A. Goldsztejn, G. Chabert: *A Generalized Interval LU Decomposition for the Solution of Interval Linear Systems*, in: Aug. 2006, pp. 312–319, isbn: 978-3-540-70940-4, doi: 10.1007/978-3-540-70942-8_37.

[2] E. Hansen, G. Walster: *Bounds for Lagrange multipliers and optimal points*, Computers & Mathematics with Applications 25.10 (1993), pp. 59–69, issn: 0898-1221, doi: 10.1016/0898-1221(93)90282-Z.

[3] E. Hansen, G. W. Walster: *Global Optimization Using Interval Analysis: Revised And Expanded*, CRC Press, 2003, p. 728, isbn: 0824740599, doi: 10.1201/9780203026922.

[4] T. M. Inc.: *MATLAB version: 9.13.0 (R2022a)*, Natick, Massachusetts, United States, 2022, url: https://www.mathworks.com.

[5] L. Jaulin, M. Kieffer, O. Didrit, E. Walter: *Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics*, Aug. 2001, isbn: 1852332190, doi: 10.1108/k.2002.06731eae.002.

[6] R. B. Kearfott: *An Interval Branch and Bound Algorithm for Bound Constrained Optimization Problems*, Journal of Global Optimization 2 (1992), pp. 259–280, doi: 10.1007/BF00171829.

[7] R. B. Kearfott: *Preconditioners for the Interval Gauss–Seidel Method*, SIAM Journal on Numerical Analysis 27.3 (1990), pp. 804–822, doi: 10.1137/0727047.

[8] R. Kearfott, M. Nakao, A. Neumaier, S. Rump, S. Shary, P. Van Hentenryck: *Standardized notation in interval analysis*, Vychislitel'nye Tekhnologii 15 (Jan. 2010).

[9] O. Mangasarian, S. Fromovitz: *The Fritz John necessary optimality conditions in the presence of equality and inequality constraints*, Journal of Mathematical Analysis and Applications 17.1 (1967), pp. 37–47, doi: 10.1016/0022-247X(67)90163-1.

[10] L. Pál, T. Csendes: *INTLAB implementation of an interval global optimization algorithm*, Optimization Methods and Software 24.4-5 (2009), pp. 749–759, doi: 10.1080/10556780902753395.

[11] S. Rump: *INTLAB - INTerval LABoratory*, in: Developments in Reliable Computing, ed. by T. Csendes, http://www.tuhh.de/ti3/rump/, Dordrecht: Kluwer Academic Publishers, 1999, pp. 77–104.

[12] H. J. S., A. Griewank, G. F. Corliss: *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, Mathematics of Computation 62.205 (Jan. 1994), p. 434, doi: 10.2307/2153424.

[13] S. P. Shary: *Interval Gauss-Seidel Method for Generalized Solution Sets to Interval Linear Systems*, Reliable Computing 7.2 (Apr. 2001), pp. 141–155, issn: 1573-1340, doi: 10.1023/A:1011422215157.

Annales Mathematicae et Informaticae
58 (2023) pp. 69–80
DOI: 10.33039/ami.2023.08.009
URL: https://ami.uni-eszterhazy.hu

A pseudonymization tool for Hungarian

Péter Hatvani[a b], **László János Laki**[b], **Zijian Győző Yang**[b]

[a]Pázmány Péter Catholic University Faculty of Humanities and Social Sciences, Doctoral School of Linguistics
hatvani.peter@hallgato.ppke.hu

[b]Hungarian Research Centre for Linguistics
{yang.zijian.gyozo,laki.laszlo}@nytud.hu

Abstract. In today's world, the volume of documents being generated is growing exponentially, making the protection of personal data an increasingly crucial task. Anonymization plays a vital role in various fields, but its implementation can be challenging. While advancements in natural language processing research have resulted in more accurate named entity recognition (NER) models, relying on an NER system to remove names from a text may compromise its fluency and coherence. In this paper, we introduce a novel approach to pseudonymization, specifically tailored for the Hungarian language, which addresses the challenges associated with maintaining text fluency and coherence. Our method employs a pipeline that integrates various NER models, morphological parsing, and generation modules. Instead of merely recognizing and removing named entities, as in conventional approaches, our pipeline utilizes a morphological generator to consistently replace names with alternative names throughout the document. This process ensures the preservation of both text coherence and anonymity. To assess the efficacy of our method, we conducted evaluations on multiple corpora, with results consistently indicating that our pipeline surpasses traditional approaches in performance. Our innovative approach paves the way for new pseudonymization possibilities across a diverse range of fields and applications.

Keywords: Pseudonymization, Named entity recognition (NER), Morphological generation

AMS Subject Classification: 68T50, 68T07

Submitted: July 31, 2023
Accepted: August 7, 2023
Published online: August 19, 2023

1. Introduction

The GDPR (General Data Protection Act) [3] of the European Union enforces stricter than ever rules on handling personal information and information that can be traced back to the subject. Luckily, data can not only be stored in a completely anonymised way but also in a pseudoanonymised way, still, as per the definition of the law:

> 'pseudonymization' means the processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information, provided that such additional information is kept separately and is subject to technical and organisational measures to ensure that the personal data are not attributed to an identified or identifiable natural person;

The most important aspect of the pseudoanonymisation is that after the process direct identification can not remain in the text. As for indirect identification - location, title, martial status - these information can remain and will remain in the text in our implementation.

The paper introduces a pseudononymization tool for Hungarian that integrates different named entity recognition, morphological parsing and generation modules. Instead of simply recognizing and removing named entities, the tool replaces found names with other names consistently throughout a given document. The tool uses huSpacy [14] and emMorph [11], a Hungarian morphological analyzer, to ensure that the results are consistent in several testing corpora. The tool is designed to be used for various use cases such as legal, medical documents and other types of sensitive texts. However, the testing of the model has been limited to crawled comments, excerpt from programmes of the Hungarian Kossuth Radio, excerpts from NerKor [17] from the news and the wikipedia parts and Hungarian literature, which serves as a proof of concept for the ability of the tool to maintain coherence and fluency in the anonymized text.

Our pseudononymization pipeline tool is freely available in our github site[1] with Apache 2.0 license.

2. Related works

Anonymization is an important task in many fields, with healthcare being one of the main areas where it is essential. In medical documents, anonymization methods need to be applied to protect patients' privacy [1, 2, 15, 16]. For Hungarian, Kinga Mátyus [7] conducted rule-based anonymization in a sociolinguistic research.

Various approaches have been proposed for anonymizing medical data. In one study [1], an enhanced method utilizing asymmetric encryption was suggested to separate the duties of pseudonymization and de-pseudonymization. This method

[1] https://github.com/nytud/pseudo-anonimization

proposed a secure and controlled process that allows authorized third parties (ombudsmen) to de-pseudonymize patients when necessary, thereby bridging the gap between bench and bedside in translational research while preserving patient privacy.

Another study [16] focused on the use of pseudonymization for retrospective research, quality assurance, and education. By replacing all person-related data within a data record with an artificial identifier, pseudonymization allows for the linking of medical data and patient identification data under specific, predefined, and controllable conditions. Consequently, medical data can be shared with third parties without enabling them to identify the individual patients.

A third study [15] introduced a system called PIPE (pseudonymization of information for privacy in e-health), which securely integrates primary and secondary usage of health data, addressing the shortcomings of existing approaches. PIPE can serve as a foundation for implementing secure electronic health record (EHR) architectures or as an extension to current systems, effectively preventing health data misuse while enhancing communication between healthcare providers and, in turn, improving patient care quality and reducing costs.

Anonymisation based on NER model and morphological tools are rare. For Hungarian, HuSpacy [14] is a spaCy library providing industrial-strength Hungarian language processing facilities. The huSpacy pipeline contains a tokenizer, a sentence splitter, a lemmatizer, a morphological tagger, a dependency parser and a named entity recognition module. The morphological analyzer achieved approximately 90% accuracy in token classification tasks, and the NER module achieved 80.75 F1 score on NerKor [17]. From the heaps of morphological analyzers for Hungarian we have chosen to use huSpacy and emMorph (eMagyar[2]). The emMorph [11] is an Hungarian morphological analyzer that uses Humor [10] unification morphology. The advantage of the spaCy tool is that it is fast and it can produce results without GPU acceleration as well as with it. Three different models are present of HuSpacy, two of them are based on the huBERT [9] model. One of the best Hungarian morphological annotation tool is the PurePos 2.0 [13] that uses emMorph morphology analyzer. The PurePos achieved 96.72% accuracy on part-of-speech recognition tasks. Also emBERT [8] is a framework that was used with a multlimodal BERT model to finetune NER and noun phase recognition (NP) models. On NerKor, the emBERT could achieve 92.09 F1 score, due to time constraints emBERT fell out of this round of comparison, but it showed promise being a freely available NER model that finetuned by [22] on NerKor and gained a 90.18 F1 score.

There are some morphological analyzers for Hungarian that can be used for generation as well. HunSpell can be used for this task in two ways: it generates word forms by typing the lemma and the features, or typing the lemma and an example word. Hunmorph [19] and Morphdb.hu [20] are also suitable for morphological generation. Hunmorph-foma[3] uses the morphological tagset of HunMorph and it is

[2]`https://e-magyar.hu`

[3]`https://github.com/r0ller/hunmorph-foma`

based on the foma generator [4]. The main problem of these tools that they are not freely available or use a different tagset than emMorph or Universal Dependencies (UD).

In our research, we used two neural morphological generators [6] (an emMorph and an UD model) that was trained by the Hungarian Research Centre for Linguistics.

3. Corpora

Names corpus: The names were gathered from the list of names that can be registered as given names [12]. This list is updated monthly and the local version of the corpus was cached in October of 2022. The corpus itself is divided into two parts, male and female given names. This distinction is important, because certain gender implying words, such as father, mother etc. in the sentence can be connected to the swapped name and in many cases the gender of the word can disambiguate the context for the conversation.

Family names: Unfortunately no ready made corpora are available of Hungarian family names, Only the most used one hundred [5] second names are published. As a fall back if an unknown family name is present the replacing algorithm was tasked to replace the unknown name with a male name rather than guessing a family name.

(1) a. ***Maga***
FORMAL PERSONAL PRONOUN THIRD PERSON SINGULAR

‘Oneself’

b. ***Fodor Kriszta***
NAME THAT CAN BE INTERPRETED AS TWO GIVEN NAMES

c. ***Budai Krisz***
NICKNAME

d. ***Pálma Veresné Berta***
OUT OF ORDER NAME

e. ***Bihar Megyei Tanács***
NAME OF AN INSTITUTION

f. ***BRÜLL ADÉLNAK***
ALL CAPITAL NAME

g. ***Aranycsapat***
NAME OF A COLLECTIVE

Evaluation corpora: Three texts were chosen for evaluating the pipeline. All of them are contemporary variations of the language and are in line with the use cases connected to journalism and the media:

- reporting on social media: “comments”: scraped comments from a social media post, especially hard text for its noisy nature
- reporting on legal documents “Ady letters”: letters from Endre Ady, selected for the usage of capitalized pronouns (1a) and full capital names (1f)
- official reporting “spok”: excerpt from MNSZ [21]
- reporting on historical documents “huwiki”: mixed sentences from the Hungarian Wikipedia
- reporting on not spoken news “newscrawl”: crawled news from reputable journalists collected in NerKor [17]

These corpora contained many opportunities for the models to fail: capitalized personal pronouns (1a), reverse ordered names (1d), nicknames (1c), names that have given names as first name (1b), all capitalized names (1f) names of collective (1g), place and institution names (1e).

4. Pipeline architecture and modules

4.1. Modules

In our pipeline we have five main modules:

1. **Preprocessing:** For preprocessing, HuSpaCy and emMorph was used. The input text is splitted into sentences and tokenized.
2. **Morphological analyzer**: HuSpaCy and emMorph modules were integrated. Morphological analysis can be performed with either HuSpaCy in UD format or EmMorph in emMorph code format.
3. **NER:** For named entity recognition, a fine-tuned huBERT model [22] was used. The model was tested on NerKor that achieved 90.04 precision, 91.17 recall and 90.60 F1 scores.
4. **Name databases:** The names were collected from the official list of Hungarian surnames that are recognised [12] and the first one hundred most used family names [5].
5. **Morphology genarator:** Two neural-based Hungarian morphology generators were used that provided by the Hungarian Research Centre for Linguistics [6]. One emMorph and one UD morphology generator. The generator models were trained with Marian neural machine translation system. The eMagyar model could gain more than 96% accuracy and the UD model could achieve more than 94% accuracy.

4.2. Architecture

The pipeline serves as an integration of various modules, which are introduced in the Modules section. This section provides an overview of the pipeline architecture. As illustrated in Figure 1, a module may be utilized multiple times throughout the process. For example, the morphological analyzers are employed for both tokenization of the raw text and subsequent morphological analysis.

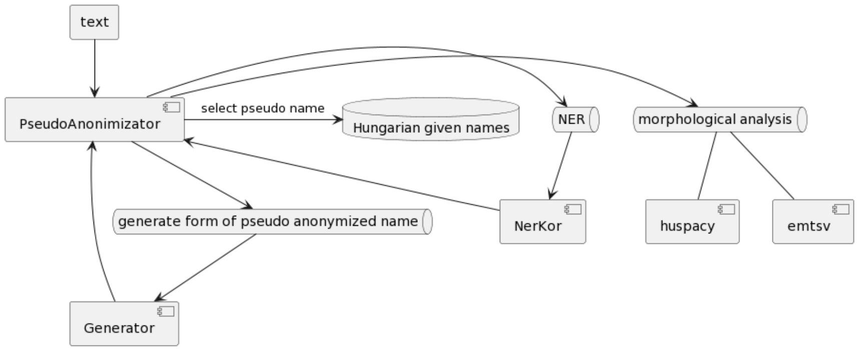


Figure 1. Structure diagram of the pseudononymization Tool.

As illustrated in Figure 2, the anonymization process is inherently sequential and challenging to parallelize. The names must first be identified, followed by selecting suitable replacement names, and finally performing the name swapping. This swapping process introduces a conundrum: not only is the identification of names potentially inaccurate, but the original and new names may also have different lengths, requiring the delta of lengths to be stored alongside with the mapping of the name to the pseudonym which will be mapped in the text.

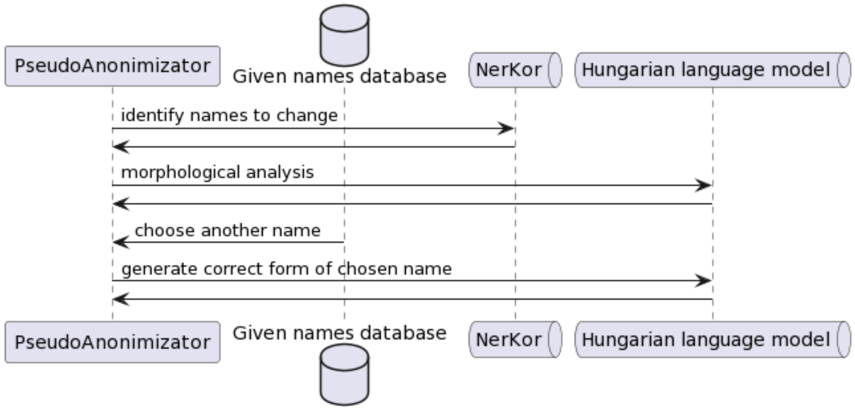


Figure 2. Activity diagram of the pseudononymization Tool.

Furthermore, neural models typically perform best when processing smaller text portions, similar to the complete sentences they were trained on. Consequently, the input text is first divided into sentences, which are then processed in a paginated manner using the NER system to identify potential named entities and their positions.

5. Results and evaluation

In our first experiment, we evaluated the performance of our NER model on person names. Using NerKor corpus, the NER model achieved 96.25 F1 score on person names (words tagged with [PER]).

In our second experiment, we evaluated the performance of eMagyar and UD morphology generators on named entities using the NerKor corpus. We collected the morphologically analyzed tokens from NerKor, filtered and extracted the unique names, and applied morphology generators to these names. The results of our evaluation are presented in Table 1. The 'all' column shows the performance of the models on all names, but this result may be biased as nearly 86% (eMagyar: 85.62%; UD: 86.07%) of names are in nominative case. Therefore, we also evaluated the performance of the models on non-nominative cases, which is shown in the 'filtered' column.

Table 1. Performance of morphology generators on named entities.

	all	filtered
emMorph	95.10%	84.51%
UD	92.85%	79.39%

For further evaluation we have established an ideal scenario, when all names are successfully replaced and no false positives are present, and a worst case, when no names are recognized and every other word gets replaced but the names. These scenarios are significantly distinct from either the eMagyar or the huSpacy morphological analyzers when compared with Student's t-test [18]. The evaluation metrics used are as follows:

1. True Positive: The number of actual named entities (real names) found by the pipeline.
2. False Positive: The number of incorrect named entities identified by the pipeline that are not real names.
3. False Negative: The number of real named entities that were not identified by the pipeline.
4. All: The total number of named entities (both correctly identified and incorrectly identified) in the text.

The p-values indicate the probability of obtaining the observed results if there is no significant difference between the pipelines. The hypothesis being tested is whether there is a significant difference in performance between the two pipelines. In the spok corpus with the eMagyar analyzer the distinction from the worst ($p = 0.0001$) and the ideal ($p = 0.0029$) are hairly distinct compared to the huSpacy analyzed which is not significantly distinct ($p = 0.5933$). The same can be observed with the analysis of the two other corpora. However, the similarity of the same pipeline on different texts is not as close as one might think eMagyar "spok" and "comments" are not close ($p = 0.0001$), only the Ady letter text and the spok corpus with the huSpacy pipeline was close ($p = 0.0454$), but they still differed significantly.

The pipelines yielded results with many false positives, such as reporting a name in the middle of a word or finding the name in two separate parts, but in such a way that the two findings are next to each other without even a character of difference. In such cases, as a remedy, a unification algorithm was used, which extended the first hit and deleted the second. Even with this measure, the false positive percentage of all found names is high, especially in the "comments" corpus (Table 2).

Table 2. Performance of pseudo anonymization or eMagyar morphological analyzer.

text	True Positive	False Positive	False Negative	All
spok	7	8	2	13
comments	42	25	9	67
Ady letters	11	4	9	20
huwiki	85	2	0	87
newscrawl	103	2	0	105

Table 3. Performance of pseudo anonymization or huSpacy morphological analyzer.

text	True Positive	False Positive	False Negative	All
spok	6	5	5	11
comments	58	19	3	77
Ady letters	6	5	9	15
huwiki	85	0	4	86
newscrawl	103	2	1	128

When analyzing the data from the huwiki and newscrawl corpora, both are collected from NerKor [17], it is evident that the false positive rates for these datasets are generally lower than those observed in the "comments" corpus. For instance, the eMagyar pipeline had only 2 false positives in both huwiki and newscrawl corpora, while the huSpacy pipeline showed no false positives in the huwiki corpus and only 2 in the newscrawl corpus (Tables 2 and 3). This suggests that the performance of both pipelines in terms of false positives may vary depending on the specific corpus being analyzed, and further research or fine-tuning may be required to optimize the pipelines for each individual corpus.

Table 4. Performance of NerKor with different tokenization pipelines.

corpus	pipeline	All Positives (True + False)	All real names	All words
spok	emagyar	24	10	735
spok	HuSpacy	24	10	735
comments	emagyar	128	67	876
comments	huspacy	126	67	876
Ady letters	emagyar	37	20	1369
Ady letters	huspacy	32	20	1369
huwiki	emagyar	85	87	2785
huwiki	huspacy	85	86	2785
newscrawl	emagyar	103	105	4551
newscrawl	huspacy	103	128	4551

Table 4 shows the performance of NerKor with two tokenization pipelines on different datasets. In general, both pipelines perform similarly, identifying named entities in the corpora. However, in the "newscrawl" dataset, the "Huspacy" pipeline outperforms "emagyar" in identifying named entities (128 vs. 103). The total word count remains consistent across all corpora and tokenization pipelines.

(2) a.

751	*SCHÖPFLIN*	***ALADÁRNAK***
751	SCHÖPFLIN	**to Aladar**
751	Schöpflin	**Ilájnak**
751	Schöpflin	**to Ilja**

b.

Édes,	*Drága,*	*áldott*	***Adélom,***
Sweet,	Precious,	blessed	**my Adel**,
Édes,	Drága,	áldott	**Darlám**,
Sweet,	Precious,	blessed	**my Darla**,

c.

ima	*által*	*kapcsolatba*	*lépünk*	***Istennel,***
prayer	through	get	connected	**with God**,
ima	által	kapcsolatba	lépünk	**Gyárfással**,
prayer	through	get	connected	**with Gyárfás**,

In Example 2 the proper form generation can be observed in action. These examples are from the 'Ady letters' corpus and they were generated from form tokenized and analyzed by the eMagyar pipeline. As you can see the case stayed the same and even the sound assimilation was generated properly. The last example showcases a certain error that poses a great challenge whereas the NER system identifies collectives or supernatural beings as persons and treats them with great respect regarding their privacy.

> '**Benjámin** Somogyi'
> '**BENJÁMIN** SOMOGYI'
> 'Magatokon lehetne a legtöbbet spórolni, de arról hallani sem akartok! Az egyes élelmiszer hatósági árát, a kereskedőknek kell köszönni, nem nektek!,'
> 'IT WOULD BE POSSIBLE TO SAVE THE MOST ON YOURSELVES, BUT YOU DON'T EVEN WANT TO HEAR ABOUT IT! THE REGULATORY PRICE OF INDIVIDUAL FOOD PRODUCTS SHOULD BE CREDITED TO THE TRADERS, NOT YOU!'
> 'Hegedűsné Krizsák **Barbara**',
> 'MRS. HEGEDŰSNÉ KRIZSÁK **BARBARA**',
> '**Benjámin** Somogyi így igaz.',
> '**BENJÁMIN** SOMOGYI THAT'S TRUE.',
> 'Száváné **nagy Balzsam**',
> 'SZÁVÁNÉ **NAGY BALZSAM**',
> '**Benjámin** Somogyi A kereskedők nem maguktól találták ezt ki. Majd hülyék lennének maguk ellen dolgozni.'
> '**BENJÁMIN** SOMOGYI THE TRADERS DIDN'T COME UP WITH THIS ON THEIR OWN. THEY WOULD BE FOOLS TO WORK AGAINST THEMSELVES.'

The previous quote demonstrates the pipeline in action on the comments corpus. One of the main features is the consistency of names. All the given names were typeset bold for the quote to show how the same person, who should be known as Benjámin according to the pipeline and wears the family name of Somogyi, is consistently swapped to be Benjámin. However, an error is also present in this presentation. The family name "Nagy" is mistakenly replaced with all lower case and part of the tagging is also present for now, as this error is under investigation.

6. Conclusion

We have presented a pseudononymization pipeline with a command line interface and a web service, tailored for the Hungarian language. All the tools used in the pipeline, as well as the pipeline itself, are freely available and open source. Instructions for local deployment can be found in the git repository, ensuring easy reproducibility. Our approach achieved impressive results as the first of its kind, extending beyond traditional use cases in medicine and legal documents. We believe that anonymization is equally important in media and business contexts, where noise is more prevalent than in the aforementioned fields. The testing corpora posed significant challenges, yet our models were able to effectively handle them, demonstrating the potential for broader applications of our pseudononymization approach.

7. Further work

In the future we want to expand this tool with the ability to create a database with which the pseudonymized text can be deanonymized with ease, this should be provided with a switch to the command line interface and a separate endpoint for the web server. Additionally more models could be incorporated to expand the possibility to find the best model for this task.

References

[1] H. Aamot, C. D. Kohl, D. Richter, P. Knaup-Gregori: *Pseudonymization of patient identifiers for translational research*, BMC Medical Informatics and Decision Making 13 (2013), pp. 1472–6947.

[2] H. Dalianis: *Pseudonymisation of Swedish Electronic Patient Records Using a Rule-Based Approach*, in: Proceedings of the Workshop on NLP and Pseudonymisation, Turku, Finland: Linköping Electronic Press, Sept. 2019, pp. 16–23, url: https://aclanthology.org/W19-6503.

[3] European Commission: *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance)*, 2016, url: https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[4] M. Hulden: *Foma: a Finite-State Compiler and Library*, in: Proceedings of the Demonstrations Session at EACL 2009, Athens, Greece: Association for Computational Linguistics, Apr. 2009, pp. 29–32, url: https://aclanthology.org/E09-2008.

[5] M. of Interior Deputy State Secretariat for Data Registers: *Most common 100 family names*, data retrieved from Ministry of Interior Deputy State Secretariat for Data Registers, https://www.nyilvantarto.hu/letoltes/statisztikak/kozerdeku_csaladnev_2022.xlsx, 2022.

[6] L. J. Laki, N. Ligeti-Nagy, N. Vadász, Z. Gy. Yang: *Neural Morphological Generators for Hungarian*, in: XIX. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2023), Szeged, Hungary: Szegedi Tudományegyetem, Informatikai Intézet, 2023, pp. 331–340.

[7] K. Mátyus: *Anonimizálási gyakorlat?*, in: IX. Magyar Számítógépes Nyelvészeti Konferencia, Szeged: Szegedi Tudományegyetem, 2013, pp. 338–342.

[8] D. M. Nemeskey: *Egy emBERT próbáló feladat*, in: XVI. Magyar Számítógépes Nyelvészeti Konferencia, Szeged: Szegedi Tudományegyetem, 2020, pp. 409–418.

[9] D. M. Nemeskey: *Introducing huBERT*, in: XVII. Magyar Számítógépes Nyelvészeti Konferencia, Szeged, Magyarország: Szegedi Tudományegyetem, Informatikai Intézet, 2021, pp. 3–14.

[10] A. Novák: *A New Form of Humor – Mapping Constraint-Based Computational Morphologies to a Finite-State Representation*, in: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), ed. by N. Calzolari, K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, S. Piperidis, Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, isbn: 978-2-9517408-8-4.

[11] A. Novák, B. Siklósi, Ch. Oravecz: *A New Integrated Open-source Morphological Analyzer for Hungarian*, in: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), ed. by N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis, Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, isbn: 978-2-9517408-9-1.

[12] N. K. Nyelvművelő és Nyelvi Tanácsadó Kutatócsoport: *Bejegyzésre alkalmasnak minősített utónevek jegyzéke*, 2022, url: http://www.nytud.hu/oszt/nyelvmuvelo/utonevek/index.html (visited on 10/25/2022).

[13] Gy. Orosz, A. Novák: *PurePos 2.0: a hybrid tool for morphological disambiguation*, in: Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013, Hissar, Bulgaria: INCOMA Ltd. Shoumen, BULGARIA, Sept. 2013, pp. 539–545, url: https://aclanthology.org/R13-1071.

[14] Gy. Orosz, Z. Szántó, P. Berkecz, G. Szabó, R. Farkas: *HuSpaCy: an industrial-strength Hungarian natural language processing toolkit*, Szeged, 2022.

[15] B. Riedl, V. Grascher, S. Fenz, T. Neubauer: *Pseudonymization for improving the Privacy in E-Health Applications*, in: Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008), 2008, pp. 255–255, doi: 10.1109/HICSS.2008.366.

[16] B. Schütze: *Use of medical treatment data outside of the patient supply: best way pseudonymisation*, Dtsch Med Wochenschr 137(16) (2012), pp. 844–850.

[17] E. Simon, N. Vadász: *Introducing NYTK-NerKor, A Gold Standard Hungarian Named Entity Annotated Corpus*, in: Text, Speech, and Dialogue - 24th International Conference, TSD 2021, Olomouc, Czech Republic, September 6-9, 2021, Proceedings, ed. by K. Ekstein, F. Pártl, M. Konopík, vol. 12848, Lecture Notes in Computer Science, Springer, 2021, pp. 222–234, doi: 10.1007/978-3-030-83527-9_19.

[18] Student: *The probable error of a mean*, Biometrika (1908), pp. 1–25.

[19] V. Trón, G. Gyepesi, P. Halácsy, A. Kornai, L. Németh, D. Varga: *Hunmorph: open source word analysis*, in: Proceedings of the ACL 2005 Software Workshop, ed. by M. Jansche, Ann Arbor: ACL, 2005, pp. 77–85.

[20] V. Trón, P. Halácsy, P. Rebrus, A. Rung, P. Vajda, E. Simon: *Morphdb.hu: Hungarian lexical database and morphological grammar*, in: Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06), Genoa, Italy: European Language Resources Association (ELRA), May 2006, url: http://www.lrec-conf.org/proceedings/lrec2006/pdf/683_pdf.pdf.

[21] T. Váradi: *The Hungarian National Corpus*, in: Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002), Las Palmas de Gran Canaria: European Language Resources Association, 2002, pp. 385–389.

[22] Z. Gy. Yang, T. Váradi: *Training language models with low resources: RoBERTa, BART and ELECTRA experimental models for Hungarian*, in: Proceedings of 12th IEEE International Conference on Cognitive Infocommunications (CogInfoCom 2021), Online: IEEE, 2021, pp. 279–285.

Annales Mathematicae et Informaticae
58 (2023) pp. 81–91
DOI: 10.33039/ami.2023.11.002
URL: https://ami.uni-eszterhazy.hu

CAPTCHA recognition using machine learning algorithms with various techniques

Ádám Kovács[ab], **Tibor Tajti**[a]

[a]Eszterházy Károly Catholic University
kovacs2.adam@uni-eszterhazy.hu
tajti.tibor@uni-eszterhazy.hu

[b]University of Debrecen, Doctoral School of Informatics

Abstract. In this paper, we present research results on the recognition of text-based CAPTCHA tests using advanced machine learning algorithms and techniques. Text-based CAPTCHAs serve as a crucial security measure to prevent automated access to various web services, but their effectiveness depends on their resistance to sophisticated recognition techniques. To this end, we focus on evaluating and enhancing the performance of recognition models using a Convolutional Neural Network (CNN) as the base model. We propose an integrated approach, which incorporates a systematic parameter optimization strategy using Grid Search Cross-Validation (Grid Search CV) and the Ensemble Voting Method to improve the performance of the recognition model. The use of Grid Search CV enables us to fine-tune the hyperparameters of the CNN model, leading to an optimal configuration. Further, we investigate the effectiveness of the Ensemble Voting Method to aggregate the predictions from multiple CNN models, each with a set of the optimal parameters obtained from the Grid Search CV. The methods' performance was evaluated through multiple learning sessions, assessing their effectiveness in recognizing text-based CAPTCHAs under various scenarios.

Keywords: Machine learning, CAPTCHA recognition, neural networks, hyperparameter optimization, ensemble methods

1. Introduction

CAPTCHA, or Completely Automated Public Turing Test to tell Computers and Humans Apart, is a widely used security measure designed to differentiate between

Submitted: August 31, 2023
Accepted: November 9, 2023
Published online: November 10, 2023

human and machine users [18]. However, with recent advancements in artificial intelligence, traditional CAPTCHAs are becoming increasingly susceptible to automated system bypassing.

Convolutional Neural Networks (CNNs) are a category of deep learning algorithms that are generally used for processing and analyzing visual data, such as images and videos. Their distinctive architecture leverages spatial hierarchies and local patterns within the data, enabling the automatic learning of complex and abstract features. While CNNs are highly applicable to a range of computer vision tasks, including image recognition, object detection, and segmentation, they can also be employed in other domains, such as time series prediction and speech recognition. The use of CNNs to recognize distorted characters has exposed the vulnerability of existing CAPTCHA systems, emphasizing the need for more sophisticated and resilient alternatives [7].

Grid Search Cross-Validation (Grid Search CV) is a hyperparameter optimization technique in machine learning models [9]. The use of Grid Search CV entails a comprehensive search across a defined range of hyperparameter values, with the performance of each combination assessed via cross-validation (CV). This approach aids in determining the optimal set of hyperparameters, resulting in superior model performance. Grid Search CV is crucial for developing robust models, as it ensures that they are fine-tuned and capable of generalizing effectively to unseen data.

Ensemble methods comprise a collection of powerful machine learning techniques that focus on integrating multiple models to achieve enhanced predictive performance compared to individual models. The core concept underlying ensemble methods are to exploit diversity among various models, which assists in reducing prediction errors, increasing stability, and bolstering generalization capabilities. By aggregating the predictions of several models, ensemble methods can counterbalance the limitations of individual models, ultimately yielding more accurate and robust predictions. Ensemble voting can be effective using learners with the same model [13], but also using various models for the voters [6].

1.1. Dataset

Figure 1 illustrates two text-based CAPTCHAs from the dataset, each subjected to different noise levels and distortions.

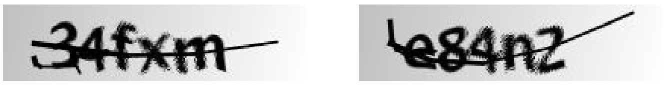


Figure 1. Random elements of the dataset.

These alterations serve to increase the CAPTCHAs' complexity for automated systems, consequently augmenting the security of the protected system. The presence of diverse distortions and noise levels in the dataset challenges the automated systems to adapt and recognize characters under varying conditions, thereby testing their robustness and reliability in solving CAPTCHAs [3].

The dataset consists of a total of 1,070 images, predominantly in the PNG format, with a few files in the JPG format [17]. Each of the images is in grayscale, featuring five alphanumeric characters that may include both letters and numbers. The dimensions of these images are 200 pixels in width and 50 pixels in height.

1.2. Model

Figure 2 shows the pre-existing model that we utilized for text-based CAPTCHA prediction, with an input layer for 50 × 200 grayscale images [10].

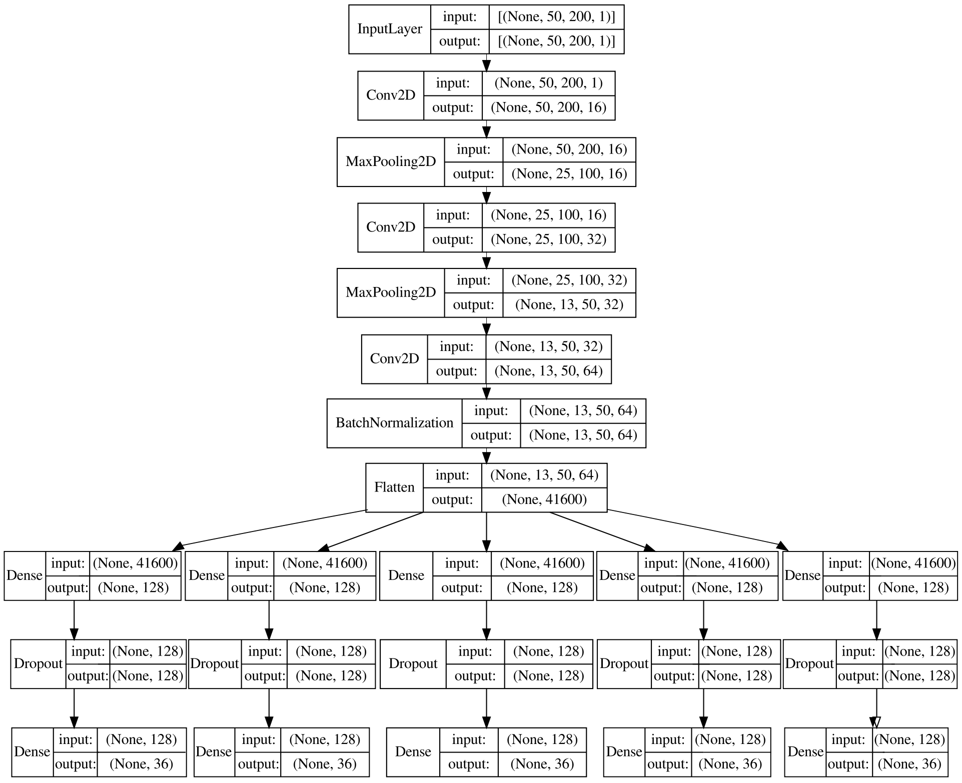

InputLayer input: [(None, 50, 200, 1)] output: [(None, 50, 200, 1)]
Conv2D input: (None, 50, 200, 1) output: (None, 50, 200, 16)
MaxPooling2D input: (None, 50, 200, 16) output: (None, 25, 100, 16)
Conv2D input: (None, 25, 100, 16) output: (None, 25, 100, 32)
MaxPooling2D input: (None, 25, 100, 32) output: (None, 13, 50, 32)
Conv2D input: (None, 13, 50, 32) output: (None, 13, 50, 64)
BatchNormalization input: (None, 13, 50, 64) output: (None, 13, 50, 64)
Flatten input: (None, 13, 50, 64) output: (None, 41600)
Dense input: (None, 41600) output: (None, 128)
Dropout input: (None, 128) output: (None, 128)
Dense input: (None, 128) output: (None, 36)

Figure 2. Representation of the layers.

It features three Conv2D layers, each followed by a MaxPooling2D layer for feature extraction, and a BatchNormalization layer for improved stability [16]. The output is flattened, and the model branches into five separate paths, each responsible for predicting one CAPTCHA character. Each branch consists of a Dense layer, a Dropout layer, and a final Dense layer with output neurons matching the number of possible characters. The activation function for the last Dense layer is the softmax function, which calculates probabilities for each possible character.

1.3. Voting method

One of the most prominent ensemble methods is the voting method [2, 5, 15]. In this approach, multiple model instances are trained on the same dataset. These trained models are then used to make predictions for new data points, and the final prediction is determined by aggregating the individual models' predictions using a voting scheme.

Various voting schemes can be employed in the voting method, such as:

- Plurality voting: The final prediction is the class (or value) that receives the most votes from the individual models.
- Fuzzy average voting: For classification tasks, the final prediction is determined by averaging the predicted class probabilities from each model and selecting the class with the highest average probability.

In practical applications, the voting method has been shown to be highly effective in a wide range of problems, such as image and speech recognition, natural language processing, and bioinformatics [1, 14]. Besides the Voting approach, other ensemble methods like Bagging and Boosting can also be applied to improve predictive performance [8].

2. Experiments and results

2.1. Performance evaluation framework

We conducted our evaluation utilizing AMD GPU in conjunction with the TensorFlow framework. Additionally, we employed the Keras library alongside TensorFlow to facilitate more straightforward and rapid implementation of neural networks.

To address the stochastic nature of the algorithms and potential discrepancies across learning sessions, we employed the k-fold Cross Validation method. We divided the dataset into 10 equal-sized subsets, and for each learning session, one fold served as the validation set while the other nine were used for training. In each learning session, every model with different parameter combinations was run 10 times and then the average performance was calculated across the 10 validation sets. To further enhance the reliability of our results, we repeated the entire procedure ten times, each time using a different random seed to shuffle the dataset before dividing it into folds. This generated a total of 100 validation sets (10 folds $\times$ 10 repetitions), and we tested all parameter combinations on these sets.

This rigorous validation technique ensured scientifically accurate and statistically reliable results, minimizing random variations and providing a solid basis for our conclusions.

For the analysis, we leveraged the Python-based Numpy and Pandas libraries to handle and manipulate the data. In addition to these libraries, we employed the matplotlib library for data visualization purposes, enabling a more comprehensive understanding of the model's performance and facilitating the evaluation of the results.

Throughout our experiments, we utilized a CNN with the optimal set of parameters to achieve the best possible performance. Initially, we conducted a thorough search for the best parameters for the CNN model, ensuring that our chosen model exhibited the highest performance. After identifying the optimal parameters, we proceeded to apply the selected model for the voting method.

In the voting method, we employed various numbers of models for prediction, which allowed us to evaluate the performance of our approach across different ensemble sizes. This strategy not only provided valuable insights into the robustness and reliability of our selected CNN model but also enabled us to identify the optimal number of models to use in our ensemble for achieving the best possible results.

2.2. Hyperparameter optimization

To ensure that our CNN model achieved the best possible performance, we conducted an extensive hyperparameter optimization process. We employed the Grid Search CV technique from the sci-kit learn library to systematically explore the hyperparameter space and identify the optimal combination of hyperparameters for our model. The following hyperparameters and their respective candidate values were included in the grid search:

- Batch size: 16, 32, 64
- Epochs: 50
- Activation function for the convolutional layers: ReLU, ReLU6, Swish
- Number of neurons used in the penultimate dense layer: 32, 64, 128
- Activation function for the first output layer: ReLU, ReLU6, Swish
- Dropout rate: 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8
- Activation function for the second output layer: Softmax
- Optimizer: Adam

The grid search was conducted with 10-fold CV to ensure that the selected hyperparameters were robust and generalizable across different subsets of the dataset. This approach provided a reliable estimate of the model's performance and reduced the risk of overfitting.

Table 1 shows the best performance with the Rectified Linear Unit (ReLU) and Rectified Linear Unit 6 (ReLU6) activation functions. Thus, we focused on them, excluding the Swish function due to its lower performance. Based on this, we chose the optimal hyperparameters for our CNN model, which guided the experiments and ensemble methods.

Table 1. The 5 best performing models with varying parameter combinations using Grid Search CV.

Function	Batch Size	Dropout Rate	Units	Mean Test Score
ReLU6	64	0.5	128	0.841061
ReLU6	32	0.4	64	0.840389
ReLU	16	0.5	64	0.840010
ReLU	16	0.4	64	0.839632
ReLU	32	0.4	64	0.839001

The outcomes of our experiments with different batch sizes, neuron numbers, and dropout rates are illustrated in Figures 3, 4, and 5.

These figures provide a comprehensive overview of the mean test scores achieved with various combinations of these hyperparameters, assessed using Grid Search CV. In these results, the ReLU6 is used as an activation function for the convolutional layers and the first output layer. This choice was based on our preliminary analysis, which indicated that ReLU6 outperformed other activation functions in our specific problem setting.

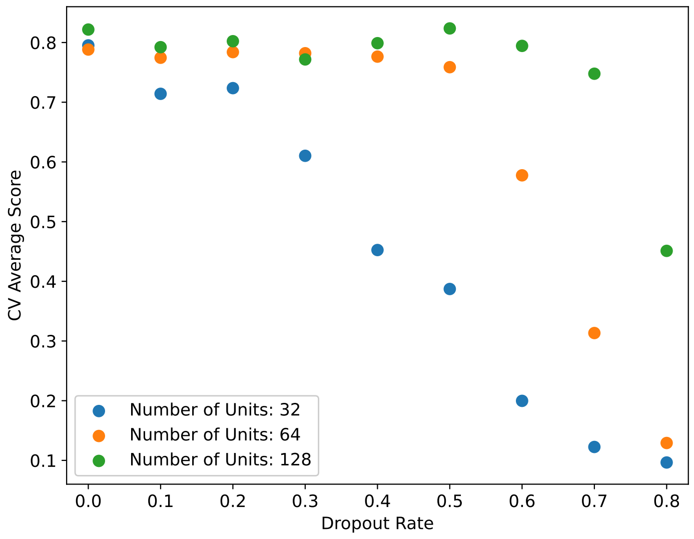


Figure 3. Mean test scores with batch size of 16 using Grid Search CV.

Figure 4 shows that as the batch size increases, the model can learn with a higher dropout rate, enabling more effective regularization and improved generalization performance. This observation is consistent with the idea that a larger batch size provides more accurate gradient estimates, allowing the model to handle the higher levels of noise introduced by dropout.

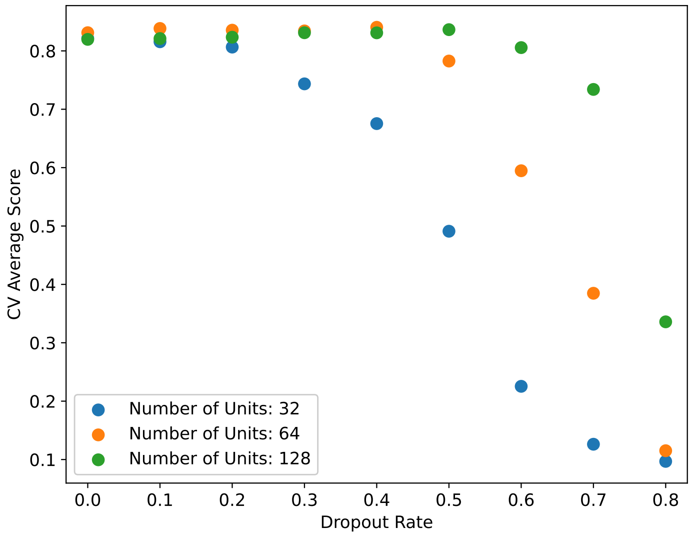


Figure 4. Mean test scores with batch size of 32 using Grid Search CV.

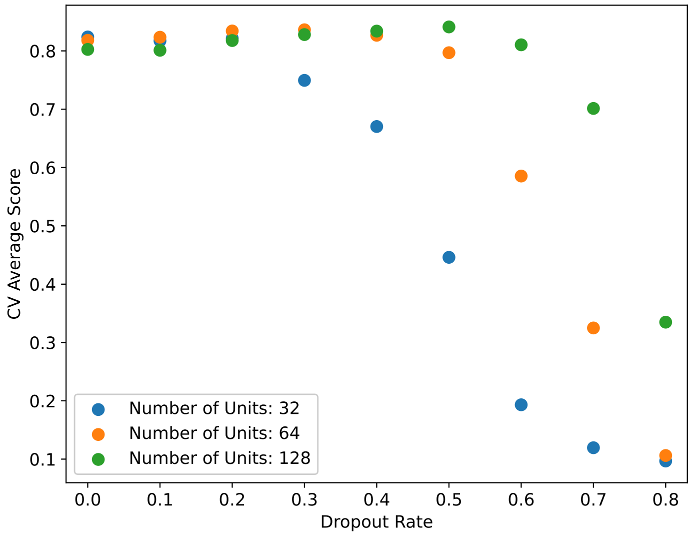


Figure 5. Mean test scores with batch size of 64 using Grid Search CV.

The higher the number of units in the model, the more effectively it can learn and utilize a higher dropout rate. This is likely because a larger number of neurons enable the model to represent more complex functions, counterbalancing the effect of dropout.

Figure 4 and 5 show that using a dropout rate equal to or greater than 0.7 does not result in any significant improvement in the model's performance. In fact, dropout rates of 0.7 or higher may lead to degraded performance due to excessive noise in the learning process, which could hinder the model from capturing important patterns in the data.

The 0.4 and 0.5 dropout rates produced the best results, providing a good balance between introducing noise to promote generalization and maintaining sufficient signals for the model to learn the underlying patterns. The optimal neuron numbers for our model were 64 and 128. These values yielded the best results across various batch sizes and dropout rates, indicating that they provide an appropriate level of model complexity to learn from the data without overfitting.

2.3. Performance of voting functions

By employing various voting schemes to combine the predictions of various models trained on the same dataset, the ensemble approach effectively reduces errors, increases stability, and enhances generalization capabilities. It is important to note that the models involved in this ensemble approach do not differ in their architecture or parameters; the differences between them arise from the individual training processes they undergo. As previously discussed in Section 1.3, the plurality voting function selects the prediction with the highest number of votes, while the fuzzy average voting function calculates the average of all predictions.

Table 2 presents the results of an experiment conducted to evaluate the performance of two voting schemes, plurality and fuzzy average voting functions. The experiment was performed 10,000 times, with each iteration involving a varying number of models, ranging from 2 to 30. The number of models used in the experiment increased incrementally by two in each iteration, providing a detailed view of the performance trends as the ensemble size grew.

In total, 60 models were used, and for each iteration, a random selection of the required number of models was made from these models. The models' predictions were based on the same dataset.

From the analysis of Figure 6, it can be observed that there is a gradual improvement in the performance of both the plurality voting function and the fuzzy average voting function as the number of models increases. This indicates that the ensemble of models can effectively leverage the strengths of individual models to reduce errors and increase stability. The voting functions tend to perform better when there is more diversity among the models, as it allows for a more robust decision-making process.

Further analysis reveals that as the number of models increases, the achieved results also improve, indicating higher accuracy in predictions. The fuzzy average voting function consistently outperforms the plurality voting function. This advantage is more noticeable with fewer models, but beyond 14–16 models, this

Table 2. Performance comparison of plurality and fuzzy average voting functions across different numbers of models.

Models	Plurality Voting Function	Fuzzy Average Voting Function
2	0.887717	0.906253
4	0.908664	0.915228
6	0.914723	0.918222
8	0.917318	0.919663
10	0.918746	0.920627
12	0.919694	0.921418
14	0.920391	0.922023
16	0.920961	0.922385
18	0.921416	0.922765
20	0.921691	0.923070
22	0.921951	0.923260
24	0.922246	0.923641
26	0.922408	0.923862
28	0.922506	0.924067
30	0.922789	0.924160

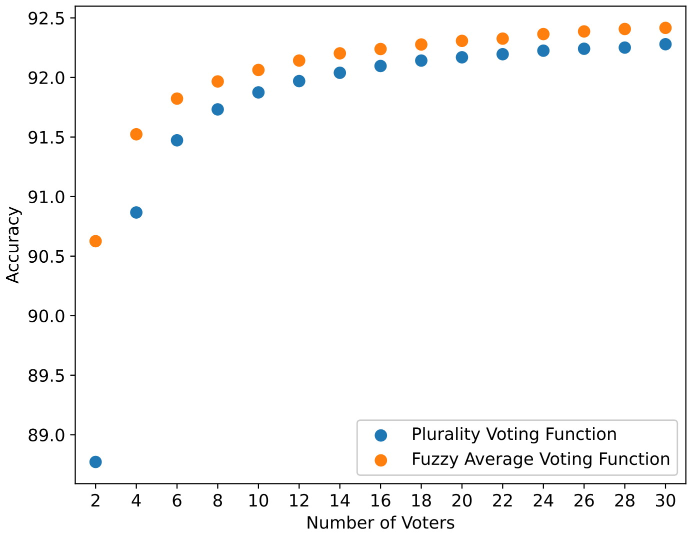


Figure 6. The performance results of voting functions by 2–30 voters on test data.

superiority stabilizes to a consistent advantage of approximately 0.001 to 0.0015 in favor of the fuzzy average voting function.

The stable advantage suggests that the fuzzy average voting function, by considering the average of predictions rather than just the most frequent one, provides more accurate results. However, from 14–16 models onwards, the rate of improvement in the results seems to decrease somewhat for both voting functions, suggesting that while incorporating more models can enhance performance, there may be diminishing returns beyond a certain point. The greater accuracy of the fuzzy average voting function may be attributed to its ability to incorporate more information from the models' outputs compared to the plurality voting function.

3. Conclusions

In conclusion, our results suggest that a careful choice of batch size, number of neurons in the penultimate dense layer, and dropout rates can significantly impact the performance of deep learning models. By employing grid search cross-validation, we were able to identify optimal combinations of these hyperparameters, leading to improved generalization and higher mean test scores. Furthermore, the experiment demonstrates that the choice of the voting scheme can have a considerable impact on the performance of an ensemble of models trained on the same dataset. While both plurality and fuzzy average voting functions can provide some benefits, the plurality voting function appears to offer more consistent improvements in performance as the number of models increases.

To further advance the field and enhance model performance, future research could explore the following developments: employing segmentation techniques to refine the input data, exploring the potential benefits of using fuzzification techniques for refining binary class membership values during model training, investigating alternative ensemble methods that may provide additional benefits, experimenting with different datasets to assess the robustness of the models, and incorporating various machine learning models, such as recurrent neural networks, to address the specific challenges of the task [4, 11, 12]. While the results presented in this study are promising, it is important to conduct additional research and analysis to fully comprehend the behavior and potential of these models, as this understanding can ultimately lead to more accurate and reliable methods.

References

[1] R. Atallah, A. Al-Mousa: *Heart Disease Detection Using Machine Learning Majority Voting Ensemble Method*, in: 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS), 2019, pp. 1–6, doi: 10.1109/ICTCS.2019.8923053.

[2] M. Brill, R. Freeman, S. Janson, M. Lackner: *Phragmén's voting methods and justified representation*, Mathematical Programming (2023), doi: 10.1007/s10107-023-01926-8.

[3] E. Bursztein, M. Martin, J. Mitchell: *Text-based CAPTCHA strengths and weaknesses*, in: Proceedings of the 18th ACM conference on Computer and communications security, 2011, pp. 125–138.

[4] J. Chen, X. Luo, Y. Liu, J. Wang, Y. Ma: *Selective Learning Confusion Class for Text-Based CAPTCHA Recognition*, IEEE Access 7 (2019), pp. 22246–22259, doi: 10.1109/ACCESS.2019.2899044.

[5] T. G. Dietterich: *Ensemble Methods in Machine Learning*, in: Multiple Classifier Systems, Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 1–15, isbn: 978-3-540-45014-6, doi: 10.1007/3-540-45014-9_1.

[6] I. Fazekas, A. Barta, L. Fórián: *Ensemble noisy label detection on MNIST*, Annales Mathematicae et Informaticae 53 (2021), pp. 125–137, doi: 10.33039/ami.2021.03.015.

[7] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al.: *Recent advances in convolutional neural networks*, Pattern Recognition 77 (2018), pp. 354–377.

[8] L. Kabari, U. Onwuka: *Comparison of Bagging and Voting Ensemble Machine Learning Algorithm as a Classifier*, International Journal of Computer Science and Software Engineering 9 (Mar. 2019), pp. 19–23.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al.: *Scikit-learn: Machine learning in Python*, the Journal of Machine Learning Research 12 (2011), pp. 2825–2830.

[10] A. Shawon: *Captcha Recognition*, Accessed: 2023-04-02, 2023, url: https://www.kaggle.com/code/shawon10/captcha-recognition.

[11] Y. Shu, Y. Xu: *End-to-End Captcha Recognition Using Deep CNN-RNN Network*, in: 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2019, pp. 54–58, doi: 10.1109/IMCEC46724.2019.8983895.

[12] T. Tajti: *Fuzzification of training data class membership binary values for neural network algorithms*, Annales Mathematicae et Informaticae 2020 (Oct. 2020), doi: 10.33039/ami.2020.10.001.

[13] T. Tajti: *New voting functions for neural network algorithms*, Annales Mathematicae et Informaticae 52 (2020), doi: 10.33039/ami.2020.10.003.

[14] E. Tasci, C. Uluturk, A. Ugur: *A voting-based ensemble deep learning method focusing on image augmentation and preprocessing variations for tuberculosis detection*, Neural Computing and Applications 33 (2021), pp. 15541–15555, doi: 10.1007/s00521-021-06177-2.

[15] S. Wan, H. Yang: *Comparison among Methods of Ensemble Learning*, in: 2013 International Symposium on Biometrics and Security Technologies, 2013, pp. 286–290, doi: 10.1109/ISBAST.2013.50.

[16] J. Wang, J. Qin, X. Xiang, Y. Tan, N. Pan: *CAPTCHA recognition based on deep convolutional neural network*, Mathematical Biosciences and Engineering 16.5 (2019), pp. 5851–5861, issn: 1551-0018, doi: 10.3934/mbe.2019292.

[17] R. Wilhelmy, H. Rosas: *captcha dataset*, July 2013, url: https://www.researchgate.net/publication/248380891_captcha_dataset.

[18] Y. Zhang, H. Gao, G. Pei, S. Luo, G. Chang, N. Cheng: *A Survey of Research on CAPTCHA Designing and Breaking Techniques*, in: 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), 2019, pp. 75–84, doi: 10.1109/TrustCom/BigDataSE.2019.00020.

Annales Mathematicae et Informaticae
58 (2023) pp. 92–109
DOI: 10.33039/ami.2023.08.011
URL: https://ami.uni-eszterhazy.hu

Using extended resolution to represent strongly connected components of directed graphs

Gábor Kusper[ab], **Zijian Győző Yang**[d], **Benedek Nagy**[ac]

[a]Eszterházy Károly Catholic University,
Faculty of Informatics,
Eger, Hungary

[b]University of Debrecen,
Faculty of Informatics,
Debrecen, Hungary

[c]Eastern Mediterranean University,
Faculty of Arts and Sciences,
Department of Mathematics,
Famagusta, North Cyprus, Mersin-10, Türkiye

[d]Hungarian Research Centre for Linguistics,
Budapest, Hungary

Abstract. In this paper, we study how to represent a directed graph as a SAT problem. We study those directed graphs which consists of two strongly connected components (SCC). We reuse the SAT models which are known as the Black-and-White SAT representations. We present the so-called 3rd Solution Lemma: If a directed graph consists of two SCCs, A and B, and there is an edge from A to B, then the corresponding SAT representation has 3 solutions: the black assignment, the white assignment, and the 3rd solution can be written as $\neg A$ union B. Using this result, we present an important negative result: We cannot represent all SAT problems as directed graphs using the Black-and-White SAT representations. Furthermore, we study the question how to represent an SCC by one Boolean variable to maintain the 3rd Solution Lemma. For that we use extended resolution.

Keywords: SAT problem, boolean logic, directed graphs

AMS Subject Classification: 03B05, 05C20

Submitted: July 31, 2023
Accepted: August 7, 2023
Published online: August 24, 2023

1. Introduction

There are various interesting links between directed graphs and SAT problems [11]. In this paper we are interested in such type of relationship. On the one hand, directed graphs could represent many types of objects, but in many cases, it is not so straightforward how and what type of representation leads to some advantages. Since, this problem generally seems to be very difficult, we work on a related one: we represent a directed graph as a propositional logical formula or, in fact, as a SAT problem.

We should mention here truth-teller–liar logical puzzles, when some people are having statements about the types of subsets of people and this information can be represented by graphs where the vertices representing the people. Depending on the various subtypes of the puzzles, the statements could produce a symmetric relation (strong truth-teller–strong liar, shortly SS puzzles [28]), meaning e.g., that the type of the persons are the same or the opposite. In many other puzzles, however, the statements are represented by directed edges and they represent implication type relations [27, 29, 30]. In some puzzles self-reference statements can also play important role [2, 3], and these puzzles are related also to the liar paradox [5]. The graph model of these puzzles usually used to infer some additional information and in many cases also the solution of the puzzle (i.e., assigning the types, like truth-teller and liar to each vertex in such a way that the statements match). Some puzzles are also analysed from the information flow point of view [6, 31].

There are also several other models to connect directed graphs to the SAT problem [20]. Each of these models has the following property: If the represented directed graph is strongly connected, then its SAT representation has only two solutions, the one where all variables are false, called the black assignment, and the one where all variables are true, called the white assignment. These models are called the Black-and-White SAT representations [9]. In this paper, we make a step further: we study those directed graphs which consist of not only one strongly connected component (SCC), but more. In this work we show that if a directed graph consists of two SCC components, A and B, and there is an edge from A to B, then the corresponding SAT representation has a third solution which is $\neg A$ union B. we call this result as the 3rd Solution Lemma.

This result a step forward in our main goal: Be able to represent any SAT problem as directed graph. Unfortunately, the final result is a negative one, we cannot achieve this goal based on Black-and-White SAT representations, as it is presented in Section 6.

We think that a graph representation conveys more intuitions than a logical formula. There are also neural networks which works on graphs. In the field of natural language processing, there are two main recent topics: large language models [32] and knowledge graphs [36]. Integration of knowledge graphs into language modeling became one of the most important research field [1, 35, 39]. Furthermore, Jiang, Gurajada et al. [16] takes an unorthodox view on the problem: combining textual heuristics together with neural features in a weighted rule-based framework based on first-order logic. This research based on the Logical Neural Networks [34], which is also becoming an increasingly researched area [24, 25].

Furthermore, we study the question how to represent an SCC by one Boolean variable. We found out that extended resolution [12, 38] is a suitable tool for that. To represent the SCC which consists of only two vertices a and b, we have to add to its model the following formula: $a \wedge b$ equals x, i.e., the following clauses: $\neg a \vee \neg b \vee x$, $\neg x \vee a$, and $\neg x \vee b$. This is the classical example of extended resolution, where x is a new variable. Although, the original problem is still very difficult, this work helps us to understand better what extended resolution means, and how to represent extended resolution graphically.

2. Formal definitions

As usual in logic and in research about SAT [7, 8, 33] we define how our objects (i.e. formulae) and their representations build up.

A *literal* is a Boolean variable, called positive literal, or the negation of a Boolean variable, called negative literal. Examples for literals are: $a, \neg a, b, \neg b, \ldots$.

A *clause* is a set of literals. A *clause set* is a set of clauses. A *SAT problem* is a clause set. An *assignment* is a set of literals. In a clause or in an assignment, a variable may occur either as a positive literal or as a negative literal, but not as both, or it may not occur at all.

Clauses are interpreted as disjunction of their literals. Assignments are interpreted as conjunction of their literals. Clause sets are interpreted as conjunction of their clauses.

If a clause or an assignment contains exactly k literals, then we say it is a *k-clause* or a *k-assignment*, respectively. A 1-clause is called to be a *unit*, a 2-clause is called to be a *binary clause*. A *k-SAT problem* is a clause set where its clauses have at most k literals. A clause from a clause set is a *full-length clause* iff it contains all variables from the clause set.

We use two intuitive notions: NNP clause, and NPP clause. A clause is an NNP clause iff it contains exactly one positive literal. A clause is an NPP clause iff it contains exactly one negative literal.

Negation of a set H is denoted by $\neg H$ which means that all elements in H are negated. Note that $\neg\neg H = H$.

Let $Vars$ be the set of variables of a clause set. We say that WW is the *white clause* or the *white assignment* iff $WW = Vars$. We say that BB is the *black clause* or the *black assignment* iff $BB = \neg Vars$. For example if $Var = \{a, b, c\}$, then $WW = \{a, b, c\}$, and $BB = \{\neg a, \neg b, \neg c\}$.

We say that clause C *subsumes* clause D iff C is a subset of D.

We say that clause set S *subsumes* clause C iff there is a clause in S which subsumes C. Formally: S *subsumes* $C \iff \exists D(D \in S \wedge D \subseteq C)$.

We say that assignment M is a *solution* for clause set S iff for all $C \in S$ we have $M \cap C \neq \{\ \}$.

We say that the clause set S is a *Black-and-White SAT problem* iff it has only two solutions, the white assignment (WW) and the black one (BB).

We say that clause sets A and B are equivalent, denoted by $A \equiv B$, iff A and B have the same set of solutions. We say that clause set A entails clause set B iff the set of solutions of A is a subset of the set of solutions of B, i.e., A may have no other solutions than B. This notion is denoted by $A \geq B$. Note that if A subsumes all clauses of B, then $A \geq B$.

We say that A is stronger than B iff $A \geq B$ and A and B are not equivalent. This notion is denoted by $A > B$.

Resolution $(\{a\} \cup A, \{\neg a\} \cup B) = A \cup B$, if $\{a\} \cup A$, $\{\neg a\} \cup B$, and $A \cup B$ are clauses. For example Resolution$(\{a, b\}, \{\neg a, c\}) = \{b, c\}$. But we cannot do resolution on $\{a, b\}$ and $\{\neg a, \neg b\}$, because $\{b, \neg b\}$ is not a clause. We say that literal c is blocked in clause C within clause set S iff for all $D \in S$ we have that if $\neg c \in D$ then $C \cup D \setminus \{c, \neg c\}$ is not a clause. In this case we say that C is blocked in S. It is well-known that a blocked clause can be deleted or added to a clause set without changing its satisfiability [17, 18] (but the set of solutions might be changed). If a is a literal in clause set S, and $\neg a$ is not a literal in S, then we say that a is a pure literal in S. Any pure literal is a blocked literal at the same time.

Extended resolution [12, 38] is SAT solver preprocessing technique to add blocked clauses to a clause set, so that the satisfiability of the original clause set is not changed. We use this technique to add (usually short) clauses which speed-up the search. Extended resolution adds clauses which are equal to $x \leftrightarrow f(Var)$, where x is a new variable, and $f(Var)$ is a Boolean function on some variables. The most widely used form is $x \leftrightarrow a \wedge b$, which is equivalent to the clause set: $\{\{\neg a, \neg b, x\}, \{a, \neg x\}, \{b, \neg x\}\}$. In this way x is blocked in all new clauses, so the new clauses are blocked.

Now, we recall some concepts of graph theory [4, 13].

The construction $D = (V, E)$ is a directed graph, where V is the set of vertices, and E is the set of edges. An edge is an ordered pair of vertices. The edge (a, b) is depicted by $a \rightarrow b$, and we can say that a has a child b. If (a, b) is an element of E, then we may say that (a, b) is an edge of D.

We say that $D = (V, E)$ is a communication graph iff for all a in V have that (a, a) is not in E, and if x is an element of V, then $\neg x$ must not be an element of V. We need this constraint because we generate a logical formula out of D. If we speak about a communication graph, then we may use the word node as a synonym of vertex.

A path from a_1 to a_j in directed graph D is a sequence of vertices $a_1, a_2, \ldots, a_j$ such that for each $i \in \{1, \ldots, j-1\}$ we have that (a_i, a_{i+1}) is an edge of D. A path from a_1 to a_j in directed graph D is a *cycle* iff (a_j, a_1) is an edge of D. The cycle $a_1, a_2, \ldots, a_j, a_1$ is represented by the following tuple: $(a_1, a_2, \ldots, a_j)$. This tuple can be used as a set of its elements as we shall define it formally later. Note that in the representation of a cycle the first and the last element must not be the same vertex.

If we have a cycle $(a_1, a_2, \ldots, a_n)$, then b is an exit point of it iff for some

$j \in \{1, 2, \ldots, n\}$ we have that (a_j, b) is an edge and $b \notin \{a_1, a_2, \ldots, a_n\}$.

A directed graph is complete iff every pair of distinct vertices is connected by a pair of unique edges (one in each direction). A directed graph is strongly connected iff there is a path from each vertex to each other vertex. Note that a complete graph is also strongly connected. Note that a strongly connected graph contains a cycle which contains all vertices.

The directed graph $G' = (V', E')$ is a subgraph of $G = (V, E)$ iff V' is a subset of V and E' is a subset of E. A subgraph of a directed graph G is a strongly connected component (SCC) iff it is strongly connected, and is maximal with this property: no additional edges or vertices from G can be included in the subgraph without breaking its property of being strongly connected. It is possible to test the strong connectivity of a graph, or to find its strongly connected components, in linear time, $O(|V| + |E|)$. The collection of strongly connected components forms a partition of the set of vertices of G.

We can create the so called condensation graph of a directed graph G by substituting each SCC of G by a single new vertex. The condensation graph is always a directed acyclic graph (DAG).

Somewhat similar technique was also used to solve truth-teller–liar puzzles by simplifying their graphs.

A SAT model (or shortly a model) of a communication graph is a mapping which maps nodes to boolean variables by a bijection, and which maps edges, cycles and possibly other parts of the graph to clauses, which creates a SAT problem, and from which the communication graph can be reconstructed up to those parts which are encoded. Generally, we use the function $MM(X)$ to assign a model to the communication graph X.

Especially, the Strong Model, our first model, was defined formally in our first paper [9]. First, we recall its definition.

Let D be a communication graph, then the Strong Model of D is denoted by $SM(D)$, and defined as follows:

$$SM(D) := \{\{\neg a,\ b\} \mid D = (V, E) \wedge (a, b) \in E\}.$$

Note that the the Strong Model can be the empty set. For example, if $D = (\{a, b\}, \{\})$, then $SM(D) = \{\}$.

The Strong Model of any communication graph must have the black solution (when every variable has the value true) and the white solution (when every variable has the value false), see Lemma 3.1.

In this way, actually, the Strong Model assigns an implication formula for each of the directed edges somewhat similarly as edges represent implications in some of the truth-teller–liar puzzles.

The Weak Model was defined formally in our second paper [21]. First, we recall its definition.

Let $D = (V, E)$ be a communication graph. Then we define the following notions:

$$OutE(a, E) := \{b \mid (a, b) \in E\}.$$

$$NodeRep(a, E) := \{\neg a\} \cup OutE(a, E).$$

$$NodeRep(D) := \{NodeRep(a, E) \mid D = (V, E) \wedge a \in V \wedge OutE(a, E) \neq \emptyset\}.$$

$$Cycles(D) := \{\{a_1, a_2, \ldots, a_k\} \mid D = (V, E) \wedge k = 1 \vee$$

$$\forall_{i=1\ldots k}(a_{(i \bmod k)+1} \in OutE(a_i, E))\}.$$

$$ExitPoints(\{a_1, a_2, \ldots a_k\}, E) := \{b \mid \exists_{i=1\ldots k}(b \in OutE(a_i, E)) \wedge$$

$$\neg\exists_{j=1\ldots k}(b = a_j)\}.$$

$$CycleRep(D) := \{\neg C \cup ExitPoints(C, E) \mid D = (V, E) \wedge$$

$$C \in Cycles(D) \wedge ExitPoints(C, E) \neq \emptyset\}.$$

$$WM(D) := NodeRep(D) \cup CycleRep(D).$$

We say that $WM(D)$ is the Weak Model of D.

3. The Black-and-White SAT representations

We recall some important results from [21]. We do not recall the proofs, only the theorems. Then we define the notion of Black-and-White SAT representations, and give a new theorem which helps us to prove the 3rd Solution Lemma.

Lemma 3.1. *Let D be a communication graph. Then $WM(D)$ has at least two solutions, namely the white assignment (WW) and the black assignment (BB). The same is true for $SM(D)$.*

Theorem 3.2. *Let D be a communication graph. Then $SM(D)$ is a Black-and-White 2-SAT problem iff the graph D is strongly connected.*

Theorem 3.3. *Let D be a communication graph. Then $WM(D)$ is a Black-and-White SAT problem iff D is strongly connected.*

Theorem 3.4 (Transitions Theorem)**.** *If $MM(X)$ is an arbitrary but fixed model of communication graphs, such that for any communication graph D we have that $SM(D) \geq MM(D) \geq WM(D)$, then $MM(D)$ is a Black-and-White SAT problem iff D is strongly connected.*

Until this point we have recalled some results from [21]. Now we give a new definition and a theorem. We shall define the notion of *Black-and-White SAT representations*, which is clearly motivated by the Transitions Theorem.

Definition 3.5. We say that function $MM(X)$, which generates a SAT problem from a communication graph, is a Black-and-White SAT representation iff for all communication graphs D we have that $SM(D) \geq MM(D) \geq WM(D)$.

There are some known example between the Strong and the Weak Model, for example: the Balatonboglár Model [21], and the Simplified Balatonboglár Model [22].

The Transitions Theorem can be extended easily for those cases where the two extreme cases, the Strong Model, and the Weak Model generate SAT problems with the same solution set. We give also the proof of this extended theorem because it is a new one.

Theorem 3.6. *Let $MM(X)$ be a Black-and-White SAT representation. Let D be an arbitrary but fixed communication graph. If $SM(D)$ and $WM(D)$ have the same set of solutions, then $SM(D)$, $MM(D)$, and $WM(D)$ have the same set of solutions.*

Proof. Let $MM(X)$ be a Black-and-White SAT representation. Let D be an arbitrary but fixed communication graph. From these we know that $SM(D) \geq MM(D) \geq WM(D)$. Assume that $SM(D)$ and $WM(D)$ have the same set of solutions. Then, by definition of clause set equivalence, we know that $SM(D) \equiv WM(D)$. From this and from $SM(D) \geq MM(D) \geq WM(D)$ we obtain that $SM(D) \equiv MM(D) \equiv WM(D)$. Hence, $SM(D)$, $MM(D)$, and $WM(D)$ have the same set of solutions. □

The message of this theorem is the following. If the two extreme cases, the Strong Model, and the Weak Model share a property on the set of solutions, then any other model between them should also have that property. We are going to use this result to proof the 3rd Solution Lemma in the next section.

4. The 3rd Solution Lemma

In this section we give and prove the so called 3rd Solution Lemma. This lemma states the following. If we use a Black-and-White SAT representation to represent a communication graph, i.e., we have a property that an SCC with vertices A represented as a SAT problem has exactly two solutions: the black, and the white assignment, then a communication graph with exactly two SCCs, with vertex sets A and B, where we have some edges from A to B, has exactly 3 solutions: $A \cup B$, $\neg A \cup \neg B$, and $\neg A \cup B$.

This result a step forward in our main goal: Be able to represent any SAT problem as directed graph. Unfortunately, the final result is a negative one, we cannot achieve this goal based on Black-and-White SAT representations, as it is presented and proven in Section 6.

First we present an auxiliary lemma, where there is no edge between the two SCCs.

Lemma 4.1. *Let C be a communication graph which consists of exactly two separate SCCs: A, and B, such that there is no edge between them. Let S be a SAT model of C generated by a Black-and-White SAT representation. Then S has exactly 4 solutions: $A \cup B$ (which is the white assignment), $\neg A \cup \neg B$ (which is the black one), $\neg A \cup B$, and $A \cup \neg B$.*

Proof. We know that A and B are distinct sets of vertices, because otherwise they would not be separate SCCs.

First, let us assume that S has been generated by the Strong Model. Then, by Theorem 3.2, we know that the representation of A has two models A and $\neg A$, and the representation of B has two models B and $\neg B$. Since the set of vertices of C equals $A \cup B$ and there is no edge between A and B, and, furthermore, S is generated from C by the Strong Model, we obtain that solution of S are the combinations of the solutions of the representations of A and B, which are the following 4 ones: $A \cup B$, $A \cup \neg B$, $\neg A \cup B$, and $\neg A \cup \neg B$.

Secondly, let us assume that S is generated by the Weak Model. Then, by Theorem 3.3, we obtain that S has the same set of solutions using the same proof steps.

Now let us assume that S is generated by any other Black-and-White SAT representation. Then, from the previous two cases and from Theorem 3.6, we obtain that S has the same set of solutions.

Hence, if C is a communication graph which consists of exactly two separate SCCs: A, and B, such that there is no edge between them, and S is a SAT model of C generated by a Black-and-White SAT representation, then S has exactly 4 solutions: $A \cup B$ (which is the white assignment), $\neg A \cup \neg B$ (which is the black one), $\neg A \cup B$, and $A \cup \neg B$. □

Now let us study the case that we have some edges between the two SCCs. We show that in this case the Black-and-White SAT representations results in 3 solution. This theorem will be called the 3rd Solution Lemma. This is our first main result.

Theorem 4.2 (3rd Solution Lemma)**.** *Let C be a communication graph which consists of exactly two separate SCCs: A, and B, such that there is at least one edge from A to B. Let S be a SAT model of C generated by a Black-and-White SAT representation. Then S has exactly 3 solutions: $A \cup B$ (which is the white assignment), $\neg A \cup \neg B$ (which is the black one), and $\neg A \cup B$.*

Proof. We know from our auxiliary Lemma 4.1, that if there were no edges between A and B, then we would have 4 solutions: $A \cup B$, $A \cup \neg B$, $\neg A \cup B$, and $\neg A \cup \neg B$.

It is enough to show that solution $A \cup \neg B$ is excluded, and $\neg A \cup B$ is not excluded because of the edges from A to B.

We do not have to consider the solutions $A \cup B$, which is the white assignment, and $\neg A \cup \neg B$, which is the black assignment, because they are always solutions, see Lemma 3.1 and Theorem 3.4.

Let as assume that we have only one edge from A to B, the edge (a_i, b_j), where $a_i \in A$, and $b_j \in B$. So (a_i, b_j) is an edge of C.

First, let us assume that S is generated from C by the Strong Model. Since (a_i, b_j) is an edge of C, we know, by definition of the Strong Model, that $\{\neg a_i, b_j\}$ is a clause in S. We know that $A \cup \neg B$ does not satisfy this clause, on the other hand $\neg A \cup B$ satisfies it. Therefore, S has 3 solutions: $A \cup B$, $\neg A \cup B$, and $\neg A \cup B$.

If we have more than one edge from A to B, then the clauses generated from those edges have the same property, thus $\neg A \cup B$ satisfies them.

Secondly, let us assume that S is generated from C by the Weak Model. Again, let us start the discussion with the case of a sole edge from A to B. Since (a_i, b_j) is an edge of C, where a_i is part of the cycle A, and where b_j is the only one exit point of this cycle, because A, and B are two separate SCCs. From this we know, by definition of the Weak Model, that $\{\neg A, b_j\}$ is a clause in S. We know that $A \cup \neg B$ does not satisfy this clause, on the other hand $\neg A \cup B$ satisfies it. There might be other cycles which contains a_i, but those are subsets of A, and they have at least one exit point, b_j. Therefore, the clauses generated from those cycles are satisfied by $\neg A \cup B$. If we have more than one edge from A to B, then the cycles from A have more exit points to B, but still, those clauses generated from those cycles have the same property, thus $\neg A \cup B$ satisfies them.

Now let us assume that S is generated by any other Black-and-White SAT representation. Then, from the previous two cases and from Theorem 3.6, we obtain that S has the same set of solutions.

Hence, if C is a communication graph which consists of exactly two separate SCCs: A, and B, such that there is at least one edge from A to B, and S is a SAT model of C, then S has exactly 3 solutions: $A \cup B$ (which is the white assignment), $\neg A \cup \neg B$ (which is the black one), and $\neg A \cup B$. □

Notice here that if the graph has two strongly connected components, then either there is no edge between them in any directions, or there is exactly one direction in which there is/are edge(s) between them in the graph. In this way, we have provided results for every graph that has exactly two SCCs.

5. An example with extended resolution

The 3rd Solution Lemma (Theorem 4.2) allows us to study not only strongly connected graphs, but also more general ones. In this section we give an example and some interesting connections to extended resolutions. We also raise some interesting questions which might highlight future ways of investigations.

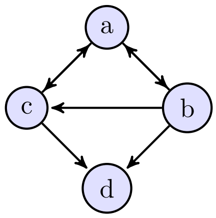


Figure 1. A communication graph with 4 vertices, 3 cycles.

Figure 1 shows a communication graph with 4 vertices, and 3 cycles.

As an example we show the Strong Model of the communication graph of Figure 1:

$$
\begin{aligned}
SM(D) = \{&\{\neg a, b\}, \{\neg a, c\}, \{\neg b, a\}, \{\neg b, c\},\\
&\{\neg b, d\}, \{\neg c, a\}, \{\neg c, d\}\}.
\end{aligned} \tag{5.1}
$$

Note that since the communication graph in Figure 1 is not strongly connected, its Strong Model (5.1) is not a Black-and-White SAT problem. Therefore, additionally to the black and white solutions, for example, it has the solution $\{\neg a, \neg b, \neg c, d\}$, as we expected because of the 3rd Solution Lemma (Theorem 4.2).

Note that, actually, this communication graph consists of two Strongly Connected Components (SCCs), which are: $\{a, b, c\}$ and $\{d\}$. Furthermore, there are edges from the first SCC to the second one, but not in the opposite direction. Hence, its model has to have, because of the 3rd Solution Lemma (Theorem 4.2), the solution: $\{\neg a, \neg b, \neg c, d\}$, which is the extra solution of our example.

Now, we can use extended resolution to introduce x instead of the first SCC. Thus, let x denote $a \wedge b \wedge c$. In this case we have to add the following clauses to the Strong Model of the graph: $\neg a \vee \neg b \vee \neg c \vee x$, $\neg x \vee a$, $\neg x \vee b$, and $\neg x \vee c$. It is easy to check that the new model still has only 3 solutions: $\{\neg a, \neg b, \neg c, \neg x, d\}$, $\{a, b, c, x, d\}$, and $\{\neg a, \neg b, \neg c, \neg x, \neg d\}$, the ones corresponding to the solutions of our original problem. This means that the first SCC can be substituted by the single variable x which greatly simplifies the graph and also its model.

Before we are going forward and show our other main result in the next section, we recall the following about another SAT representation technique: We know that resolvable networks can represent any SAT problem, see Lemma 3 in [23].

Now, some very interesting questions are arising. How to generalize the notion of SCC for resolvable networks? How can we recognize extended resolution? Especially, how can we recognize the new variable of extended resolution in a resolvable network?

Now, we switch back to the main topic of the present paper, the relation of communication graphs and SAT.

6. A negative result

Based on the 3rd Solution Lemma (Theorem 4.2), in this section, we concentrate on the question which SAT problems can be represented by communication graphs. Indeed, we can reverse engineering the directed graph representation of a SAT problem based on the solutions of the SAT problem.

We give all the examples with $N = 2$, and $N = 3$ variables. Note, that we do not list the black and the white assignments in set of solutions. (Technical note: We also used Wolfram Alpha to test our work.)

Examples with $N = 2$:

- Set of solutions: $\{\{\neg a, b\}\}$; the corresponding directed graph: $(\{a, b\}, \{(a, b)\})$.

- Set of solutions: $\{\{a, \neg b\}\}$; the corresponding directed graph: $(\{a, b\}, \{(b, a)\})$. Note that this is the same as the previous one up to variable renaming. So the resulting graphs are isomorphic.

- Set ofsolutions: $\{\{\neg a, b\}, \{a, \neg b\}\}$; the corresponding directed graph: $(\{a, b\}, \{\})$. Note that the set of edges is empty.

Examples with $N = 3$ (we do not list isomorphic examples):

- Set of solutions: $\{\{\neg a, \neg b, c\}\}$; the corresponding directed graph: $(\{a, b, c\}, \{(a, b), (b, a), (a, c)\})$. Note that we have two SCCs: $\{a, b\}$ and $\{c\}$, and an edge (a, c) between them. Further, we could also add the edge (b, c), because it is all the same which edge do we have between two SCCs. See the 3rd Solution Lemma (Theorem 4.2).

- Set of solutions: $\{\{\neg a, b, c\}\}$; the corresponding directed graph: $(\{a, b, c\}, \{(a, b), (b, c), (c, b)\})$. Note that we have two SCCs: $\{a\}$ and $\{b, c\}$, and an edge (a, b) between them.

- Set of solutions: $\{\{\neg a, \neg b, c\}, \{\neg a, b, c\}\}$; the corresponding directed graph: $(\{a, b, c\}, \{(a, b), (b, c)\})$. Note that we have three SCCs: $\{a\}$, $\{b\}$, and $\{c\}$, and there is a single path which consist of the edges: $(a, b), (b, c)$.

- Set of solutions: $\{\{\neg a, \neg b, c\}, \{\neg a, b, \neg c\}\{\neg a, b, c\}\}$; the corresponding directed graph: $(\{a, b, c\}, \{(a, b), (a, c)\})$. Note that we have three SCCs: $\{a\}$, $\{b\}$, and $\{c\}$, and they form a tree, where the root is a and the two leaves are b, and c.

- Set of solutions: $\{\{\neg a, \neg b, c\}, \{\neg a, b, \neg c\}\}$; the corresponding directed graph: Unfortunately, we cannot construct this. Based on this observation we have created Theorem 6.1, see below.

Now, we have arrived to the other of our main results. By listing some of the possibilities, it can clearly be seen that we have more possible SAT problems than how many communication graphs we have with the same number of variables (e.g., vertices). Thus, based on this counting argument, we can state our second main result:

Theorem 6.1. *There exists a SAT problem S for which there is no communication graph C such that $MM(C)$ is equivalent to S, if $MM(X)$ is a Black-and-White SAT representation.*

Proof. Let us consider all communication graph with 3 nodes. There are 16 possible ones up to isomorphism, see, e.g., `https://mathinsight.org/image/three_node_motifs`. We give the Strong and the Weak Model for all of them. Then we show that there is a SAT problem which cannot be represented, if we use a Black-and-White SAT representation.

1. $D_1 = (\{a, b, c\}, \{\})$,
   $SM(D_1) = \{\}$, $WM(D_1) = \{\}$, i.e.,
   $SM(D_1) \equiv WM(D_1)$. Number of SCCs is 3. All eight possible assignments are solutions, i.e., they satisfy the 'empty' condition.

2. $D_2 = (\{a, b, c\}, \{(a, b)\})$,
   $SM(D_2) = \{\{\neg a, b\}\}$, $WM(D_2) = \{\{\neg a, b\}\}$, i.e.,
   $SM(D_2) \equiv WM(D_2)$. Number of SCCs is 3.

3. $D_3 = (\{a, b, c\}, \{(a, b), (b, a)\})$,
   $SM(D_3) = \{\{\neg a, b\}, \{\neg b, a\}\}$, $WM(D_3) = \{\{\neg a, b\}, \{\neg b, a\}\}$, i.e.,
   $SM(D_3) \equiv WM(D_3)$. Number of SCCs is 2.

4. $D_4 = (\{a, b, c\}, \{(a, b), (a, c)\})$,
   $SM(D_4) = \{\{\neg a, b\}, \{\neg a, c\}\}$, $WM(D_4) = \{\{\neg a, b, c\}\}$, i.e.,
   $SM(D_4) \geq WM(D_4)$,
   $SM(D_4) \equiv WM(D_4) \cup \{\{\neg a, b, \neg c\}, \{\neg a, \neg b, c\}\}$. Number of SCCs is 3.

5. $D_5 = (\{a, b, c\}, \{(b, a), (c, a)\})$,
   $SM(D_5) = \{\{\neg b, a\}, \{\neg c, a\}\}$, $WM(D_5) = \{\{\neg b, a\}, \{\neg c, a\}\}$, i.e.,
   $SM(D_5) \equiv WM(D_5)$. Number of SCCs is 3.

6. $D_6 = (\{a, b, c\}, \{(c, a), (a, b)\})$,
   $SM(D_6) = \{\{\neg c, a\}, \{\neg a, b\}\}$, $WM(D_6) = \{\{\neg c, a\}, \{\neg a, b\}\}$, i.e.,
   $SM(D_6) \equiv WM(D_6)$. Number of SCCs is 3.

7. $D_7 = (\{a, b, c\}, \{(c, a), (a, b), (b, a)\})$,
   $SM(D_7) = \{\{\neg c, a\}, \{\neg a, b\}, \{\neg b, a\}\}$,
   $WM(D_7) = \{\{\neg c, a\}, \{\neg a, b\}, \{\neg b, a\}\}$, i.e.,
   $SM(D_7) \equiv WM(D_7)$. Number of SCCs is 2.

8. $D_8 = (\{a, b, c\}, \{(a, c), (a, b), (b, a)\})$,
   $SM(D_8) = \{\{\neg a, c\}, \{\neg a, b\}, \{\neg b, a\}\}$,
   $WM(D_8) = \{\{\neg a, b, c\}, \{\neg b, a\}, \{\neg a, \neg b, c\}\}$, i.e.,
   $SM(D_8) \geq WM(D_8)$, $SM(D_8) \equiv WM(D_8) \cup \{\{\neg a, b, \neg c\}\}$. Number of SCCs is 2.

9. $D_9 = (\{a, b, c\}, \{(a, c), (c, a), (a, b), (b, a)\})$,
   $SM(D_9) = \{\{\neg a, c\}, \{\neg c, a\}, \{\neg a, b\}, \{\neg b, a\}\}$,
   $WM(D_9) = \{\{\neg a, b, c\}, \{\neg c, a\}, \{\neg b, a\}, \{\neg a, \neg b, c\}, \{\neg a, \neg c, b\}\}$, i.e.,
   $SM(D_9) \equiv WM(D_9)$, because of resolution. Number of SCCs is 1.

10. $D_{10} = (\{a, b, c\}, \{(a, b), (a, c), (c, b)\})$,

    $SM(D_{10}) = \{\{\neg a, b\}, \{\neg a, c\}, \{\neg c, b\}\}$, $WM(D_{10}) = \{\{\neg a, b, c\}, \{\neg c, b\}\}$, i.e.,

    $SM(D_{10}) \geq WM(D_{10})$, $SM(D_{10}) \equiv WM(D_{10}) \cup \{\{\neg a, \neg b, c\}\}$. Number of SCCs is 3.

11. $D_{11} = (\{a, b, c\}, \{(a, b), (b, c), (c, a)\})$,

    $SM(D_{11}) = \{\{\neg a, b\}, \{\neg b, c\}, \{\neg c, a\}\}$,

    $WM(D_{11}) = \{\{\neg a, b\}, \{\neg b, c\}, \{\neg c, a\}\}$, i.e.,

    $SM(D_{11}) \equiv WM(D_{11})$. Number of SCCs is 1.

12. $D_{12} = (\{a, b, c\}, \{(a, b), (b, a), (c, a), (c, b)\})$,

    $SM(D_{12}) = \{\{\neg a, b\}, \{\neg b, a\}, \{\neg c, a\}, \{\neg c, b\}\}$,

    $WM(D_{12}) = \{\{\neg a, b\}, \{\neg b, a\}, \{\neg c, a, b\}\}$, i.e.,

    $SM(D_{12}) \equiv WM(D_{12})$, because of resolution. Number of SCCs is 2.

13. $D_{13} = (\{a, b, c\}, \{(a, b), (b, a), (a, c), (c, b)\})$,

    $SM(D_{13}) = \{\{\neg a, b\}, \{\neg b, a\}, \{\neg a, c\}, \{\neg c, b\}\}$,

    $WM(D_{13}) = \{\{\neg a, b, c\}, \{\neg b, a\}, \{\neg c, b\}, \{\neg a, \neg b, c\}\}$, i.e.,

    $SM(D_{13}) \equiv WM(D_{13})$, because of resolution. Number of SCCs is 1.

14. $D_{14} = (\{a, b, c\}, \{(a, b), (b, a), (a, c), (b, c)\})$,

    $SM(D_{14}) = \{\{\neg a, b\}, \{\neg b, a\}, \{\neg a, c\}, \{\neg b, c\}\}$,

    $WM(D_{14}) = \{\{\neg a, b, c\}, \{\neg b, a, c\}, \{\neg a, \neg b, c\}\}$, i.e.,

    $SM(D_{14}) \geq WM(D_{14})$, $SM(D_{14}) \equiv WM(D_{14}) \cup \{\{\neg a, b, \neg c\}, \{\neg b, \neg c, a\}\}$. Number of SCCs is 2.

15. $D_{15} = (\{a, b, c\}, \{(a, b), (b, a), (a, c), (c, a), (c, b)\})$,

    $SM(D_{15}) = \{\{\neg a, b\}, \{\neg b, a\}, \{\neg a, c\}, \{\neg c, a\}, \{\neg c, b\}\}$,

    $WM(D_{15}) = \{\{\neg a, b, c\}, \{\neg b, a\}, \{\neg c, a, b\}, \{\neg a, \neg b, c\}, \{\neg a, \neg c, b\}\}$, i.e.,

    $SM(D_{15}) \equiv WM(D_{15})$, because of resolution. Number of SCCs is 1.

16. $D_{16} = (\{a, b, c\}, \{(a, b), (b, a), (a, c), (c, a), (b, c), (c, b)\})$,

    $SM(D_{16}) = \{\{\neg a, b\}, \{\neg b, a\}, \{\neg a, c\}, \{\neg c, a\}, \{\neg b, c\}, \{\neg c, b\}\}$, $WM(D_{16}) = \{\{\neg a, b, c\}, \{\neg b, a, c\}, \{\neg c, a, b\}, \{\neg a, \neg b, c\}, \{\neg a, \neg c, b\}, \{\neg b, \neg c, a\}\}$, i.e.,

    $SM(D_{16}) \equiv WM(D_{16})$, because of resolution. Number of SCCs is 1.

Now, let us consider the special case when $S = \{\{\neg a, b\}, \{\neg a, c\}, \{a, \neg b, \neg c\}\}$. The solution set of S is $\{\{\neg a, \neg b, c\}, \{\neg a, b, \neg c\}\} \cup \{BB, WW\}$.

If we check, then neither of the above communication graph makes this property true: $SM(D_i) \geq S \geq WM(D_i)$, $i = 1 \dots 16$. To be more formal, we have to check all 16 cases. We do this in the following few paragraphs.

It is clear that there is no equivalent 2-SAT problem to S because all literals in its last clause are blocked. So we do not need to check those communication graphs where $SM(D_i) \equiv WM(D_i)$, since the Strong Model is always a 2-SAT problem. So we need to check only the following cases: $i = \{4, 8, 10, 14\}$.

By definition of clause set equality it is enough to show that either the solution set of $SM(D_i)$ is not a subset of solution set of S, or the solution set of S is not a subset of the solution set of $WM(D_i)$, where $i = \{4, 8, 10, 14\}$. We show all the 4 cases, $i = \{4, 8, 10, 14\}$, that this is true. Since the black and the white assignments are always solutions, we do not show them in the solution sets.

In the following 4 lines the first set is the set of solutions of $SM(D_i)$. the second one is the solution set of S, end the last one is the solution set of $WM(D_i)$.

$i = 4$: $\{\{\neg a, \neg b, c\}, \{\neg a, b, \neg c\}, \{\neg a, b, c\}\} \nsubseteq \{\{\neg a, \neg b, c\}, \{\neg a, b, \neg c\}\}$
$\subseteq \{\{\neg a, \neg b, c\}, \{\neg a, b, \neg c\}, \{\neg a, b, c\}, \{\neg a, b, \neg c\}, \{a, \neg b, c\}, \{a, b, \neg c\}\}$.

$i = 8$: $\{\{\neg a, \neg b, c\}\} \subseteq \{\{\neg a, \neg b, c\}, \{\neg a, b, \neg c\}\} \nsubseteq \{\{\neg a, \neg b, c\}, \{a, \neg b, c\}\}$.

$i = 10$: $\{\{\neg a, b, \neg c\}, \{\neg a, b, c\}\} \nsubseteq \{\{\neg a, \neg b, c\}, \{\neg a, b, \neg c\}\}$
$\nsubseteq \{\{\neg a, b, \neg c\}, \{\neg a, b, c\}, \{a, b, \neg c\}$.

$i = 14$: $\{\{\neg a, \neg b, c\}\}\} \subseteq \{\{\neg a, \neg b, c\}, \{\neg a, b, \neg c\}\}$
$\nsubseteq \{\{\neg a, \neg b, c\}, \{\neg a, b, c\}, \{a, \neg b, c\}$.

Therefore, for S there exists no communication graph D_i, $i = 1 \dots 16$ such that $MM(D_i)$ is equivalent to S, if $MM(X)$ is a Black-and-White SAT representation.

There are two isomorphic clause set for S: $S' = \{\{a, \neg b\}, \{\neg b, c\}, \{\neg a, b, \neg c\}\}$; and $S'' = \{\{b, \neg c\}, \{a, \neg c\}, \{\neg a, \neg b, c\}\}$.

By similar steps we can show that there exists no communication graph D_i, $i = 1 \dots 16$ such that $MM(D_i)$ is equivalent to S' or S'', if $MM(X)$ is a Black-and-White SAT representation.

Hence, there exists a SAT problem S for which there exists is no communication graph C such that $MM(C)$ is equivalent to S, if $MM(X)$ is a Black-and-White SAT representation. □

The above proof is rather technical, but we decided to present it, because it was not easy to construct it.

We present a more intuitive proving argument as a second proof of Theorem 6.1.

Proof. Let us count first the number of different communication graphs (up to isomorphism). In case of 3 vertices there are 16 different communication graphs (up to isomorphism), see `https://mathinsight.org/image/three_node_motifs`. Actually, the number of strongly connected graphs with up to isomorphism is known for many decades ago [26] and shown in various textbooks, e.g., [10, 14, 15]. Various combinatorial relations and the integer sequence of the number of directed graphs as a function of the number of vertices is listed in [37] at `https://oeis.org/A000273`.

Now, let us consider the 3-variable SAT problems. They may have at least the following 17 different solution sets (up to isomorphism). The black and white solutions, i.e., ppp and nnn are not listed, but should be added to each set below.

There is a case, when no extra solution exists (the Black and White case).

The cases with a sole extra solution:

$npp \qquad pnn.$

With two extra solutions:

$npp, pnp \qquad npp, pnn \qquad npp, npn \qquad pnn, npn.$

With three extra solutions:

$npp, pnp, ppn \qquad npp, pnp, pnn \qquad pnn, npn, npp \qquad pnn, npn, nnp$

$npp, pnp, nnp \qquad pnn, npn, ppn.$

With four extra solutions:

$npp, pnp, ppn, pnn \qquad npp, pnp, pnn, npn \qquad npp, pnp, pnn, nnp \qquad pnn, npn, nnp, npp.$

(And we have some cases with 5 or 6 extra solutions as well...)

Thus the number of possible solution sets is larger than the number of communication graphs and thus, the proof of the result is finished. □

We decided to keep both proofs, because their structure are very different, and we twink that both of them can be interesting for the readers. These two alternative proofs suggest that if we would like save our original goal, i.e., to find a graph representation of SAT problems, then we have to decrease the number of possible SAT instances, or we have to weaken the Weak Model or find some other representation.

We can decrease the number of possible SAT instances if we delete the blocked clauses before creating the corresponding directed graph. It is not so difficult to detect all blocked literal in case of 3-SAT, see [19], but in general, we have no idea how to show that this direction could work.

We can also use other representations. Actually, we have another candidate, the so called resolvable networks [23]. Its idea is that each clause can be represented by an $A \to B$ edge, where A represents the negative literals of the clause, and B represents the positive liters of it. A and B are called subnetworks, and they might have their own inner structure.

7. Conclusions

We have proven the 3rd Solution Lemma by showing that communication graphs with exactly two SSCs must have a 3rd solution in case of Black-and-White SAT representations. They must have 4 solution if there is no edges between the two SCCs, and they have 3 ones, if there are some edges (but not in both directions).We have also used a counting argument to prove that communications graphs cannot represent all possible SAT problems (if each variable is represented by a vertex). One may feel that this negative result gives a large imitation of our study, but this is actually, not the case. On the one hand, we believe that it gives a new motivation also to find and work with other type of graph representations of SAT, and not only with pure communication graphs. On the other hand, the subclass of SAT problems that can be represented by communication graphs can still be interesting to find out and graphically, visually show how some techniques could help us to solve some of the SAT instances.

Acknowledgements. The authors would like to thank to the project "Quality insurance automatization services for IT development by analysing and refactoring of decision structures of source codes based on expert systems", project ID: 2018-1.1.1-MKI-2018-00200, to support this research.

References

[1] M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, S. Sharifzadeh, V. Tresp, J. Lehmann: *PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings*, Journal of Machine Learning Research 22.82 (2021), pp. 1–6, URL: http://jmlr.org/papers/v22/20-825.html.

[2] L. Alzboon, B. Nagy: *Crazy Truth-Teller–Liar Puzzles*, Axiomathes 32.4 (2022), pp. 639–657.

[3] L. Alzboon, B. Nagy: *Truth-Teller–Liar Puzzles with Self-Reference*, Mathematics 8.2 (2020), paper 190, DOI: 10.3390/math8020190.

[4] J. Bang-Jensen, G. Gutin: *Digraphs: Theory, Algorithms and Applications*, Springer, Berlin, Heidelberg, Budapest, 2007.

[5] J. Barwise, L. S. Moss: *Vicious Circles: On the Mathematics of Non-Wellfounded Phenomena*, CLSI Publications, 1996.

[6] J. Barwise, J. Seligman: *Information Flow: the Logic of Distributed Systems*, Cambridge University Press, 1997.

[7] M. Ben-Ari: *Mathematical Logic for Computer Science*, Springer, London, 2012.

[8] A. Biere, M. Heule, H. Van Maaren, T. Walsh, eds.: *Handbook of Satisfiability*, IOS Press, 2021.

[9] C. Biró, G. Kusper: *Equivalence of Strongly Connected Graphs and Black-and-White 2-SAT Problems*, Miskolc Mathematical Notes 19.2 (2018), pp. 755–768, DOI: 10.18514/MMN.2018.2140.

[10] C. J. Colbourn, J. H. Dinitz, eds.: *CRC Handbook of Combinatorial Designs*, CRC Press, 1996.

[11] S. A. Cook: *The complexity of theorem proving procedures*, in: Proceedings of the Third Annual ACM Symposium, ACM, 1971, pp. 151–158.

[12] S. A. Cook: *A Short Proof of the Pigeon Hole Principle Using Extended Resolution*, SIGACT News 8.4 (1976), pp. 28–32, ISSN: 0163-5700, DOI: 10.1145/1008335.1008338.

[13] R. Diestel: *Graph Theory*, Springer, 2017.

[14] J. L. Gross, J. Yellen, eds.: *Handbook of Graph Theory*, CRC Press, 2004.

[15] F. Harary, E. M. Palmer: *Graphical Enumeration*, Academic Press, NY, 1973.

[16] H. Jiang, S. Gurajada, Q. Lu, S. Neelam, L. Popa, P. Sen, Y. Li, A. Gray: *LNN-EL: A Neuro-Symbolic Approach to Short-text Entity Linking*, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Online: Association for Computational Linguistics, Aug. 2021, pp. 775–787, DOI: 10.18653/v1/2021.acl-long.64, URL: https://aclanthology.org/2021.acl-long.64.

[17] O. Kullmann: *New methods for 3-SAT decision and worst-case analysis*, Theoretical Computer Science 223.1-2 (1999), pp. 1–72.

[18] O. Kullmann: *On a Generalization of Extended Resolution*, Discrete Applied Mathematics 96-97.1-3 (1999), pp. 149–176.

[19] G. Kusper: *Finding Models for Blocked 3-SAT Problems in Linear Time by Systematical Refinement of a Sub-model*, in: KI 2006: Advances in Artificial Intelligence, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 128–142, isbn: 978-3-540-69912-5.

[20] G. Kusper, T. Balla, C. Biró, T. Tajti, Z. G. Yang, I. Baják: *Generating Minimal Unsatisfiable SAT Instances from Strong Digraphs*, in: 2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2020, pp. 84–92, doi: 10.1109/SYNASC51798.2020.00024.

[21] G. Kusper, C. Biró: *Convert a Strongly Connected Directed Graph to a Black-and-White 3-SAT Problem by the Balatonboglár Model*, Algorithms 13.12 (2020), issn: 1999-4893, doi: 10.3390/a13120321, url: https://www.mdpi.com/1999-4893/13/12/321.

[22] G. Kusper, C. Biró, T. Balla: *Representing Directed Graphs as 3-SAT Problems using the Simplified Balatonboglár Model*, in: The 11th International Conference on Applied Informatics (ICAI-2020), 2020, poster presented at ICAI-2020.

[23] G. Kusper, C. Biró, B. Nagy: *Resolvable Networks—A Graphical Tool for Representing and Solving SAT*, Mathematics 9.20 (2021), issn: 2227-7390, doi: 10.3390/math9202597, url: https://www.mdpi.com/2227-7390/9/20/2597.

[24] T. Lebese, N. Makondo, C. Cornelio, N. Khan: *Proof Extraction for Logical Neural Networks*, in: Advances in Programming Languages and Neurosymbolic Systems workshop at NeurIPS 2021, 2021.

[25] S. Lu, N. Khan, I. Y. Akhalwaya, R. Riegel, L. Horesh, A. Gray: *Training Logical Neural Networks by Primal–Dual Methods for Neuro-Symbolic Reasoning*, in: ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 5559–5563, doi: 10.1109/ICASSP39728.2021.9415044.

[26] M. D. McIlroy: *Calculation of numbers of structures of relations on finite sets*, Massachusetts Institute of Technology, Research Laboratory of Electronics, Quarterly Progress Reports, No. 17 (1955), pp. 14–22.

[27] B. Nagy: *Duality of logical puzzles of type SW and WS—their solution using graphs*, Pure Math. Appl. 15.2-3 (2004), pp. 235–252.

[28] B. Nagy: *SS-type truthteller-liar puzzles and their graphs. (Hungarian)*, Alkalmaz. Mat. Lapok 23.1 (2006), pp. 59–72.

[29] B. Nagy: *SW-type puzzles and their graphs*, Acta Cybern. 16.1 (2003), pp. 67–82, url: https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3611.

[30] B. Nagy: *Truth-teller and liar puzzles and their graphs*, CEJOR: Cent. Eur. J. Oper. Res. 11.1 (2003), pp. 57–72.

[31] B. Nagy, G. Allwein: *Diagrams and Non-monotonicity in Puzzles*, in: Diagrammatic Representation and Inference, Third International Conference, Diagrams 2004, Cambridge, UK, March 22-24, 2004, Proceedings, ed. by A. F. Blackwell, K. Marriott, A. Shimojima, vol. 2980, Lecture Notes in Computer Science, Springer, 2004, pp. 82–96, doi: 10.1007/978-3-540-25931-2_10.

[32] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Gray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, R. Lowe: *Training language models to follow instructions with human feedback*, in: Advances in Neural Information Processing Systems, ed. by A. H. Oh, A. Agarwal, D. Belgrave, K. Cho, 2022.

[33] K. Pásztorné-Varga, M. Várterész: *A matematikai logika alkalmazásszemléletű tárgyalása (Application-oriented Study of Mathematical Logics, Hunagrian)*, PANEM, Budapest, 2003.

[34] R. Riegel, A. G. Gray, F. P. S. Luus, N. Khan, N. Makondo, I. Y. Akhalwaya, H. Qian, R. Fagin, F. Barahona, U. Sharma, S. Ikbal, H. Karanam, S. Neelam, A. Likhyani, S. K. Srivastava: *Logical Neural Networks*, CoRR abs/2006.13155 (2020), arXiv: 2006.13155.

[35] A. Saxena, A. Kochsiek, R. Gemulla: *Sequence-to-Sequence Knowledge Graph Completion and Question Answering*, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 2814–2828, doi: 10.18653/v1/2022.acl-long.201, url: https://aclanthology.org/2022.acl-long.201.

[36] P. Schneider, T. Schopf, J. Vladika, M. Galkin, E. Simperl, F. Matthes: *A Decade of Knowledge Graphs in Natural Language Processing: A Survey*, in: Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Online only: Association for Computational Linguistics, Nov. 2022, pp. 601–614, url: https://aclanthology.org/2022.aacl-main.46.

[37] N. J. A. Sloane, S. Plouffe: *The Encyclopedia of Integer Sequences (and its online version on the link)*, Academic Press, 1995, url: https://oeis.org/.

[38] G. S. Tseitin: *On the complexity of derivation in propositional calculus*, Structures in Constructive Mathematics and Mathematical Logic (1968), pp. 115–125.

[39] R. Wang, D. Tang, N. Duan, Z. Wei, X. Huang, J. Ji, G. Cao, D. Jiang, M. Zhou: *K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters*, in: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, Online: Association for Computational Linguistics, Aug. 2021, pp. 1405–1418, doi: 10.18653/v1/2021.findings-acl.121, url: https://aclanthology.org/2021.findings-acl.121.

Annales Mathematicae et Informaticae
58 (2023) pp. 110–120
DOI: 10.33039/ami.2023.08.004
URL: https://ami.uni-eszterhazy.hu

On language classes accepted by stateless $5' \to 3'$ Watson-Crick finite automata

Benedek Nagy[ab]

[a]Eastern Mediterranean University,
Faculty of Arts and Sciences,
Department of Mathematics,
Famagusta, North Cyprus, Mersin-10, Turkey

[b]Department of Computer Science,
Institute of Mathematics and Informatics,
Eszterházy Károly Catholic University,
Eger, Hungary
nbenedek.inf@gmail.com

Abstract. Watson-Crick automata are belonging to the natural computing paradigm as these finite automata are working on strings representing DNA molecules. Watson-Crick automata have two reading heads, and in the $5' \to 3'$ models these two heads start from the two extremes of the input. This is well motivated by the fact that DNA strands have $5'$ and $3'$ ends based on the fact which carbon atoms of the sugar group is used in the covalent bonds to continue the strand. However, in the two stranded DNA, the directions of the strands are opposite, so that, if an enzyme would read the strand it may read each strand in its $5'$ to $3'$ direction, which means physically opposite directions starting from the two extremes of the molecule. On the other hand, enzymes may not have inner states, thus those Watson-Crick automata which are stateless (i.e. have exactly one state) are more realistic from this point of view. In this paper these stateless $5' \to 3'$ Watson-Crick automata are studied and some properties of the language classes accepted by their variants are proven. We show hierarchy results, and also a "pumping", i.e., iteration result for these languages that can be used to prove that some languages may not be in the class accepted by the class of stateless $5' \to 3'$ Watson-Crick automata.

Keywords: Bio-computing, stateless finite automata, linear languages

AMS Subject Classification: AMS 68Q45: Formal languages and automata, 68Q07 Biologically inspired models of computation

Submitted: July 14, 2023
Accepted: August 7, 2023
Published online: August 16, 2023

1. Introduction

Finite automata are one of the oldest models of computing. The classes of both deterministic and nondeterministic variants are able to accept exactly the class of regular languages. Amar and Putzolu have generalised the concept and they have defined a special class of linear context-free language class, the so-called even-linear grammars and languages [1, 2].

Finite automata are very popular since they are very simple comparing them to other more sophisticated models. During the last decades, many kinds of extensions of finite automata are studied and proven to be applicable to accept larger classes of languages than the class of regular languages, but still have a moderate complexity. One of the branches of DNA computing is working with automata models accepting DNA molecules (or their formal representations), these automata are named as Watson-Crick automata [3, 6, 28]. These automata have two reading heads, one for each strand of the double stranded DNA. They can be used also for bioinformatic problems [30]. On the other hand, the strings have two extremes, namely their beginning and ends, which gives the rise of the 2-head models processing the input from their beginning and their end in a kind of parallel manner [13, 16]. Some of these 2-head models are known as $5' \rightarrow 3'$ Watson-Crick automata by a biological motivation describing these models to accept DNA molecules instead of ordinary words [17–19, 24, 26]. As usual at Watson-Crick automata, various extensions/restrictions could be applied on the model, e.g., string reading feature, or having only accepting states or having only one state. Generally, this 2-head model of computing, by finishing the computation at latest when the heads are in the same position, characterizes exactly the class of linear context-free languages [13, 16, 17, 26]. Regular-like expressions for linear context-free languages are shown in [29] to suggest the feeling that linear context-free languages can really be imagined as a superclass of the regular languages. Special variants capable to accept some special subclasses of the class of linear context-free languages of these automata models are also studied, e.g., the so-called even-linear languages (see, e.g., their importance in various applications [31, 32]) are accepted by a model, the so-called both-head stepping $5' \rightarrow 3'$ Watson-Crick finite automata, in which the two heads must move together in a synchronous way [14]. On the other hand, opposite to the ordinary finite state automata, the deterministic counterpart of the 2-head model is weaker, and the language class 2detLIN is accepted by them [25, 27]. Recently two more variants of the model have been investigated: In the state deterministic $5' \rightarrow 3'$ Watson-Crick automata the state of the next configuration depends only on the actual state and it does not depend on the read symbol(s) [22]. These automata can easily be characterized by their graphs. On the other hand, in quasi-deterministic $5' \rightarrow 3'$ Watson-Crick automata, in any computation, the state of the next configuration is deterministically computed, however the configuration itself is not [21]. These automata behave somewhat between the classical deterministic and nondeterministic models of the $5' \rightarrow 3'$ Watson-Crick automata. In [12], the non-sensing $5' \rightarrow 3'$ Watson-Crick automata are studied, in which both heads read the entire input (but, of course, in opposite direction). It was shown that Turing machine computations can be coded into the input which gives, on the one hand, a characterization of the class of recursively enumerable class of languages by the model, and on the other hand, the undecidability of some of the simple problems for the accepted languages. Moreover, infinite hierarchies of language classes were shown according to the number of allowed runs on the input, when in each run the entire input is processed by both heads.

From a biological point of view, the stateless variants, i.e., $5' \to 3'$ Watson-Crick automata with a sole state make more sense than models with several states. Thus, in this paper, we consider these variants. Usually, finite state automata store the information about the already processed input in their states. To store one bit information, one needs two states. Automata with a sole state referred as stateless, as they cannot store any information in their state. Thus, usually, not to allow to accept all possible inputs, these automata are incomplete, in the sense that they are not accepting those inputs that cannot be processed, i.e., they get stuck during (all) the computation(s) on these inputs. Actually, the things are a little bit more complex here, since actually, stateless automata at least have the information that the already processed part of the input can be processed.

We recall the well-known fact that stateless automata are as efficient as automata with any finitely many number of states in case of pushdown automata [9]. Furthermore, a similar fact has been proven for the $5' \to 3'$ Watson-Crick pushdown automata [15]. On the other hand, stateless variants of $5' \to 3'$ Watson-Crick multicounter machines were studied in [4, 5, 7, 8] by obtaining various hierarchies of the accepted language classes.

Pumping and iteration lemmas are well-known for various subclasses of the context-free languages [9]. In general, they give necessary conditions for the languages belonging to a given class, and thus, by their help, we may prove that a given language is definitely not belonging to the class we are interested in. They are usually proven by considering derivation trees, or for many subclasses, including, e.g., the class of regular languages, by arguments based on the finite automata model. There are variants of these theorems for some special subclasses of the class of linear context-free languages [10, 20]. In this paper, as one of our main results, we provide an iteration result for the languages of stateless $5' \to 3'$ Watson-Crick automata.

In the next section, we formally define our model. In Section 3 we show some examples. In Section 4 we present our main theorems and also we show how they can be applied. Conclusions and some future topics of research close the paper.

2. Formal definitions

In this section we define formally our model. We note here that in the literature, the definition may also include the so-called Watson-Crick complementarity relation defined on the alphabet. Since in the nature, it is a symmetric bijective relation, we simplify our model not to play with it. This can be done, since, on the one hand, in [11] it is shown that at Watson-Crick automata this relation does not play any important role, as the same language class can be accepted by using the identity instead of a more general complementarity relation. On the other hand, as we will see, in the sensing $5' \to 3'$ Watson-Crick automata, every position of the double stranded DNA is read by at most one of the heads, and thus, the relation on the letters at the same position of the two strands cannot play any role in the accepted languages.

Definition 2.1. A Watson-Crick finite automaton (a WK automaton) is a 5-tuple $A = (T, Q, q_0, F, \delta)$, where:

- T is the (input) alphabet, (e.g., the letters standing for possible bases of the nucleotides),
- the finite set of states Q, the initial state $q_0 \in Q$ and the set of final (also called accepting) states $F \subseteq Q$,
- the transition mapping δ is of the form $\delta : Q \times T^* \times T^* \to 2^Q$, such that it is non-empty only for finitely many triplets $(q, u, v), q \in Q, u, v \in T^*$.

The computation by WK automata goes through configurations as follows.

Definition 2.2. A configuration is a pair (q, w) containing q, the current state and w, the unprocessed part of the input.

In sensing $5' \to 3'$ WK automata, for any $w', x, y \in T^*, q, q' \in Q$, we write a step of the computation between two configurations as follows: $(q, xw'y) \Rightarrow (q', w')$ if and only if $q' \in \delta(q, x, y)$.

We denote the reflexive and transitive closure of the relation $\Rightarrow$ by $\Rightarrow^*$, and this is the computation relation on configurations.

Further, for an input $w \in T^*$, an accepting computation is a sequence of steps $(q_0, w) \Rightarrow^* (q_f, \lambda)$ for some $q_f \in F$.

As usual, we use automata for accepting languages, thus we have:

Definition 2.3. The language accepted by a sensing $5' \to 3'$ WK automaton consists of all words that are accepted by the automaton.

By comparing traditional finite state automata with sensing $5' \to 3'$ WK automata, there are two main differences. Both of those can be seen in the transition function. The first difference, as we have already mentioned, is that the sensing $5' \to 3'$ WK automata have two reading heads, thus the domain of the transition function contains triplets. The other difference, coming from biological motivations, is that the WK automata may read strings in a transition, not only letters. This is motivated by the fact that enzymes may be attached to the strands, and thus, read a longer part of the input in a computation step. On the other hand, to keep the model still finite, it is allowed to have transitions only for finitely many triplets of the domain, since it is not feasible to allow to read strings with an unlimited length, as enzymes must also have a finite size.

The above definitions can be used in general for any $5' \to 3'$ WK automata. However, there are some restricted variants, and actually, in this paper, we are focusing on some of these variants.

Definition 2.4. A Watson-Crick finite automaton is STATELESS if $Q = F = \{q_0\}$.

A Watson-Crick finite automaton is SIMPLE if $\delta : (Q \times ((\lambda, T^*) \cup (T^*, \lambda))) \to 2^Q$, i.e., at most one heads reads in a step.

A Watson-Crick finite automaton is ONE-LIMITED if $\delta : (Q \times ((\lambda, T) \cup (T, \lambda))) \to 2^Q$, i.e., exactly one letter is being read in each step.

The notation NWK is used for the stateless automata, as N stands for "no states". Further, the notation NSWK and N1WK is used for stateless simple and stateless one-limited automata, respectively.

By definition, clearly all N1WK automata are NSWK automata, and all NSWK automata are NWK automata at the same time.

We may also have other types of restrictions based on the sequences of computation steps:

Definition 2.5. A Watson-Crick finite automaton is DETERMINISTIC, if for any of its possible configurations there is at most one possible step to continue the computation.

A Watson-Crick finite automaton is STATE-DETERMINISTIC, if for each of its states $q \in Q$, if there is a transition from q and it goes to state p, i.e., $p \in \delta(q, u, v)$, then every transition from q goes to p.

A Watson-Crick finite automaton is QUASI-DETERMINISTIC, if for each possible configuration (q, w), if $(q, w) \Rightarrow (p, u)$ and also $(q, w) \Rightarrow (r, v)$, then $p = r$ must hold.

We note here that in some cases, e.g., [13, 16] the 2-head automata are defined in a way that they may able to read the input only letter by letter. Generally, if the automaton could have many states, that is not a problem, the string-reading feature of our model can be resolved by adding some new states and doing the computation on the input letter by letter. This can be done also in the deterministic case as proven in [25]. On the other hand, if we consider only automata with a sole state, the string-reading feature becomes essential in our models. Without allowing to read strings in a transition only very limited number of languages would be accepted by 2-head stateless automata.

3. Examples

In this section, for better understanding these computational models, we give some examples.

Example 3.1. The regular language $(01)^*$ is accepted by the deterministic NSWK automaton with state q having only transition $q \in \delta(q, 01, \lambda)$ (since the automaton has only a sole state, we briefly say that it has a transition with $(01, \lambda)$ without mentioning its state). This automaton uses only its left head during the entire computation on its input. Clearly the whole input can be processed if and only if it is in the regular language $(01)^*$.

Example 3.2. The deterministic NWK accepts the language $\{0^n 1^{3n}\}$ having only transition by $(0, 111)$. In each computation step, the left head (starting from the beginning of the input) is reading a 0, while the right head (starting from the end of the input) is reading 111. Consequently, when the heads meet and the computation is finished, any nonempty input accepted must have the form that all 0s precede all the 1s, and the number of 1s is exactly three times as many as the number of 0s. This language is a non-regular linear context-free language.

Example 3.3. The deterministic NWK with two transitions $(0, 0)$ and $(1, 1)$ accepts the language of even palindromes over $\{0, 1\}$, i.e., the language $\{u \cdot u^R \mid u \in \{0, 1\}^*\}$ where u^R is the reversal of the word u. This language is a well-known non-regular linear context-free language.

Example 3.4. The regular language 0^*1^* is accepted by the nondeterministic N1WK automaton having two transitions by $(0, \lambda)$ and by $(\lambda, 1)$. In each step of the computation, either the left head reads a 0 (from the beginning of the remaining input) or the right head reads a 1 (from the end of the remaining input). Observe that in fact, this automaton is not deterministic.

From the definitions of state-deterministic, quasi-deterministic and deterministic variants (see also [21, 22, 25, 27]) we can infer the following:

Proposition 3.5. *Every NWK automaton is state-deterministic and quasi-deterministic. Further, the class of NSWK automata coincides with the class of state-deterministic NSWK automata and also with the class of quasi-deterministic NSWK automata. Moreover, the class of N1WK automata coincides with the class of state-deterministic N1WK automata and with the class of quasi-deterministic N1WK automata.*

On the other hand, based on the examples shown above, we can infer also the following result about these models.

Proposition 3.6. *There are NWK, NSWK and N1WK automata that are not deterministic.*

Thus actually, we can consider six classes of stateless $5' \to 3'$ WK automata in the sequel. We show the hierarchy of the language classes accepted by them in Figure 1. However, to put also the class of regular languages into this hierarchy we may use our new results presented in the next section.

4. Main results

In this section, we concentrate on the NWK automata in general, thus the results of this section are applicable for each of the above mentioned subcases of the model as well.

Based on the transitions used in an NWK automaton we can define some further concepts.

Definition 4.1. Let an NWK automaton $A = (T, \{q\}, q, \{q\}, \delta)$ be given. By definition, it has finitely many transitions $\delta(q, \ell_i, r_i) = \{q\}$ defined, let denote this number by n. Let us have an alphabet $V = \{v_1, \dots, v_n\}$ with n elements, and let us assign the elements of V to the transitions of the automaton in a bijective way: $v_i \leftrightarrow (\ell_i, r_i)$. Let $\varphi, \mu : V \to T^*$ be the mappings defined as $\varphi(v_i) = \ell_i$ and analogously, $\mu(v_i) = r_i^R$, where R stand for the reversal of the word.

We refer to φ and μ as the FORWARD and the BACKWARD MORPHISMS of the automaton A and its accepted language L.

Now we are ready to claim one of our new results about the languages accepted by these models.

Theorem 4.2. *Let A be an NWK automaton over alphabet T. Then there is a finite alphabet V, and there exist the forward and backward morphisms $\varphi, \mu : V \to T^*$ such that the language accepted by A can be written as $\{\varphi(x)\mu(x^R) \mid x \in V^*\}$, where x^R is the reversal of the word x.*

Proof. Clearly, for each word $w \in T^*$ accepted by the automaton, there exists an accepting computation that can be described by the sequence of transitions $x \in V^*$. Moreover, in stateless automata every word of V^* is describing an accepting computation (of some input word w). In this computation the left head is reading the word defined by $\varphi(x)$ and as the right head is reading from the right, it is reading $\mu(x^R)$ during the computation. □

Furthermore, we state the following "pumping"-like theorem.

Theorem 4.3. *Let A be an NWK automaton over T. For any word w accepted by A, there is a factorisation $w = u \cdot v$, such that $u^i v^i$ is also accepted by A for any $i \in \mathbb{N}$.*

Proof. Let us consider any word w accepted by A. Then, by Theorem 4.2, an/the accepting computation on w can be described by $x \in V^*$. Further, $w = \varphi(x)\mu(x^R)$ with the associated morphisms. Considering the words of the form $x^i \in V^*$, they describe accepting computations of the words of the form $\varphi(x^i) \cdot (\mu((x^i)^R) = (\varphi(x))^i \cdot (\mu(x^R))^i$ which, with the choice of $u = \varphi(x)$ and $v = \mu(x^R)$, can be written as $u^i v^i$ as the theorem states. □

The theorem can also be seen as follows: the repetition of the computation implies a kind of insertion operation on the accepted words.

As we have seen, there are some non-regular languages that are accepted with deterministic NWK automata. Let us see an example what Theorem 4.3 means for an accepted language.

Example 4.4. Let us consider the language L of even palindromes shown in Example 3.3. Let $V = \{a, b\}$, $\varphi(a) = 0$, $\varphi(b) = 1$ and $\mu(a) = 0$, $\mu(b) = 1$. Then

$w = 00100100 \in L$, and in fact, $x = aaba \in V^*$ has the property that $\varphi(x) = 0010$ and $\mu(x^R) = 0100$. Therefore $w = \varphi(x) \cdot \mu(x^R)$. We may obtain the words
$(\varphi(x))^2 \cdot (\mu(x^R))^2 = 0010001001000100$,
$(\varphi(x))^3 \cdot (\mu(x^R))^3 = 001000100010010001001000$,
$(\varphi(x))^i \cdot (\mu(x^R))^i$ (for any $i \in \mathbb{N}$) based on the words x^2, x^3, x^i.

On the other hand, now we present another possible, maybe more useful application of Theorem 4.3.

Proposition 4.5. *The regular language a^*bba^* is not accepted by any NWK automata.*

Proof. As a^*bba^* does not satisfy the conditions of the previous theorem, as the number of bs cannot be "pumped", thus, obviously it cannot be a language that is accepted by any NWK automata. □

Thus, our result can efficiently be used to show that some languages are not acceptable by any NWK automata. From, e.g., Example 3.3 and Proposition 4.5 we can infer the incomparability of the class of regular languages and the class of languages accepted by stateless WK automata under set theoretical inclusion.

5. Conclusion and future work

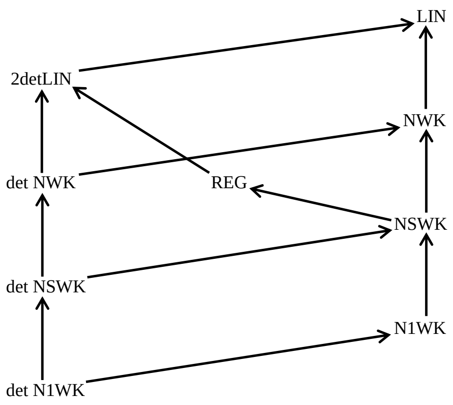


Figure 1. A hierarchy of language classes accepted by stateless sensing $5' \to 3'$ WK automata.

Figure 1 shows the hierarchy of the language classes of our model in a Hasse diagram (the automaton class here denoting the accepted language class). Classes not having directed path between them are incomparable under set theoretic inclusion relation. REG denotes the class of regular, LIN, the class of linear context-free languages (this is the class that is accepted by sensing $5' \to 3'$ WK automata) and 2detLIN the class of languages accepted by deterministic sensing $5' \to 3'$ WK automata.

In this paper, we have shown a new iteration theorem for the languages accepted by stateless $5' \to 3'$ Watson-Crick automata. This theorem is based on two newly defined morphisms. This approach could also be fruitful to analyse further properties of the corresponding language classes. We recall that a somewhat related topic, finite state $5' \to 3'$ Watson-Crick transducers (automata with output) were discussed in [23], where the description was also used some special functions that can be in relation to our newly defined morphisms.

It is a task of a future work to develop other specific iteration theorems and for other specific classes of languages accepted by variants of Watson-Crick automata and to describe some new properties of those language classes based on these and related results.

References

[1] V. Amar, G. R. Putzolu: *Generalizations of Regular Events*, Inf. Control. 8.1 (1965), pp. 56–63, doi: 10.1016/S0019-9958(65)90275-5.

[2] V. Amar, G. R. Putzolu: *On a Family of Linear Grammars*, Inf. Control. 7.3 (1964), pp. 283–291, doi: 10.1016/S0019-9958(64)90294-3.

[3] E. Czeizler, E. Czeizler: *A Short Survey on Watson-Crick Automata*, Bull. EATCS 88 (2006), pp. 104–119.

[4] Ö. Egecioglu, L. Hegedüs, B. Nagy: *Hierarchies of Stateless Multicounter $5' \to 3'$ Watson-Crick Automata Languages*, Fundam. Informaticae 110.1-4 (2011), pp. 111–123, doi: 10.3233/FI-2011-531.

[5] Ö. Egecioglu, L. Hegedüs, B. Nagy: *Stateless multicounter $5' \to 3'$ Watson-Crick automata*, in: Fifth International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2010, University of Hunan, Liverpool Hope University, Liverpool, United Kingdom / Changsha, China, September 8-10 and September 23-26, 2010, IEEE, 2010, pp. 1599–1606, doi: 10.1109/BICTA.2010.5645263.

[6] R. Freund, G. Paun, G. Rozenberg, A. Salomaa: *Watson-Crick finite automata*, in: DNA Based Computers, Proceedings of a DIMACS Workshop, Philadelphia, Pennsylvania, USA, June 23-25, 1997, ed. by H. Rubin, D. H. Wood, vol. 48, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, DIMACS/AMS, 1997, pp. 297–327, doi: 10.1090/dimacs/048/22.

[7] L. Hegedüs, B. Nagy: *On String Reading Stateless Multicounter $5' \to 3'$ Watson-Crick Automata - (Extended Abstract)*, in: Unconventional Computation and Natural Computation - 12th International Conference, UCNC 2013, Milan, Italy, July 1-5, 2013. Proceedings, ed. by G. Mauri, A. Dennunzio, L. Manzoni, A. E. Porreca, vol. 7956, Lecture Notes in Computer Science, Springer, 2013, pp. 257–258, doi: 10.1007/978-3-642-39074-6_29.

[8] L. Hegedüs, B. Nagy, Ö. Egecioglu: *Stateless multicounter $5' \to 3'$ Watson-Crick automata: the deterministic case*, Nat. Comput. 11.3 (2012), pp. 361–368, doi: 10.1007/s11047-011-9290-9.

[9] J. E. Hopcroft, J. D. Ullman: *Introduction to Automata Theory, Languages and Computation, Second Edition*, Addison-Wesley, 2000.

[10] G. Horváth, B. Nagy: *Pumping lemmas for linear and nonlinear context-free languages*, Acta Univ. Sapientiae, Informatica 2.2 (2010), pp. 194–209, arXiv: 1012.0023, url: http://arxiv.org/abs/1012.0023.

[11] D. Kuske, P. Weigel: *The Role of the Complementarity Relation in Watson-Crick Automata and Sticker Systems*, in: Developments in Language Theory, 8th International Conference, DLT 2004, Auckland, New Zealand, December 13-17, 2004, Proceedings, ed. by C. Calude, E. Calude, M. J. Dinneen, vol. 3340, Lecture Notes in Computer Science, Springer, 2004, pp. 272–283, doi: 10.1007/978-3-540-30550-7_23.

[12] P. Leupold, B. Nagy: *$5'{\to}3'$ Watson-Crick Automata With Several Runs*, Fundam. Informaticae 104.1-2 (2010), pp. 71–91, doi: 10.3233/FI-2010-336.

[13] R. Loukanova: *Linear Context Free Languages*, in: Theoretical Aspects of Computing - ICTAC 2007, 4th International Colloquium, Macau, China, September 26-28, 2007, Proceedings, ed. by C. B. Jones, Z. Liu, J. Woodcock, vol. 4711, Lecture Notes in Computer Science, Springer, 2007, pp. 351–365, doi: 10.1007/978-3-540-75292-9_24.

[14] B. Nagy: $5' \to 3'$ *Sensing Watson-Crick Finite Automata*, in: Sequence and Genome Analysis II – Methods and Applications, ed. by G. Fung, iConcept Press, 2010, pp. 39–56.

[15] B. Nagy: *$5'{\to}3'$ Watson-Crick pushdown automata*, Inf. Sci. 537 (2020), pp. 452–466, doi: 10.1016/j.ins.2020.06.031.

[16] B. Nagy: *A class of 2-head finite automata for linear languages*, Triangle 8 (Languages. Mathematical Approaches) (2012), pp. 89–99.

[17] B. Nagy: *On* $5' \to 3'$ *Sensing Watson-Crick Finite Automata*, in: DNA Computing, 13th International Meeting on DNA Computing, DNA13, Memphis, TN, USA, June 4-8, 2007, Revised Selected Papers, ed. by M. H. Garzon, H. Yan, vol. 4848, Lecture Notes in Computer Science, Springer, 2008, pp. 256–262, doi: 10.1007/978-3-540-77962-9_27.

[18] B. Nagy: *On a hierarchy of* $5' \to 3'$ *sensing Watson-Crick finite automata languages*, J. Log. Comput. 23.4 (2013), pp. 855–872, doi: 10.1093/logcom/exr049.

[19] B. Nagy: *On a hierarchy of* $5' \to 3'$ *sensing WK finite automata languages*, in: Mathematical Theory and Computational Practice, CiE, Abstract Booklet, Heidelberg, Germany, 2009, pp. 266–275.

[20] B. Nagy: *Pumping lemmas for special linear languages*, in: ICAI 2010, Eger, Hungary, vol. II. 2010, pp. 73–81.

[21] B. Nagy: *Quasi-deterministic* $5' \to 3'$ *Watson-Crick Automata*, in: Proceedings 12th International Workshop on Non-Classical Models of Automata and Applications, NCMA 2022, Debrecen, Hungary, August 26-27, 2022, ed. by H. Bordihn, G. Horváth, G. Vaszil, vol. 367, EPTCS, 2022, pp. 160–176, doi: 10.4204/EPTCS.367.11.

[22] B. Nagy: *State-deterministic* $5' \to 3'$ *Watson-Crick automata*, Nat. Comput. 20.4 (2021), pp. 725–737, doi: 10.1007/s11047-021-09865-z.

[23] B. Nagy, Z. Kovács: *On deterministic 1-limited $5' \to 3'$ sensing Watson-Crick finite-state transducers*, RAIRO Theor. Informatics Appl. 55 (2021), p. 5, doi: 10.1051/ita/2021007.

[24] B. Nagy, S. Parchami: $5' \to 3'$ *Watson-Crick automata languages-without sensing parameter*, Nat. Comput. 21.4 (2022), pp. 679–691, doi: 10.1007/s11047-021-09869-9.

[25] B. Nagy, S. Parchami: *On deterministic sensing* $5' \to 3'$ *Watson-Crick finite automata: a full hierarchy in 2detLIN*, Acta Informatica 58.3 (2021), pp. 153–175, doi: 10.1007/s00236-019-00362-6.

[26] B. Nagy, S. Parchami, H. M. M. Sadeghi: *A New Sensing* $5' \to 3'$ *Watson-Crick Automata Concept*, in: Proceedings 15th International Conference on Automata and Formal Languages, AFL 2017, Debrecen, Hungary, September 4-6, 2017, ed. by E. Csuhaj-Varjú, P. Dömösi, G. Vaszil, vol. 252, EPTCS, 2017, pp. 195–204, doi: 10.4204/EPTCS.252.19.

[27] S. Parchami, B. Nagy: *Deterministic Sensing* $5' \to 3'$ *Watson-Crick Automata Without Sensing Parameter*, in: Unconventional Computation and Natural Computation - 17th International Conference, UCNC 2018, Fontainebleau, France, June 25-29, 2018, Proceedings, ed. by S. Stepney, S. Verlan, vol. 10867, Lecture Notes in Computer Science, Springer, 2018, pp. 173–187, doi: 10.1007/978-3-319-92435-9_13.

[28] G. Paun, G. Rozenberg, A. Salomaa: *DNA Computing - New Computing Paradigms*, Texts in Theoretical Computer Science. An EATCS Series, Springer, 1998, isbn: 978-3-540-64196-4, doi: 10.1007/978-3-662-03563-4.

[29] J. M. Sempere: *On a Class of Regular-like Expressions for Linear Languages*, J. Autom. Lang. Comb. 5.3 (2000), pp. 343–354, doi: 10.25596/jalc-2000-343.

[30] J. M. Sempere: *On the application of Watson-Crick finite automata for the resolution of bioinformatic problems*, in: Tenth Workshop on Non-Classical Models of Automata and Applications, NCMA 2018, Košice, Slovakia, August 21-22, 2018, ed. by R. Freund, M. Hospodár, G. Jirásková, G. Pighizzini, Österreichische Computer Gesellschaft, 2018, pp. 29–30.

[31] J. M. Sempere, P. García: *A Characterization of Even Linear Languages and its Application to the Learning Problem*, in: Grammatical Inference and Applications, Second International Colloquium, ICGI-94, Alicante, Spain, September 21-23, 1994, Proceedings, ed. by R. C. Carrasco, J. Oncina, vol. 862, Lecture Notes in Computer Science, Springer, 1994, pp. 38–44, doi: 10.1007/3-540-58473-0_135.

[32] J. M. Sempere, P. García: *Learning Locally Testable Even Linear Languages from Positive Data*, in: Grammatical Inference: Algorithms and Applications, 6th International Colloquium: ICGI 2002, Amsterdam, The Netherlands, September 23-25, 2002, Proceedings, ed. by P. W. Adriaans, H. Fernau, M. van Zaanen, vol. 2484, Lecture Notes in Computer Science, Springer, 2002, pp. 225–236, doi: 10.1007/3-540-45790-9_18.

Annales Mathematicae et Informaticae
58 (2023) pp. 121–132
DOI: 10.33039/ami.2023.08.012
URL: https://ami.uni-eszterhazy.hu

Vibronic spectra of molecules – an experiment with a quantum computer simulator[*]

András Németh[a], Tamás Kozsik[a], Zoltán Zimborás[a b]

[a]Department of Programming Languages and Compilers, Faculty of Informatics, ELTE Eötvös Loránd University, Budapest, Hungary
nemethandras@inf.elte.hu, ORCID: 0000-0002-4187-0673
kozsik.tamas@inf.elte.hu, ORCID: 0000-0003-4484-9172

[b]Quantum Computing and Information Research Group, Wigner Research Centre for Physics, Budapest, Hungary
zimboras.zoltan@wigner.hu, ORCID: 0000-0002-2184-526X

Abstract. In addition to the exciting fundamental questions of quantum computing and implementation possibilities of quantum computers, it is important to look for application areas of quantum computing, to point out practical cases which justify the need for this technology. Besides the well-known qubit-based quantum computers, there are also devices based on other foundations that can surpass the capabilities of classical computers. Among these, devices operating on the basis of boson sampling have a naturally occurring application: the approximate calculation of the vibrational spectrum of molecules. Two separate research groups, Huh et al. [14] and Wang et al. [22] created quantum simulators based on boson sampling, which were successfully used to calculate the transition probabilities between specific states of small molecules. Following the methodology found in these two articles, this paper presents how the calculations of transition probabilities can be performed on a classical computer using a quantum computer simulator which is based on Gaussian boson sampling.

Keywords: Quantum Computing, Gaussian Boson Sampling, Vibronic Spectra of Molecules, Quantum Chemistry, Spectroscopy

AMS Subject Classification: 68Q12, 81P68

[*]This research was supported by the Ministry of Culture and Innovation and the National Research, Development and Innovation Office within the Quantum Information National Laboratory of Hungary (Grant No. 2022-2.1.1-NL-2022-00004).

Submitted: July 25, 2023
Accepted: August 7, 2023
Published online: August 30, 2023

1. Introduction

The topic of this paper lies in the intersection of two huge research areas. The first of these is currently one of the hottest, most exciting technology: quantum informatics and quantum computers. Among the countless possible directions, this paper addresses boson sampling, which became the focus of attention after the famous article by Aaronson and Arkhipov [1], in which they proved that the boson sampling problem is difficult to calculate on a classical computer. There are multiple physical implementations of boson sampling, but the most common is the photonic solution, where one-photon states are used as input to an interferometer, and the number of photons is measured at the output. The interferometer is actually the simplest device in the setup; the devices for producing single-photons and for measuring the number of photons are the pinnacle of today's technology, and are constantly being developed. Therefore, it is a much simpler option if we do not use single photons as input, but coherent light, i.e. laser. Since coherent states can be described by Gaussian distributions, this type of boson sampling is called Gaussian boson sampling.

The other major area is – in contrast to quantum informatics – an area that has been actively researched for more than a hundred years, namely spectroscopy, including the examination of the vibrational spectra of molecules. Its importance does not need to be explained, since it has a great influence on our everyday life, for example, on the discovery of novel and more suitable materials for specific scientific, industrial or medical tasks.

There exists an approximate method for determining the vibration spectrum of molecules, in which the probability of transitions between different charged states is calculated. This method can be implemented naturally using Gaussian boson sampling. Our research focuses on this opportunity. Based on Wang et al. [22] and Huh et al. [14], an implementation is presented here which is running on a boson sampling-based simulator, which, in turn, is executed on a classical computer.

The contributions of this paper are (1) an implementation of the algorithm for calculating an approximation of the vibronic spectra of molecules on a photonic quantum computer simulator; (2) the proof that for small molecules this algorithm can be executed on classical computers, which makes testing of photonic quantum computer programs in a simulator possible; and (3) the validation of the results presented in earlier works on the computation of the vibronic spectra of several molecules.

The rest of the paper is structured as follows. Section 2.1 provides a brief introduction to boson sampling and Gaussian boson sampling. Section 2.2 overviews a possible approximate calculation of the vibrational spectra of molecules relying on Gaussian boson sampling. Section 3 presents the implementation of this calculation on a photonic quantum computer simulator. The calculation results are evaluated in Section 4. The achievements of Xanadu, based on the work of Huh et al. [14], are covered in Section 5. Finally, Section 6 sets out future research directions and concludes the paper.

2. Theoretical background

Before elaborating on our experiment with the photonic quantum computer simulator, some basic information of the two aforementioned research areas shall be provided, with a special emphasis on the connection of the two. First the idea of Gaussian boson sampling is introduced, which is one of the main interests of the used simulator, and which gives the algorithmic background for the second component, the approximate computation of the vibronic spectra of molecules.

2.1. Gaussian boson sampling

When talking about quantum computing, discussions tend to begin with the concept of the *qubit*, since it is a widely recognized concept not only among scientists but also among the general public. A qubit is an abstract two-dimensional quantum physical system that can be implemented in various physical forms. It can take on two values (base states), with the Dirac-notation $|0\rangle$ and $|1\rangle$, and can be described using elements of quantum physics, such as a Hilbert space. Qubits can exist in superposition and entangled states with each other. However, these concepts can be extended to larger dimensions, such as the three-dimensional, or any finite-dimensional *qudit*, i.e. $|n\rangle$. Computations are described as unitary operators which act on these states. Moreover, there are two special, non-unitary operators with special purposes: the *annihilation* and *creation operators* (together called as *ladder operators*). These move the physical state from one state to another. For example, if the annihilation operator is denoted as $\hat{a}$ then $\hat{a}|1\rangle = |0\rangle$. The creation operator $\hat{a}^\dagger$ is the adjoint of the annihilation operator. Other important theorems for any finite d-dimensional quantum physical system are the following.

$$\begin{aligned} \hat{a}|n\rangle &= \sqrt{n}|n-1\rangle \\ \hat{a}^\dagger|n\rangle &= \sqrt{n+1}|n+1\rangle \\ \hat{a}^\dagger\hat{a}|n\rangle &= n|n\rangle \\ \hat{a}|0\rangle = 0 \quad &\text{and} \quad \hat{a}^\dagger|d-1\rangle = 0 \end{aligned} \tag{2.1}$$

The product of ladder operators $\hat{N} = \hat{a}^\dagger\hat{a}$ is the number operator. The third equation of (2.1) also shows that quantum states are eigenstates of this number operator in a natural way.

The above can be further generalized to a countably infinite dimension (note that in this case the last equation of (2.1) will make no sense: there will be no state that the creation operator takes to the null vector of the Hilbert space). Bosonic particles are an example of this type of quantum physical system, where unbounded number of particles can be counted in a given state. Such a system is still quantum, i.e. it has superposition and entanglement properties. In the countably infinite dimensional case the quantum physical states are called the *qumodes*.

There is a distribution of states and physical quantities in the case of an arbitrary quantum physical system which can only be determined by multiple measurements, so in fact every quantum computer is a physical implementation of the sampling problem. In the countably infinite-dimensional case this is called boson sampling [10].

Note that the physically feasible operators are all unitary, in the two-dimensional case, for example, the well-known Pauli matrices, the Hadamard gate, or the Controlled Not gate. The physical implementation of boson sampling can be various, but the simplest is still the photonic solution. Quantum optical devices act as unitary operators on the qumode(s). One can find a detailed description of these in quantum optics. The theoretical formula of such devices used in practice with the help of ladder operators is as follows [25].

$$\hat{\mathcal{P}}_i(\phi) = e^{-i\phi\hat{a}_i^\dagger\hat{a}_i} \tag{2.2}$$

$$\hat{\mathcal{D}}_i(\alpha) = e^{\alpha\hat{a}_i^\dagger-\alpha^*\hat{a}_i} \tag{2.3}$$

$$\hat{\mathcal{S}}_i(\zeta) = e^{\frac{1}{2}(\zeta^*\hat{a}_i^2-\zeta\hat{a}_i^{\dagger 2})} \tag{2.4}$$

$$\hat{\mathcal{B}}_{ij}(\theta) = e^{\theta(\hat{a}_i^\dagger\hat{a}_j-\hat{a}_i\hat{a}_j^\dagger)} \tag{2.5}$$

In order of appearance, the phase shifter, displacement and squeezing operators are acting on one-qumode, and the beam-splitter operator is acting on two-qumodes. These are simple devices from which any linear quantum optical circuit can be assembled. Moreover, any unitary operator can be built using beam-splitters and phase shifters, as an interferometer.

In fact, a pure qumode, i.e. $|n\rangle$ states are technically difficult to produce. The $|1\rangle$ means one photon in a specific qumode. Producing exactly one photon is really a big challenge. For this reason, several easier solutions have been developed to achieve this heavy task [3]. The Gaussian version [12] is particularly important for our research: here, coherent light, viz. a laser, is used as input. A coherent light is nothing more than the sum of countably infinite states for each qumode. For example, if the displacement operator is applied on the vacuum state ($|0\rangle$), the following holds.

$$|\alpha\rangle = e^{-\frac{1}{2}\sum_{n=0}^{\infty}\frac{\alpha_n}{\sqrt{n!}}|n\rangle} \tag{2.6}$$

The operators (2.2), (2.3), (2.4) and (2.5) are also called Gaussian operators because they transform coherent states into coherent states. The general coherent state can be achieved starting with a vacuum state and transformed by any product of Gaussian operators above.

The mathematical apparatus for qumodes requires different tools than that of the one-photon states. Since such a system can be described with a Gaussian distribution, the quantum simulator built in this way is called Gaussian boson sampling [24, 25]. Although not used here, note that this can be further extended to continuous-variable systems. More information on Gaussian states and this topic can be found in many articles [11, 18].

2.2. Vibronic spectra of molecules

Besides the brief description of the vibration spectrum of molecules given here, more detailed descriptions can be found in Huh's thesis [13] and many other sources [15, 21, 27]. One of the main issues of spectroscopy is the examination of the electrical transitions of atoms and molecules, whether they take place between states of the same or different electrical charges. However, with the exception of the hydrogen atom, which can be solved exactly, the quantum mechanical equations become exponentially more complex with the number of electrons, and their calculation becomes expensive or practically impossible. Therefore, approximate methods must be found; here, for instance, the path leading to the method of calculating coherent states with the help of operators acting on them is presented.

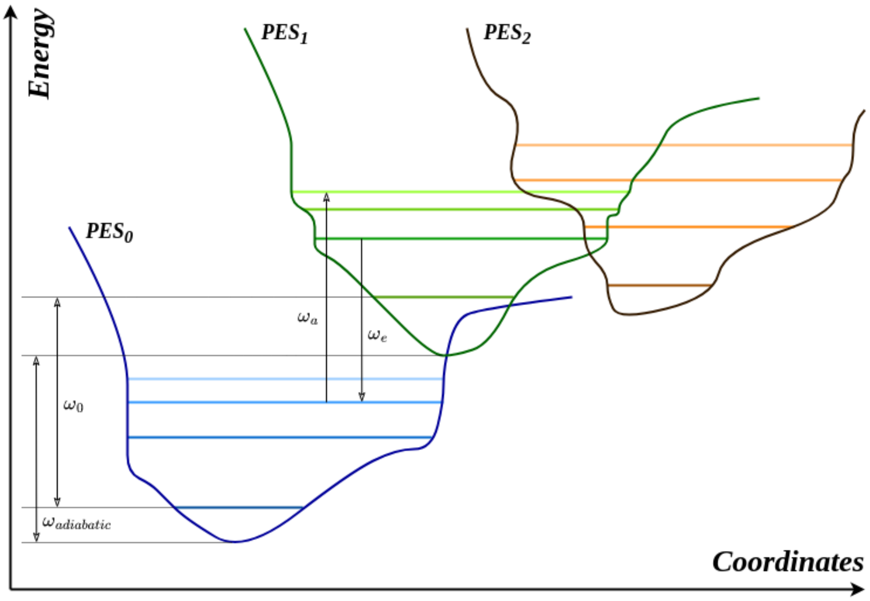


Figure 1. Transitions between molecular states. The curves represent different electronic *potential energy surfaces (PES)*. Transitions are possible between any two PES, however the picture takes into account only the transitions for our interest, i.e. the states of q molecule with different electronic charges. Here ω_a is the angular frequency of one-photon absorption, while ω_e stands for the angular frequency of one-photon emission.

A good approximation of enormously complex system of molecules, if the nuclear and electron states are assumed to be independent, is called the Born-Oppenheimer approximation [2]. Also, due to their mass, atomic nuclei react more slowly to changes in the electron shell, so the nuclei can be considered as a constant background in Franck-Condon principle [6, 9].

With approximating the vibration of electrons with simple quantum harmonic oscillators, one can write the Hamiltonian operator of electrons as a function of the reaction coordinates. Then the potential electron surface can already be described by a parabola.

$$\hat{\mathcal{H}} = \hat{\mathbf{p}}^2 + \hat{\mathbf{q}}^2 \tag{2.7}$$

Here $\hat{\mathbf{q}}$ are the mass-weighted normal coordinates and $\hat{\mathbf{p}}$ are the canonical moments. Duschinsky [7] used another approximation, assuming a linear relationship between the normal coordinates of the different electronic states.

$$\hat{\mathbf{q}}^{'} = \mathcal{U}_{Dusch}\hat{\mathbf{q}} + \hat{\mathbf{d}} \tag{2.8}$$

This introduces the real $\mathcal{U}_{Dusch}$ mixing matrix, as well as the real $\hat{\mathbf{d}}$ displacement operator.

Doktorov and et al. [8] showed that under these conditions, for the states of such linearly connected quantum harmonic oscillators the relation between quantum states $|\phi\rangle$ and $|\phi'\rangle$ of different *PES* can be given in the following way.

$$|\phi'\rangle = U_{\mathrm{Dokt}}|\phi\rangle \tag{2.9}$$

Here the so-called Doktorov-operator is defined as follows.

$$\hat{U}_{Dok} = \hat{D}(\hat{\mathbf{d}})\hat{S}^{\dagger}(\Omega')\hat{R}(U_{\mathrm{Dusch}})\hat{S}(\Omega) \tag{2.10}$$

That is, the Doktorov operator can be written as a product of unitary Gaussian operators, with the help of one-mode squeezings $\hat{S}$, the rotation $\hat{R}$ and the displacement operator $\hat{D}$. The input parameter of the rotation operator is the Duschinsky mixing matrix itself, and the argument of the displacement operator $\hat{D}$ is the Duschinsky displacement from equation (2.8).[1] The squeezings are derived from the physical characteristics of the molecule, specifically Ω and Ω' are the harmonic angular frequencies of the atoms within the molecule in pre- and post-transition states.

The solution of the (2.7) eigenvalue problem is the coherent states in Fock-space. The peculiarity of the Gaussian operators acting on them is that they transform a coherent state to a coherent state.

The transition probabilities – the so-called Franck-Condon factors – between the different electronic states can be approximated using the previous definitions as follows.

$$FCF_{\mathbf{n}',\mathbf{n}} = |\langle\phi'|\phi\rangle|^2 = |\langle\mathbf{n}'|U_{Dokt}|\mathbf{n}\rangle|^2 \tag{2.11}$$

The power of boson sampling is the implementation of a rotation matrix acting on qumodes. In our case of calculating the vibonic spectra, the Doktorov operator (2.10) is used, containing the $\hat{R}(U_{\mathrm{Dusch}})$ rotation operator. Thus, the transition probabilities (2.11) can be obtained by passing a properly prepared coherent state through a boson sampling device, and after applying another squeezing, the modes and thus the factors can be measured.

[1]Note that the Duschinsky displacement added to the normal coordinates and the Gaussian displacement acting on the quantum states are different concepts.

3. Implementation of the experiment

For a Gaussian boson sampling task, building a quantum computer operating on such a principle is of course the most adequate approach. Although this is no longer an impossible task, it is still difficult and expensive. Huh et al. [14] and Wang et al. [22] implemented these measurements in hardware. The former group used a photonic solution, where the rotation operation was provided by an interferometer, but in a modified form, since the photons coming out of the interferometer could be measured directly by the photon number detectors. The other group implemented the calculation of (2.11) with trapped ions (also Shen et al. [20] created such a device), so they could directly implement the resolution of the Doktorov transformation by (2.10) Gaussian operators.

Another possible, albeit limited solution is to create a simulator based on boson sampling which can be executed on a classic computer. In this experiment Piquasso [5], an open-source photonic quantum computer simulator was used. Piquasso provides a domain-specific programming language to describe quantum optical circuits. This language is embedded into Python, and the simulator front-end is also implemented in Python. The back-end of the simulator can either be executed in Python, or, in the case of Piquasso Boost, in C++.

```
with pq.Program() as program:
    for i in range(modes):
        pq.Q(i) | pq.Squeezing(r = pre_transition_squeezing[i])
    pq.Q() | pq.Interferometer(U_duschinsky)
    for i in range(modes):
        pq.Q(i) | pq.Squeezing(r = (-1) * post_transition_squeezing[i])
    for i in range(modes):
        pq.Q(i) | pq.Displacement(a = displacement[i])
    pq.Q() | pq.ParticleNumberMeasurement()

config = pq.Config(cutoff = cutoff, measurement_cutoff = measurement_cutoff)
simulator = pq.GaussianSimulator(d = modes, config = config)
result = simulator.execute(program, shots = shots).samples
```

Figure 2. Code snippet about circuit of Gaussian boson sampling calculating vibronic spectra of molecules.

In Figure 2 the photonic quantum computer code for calculating vibronic spectra can be seen in Piquasso. Here *pq* is the abbreviation of the Piquasso package, and the operators provided by the package are easy and clear to interpret and use. With the help of Piquasso, the circuit seems quite simple, but of course behind the application of every Gaussian operator there is a multiplication of matrices with exponentialized matrices, as one expects due to (2.2), (2.3), (2.4) and (2.5). Because of the infinity power series of exponential, we need to introduce finite cutoffs. The dimensionless arguments of the operators are derived from the physical parameters of the given molecule. To determine the distribution more precisely, several measurements are required, typically a few thousand runs.

The calculation of Franck-Condon factors (2.11) at temperature 0 K, i.e. $|\mathbf{n}\rangle = |\mathbf{0}\rangle$ is calculated on the pre-transition base state. This can be expanded to any

$|\mathbf{n}\rangle$ also for pre-transition modes, using a new one-mode squeezing operator at the beginning of the calculations, see [19, Equation 51].

4. Result

Now the calculations of transitions of three molecules, namely the formic acid (Figure 3), ozone (Figure 4) and nitrogen dioxide (Figure 5) are presented. The calculation of the Franck-Condon factors of the first molecule on a boson sampling-based quantum simulator was performed by Huh et al. [14]. The calculation of the Franck-Condon factors of the other two molecules was carried out by Wang et al. [22], and in their case, simulation was also measured with single-bit extraction. Furthermore, their article contains the values of transition probabilities calculated classically and with master equation as well. These data can be found in the Supplemental Material of their article on arXiv [23]. The physical parameters of the molecules can be found in the above articles, and also in the Nist database [17].

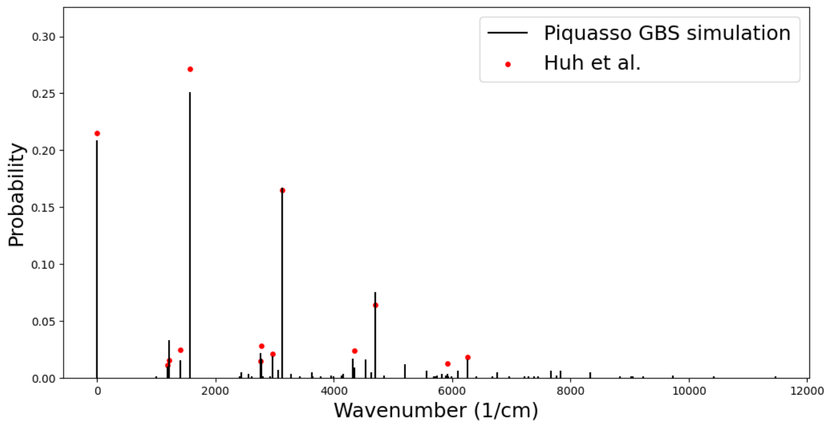


Figure 3. Formic acid $1^1A^{'}$ $(n = 0, m = 0) \longrightarrow 1^2A^{''}$ transition.

The calculation time of transition probabilities with the Piquasso [5] boson sampling-based simulator increases drastically (exponentially) with the increase of the cutoff value. In the case of formic acid, it is sufficient to set the cutoff value for the number of qumodes, but for the other two molecules it is worthwhile to choose the total number of photons that can be measured on the output (two qumodes), as found in the Supplemental Material [23].

In the case of ozone and nitrogen dioxide, which have C_{2v} point group symmetry, a negative anion emits an electron, and the molecule becomes neutral. The formic acid molecule has C_s point group symmetry.

For the convergence of the sampling process, the formic acid case required 1 000 runs of the quantum program, but the other two molecules needed 10 000 runs. The

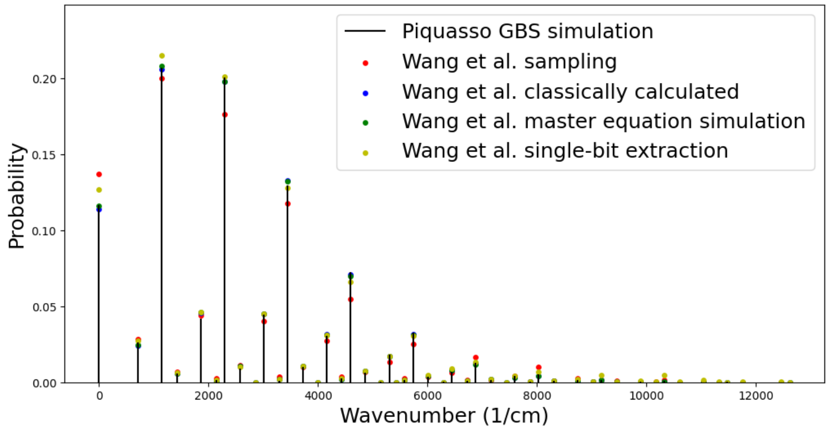


Figure 4. Spectrum of $O_3^-(n = 0, m = 0) \longrightarrow O^3 + e^-$ transition.

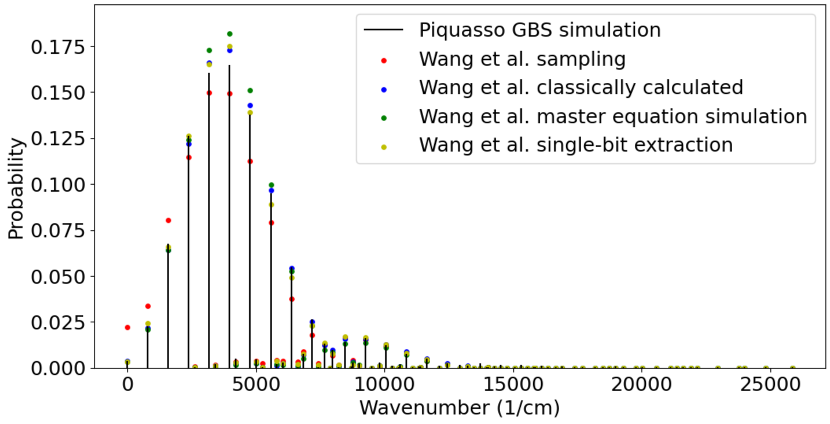


Figure 5. Spectrum of $NO_2^-(n = 0, m = 0) \longrightarrow NO_2 + e^-$ transition.

computations were performed with the Piquasso Python back-end on a standard PC (i7 1.9 GHz with 16 GB memory), and required less then 1 hour.

5. Related work

The Strawberryfields application of Xanadu [26] is another boson sampling based quantum simulator, also written in the Python programming language. This software is a stepping stone for available photonic quantum computers developed by Xanadu. Such a quantum hardware environment does not yet exist behind Piquasso, therefore the latter relies on the resources of classical computers. More information about the Xanadu Strawberryfields simulator can be found in the works of Bromley et al. [4] and Killoran et al. [16].

Similar to the Piquasso simulator, a circuit simulating the vibration spectrum of molecules can also be implemented in Xanadu's simulator. The physical data of formic acid, water and pyrrole molecules can also be found in Strawberryfields, with which the Franck-Condon factors of these molecules can be easily calculated. The simulation results for the formic acid in Piquasso and Strawberryfields perfectly match, but the Piquasso-based implementation is faster, especially with the Piquasso Boost back-end. The difference in execution time was significant for the experiments with 10 000 runs.

6. Conclusion

The circuit developed in the Piquasso photonic quantum computer simulator successfully calculated the Franck-Condon factors of the vibrational spectra of three molecules: formic acid, ozone and nitrogen dioxid, yielding similar results to those found in Huh et al. [14] and Wang et al. [22]. The simulator executed the Gaussian boson sampling algorithm on a classical computer. Although computations for larger molecules would exceed the resources of classical computers, our results show that smaller molecules can be handled with this technology. This makes it possible to experiment with affordable devices in order to further improve the algorithm and possibly the quantum chemical model. Another important consequence of the presented experiment is that programs written for quantum computing devices can be effectively tested in a simulator, hence facilitating the development of such applications.

It is important to note that the disadvantage of a simulator based on Gaussian boson sampling compared to a physically realized quantum computer built on a similar principle is its finite calculation capacity, i.e. the application of a cutoff is necessary to limit the maximum value of the qumodes in the case of the former, while the latter is only hindered by the limitations of the detectors.

There are many open research possibilities in the topic: testing the existing algorithm, extending it to calculate new transitions of existing molecules, but also to extending it to additional molecules. Furthermore, the limits of the simulators that can be run on classical computers can be measured, either depending on the number or the maximum value of the qumodes. In the more distant future, an automated search for suitable molecules with transitions optimized for a given task is conceivable.

References

[1] S. Aaronson, A. Arkhipov: *The Computational Complexity of Linear Optics*, Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing 24.10 (2011), pp. 333–342, doi: 10.1145/1993636.1993682.

[2] M. Born, R. Oppenheimer: *Zur Quantentheorie der Molekeln*, Annalen der Physik 389.20 (1927), pp. 457–484, doi: 10.1002/andp.19273892002.

[3] D. J. Brod, E. F. Galvão, A. Crespi, R. Osellame, N. Spagnolo, F. Sciarrino: *Photonic implementation of boson sampling: a review*, Advanced Photonics 1.3 (2019), p. 034001, doi: 10.1117/1.AP.1.3.034001.

[4] T. R. Bromley, J. M. Arrazola, S. Jahangiri, J. Izaac, N. Quesada, A. D. Gran, M. Schuld, J. Swinarton, Z. Zabaneh, N. Killoran: *Applications of near-term photonic quantum computers: software and algorithms*, Quantum Science and Technology 5.3 (May 2020), p. 034010, doi: 10.1088/2058-9565/ab8504.

[5] Budapest Quantum Computing Group: *Piquasso*, https://github.com/Budapest-Quantum-Computing-Group/piquasso, 2022.

[6] E. Condon: *A Theory of Intensity Distribution in Band Systems*, Phys. Rev. 28 (6 Dec. 1926), pp. 1182–1201, doi: 10.1103/PhysRev.28.1182.

[7] F. Duschinsky: *Acta Physicochim*, URSS 7 (1937), pp. 551–566.

[8] D. E, M. I, M. V: *Acta Physicochim*, Journal of Physics B: Atomic and Molecular Physics 9 (1976), p. 507.

[9] J. Franck, E. G. Dymond: *Elementary processes of photochemical reactions*, Trans. Faraday Soc. 21 (February 1926), pp. 536–542, doi: 10.1039/TF9262100536.

[10] B. T. Gard, K. R. Motes, J. P. Olson, P. P. Rohde, J. P. Dowling: *An Introduction to Boson-Sampling*, in: From Atomic to Mesoscale, WORLD SCIENTIFIC, June 2015, pp. 167–192, doi: 10.1142/9789814678704_0008.

[11] L. Hackl: *Aspects of Gaussian States Entanglement, Squeezing and Complexity*, in: 2018.

[12] C. S. Hamilton, R. Kruse, L. Sansoni, S. Barkhofen, C. Silberhorn, I. Jex: *Gaussian Boson Sampling*, Physical Review Letters 119.17 (Oct. 2017), doi: 10.1103/physrevlett.119.170501.

[13] J. Huh: *Unified description of vibronic transitions with coherent states*, 2011, url: http://publikationen.stub.unifrankfurt.de/frontdoor/index/index/docId/21033.

[14] J. Huh, G. G. Guerreschi, B. Peropadre, J. R. McClean, A. Aspuru-Guzik: *Boson Sampling for Molecular Vibronic Spectra*, Nature Photonics 9 (2014), pp. 615–620, doi: 10.1038/nphoton.2015.153.

[15] J. Huh, M. Neff, G. Rauhut, R. Berger: *Franck–Condon profiles in photodetachment-photoelectron spectra of HS_2^- and DS_2^- based on vibrational configuration interaction wavefunctions*, Molecular Physics 108.3-4 (2010), pp. 409–423, doi: 10.1080/00268970903521178.

[16] N. Killoran, J. Izaac, N. Quesada, V. Bergholm, M. Amy, C. Weedbrook: *Strawberry Fields: A Software Platform for Photonic Quantum Computing*, Quantum 3 (2019), p. 129, issn: 2521-327X, doi: 10.22331/q-2019-03-11-129.

[17] National Institute of Standards and Technology: *NIST Chemistry WebBook, SRD 69*, url: https://webbook.nist.gov/.

[18] K. Parthasarathy: *What is a Gaussian state?*, Communications on Stochastic Analysis 4 (June 2010), doi: 10.31390/cosa.4.2.02.

[19] N. Quesada: *Franck-Condon factors by counting perfect matchings of graphs with loops.* The Journal of chemical physics 150 16 (2018), p. 164113, doi: 10.1063/1.5086387.

[20] Y. Shen, Y. Lu, K. Zhang, J. Zhang, S. Zhang, J. Huh, K. Kim: *Quantum optical emulation of molecular vibronic spectroscopy using a trapped-ion device*, Chem. Sci. 9 (4 2018), pp. 836–840, doi: 10.1039/C7SC04602B.

[21] A. Toniolo, M. Persico: *Efficient calculation of Franck–Condon factors and vibronic couplings in polyatomics*, Journal of Computational Chemistry 22.9 (2001), pp. 968–975, doi: 10.1002/jcc.1057.

[22] C. S. Wang, J. C. Curtis, B. J. Lester, Y. Zhang, Y. Y. Gao, J. G. Freeze, V. S. Batista, P. H. Vaccaro, I. L. Chuang, L. Frunzio, L. Jiang, S. M. Girvin, R. J. Schoelkopf: *Efficient Multiphoton Sampling of Molecular Vibronic Spectra on a Superconducting Bosonic Processor*, Physical Review X (2019), doi: 10.1103/PhysRevX.10.021060.

[23] C. S. Wang, J. C. Curtis, B. J. Lester, Y. Zhang, Y. Y. Gao, J. G. Freeze, V. S. Batista, P. H. Vaccaro, I. L. Chuang, L. Frunzio, L. Jiang, S. M. Girvin, R. J. Schoelkopf: *Efficient Multiphoton Sampling of Molecular Vibronic Spectra on a Superconducting Bosonic Processor*, https://arxiv.org/abs/1908.03598v2, 2019.

[24] X. Wang, T. Hiroshima, A. Tomita, M. Hayashi: *Quantum information with Gaussian states*, Physics Reports 448.1-4 (Aug. 2007), pp. 1–111, doi: 10.1016/j.physrep.2007.04.005.

[25] C. Weedbrook, S. Pirandola, R. García-Patrón, N. J. Cerf, T. C. Ralph, J. H. Shapiro, S. Lloyd: *Gaussian quantum information*, Reviews of Modern Physics 84.2 (May 2012), pp. 621–669, doi: 10.1103/revmodphys.84.621.

[26] XanaduAI: *Strawberryfields*, https://github.com/XanaduAI/strawberryfields, 2018.

[27] Z.-l. Yang, Z. Zhang, S. Jiang, Y.-j. Feng, J. Liang, W. Huang: *Calculation of Franck–Condon factors and simulation of photoelectron spectra of the HCCl−anion: Including Duschinsky effects*, Journal of Electron Spectroscopy and Related Phenomena 211 (2016), pp. 41–46, issn: 0368-2048, doi: 10.1016/j.elspec.2016.06.003, url: https://www.sciencedirect.com/science/article/pii/S0368204816300718.

Annales Mathematicae et Informaticae
58 (2023) pp. 133–146
DOI: 10.33039/ami.2023.08.014
URL: https://ami.uni-eszterhazy.hu

Three level benchmarking of Singularity containers for scientific calculations

Péter Polgár, Tamás Menyhárt, Csanád Bátor Baksay, Gergely Kocsis, Tibor Gábor Tajti, Zoltán Gál[*]

University of Debrecen, Faculty of Informatics,
Debrecen, Hungary
kocsis.gergely@inf.unideb.hu
tajti.tibor@inf.unideb.hu

Abstract. In this work we present our results from benchmarking Singularity containers running scientific calculations intended for HPC at three levels of complexity, comparing them against native software environments. Our investigations were run on up-to-date hardware and the latest available software as of early 2023. To get a more detailed picture, we examined three different system aspects: we ran separate benchmarks on CPU, memory, and I/O intensive operations. These aspects were tested with microbenchmarks, and at a high level – an HPC workload pipeline which intensely loaded each aspect. As a result of our investigations we show that Singularity containerization continues to provide as good, or in some cases even better performance indicators as in a native environment – with the substantial added benefits of application flexibility and portability.

Keywords: Container benchmarking, Singularity, HPC

AMS Subject Classification: Computer Science 68U01, 68U99

1. Introduction

The tools and methods of scientific computing undergo continuous progress, providing opportunities for researchers to achieve more sophisticated and increasingly

[*]This work has been supported by the project TKP2021-NKTA of the University of Debrecen. Project no. TKP2021-NKTA-34 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the TKP2021-NKTA funding scheme.

Submitted: July 28, 2023
Accepted: August 7, 2023
Published online: September 9, 2023

accurate results using better and better facilities. In the meantime, however, history has shown that following new trends in tools and methodologies very often leads to software compatibility issues. It is also a common requirement to be able to run calculations or simulations in differing hardware and software environments. Today's answer to these issues is the use of containerization [3]. By the use of containers, researchers can build uniform software environments, sealed in portable disk images files, and run their simulations or calculations without any regard to the details of the real host's software setup. From this perspective, containers work similar to virtual machines, however, most findings have shown that they operate with much smaller overhead, increasing their efficiency. In the case of scientific calculations, Singularity (nowadays known as Apptainer or SingularityCE – of which we used the latter) is the most common containerization solution used to overcome incompatibility issues between environments. To prove this evident and conventionally accepted statement, we have checked the first 172 HPC providers from the Top500 [15] list to find out what containerization technologies they support. As one can see in Figure 1, 48 out of 172 observed providers published this information on their homepages. Note that while 3/4 of them support only Singularity and/or Apptainer, there are only about 10% which support other containerization technologies, but not Singularity or Apptainer. However, the question continues to arise in an ongoing manner, whether containerization has a performance impact on calculations, and if yes, of what sort?

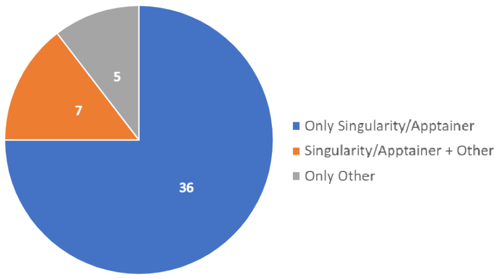


Figure 1. Containerization technologies supported by 48 Top500 HPC providers. Note the majority supports exclusively Singularity and/or Apptainer.

The overall structure of this paper from here on is as follows: In Section 2 we describe the details and the methodology of our investigations, first by presenting the employed hardware and OS configurations (Subsection 2.1) and then by showing the codes, tools and methodology on each level (Subsections 2.2, 2.3, 2.4 and 2.5). Section 3 presents our results at these respective levels. Thus Subsection 3.1 is for our results on level L1, subsection 3.2 is for level L2, while subsection 3.3 shows our results on the topmost level L3. On level L1 and L2 we sort our results regarding different computational resources into separate subsections. Finally, in Section 4 we conclude our results and in Section 5 we present our plans for our related future work.

2. Problem formulation and applied methodology

Since the rise of containerization technology, several investigations have been conducted to describe it's pros and cons. According to [1, 4, 6, 10] CPU and memory performance of containers are about the same or slightly worse compared to the performance of the underlying host system. While in most cases these investigations have shown that the use of containers does not have a significant impact on performance (in some cases they can even beat performance measures of bare metal environments) [2, 3], it is also clear that details change as new versions of underlying hardware and software systems are released. For this reason, in this work we compare scientific calculations of three levels of complexity on both bare metal and in containerized environments, from three different aspects (CPU, memory, I/O) to find out whether actual setups (in early 2023) changed the details of these results.

For the first level (L1), we do “micro-benchmarking” by repeatedly running large amounts of simple calculations. On this level we can easily find the roots of performance strengths or weaknesses of Singularity containers or native environments. The other benefit of this level is that it is very easy to scale these tests to study the size-dependence of time overheads.

Our second level of benchmarking (L2) uses well-known benchmark tools [3, 16] (listed in Subsection 2.3). This level gives us a basis of comparison against others' results. In our current work we focus on CPU, memory, and I/O intensive calculations and tools, since results from this are easy to compare to L1.

As our topmost level (L3) we run an existing pipeline of social-sciences-related scientific calculations, intended for HPC. The exact results and purpose of these calculations don't concern us, but the run-times and other performance indicators give us an understanding of how significantly the behavior of real computational scenarios differs from microbenchmarks and well-known benchmarking tools.

Note that an infinite multitude of measurements can be run with any number of results. Our point is not to declare a winner, but to establish that container portability is not at the price of performance degradation.

2.1. Specification of the applied hardware

We chose one of the most modern available laptop computers, and an LTS Linux version as the hardware and software environments for our tests, so we can say that our results to be found are valid for systems of late 2022 or early 2023. For a detailed description see Table 1 and Table 2.

2.2. About the used codes at level L1

The primary purpose of L1 measurements is to execute simple, often elementary code, repeatedly, targeting a specific piece of hardware or part of it (e.g., an on-processor cache) to measure performance. It is important to note that we cannot speak of a pure processor-memory-storage or video card comparison, because usually a given resource cannot work without the others, but still the main resource

Table 1. Specification of the used hardware.
Date of measurements: 2022–2023.

Hardware	Type / Specifications
CPU	Intel i7-12700H, 14 core, 20 threads, 5 GHz turbo
Memory	32 GB, DDR5, 4800 MHz
Storage	500 GB, Samsung 980 PRO, 7 GB/s, PCIe 4.0

Table 2. Specification of the used software.
Date of measurements: 2022–2023.

Software / property	Native (host)	Container
Userspace	22.10 UwUntu	22.04 Ubuntu
Kernel version	5.15.0.60-generic	—
File system	ext4, ext3 image	can use host's ext4, ext3 overlay
Free space	ext4 host: ~200GB	ext3 overlay: ~70GB
gcc / g++ version	Ubuntu 11.3.0	Ubuntu 11.3.0

can be identified. In addition to the test codes, there are other processes running on the system, and the operating system (although very well optimized), is still running in the background. Because of this, all measurements were performed several times to reduce the chance of measurement error.

The majority of the code is presented without optimization. When optimized, the run time for computation- and memory-intensive processes is significantly reduced in both native and container contexts. Note also that both the container and the native system used exactly the same kernel and compiler. In an HPC environment, it is not common to update these to the current release, so our code may not work with the software installed on the HPC system, or may not compile, or run much slower without certain switches. But, even if not optimized, computationally intensive code will still run significantly faster with a newer gcc or g++ compiler. It is worth using containers precisely because these problems can be completely eliminated, and CPU time is our real time and money.

2.3. Used benchmark tools at L2

In order to have results that are comparable to the findings of other works, we used existing benchmarking tools to test the performance of the native and containerized jobs. We refer to these tools as the standard benchmark tools. The applied so-called standard benchmark tools at L2 are the following:

CPU benchmarks: 7-Zip Compression 22.01 [4, 9, 11, 12]; XZ compression 5.2.4 [11]; CppPerformanceBenchmarks 9; LAME MP3 Encoding 3.100 [11]; FFTW 3.3.6; Geekbench 5.4.6 [7]; Glibc Benchmarks; Himeno Benchmark 3.0; Timed MAFFT Alignment 7.471 [11]; ACES DGEMM 1.0; libjpeg-turbo tjbench 2.1.0

Memory benchmarks: CacheBench [11]; MBW 2018-09-08 [9]; RAMspeed SMP 3.5.0 [9, 11, 12]; STREAM 2013-01-17 [11]; sysbench 1.0.20 [1]; Tinymembench 2018-05-28 [9]

Storage I/O benchmarks: Bonnie++ 2.00 [10]; fio 3.28 [5]; FS-Mark 3.3 [6]; IOzone 3.465 [4, 11, 13]; PostMark 1.51 [9, 14]; sysbench 1.0.20 [1]

Most of these benchmarks were run by the Phoronix Test Suite [11] (PTS), with the exceptions of: Geekbench; sysbench; Bonnie++; and fio. PTS ran each benchmark at least three times, and it's showed result is the average of these results.

2.4. About the used storage I/O benchmarking methodology

Files can be stored in different ways both in native and container environments. We examined three storage options for native environments:

1. using the native (host) filesystem (ext4);
2. using an ext3 image on the host filesystem;
3. using `/dev/shm`.

We found that in Singularity container environments, standard benchmark tools for storage I/O can only be run with one of the following four filesystem options:

1. using the native (host) filesystem (ext4);
2. using an external ext3 image overlay;
3. using an ext3 overlay embedded in the SIF container image;
4. using `/dev/shm`.

`/dev/shm` is a tmpfs memory-backed filesystem, whose maximum size is usually the half of the overall memory size.

For all storage I/O benchmarks there was enough free space on storage to ensure that no significant performance degradation would occur due to insufficient free space being available.

The above options are not all the possible options, however, some were left out, e.g., directory overlay, or tmpfs overlay, due to technical requirements which may not be satisfied on HPC systems, such as needing root permissions. We also didn't examine sandbox-type containers because sandboxes are intended for development purposes.

2.5. About the used pipeline code at L3

For the top level of our investigation, we used a real pipeline of network science C codes [8, 17]. The abstract structure of this pipeline is presented in Figure 2. While the exact purpose of the code is not important, in our case the significance is that load is applied to all of the resources of interest. For example, generating and "attacking" the networks is a highly CPU and memory intensive task, and between the pipeline stages, the network states are serialized to- and deserialized from storage.

Our example pipeline is just one possible scenario from those that were applied in the referred research [8, 17]. We found it important to do these investigations beside the other two levels in order to have a look how the somewhat artificial benchmarking results are related to real-world applications.

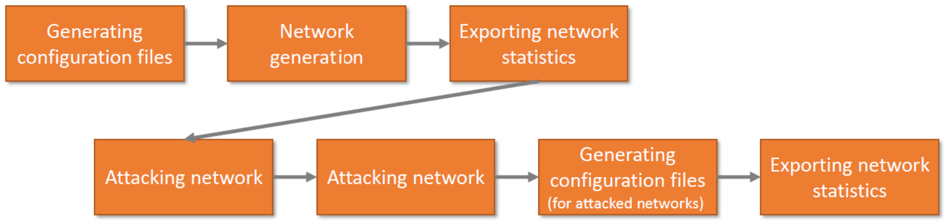

Generating configuration files
Network generation
Exporting network statistics
Attacking network
Attacking network
Generating configuration files (for attacked networks)
Exporting network statistics

Figure 2. The example social network analysis pipeline used to test the performance of native and containerized jobs [8, 17].

3. Results

3.1. Custom microbenchmarking (L1)

The results of our investigations on the lowermost level L1 are summed up in Table 3. In the following subsections we consider the three main system resources and interpret the corresponding findings for both environments as presented in the table. While at first sight it is clear that both the native and containerized environments have strengths, it also has to be noted that the differences are small.

3.1.1. CPU intensive benchmarks (L1)

There are several measurements of a processor's performance because there are many ways to measure its speed, and the result depends on the type of measurement. The measurements therefore are split into the following subcategories:

Processor threads: Single-threaded operation / Multi-threaded operation (with dynamic and static allocation!).
Data structure perspective: Computing with primitives / We use complex data structures.
Cache aspects: Level 1, 2 and 3.
Optimization: compiler flag O2, O3 / O3+ or Ofast / without optimization.

We mainly focus on binaries running on one thread without optimization. The container performed better with operations on whole numbers. No difference was observed for simpler data structures such as char arrays (strings). A Dijkstra pathfinding algorithm was also created from our own implementation. It mostly used the level 3 processor cache for data storage, and did not use non-integer numbers. The native operating system seemed a bit faster in this case.

Table 3. Benchmarking results on level L1. Colored cells mark more than 1% difference.

Microbenchmarks: (Elapsed Time)	Native (s)	Container (s)	Efficiency (%)	Main Hardware Component	file name
whole numbers	540,50	527,41	102,48%	CPU	whole.c
string formatting	473,43	472,92	100,11%	CPU, CPU CACHE	strformat.c
sqrt	413,95	433,72	95,44%	CPU	sqrt.c
sin, cos	378,36	374,84	100,94%	CPU	sincos.c
log	421,33	421,53	99,95%	CPU	log.c
matrix-Dijkstra algorithm	7406,86	7418,63	99,84%	CPU, CPU CACHE	dijkstra.cpp
Memory handling (large size)	299,08	298,95	100,05%	RAM	filereadmem.c
file writing (on ext4 host)	105,73	107,03	98,78%	SSD	filewrite.c
file reading (on ext4 host)	26,45	23,80	111,13%	SSD	fileread.c
file copy (on ext4 host)	22,43	22,02	101,86%	SSD	filecopy.sh
file writing (on ext3 overlay)	124,07	130,59	95,01%	SSD	filewrite.c
file reading (on ext3 overlay)	26,75	25,94	103,09%	SSD	fileread.c
file copy (on ext3 overlay)	33,20	22,31	148,79%	SSD	filecopy.sh

The real differences come in with floating point numbers. While one could do an almost infinite number of measurements using various mathematical functions, we preferred calculating square roots, means, sines, natural-based logarithms. In the case of the square roots, the native performed better, while in the other cases the two were considered equal, or the container was slightly better.

We found that an important thing is to increase optimization, because an optimization of at least O3 can reduce a floating point computation by up to a quarter. Note that when we optimize the root mean square, we found a difference of less than 1%. If we used a 2 years older gcc version, the runtime deteriorated by about 20%. (Which is common, especially if you want to run it on another machine or HPC where they do not update the compilers to the latest version.) So it is more important to keep the software in the container as up to date as possible, because that already has a significant impact on performance.

Summing up the results one can say that CPU usage produces results within

5% error margin, which can become an overwhelming advantage for the container if more advanced software is installed for it in terms of operations on one thread at the L1 benchmark level.

3.1.2. Memory benchmarks (L1)

Since DDR5 memory is so performant, even if there is a large amount of it, it is hard to measure with precision, thus large numbers of large memory allocations, reads, copies and free instructions were performed. In practice, a large part of the memory was allocated, overwritten and freed 256 times. The point was that the amount of data should not fit into the cache of the processor, but also should not be too big in order to avoid swapping. The difference was barely 0.05%, in favour of the container. The conclusion is that, even from the worst perspective, the container handles and uses memory as well as the native operating system at the L1 benchmark level.

3.1.3. Storage I/O benchmarks (L1)

Background storage can also be measured in many ways. At the L1 benchmark level, we look at sequential writes, reads and copies. The container can use the host's ext4 file system or its own custom ext3 (block size 4096kB) overlay file with a removable .img extension. This overlay file can also be used by the host by creating a folder and mounting it. An overlay is useful because you can, for example, install programs into it that the container can use at runtime. The container can use both the host and the overlay filesystem at the same time.

Note that the measurements were made using a PCIe 4.0 SSD drive, which is an order of magnitude faster than a regular SATA III M.2 SSD or any HDD. We performed sequential read, write and copy. (Of course, there is no regularity in the file contents.) The file size was roughly 18 GB.

Generally speaking:
Write: the container is about 2–5% slower than the host
Read: the container is about 3–11% faster than the host
Copying: the container is about 0.5% faster than the host

The host, when it wanted to copy in the overlay, was significantly slower than when the container performed the operation on its own file. For the container it took 22.31 seconds to copy the data on its ext3 filesystem while the same took 22.43 seconds for the host to over its ext4 filesystem. This is only 0.5% of difference. The root of this difference is that ext3 filesystem performs worse than on the ext4 and not the fact that we are using a container.

Of course, there are many ways to read/write copy. Multiple files can be copied in parallel, or many small files in succession, or many large files at once, etc. But for us only writing performed worse in the L1 benchmarks for containers. Of all the benchmarks, storage yielded the largest differences, but the differences are still small, not orders of magnitude. There are seven ways to manage files for the container (many of these ways are presented in section 2.4), so you can choose the most appropriate one depending on the intended use-case.

3.1.4. Conclusions (L1)

Summarizing our results on level L1, we found that the advantage of container portability does not come at the cost of sacrificing performance, as most differences were within measurement error. Regarding memory they are identical. Regarding CPU, the host performs slightly better, but the difference is not radical. Compile-time optimization, however, has a huge importance and the use of state-of-the-art software and compilers in the container is recommended. A gcc or g++ that is only 2 years old, for example, will cause a 20% performance degradation for computationally intensive operations.

3.2. Standard microbenchmarking (L2)

On L2 we present our findings obtained through the well-known benchmarking tools presented in Subsection 2.3. Since there is insufficient space to consider all of our data, we excerpt here only the most interesting but still sufficiently detailed portions. The full dataset can be found in the supplementary material of this work.

3.2.1. CPU intensive benchmarks (L2)

The portion of CPU benchmarks where the differences between native and container environment results are less than or equal to 1% is 82.5% (in this context the difference means how proportionally better one was compared to the other). We consider these differences to be within a margin of error, and in 42.42% of the benchmarks the container won. Where the differences between native and container results were greater than 1%, the differences are these in ascending order: 1.33%, 1.33%, 2.08%, 2.17%, 3.43%, 4.54% and 14.33%. It can be seen that these differences are less than 5% except one case which is 14.33%. The benchmarks producing these deviations were rerun twice. These reruns showed less than 1% differences between the environments in most cases, and the used benchmarks did not consistently judge a given environment type as the faster one. The only exception to this being the 7-Zip decompression benchmark, which reliably showed the native environment performing better, but by only 0.68% and 1.25% (the original run was 3.43%).

Thus the conclusion is that all differences in these benchmark results are within a margin of error, and our benchmarks could not significantly differentiate the two environments.

3.2.2. Memory benchmarks (L2)

The portion of memory benchmarks where the differences between native and container results are less than or equal to 1% is 86.36% (in this context the difference means how proportionally better one was compared to the other). These less than or equal to 1% differences are considered within a margin of error, and in 52.63% of the benchmarks the container won. Where the differences between native and container results were greater than 1%, the differences are these in ascending order: 1.17%, 1.21% and 1.45%. It can be seen that these differences are less than 2%. The benchmarks which produce these differences were rerun twice. These reruns showed less than 1% differences in two of three benchmarks, and the used benchmarks did not consistently judge a given environment type as the faster one.

The conclusion is that all differences in the results are within a margin of error, and the portion of container wins was similar to the portion of native wins.

3.2.3. Storage I/O benchmarks (L2)

Based on our measurement results we can conclude the following:

1. Bonnie++ is unreliable for comparing native and container environments.
2. With Fio's asynchronous non-buffered reading, ext3 container overlay (both external and embedded) wins over host ext4 and native mounted ext3 image.
3. With Fio's reading a file backward, ext3 overlay (both container and native) wins over host ext4 filesystem (both container and native).
4. With FS-Mark sync, IOzone and PostMark benchmarks, host ext4 filesystem is similar to or wins over ext3 overlay (both container and native).
5. With sysbench, host ext4 was definitely better than ext3 container overlays.
6. Except reading a file backward, in everything else native host ext4 was far better than native mounted ext3 image.

Since there are many different filesystem options in both native and container environments and each option has its own advantages, further research is needed to determine which native or container filesystem options or a mix of these are the best for the most common use cases, especially for HPC environments.

3.2.4. Conclusions (L2)

During our investigations we ran 25 different tools and got 103 different results (together with all storage runs in fact 447). To get insight on the overall picture of this large amount of results, we have plotted them on Figure 3. On the figure one can see for all tests a percentage value showing whether containerized or native runs performed better. Note that since in some cases a lower measure is better while in other cases the bigger one, this value cannot be simply written as a fraction of the results coming from native and containerized runs. Here we do not take into account the exact type of the benchmarking, our aim is only to see how native and containerized jobs behave typically. On the figure ">100" means that the difference between the results were more than 100% for either one or the other direction, but the exact value has not been plotted here in order to keep the figure informative for smaller results.

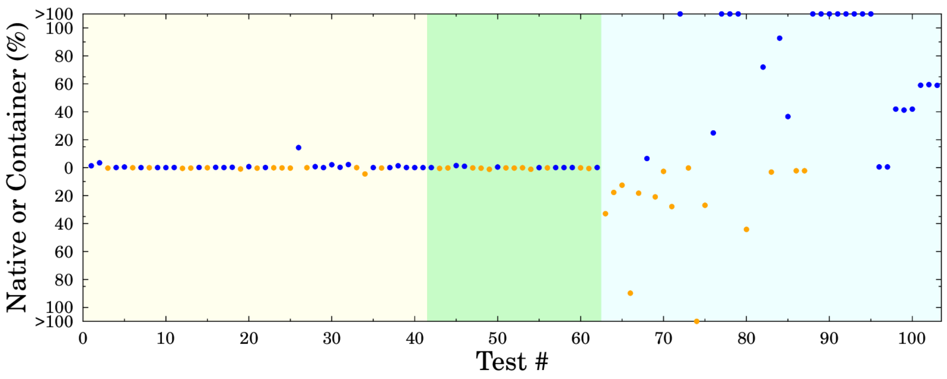


Figure 3. A summary of (L2) benchmarking results. Blue means native wins over containerized runs, orange means the opposite.

Note that in the case of memory related benchmarking we barely found any differences in performance just as for L1. Some CPU related runs, however, resulted in showing that there are cases where native (blue) or containerized (orange) runs are better. The number of cases, however, on each side is almost equal. We found the most diverse results in the case of storage I/O related benchmarking, and here also we see instances on both sides, meaning that it depends highly on the type of benchmark if containerization has a negative or positive effect on the performance. The final message of these investigations should be that one has to know well the required resources of their jobs in order to take the best advantages of containerization – but generally we can say once again that container technology by itself does not require more resources than running natively.

3.3. Benchmarking via a real scientific pipeline (L3)

The used L3 pipeline mostly consists of sequentially running C programs that use each other's inputs and outputs for calculations. These are mainly CPU intensive operations, but they also use memory and storage I/O, e.g., writing out and reading back the results of the steps. Here, along with comparing native and container performance of the unoptimized pipeline binaries, we also compared against optimized binaries, as well as comparing old and new versions of gcc. The results are summed up on Figure 4.

Here, the container has a gcc version released in 2020, while the host has the latest version of 2022. Note that the code ran longer in the older container when not optimized, but the optimization almost completely eliminated the differences (Figure 4 *(left)*). This means that it is worthwhile to install as much up-to-date software as possible in the container to make the runtime sufficiently short and to optimize the code, as this can reduce the runtime of the code by an order of magnitude and almost completely eliminate the differences between the non-matching gcc versions.

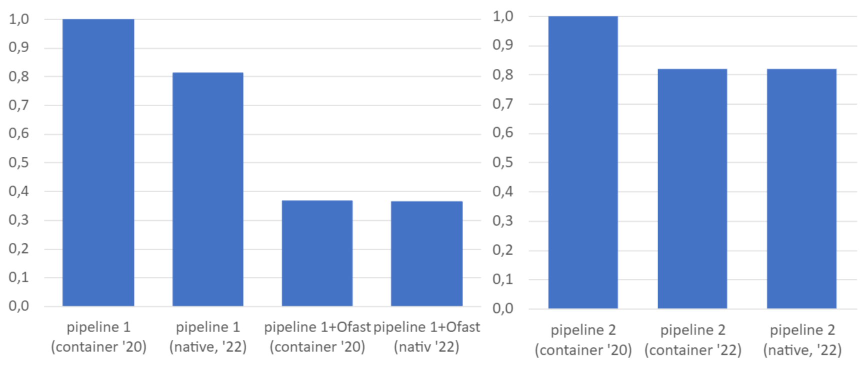


Figure 4. Runtime ratios of a real scientific pipeline. *(left)* shows the obvious profit of optimizing the code during compilation. *(right)* shows how the use of the latest OS can affect the performance.

On Figure 4 *(right)* we ran the pipeline 2 program with the state-of-the-art tools in both cases, but for comparison we also ran it using an older software setup ('20). When we used the old gcc, the runtime increased by 22%. So again the up-to-dateness of the software has more impact on runtime than whether we are looking at a host or native case for L3 Pipeline benchmarks.

4. Conclusions

As the results of our studies we have shown that the use of Singularity containers for providing uniformized, portable environments for scientific calculations does not affect the running times of the calculations in an unaffordably negative way. In some cases the use of containers can even beat the performance of native software environments. We ran our measurements at three levels of complexity. On L1 and L2 we used separate benchmarks for CPU, memory and storage I/O intensive testing. As a result of our findings we have shown with L1 and L2 that there is almost no difference in performance of native and containerized runs of calculations from the aspect of memory intensive tasks. In the case of CPU intensive tasks we found some differences, but most cases showed less than a 5% difference in performance. Storage I/O intensive runs proved to yield the most diverse results, so applications must be careful in choosing solutions appropriate for their workloads.

5. Further work

We have plans for further investigations of the I/O subsytem, and in the direction of parallelization, which we did not study deeply in this work, but is a key component of HPC computations.

Supplementary material

Supplementary material for this work can be found at the below link:
https://arato.inf.unideb.hu/kocsis.gergely/icai2023/

References

[1] S. Abraham, A. K. Paul, R. I. S. Khan, A. R. Butt: *On the Use of Containers in High Performance Computing Environments*, in: 2020 IEEE 13th International Conference on Cloud Computing (CLOUD), 2020, pp. 284–293, doi: 10.1109/CLOUD49709.2020.00048.

[2] C. Arango, R. Dernat, J. Sanabria: *Performance Evaluation of Container-based Virtualization for High Performance Computing Environments*, Microsoft Research WA 98052 (2005).

[3] N. G. Bachiega, P. S. L. Souza, S. M. Bruschi, S. d. R. S. de Souza: *Container-Based Performance Evaluation: A Survey and Challenges*, in: 2018 IEEE International Conference on Cloud Engineering (IC2E), 2018, pp. 398–403, doi: 10.1109/IC2E.2018.00075.

[4] R. K. Barik, R. K. Lenka, K. R. Rao, D. Ghose: *Performance analysis of virtual machines and containers in cloud computing*, in: 2016 International Conference on Computing, Communication and Automation (ICCCA), 2016, pp. 1204–1210, doi: 10.1109/CCAA.2016.7813925.

[5] J. Bhimani, J. Yang, Z. Yang, N. Mi, Q. Xu, M. Awasthi, R. Pandurangan, V. Balakrishnan: *Understanding performance of I/O intensive containerized applications for NVMe SSDs*, in: 2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC), 2016, pp. 1–8, doi: 10.1109/PCCC.2016.7820650.

[6] T. A. Bodhanya: *Comparing Cloud Orchestrated Container Platforms : Under the lenses of Performance, Cost, Ease-of-Use, and Reliability*, MA thesis, Uppsala University, Department of Information Technology, 2022, p. 31.

[7] J. Hadley, Y. Elkhatib, G. Blair, U. Roedig: *MultiBox: Lightweight Containers for Vendor-Independent Multi-cloud Deployments*, in: Embracing Global Computing in Emerging Economies, ed. by R. Horne, Cham: Springer International Publishing, 2015, pp. 79–90, isbn: 978-3-319-25043-4.

[8] G. Kocsis, I. Varga: *Investigating the effectiveness of advertising on declining social networks*, Creative Mathematics and Informatics 23.5 (2014).

[9] M. Lindström: *Containers & Virtual machines : A performance, resource & power consumption comparison*, 2022.

[10] P. E. N, F. J. P. Mulerickal, B. Paul, Y. Sastri: *Evaluation of Docker containers based on hardware utilization*, in: 2015 International Conference on Control Communication & Computing India (ICCC), 2015, pp. 697–700, doi: 10.1109/ICCC.2015.7432984.

[11] Y. Pan, I. Chen, F. Brasileiro, G. Jayaputera, R. Sinnott: *A Performance Comparison of Cloud-Based Container Orchestration Tools*, in: 2019 IEEE International Conference on Big Knowledge (ICBK), 2019, pp. 191–198, doi: 10.1109/ICBK.2019.00033.

[12] A. M. Potdar, N. D G, S. Kengond, M. M. Mulla: *Performance Evaluation of Docker Container and Virtual Machine*, Procedia Computer Science 171 (2020), Third International Conference on Computing and Network Communications (CoCoNet'19), pp. 1419–1428, issn: 1877-0509, doi: 10.1016/j.procs.2020.04.152, url: https://www.sciencedirect.com/science/article/pii/S1877050920311315.

[13] A. Putri, R. Munadi, R. Negara: *Performance analysis of multi services on container Docker, LXC, and LXD*, Bulletin of Electrical Engineering and Informatics 9.5 (2020), pp. 2008–2011, ISSN: 2302-9285, DOI: 10.11591/eei.v9i5.1953, URL: https://beei.org/index.php/EEI/article/view/1953.

[14] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, L. Peterson: *Container-Based Operating System Virtualization: A Scalable, High-Performance Alternative to Hypervisors*, SIGOPS Oper. Syst. Rev. 41.3 (Mar. 2007), pp. 275–287, ISSN: 0163-5980, DOI: 10.1145/1272998.1273025.

[15] E. Strohmaier, J. Dongarra, H. Simon, M. Meuer: *The 500 most powerful commercially available computer systems*, 2022, URL: https://www.top500.org/.

[16] A. Torrez, T. Randles, R. Priedhorsky: *HPC Container Runtimes have Minimal or No Performance Impact*, in: 2019 IEEE/ACM International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC), 2019, pp. 37–42, DOI: 10.1109/CANOPIE-HPC49598.2019.00010.

[17] I. Varga: *Weighted multiplex network of air transportation*, European Physical Journal B 89.6 (2016).

Annales Mathematicae et Informaticae
58 (2023) pp. 147–159
DOI: 10.33039/ami.2023.08.007
URL: https://ami.uni-eszterhazy.hu

An incremental algorithm for computing the transversal hypergraph

Laszlo Szathmary

University of Debrecen, Faculty of Informatics
Debrecen, Hungary
szathmary.laszlo@inf.unideb.hu

Abstract. In this paper we present an incremental algorithm for computing the transversal hypergraph. Our algorithm is an optimized version of Berge's algorithm [2] for solving the transversal hypergraph problem. The original algorithm of Berge is the simplest and most direct scheme for generating all minimal transversals of a hypergraph. Here we present an optimized version of Berge's algorithm that we call *BergeOpt*. We show that *BergeOpt* can significantly reduce the number of expensive inclusion tests.

1. Basic concepts

Here we recall the basic notions of hypergraph theory, frequent itemset mining, and we also point out the relation between itemsets and hypergraphs.

1.1. Hypergraphs

In this subsection we mainly rely on [3]. Hypergraph theory [2] is an important field of discrete mathematics with many relevant applications in applied computer science. A hypergraph is a generalization of a graph, where edges can connect arbitrary number of vertices. Formally:

Definition 1.1 (hypergraph)**.** A *hypergraph* is a pair $(V, \mathcal{E})$ of a finite set $V = \{v_1, v_2, \ldots, v_n\}$ and a family $\mathcal{E}$ of subsets of V. The elements of V are called vertices, the elements of $\mathcal{E}$ edges.

Note that some authors, e.g. [2], state that the edge-set as well as each edge must be non-empty and that the union of all edges results in the vertex set.

Submitted: July 30, 2023
Accepted: August 7, 2023
Published online: August 19, 2023

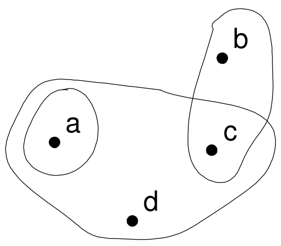


Figure 1. A sample hypergraph $\mathcal{H}$, where $V = \{a, b, c, d\}$ and $\mathcal{E} = \{\{a\}, \{b, c\}, \{a, c, d\}\}$.

Definition 1.2 (partial hypergraph)**.** Let $\mathcal{H} = \{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_m\}$ be a hypergraph. The partial hypergraph $\mathcal{H}_i$ of $\mathcal{H}$ $(i = 1, \ldots, n)$ is the hypergraph that contains the first i edges of $\mathcal{H}$, i.e. $\mathcal{H}_i = \{\mathcal{E}_1, \ldots, \mathcal{E}_i\}$.

A hypergraph is *simple* if none of its edges is contained in any other of its edges. Formally:

Definition 1.3 (simple hypergraph)**.** A hypergraph is called *simple* if it satisfies $\forall \mathcal{E}_i, \mathcal{E}_j \in \mathcal{E} : \mathcal{E}_i \subseteq \mathcal{E}_j \Rightarrow i = j$.

Example. The hypergraph $\mathcal{H}$ in Figure 1 is not simple because the edge $\{a\}$ is contained in the edge $\{a, c, d\}$.

Definition 1.4. Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph. Then $min(\mathcal{H})$ denotes the set of minimal edges of $\mathcal{H}$ w.r.t. set inclusion, i.e. $min(\mathcal{H}) = \{E \in \mathcal{E} \mid \nexists E' \in \mathcal{E} : E' \subset E\}$, and $max(\mathcal{H})$ denotes the set of maximal edges of $\mathcal{H}$ w.r.t. set inclusion, i.e. $max(\mathcal{H}) = \{E \in \mathcal{E} \mid \nexists E' \in \mathcal{E} : E' \supset E\}$.

Clearly, for any hypergraph $\mathcal{H}$, $min(\mathcal{H})$ and $max(\mathcal{H})$ are simple hypergraphs. Moreover, every partial hypergraph of a simple hypergraph is simple, too.
Example. In the case of hypergraph $\mathcal{H}$ in Figure 1, $\min(\mathcal{H}) = \{\{a\}, \{b, c\}\}$ and $\max(\mathcal{H}) = \{\{b, c\}, \{a, c, d\}\}$.

The problem that is of high interest for us concerns hypergraph transversals. A transversal of a hypergraph $\mathcal{H}$ is a subset of the vertex set of $\mathcal{H}$ which intersects each edge of $\mathcal{H}$. A transversal is *minimal* if it does not contain any transversal as proper subset. Formally:

Definition 1.5 (transversal)**.** Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph. A set $T \subseteq V$ is called a *transversal* of $\mathcal{H}$ if it meets all edges of $\mathcal{H}$, i.e. $\forall E \in \mathcal{E} : T \cap E \neq \emptyset$. A transversal T is called *minimal* if no proper subset T' of T is a transversal.

Note that Pfaltz and Jamison call transversal (resp. minimal transversal) as *blocker* (resp. *minimal blocker*) in [8]. Outside hypergraph theory, a transversal is usually called a *hitting set*.

Example. The hypergraph $\mathcal{H}$ in Figure 1 has two minimal transversals: $\{a, b\}$ and $\{a, c\}$. For instance, the sets $\{a, b, c\}$ and $\{a, c, d\}$ are transversals but they are not minimal.

Definition 1.6 (transversal hypergraph)**.** The family of all minimal transversals of $\mathcal{H}$ constitutes a simple hypergraph on V called the *transversal hypergraph* of $\mathcal{H}$, which is denoted by $Tr(\mathcal{H})$.

Example. Considering the hypergraph $\mathcal{H}$ in Figure 1, $Tr(\mathcal{H}) = \{\{a, b\}, \{a, c\}\}$.

The following propositions capture important relations between a hypergraph and its transversal hypergraph (for proofs see [2]).

Proposition 1.7. *Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph. Then $Tr(\mathcal{H})$ is a simple hypergraph, and $Tr(\mathcal{H}) = Tr(min(\mathcal{H}))$.*

Proposition 1.8. *Let $\mathcal{G}$ and $\mathcal{H}$ be two simple hypergraphs. Then $\mathcal{G} = Tr(\mathcal{H})$ if and only if $\mathcal{H} = Tr(\mathcal{G})$.*

Corollary 1.9. *Let $\mathcal{G}$ and $\mathcal{H}$ be two simple hypergraphs. Then $Tr(\mathcal{G}) = Tr(\mathcal{H})$ iff $\mathcal{G} = \mathcal{H}$.*

Corollary 1.10 (duality property)**.** *Let $\mathcal{H}$ be a simple hypergraph. Then $Tr(Tr(\mathcal{H})) = \mathcal{H}$.*

Corollary 1.10 states that calculating the transversal hypergraph $\mathcal{H}'$ of a simple hypergraph $\mathcal{H}$, and calculating once again the transversal hypergraph $\mathcal{H}''$ of $\mathcal{H}'$, we get back the original hypergraph $\mathcal{H}$, i.e. $\mathcal{H}'' = \mathcal{H}$.

Example. Consider the hypergraph $\mathcal{H}$ in Figure 1. Since $\mathcal{H}$ is not simple, let $\mathcal{G} = \min(\mathcal{H}) = \{\{a\}, \{b, c\}\}$. Then,

$$\mathcal{G}' = Tr(\mathcal{G}) = Tr(\{\{a\}, \{b, c\}\}) = \{\{a, b\}, \{a, c\}\}$$
$$\mathcal{G}'' = Tr(\mathcal{G}') = Tr(\{\{a, b\}, \{a, c\}\}) = \{\{a\}, \{b, c\}\}.$$

That is, $\mathcal{G}'' = \mathcal{G}$.

1.2. Frequent itemsets

Consider the following 5×5 sample dataset: $\mathcal{D} = \{(1,\ ACDE), (2,\ ABCDE), (3,\ AB), (4,\ D), (5,\ B)\}$. Throughout the paper, we will refer to this example as **"dataset $\mathcal{D}$"**.

Below we use standard definitions of data mining. We consider a set of *objects* or *transactions* $\mathcal{O} = \{o_1, o_2, \ldots, o_m\}$, a set of *attributes* or *items* $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$, and a relation $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$, where $\mathcal{R}(o, a)$ means that the object o has the attribute a. In formal concept analysis [4] the triple $(\mathcal{O}, \mathcal{A}, \mathcal{R})$ is called a *formal context*. A set of items is called an *itemset* or a *pattern*. Each transaction has a unique identifier (*tid*), and a set of transactions is called a *tidset*.[1] The *length* of an itemset is the cardinality of the itemset, i.e. the number of items included in the itemset. An itemset of length i is called an i-long itemset, or simply an i-itemset[2]. An itemset P is said to be *larger* (resp. *smaller*) than Q if $|P| > |Q|$ (resp. $|P| < |Q|$). We say that an itemset $P \subseteq \mathcal{A}$ is *included* in an object $o \in \mathcal{O}$, if $(o, p) \in \mathcal{R}$ for all $p \in P$. Let f be the function that assigns to each itemset $P \subseteq \mathcal{A}$ the set of all objects that include P: $f(P) = \{o \in \mathcal{O} \mid o \text{ includes } P\}$. The set of objects including the itemset is also known as the *image* of the itemset.[3] The (absolute) *support* of an itemset P indicates how many objects include the itemset, i.e. $supp(P) = |f(P)|$. The support of an itemset P can also be defined in relative value, which corresponds to the proportion of objects including P, with respect to the whole population of objects. An itemset P is called *frequent*, if its support is not less than a given *minimum support* (denoted by min_supp), i.e. $supp(P) \geq min\_supp$.

Definition 1.11 (generator)**.** An itemset G is called *generator* if it has no proper subset H ($H \subset G$) with the same support.

Definition 1.12 (closed itemset)**.** An itemset X is called *closed* if it has no proper superset Y ($X \subset Y$) with the same support.

The *closure* of an itemset X (denoted by $\gamma(X)$) is the largest superset of X with the same support. Naturally, if $X = \gamma(X)$, then X is closed. The task of frequent (closed) itemset mining consists of generating all (closed) itemsets with supports greater than or equal to a specified min_supp.

Equivalence classes. Two itemsets $P, Q \subseteq \mathcal{A}$ are said to be *equivalent* ($P \cong Q$) iff they belong to the same set of objects (i.e. $\gamma(P) = \gamma(Q)$). From this definition it follows that *equivalent itemsets have the same support values*. The set of itemsets that are equivalent to an itemset P (P's *equivalence class*) is denoted by $[P] = \{Q \subseteq \mathcal{A} \mid P \cong Q\}$. Generators are *minimal elements* in their equivalence classes (w.r.t. set inclusion), i.e. a generator $G \in [G]$ has no proper subset in $[G]$. An equivalence class has at least one generator. Closed itemsets are *maximal elements* in their equivalence classes (w.r.t. set inclusion), i.e. a closed itemset $X \in [X]$ has no proper superset in $[X]$. An equivalence class has exactly one closed itemset, which means that closed itemsets are unique elements in their equivalence classes. If an equivalence class has only one element, then the equivalence class is called *singleton*. The only element of a singleton equivalence class is closed as well as generator.

1.3. Relation between itemsets and hypergraphs

Here we show that a family of itemsets can be treated as a hypergraph, and vice versa. As seen in Def. 1.1, a hypergraph $\mathcal{H}$ is a pair $(V, \mathcal{E})$, where V is a finite

[1]For convenience, we will use separator-free set notations throughout the paper, e.g. AB stands for $\{A, B\}$, 13 stands for $\{1, 3\}$, etc.

[2]For instance, ABE is a 3-itemset.

[3]For instance, in dataset $\mathcal{D}$, the image of AB is 23.

set $\{v_1, v_2, \ldots, v_n\}$ and $\mathcal{E}$ is a family of subsets of V. The elements of V are called vertices, the elements of $\mathcal{E}$ edges. In the previous subsection we saw that a formal context is a triple $(\mathcal{O}, \mathcal{A}, \mathcal{R})$, where $\mathcal{O} = \{o_1, o_2, \ldots, o_m\}$ is a set of objects, $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ is a set of items, and $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$ is a relation between $\mathcal{O}$ and $\mathcal{A}$, where $\mathcal{R}(o, a)$ means that the object o has the item a. A set of items is called an itemset.

The set $\mathcal{A}$ can be considered as a set of vertices V. An itemset corresponds to an edge $E \in \mathcal{E}$. From this it follows that a set of itemsets can be considered as a family of edges $\mathcal{E}$.

Example. Consider the hypergraph $\mathcal{H}$ in Figure 1, where $V = \{a, b, c, d\}$ and $\mathcal{E} = \{\{a\}, \{b, c\}, \{a, c, d\}\}$. This hypergraph corresponds to the following set of itemsets: $\{\{a\}, \{b, c\}, \{a, c, d\}\}$. For convenience, we will use separator-free set notations, and we will indicate itemsets with capital letters. That is, the hypergraph $\mathcal{H}$ can be considered as the following set of itemsets: $\{A, BC, ACD\}$. This holds in the other direction too, i.e. the hypergraph representation of the family of itemsets $\{A, BC, ACD\}$ is depicted in Figure 1.

In the rest of the paper, we will treat a family of itemsets as a hypergraph and vice-versa if there is no danger of ambiguity. Thus for a set of itemsets $\{A, BC, ACD\}$, we write "the *hypergraph* $\{A, BC, ACD\}$", etc.

2. The algorithm of Berge

In this section we review the basic algorithm of Berge [2], which is the most simple and direct scheme for generating all minimal transversals of a hypergraph. First, let us see two useful operations on hypergraphs:

Definition 2.1. Let $\mathcal{H} = \{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$ and $\mathcal{G} = \{\mathcal{E}'_1, \ldots, \mathcal{E}'_{n'}\}$ be two hypergraphs. Then,

$$\mathcal{H} \cup \mathcal{G} = \{\mathcal{E}_1, \ldots, \mathcal{E}_n, \mathcal{E}'_1, \ldots, \mathcal{E}'_{n'}\}, \text{ and}$$
$$\mathcal{H} \vee \mathcal{G} = \{\mathcal{E}_i \cup \mathcal{E}'_j,\ i = 1, \ldots, n,\ j = 1, \ldots, n'\}.$$

The first operation is the *union* of $\mathcal{H}$ and $\mathcal{G}$, i.e. the hypergraph whose edges are the edges of both hypergraphs. The second operation is very similar to the *Cartesian product*, i.e. the union of all possible pairs of edges, where one element of a pair is from the first hypergraph, and the other element is from the second hypergraph.

Proposition 2.2 ([2])**.** *Let $\mathcal{H}$ and $\mathcal{G}$ be two simple hypergraphs. Then,*

$$Tr(\mathcal{H} \cup \mathcal{G}) = min(Tr(\mathcal{H}) \vee Tr(\mathcal{G})).$$

Let $\mathcal{H}_i = \{\mathcal{E}_1, \ldots, \mathcal{E}_i\},\ i = 1, \ldots, n$ be the partial hypergraph of the hypergraph $\mathcal{H}$. It holds that $\mathcal{H}_i = \mathcal{H}_{i-1} \cup \{\mathcal{E}_i\}$, for all $i = 2, \ldots, n$, where $\mathcal{H}_1 = \{\mathcal{E}_1\}$ and $\mathcal{H}_n = \mathcal{H}$. Thus, $Tr(\mathcal{H}_i) = Tr(\mathcal{H}_{i-1} \cup \{\mathcal{E}_i\})$, and by Prop. 2.2,

Equation 2.3.

$$\begin{aligned} Tr(\mathcal{H}_i) &= min(Tr(\mathcal{H}_{i-1}) \vee Tr(\{\mathcal{E}_i\})) \\ &= min(Tr(\mathcal{H}_{i-1}) \vee \{\{v\},\ v \in \mathcal{E}_i\}). \end{aligned}$$

The algorithm of Berge is based on this equation. The algorithm computes all minimal transversals of a given hypergraph $\mathcal{H}$ in two steps. First, it computes the minimal transversals of the partial hypergraph $\mathcal{H}_{i-1}$ and then it calculates the Cartesian product of the set $Tr(\mathcal{H}_{i-1})$ by the i^{th} edge $\mathcal{E}_i$ of $\mathcal{H}$. Finally, non-minimal elements are removed. Thus, the algorithm starts with the computation of $Tr(\mathcal{H}_1)$, which is a trivial case ($\mathcal{H}_1$ has one edge only, $\mathcal{E}_1$, whose minimal transversals are its vertices). Then, the algorithm adds one by one the rest of the edges, computing at each step the set of minimal transversals of the new partial hypergraph. The algorithm terminates when the last edge $\mathcal{E}_n$ is added. The algorithm of Berge outputs at the end all minimal transversals of the input hypergraph $\mathcal{H}$ [2].

3. An optimized version of Berge's algorithm

In the previous section we reviewed the algorithm of Berge, which implements the most simple and direct approach for calculating the minimal transversals of a hypergraph. Here we present an optimized version of Berge's algorithm that we call *BergeOpt*.

In [7], Le Floc'h *et al.* presented an algorithm called *JEN* whose goal is to efficiently extract generators from a concept lattice [4] for mining exact and approximate association rules [1]. As part of *JEN*, the aforementioned authors presented a simple algorithm without a name for calculating all the minimal transversals of a hypergraph. In the rest of this section we present this algorithm in an extended and completed way. In addition to [7], **(i)** we show that this algorithm is actually an optimization of Berge's original algorithm (hence the name *BergeOpt*), and **(ii)** we provide a proposition (see Prop. 3.1) and its proof.

Optimization idea. One drawback of Berge's algorithm is that after calculating the Cartesian product of the set $Tr(\mathcal{H}_{i-1})$ by the i^{th} edge $\mathcal{E}_i$ of $\mathcal{H}$ (see Equation 2.3), it stores the resulting elements together in the same set, i.e. it has no information whether an element is minimal or not. As a consequence, the filtering of non-minimal elements can be quite expensive when the resulting set has a large number of elements because the algorithm must test the minimality of all elements, including also such elements that are actually minimal.

Our optimization is based on the idea to separate minimal and potentially minimal transversals in two different lists L_1 and L_2, respectively. This way, our optimized algorithm only has to check the minimality of the potentially minimal elements in L_2. As a result, the number of expensive subset checks can be reduced.

The *BergeOpt* algorithm exploits the following proposition:

Proposition 3.1. *In the* BergeOpt *algorithm, the potentially minimal transversals stored in the list L_2 form a simple hypergraph, i.e. L_2 has no two elements e_i and e_j such that $e_i \subseteq e_j$.*

Proof. Assume $X, Y \subseteq V$ are two distinct subsets in $Tr(\mathcal{H}_{i-1}) - Tr(\mathcal{H}_i)$, i.e., they are minimal transversals of $\mathcal{H}_{i-1}$ that lost this status in the i-th partial hypergraph. Assume also that $X \cup \{a\}$ and $Y \cup \{b\}$ are two candidates for $Tr(\mathcal{H}_i)$ produced by the algorithm (i.e., $\{a, b\} \subseteq \mathcal{E}_i$ whereby $a \notin X$ and $b \notin Y$).

Notice that any element of the L_2 list will have the form $X \cup \{a\}$ for some X and a.

Now, without loss of generality we can hypothesize $X \cup \{a\} \subseteq Y \cup \{b\}$, and show this leads to a contradiction. First, notice that $a \neq b$, otherwise we would have $X \subseteq Y$ hence a contradiction with the minimal transversal status. Next, we deduce that $Y = \bar{X} \cup \bar{Y}$ where $\bar{X} = X - \{b\}$ hence $X = \bar{X} \cup \{b\}$. Yet this means that $b \in X \cap \mathcal{E}_i$ which contradicts $X \notin Tr(\mathcal{H}_i)$. □

Pseudo code. The pseudo code of the algorithm is given in Algorithm 1. Let $\mathcal{H}_i = \{\mathcal{E}_1, \ldots, \mathcal{E}_i\}$, $i = 1, \ldots, n$ be the partial hypergraph of the hypergraph $\mathcal{H}$. It holds that $\mathcal{H}_i = \mathcal{H}_{i-1} \cup \{\mathcal{E}_i\}$, for all $i = 2, \ldots, n$, where $\mathcal{H}_1 = \{\mathcal{E}_1\}$ and $\mathcal{H}_n = \mathcal{H}$. Let $\mathcal{MT}_{\mathcal{H}_i}$ denote the set of all minimal transversals of the partial hypergraph $\mathcal{H}_i$.

As input, we have a set of itemsets that we treat as a hypergraph (see Section 1.3). The goal is to compute all the minimal transversals of this hypergraph. The algorithm performs this task in an incremental way. First, the algorithm takes the first itemset $\mathcal{E}_1$ of the input and it calculates its minimal transversals. This is a trivial case; we only have to decompose the itemset into its 1-long subsets. For instance, the itemset ABC has three minimal transversals namely A, B, and C. Then, the algorithm takes the next itemset $\mathcal{E}_i$ of the input and it updates the list of minimal transversals $\mathcal{MT}_{\mathcal{H}_{i-1}}$ if necessary. This is done the following way. Each minimal transversal m found so far, i.e. each element of $\mathcal{MT}_{\mathcal{H}_{i-1}}$, is tested if it has a common part with the current itemset $\mathcal{E}_i$. If it has, then m is a minimal transversal of $\mathcal{H}_i$ too, thus m is added to the list L_1. In the list L_1 we collect those itemsets that are minimal transversals of the partial hypergraph processed so far, including the current itemset $\mathcal{E}_i$ too. Prop. 1.7 guarantees that L_1 has no two elements e_1 and e_2 such that $e_1 \subseteq e_2$. If the test was negative, i.e. m has no common part with the current itemset $\mathcal{E}_i$, then it means that m is not a transversal of $\mathcal{E}_i$, thus m must be extended to have an intersection with $\mathcal{E}_i$ (in other words, m is a transversal of $\mathcal{H}_{i-1}$, but not a transversal of $\mathcal{H}_i$). This can be done by decomposing $\mathcal{E}_i$, and generating the one-size larger supersets of m using the 1-long subsets of $\mathcal{E}_i$ (Cartesian product of m with the vertices of $\mathcal{E}_i$). For instance, if $\mathcal{E}_i = BCH$, and the minimal transversal to be updated is AD, then the following potentially minimal transversals are generated: ABD, ACD, and ADH. We call these itemsets "potentially minimal transversals", because with this extension it is guaranteed that they became transversals of $\mathcal{H}_i$, but it is not sure that they are *minimal*, thus they are put in another list L_2. It can be possible that they have

Algorithm 1 (“getMinTransversals” function):

```
Description:   BergeOpt algorithm
Input:         a hypergraph ($\mathcal{H}$)
Output:        all minimal transversals of $\mathcal{H}$ ($\mathcal{MT}$)

 1)  $\mathcal{MT} \leftarrow \emptyset$;   // initialisation; no minimal transversals are found yet
 2)  loop over the elements of $\mathcal{H}$ ($\mathcal{E}_i$)   // an element of $\mathcal{H}$ is an edge (an itemset)
 3)  {
 4)      if ($\mathcal{E}_i$ is the first element of $\mathcal{H}$) {
 5)          $\mathcal{MT} \leftarrow$ {vertices of $\mathcal{E}_i$};   // decomposition (1-itemsets of $\mathcal{E}_i$)
 6)      }
 7)      else
 8)      {
 9)          $L_1 \leftarrow \emptyset$; $L_2 \leftarrow \emptyset$;   // two empty lists
10)          loop over the elements of $\mathcal{MT}$ ($m$)
11)          {
12)              if ($m \cap \mathcal{E}_i \neq \emptyset$) {   // m has a common vertex with $\mathcal{E}_i$
13)                  $L_1 \leftarrow L_1 \cup m$;   // m is a minimal transversal of $\mathcal{E}_i$
14)              }
15)              else {
16)                  $S \leftarrow$ {one-size larger supersets of $m$ using
                          the vertices of $\mathcal{E}_i$};
17)                  $L_2 \leftarrow L_2 \cup S$;
18)              }
19)          }
20)          if ($L_1 \neq \emptyset$ and $L_2 \neq \emptyset$) {
21)              cleanSupersets($L_1$, $L_2$);   // removing non-minimal ...
22)          }                                   // ... transversals from $L_2$
23)          $\mathcal{MT} \leftarrow L_1 \cup L_2$;
24)      }
25)  }
26)
27)  return $\mathcal{MT}$;
```

subsets among the minimal transversals in L_1. When all elements of $\mathcal{MT}_{\mathcal{H}_{i-1}}$ are tested against the current itemset $\mathcal{E}_i$, the lists L_1 and L_2 are filled. At this point, there are three possibilities (lines 20–23 of Algorithm 1): **(1)** L_1 is non-empty and L_2 is empty, or **(2)** L_1 is empty and L_2 is non-empty, or **(3)** both L_1 and L_2 are non-empty. In the *first case*, L_1 contains all the minimal transversals of $\mathcal{H}_i$. By Prop. 1.7, L_1 is a simple hypergraph. In the *second case*, L_2 contains all the minimal transversals of $\mathcal{H}_i$. Since L_1 is empty, all elements in L_2 are minimal. Moreover, from Prop. 3.1 it follows that L_2 is a simple hypergraph. In the *third case*, the list L_2 must be cleaned first, i.e. if an element e_1 in L_1 is a subset of an element e_2 in L_2, then e_2 must be removed because e_2 is *not minimal.* Prop. 3.1

guarantees that the elements of L_2 are not comparable w.r.t. set inclusion. Then, taking the union of the lists L_1 and L_2, we have all the minimal transversals of $\mathcal{H}_i$.

The algorithm continues by taking the next itemset of the input set (next current itemset) and it updates again the list of minimal transversals. The algorithm terminates when all elements of the input set are processed. At this point, the algorithm collected all the minimal transversals of the input set, i.e. it calculated the transversal hypergraph of the input hypergraph.

`cleanSupersets` procedure: this method removes non-minimal transversals from the list L_2, i.e. itemsets that have subsets in L_1. The procedure works as follows. It enumerates all elements of L_2. If the current element e_2 in L_2 has a subset in L_1, then e_2 is removed from L_2. When the procedure terminates, L_2 only contains *minimal* transversals.

Running example. Consider the following hypergraph $\mathcal{H} = \{ACD, ACH, BCD, DF, FH\}$. Let $\mathcal{E}_i$ denote the i^{th} element (edge) of the hypergraph, i.e. $\mathcal{E}_1 = ACD, \mathcal{E}_2 = ACH, \dots, \mathcal{E}_5 = FH$. Let $\mathcal{H}_i$ denote the partial hypergraph that contains the first i elements of $\mathcal{H}$, i.e. $\mathcal{H}_1 = \{ACD\}$, $\mathcal{H}_2 = \{ACD, ACH\}$, ..., $\mathcal{H}_5 = \{ACD, ACH, BCD, DF, FH\} = \mathcal{H}$. The notation $\mathcal{MT}_{\mathcal{H}_i}$ denotes the set of all minimal transversals of the partial hypergraph $\mathcal{H}_i$.

Table 1. Incremental computation of the transversal hypergraph of $\mathcal{H} = \{ACD, ACH, BCD, DF, FH\}$ with the *BergeOpt* algorithm.

$\mathcal{E}_1 = ACD$	$\mathcal{MT}_{\mathcal{H}_1} = \{A, C, D\}$
$\mathcal{E}_2 = ACH$	$L_1 = \{A, C\}$
	$L_2 = \{$~~AD~~,~~CD~~,$DH\}$
	$\mathcal{MT}_{\mathcal{H}_2} = \{A, C, DH\}$
$\mathcal{E}_3 = BCD$	$L_1 = \{C, DH\}$
	$L_2 = \{AB,$~~AC~~$,AD\}$
	$\mathcal{MT}_{\mathcal{H}_3} = \{C, DH, AB, AD\}$
$\mathcal{E}_4 = DF$	$L_1 = \{DH, AD\}$
	$L_2 = \{CD, CF,$~~ABD~~$,ABF\}$
	$\mathcal{MT}_{\mathcal{H}_4} = \{DH, AD, CD, CF, ABF\}$
$\mathcal{E}_5 = FH$	$L_1 = \{DH, CF, ABF\}$
	$L_2 = \{ADF,$~~ADH~~,~~CDF~~,~~CDH~~$\}$
	$\mathcal{MT}_{\mathcal{H}_5} = \{DH, CF, ABF, ADF\} = \mathcal{MT}_{\mathcal{H}} = Tr(\mathcal{H})$

The execution of the algorithm is depicted in Table 1. First, the algorithm takes $\mathcal{E}_1$ (ACD) and computes its minimal transversals that are A, C, and D. The algorithm continues with processing $\mathcal{E}_2$ (ACH). Each time when a new element of $\mathcal{H}$ is handled, the already found minimal transversals are tested. The itemsets A and C have common parts with $\mathcal{E}_2$, thus they are minimal transversals of ACH, so they are added to the list L_1. However, D has no common part with $\mathcal{E}_2$, which means that D is a minimal transversal of $\mathcal{H}_1$, but not a transversal of $\mathcal{H}_2$. In order to make D a transversal of $\mathcal{H}_2$, D is extended with the 1-long subsets of

ACH, thus the following candidates are generated: AD, CD, and DH. These three itemsets are put in the list L_2. Then, the algorithm removes itemsets from L_2 that have subsets in L_1 since they are not minimal transversals (AD and CD). The union of L_1 and L_2, which is stored in the list $\mathcal{MT}_{\mathcal{H}_2}$, gives all the minimal transversals of $\mathcal{H}_2$. The same steps are repeated with the other elements of $\mathcal{H}$ ($\mathcal{E}_3$, $\mathcal{E}_4$, and $\mathcal{E}_5$). When the algorithm terminates, all minimal transversals of the hypergraph $\mathcal{H}$ are discovered. In this example, the transversal hypergraph of $\mathcal{H}$ is $Tr(\mathcal{H}) = \{DH, CF, ABF, ADF\}$.

4. Experimental results

The *BergeOpt* algorithm was implemented in Java in the Coron data mining platform [9].[4] The experiments were carried out on an Intel Core i7 3.5 GHz machine with 16 GB RAM running under Manjaro GNU/Linux. All times reported are real, wall clock times.

Our algorithm *BergeOpt* was used as part of another algorithm called *Snow* that we presented in [10]. In [10] we just mentioned *BergeOpt* without giving any details. A detailed presentation of *Snow* is out of the scope of this paper, but we give a short summary. Frequent closures (FCIs) and frequent generators (FGs) as well as the precedence relation on FCIs are key components in the definition of a variety of association rule bases (see [6] for a survey). The goal of the *Snow* algorithm is to extract the precedence relation from a more common mining output, i.e. closures and generators. Thus, the idea is the following. First, we extract FCIs and their associated generators, i.e. we get the frequent equivalence classes (see Section 1.2). In each equivalence class, we consider the set of FGs to be a simple hypergraph. Using *BergeOpt*, we calculate the transversal hypergraph of the generators. With this result, the order among the FCIs can be obtained very efficiently. For a detailed description of the *Snow* algorithm, please refer to [10]. To conclude, in our experiments *BergeOpt* was used to calculate the transversal hypergraph of the generators in each equivalence class in a dataset.

For the experiments, we used several real and synthetic dataset benchmarks. Database characteristics are shown in Table 2 (top). The chess and connect datasets are derived from their respective game steps. The Mushrooms database describes mushrooms characteristics. These three datasets can be found in the UC Irvine Machine Learning Database Repository.[5] The pumsb, C20D10K, and C73D10K datasets contain census data from the PUMS sample file. The synthetic datasets T20I6D100K and T25I10D10K, using the IBM Almaden generator, are constructed according to the properties of market basket data. Typically, real datasets are very dense, while synthetic data are usually sparse.

Table 2 (bottom left and right) provides a summary of the experimental results. The first column specifies the various minimum support values for each of the datasets (low for the sparse dataset, higher for dense ones). The second and third

[4] `http://coron.loria.fr`
[5] `https://archive.ics.uci.edu`

Table 2. Top: database characteristics. **Bottom:** response times of *BergeOpt*.

database name	# records	# non-empty attributes	# attributes (in average)	largest attribute
T20I6D100K	100,000	893	20	1,000
T25I10D10K	10,000	929	25	1,000
chess	3,196	75	37	75
connect	67,557	129	43	129
pumsb	49,046	2,113	74	7,116
MUSHROOMS	8,416	119	23	128
C20D10K	10,000	192	20	385
C73D10K	10,000	1,592	73	2,177

min_supp	# concepts (including top)	*BergeOpt* (seconds)
T20I6D100K		
0.75%	4,711	0.03
0.50%	26,209	0.21
0.25%	149,218	1.10
T25I10D10K		
0.40%	83,063	0.56
0.30%	122,582	0.86
0.20%	184,301	1.33
chess		
65%	49,241	0.34
60%	98,393	0.68
55%	192,864	1.28
connect		
65%	49,707	0.29
60%	68,350	0.46
55%	94,917	0.56

min_supp	# concepts (including top)	*BergeOpt* (seconds)
pumsb		
80%	33,296	0.57
78%	53,418	0.99
76%	82,539	2.04
MUSHROOMS		
20%	1,169	0.01
10%	4,850	0.04
5%	12,789	0.15
C20D10K		
0.50%	132,952	1.12
0.40%	151,394	1.18
0.30%	177,195	1.45
C73D10K		
65%	47,491	0.53
60%	108,428	1.26
55%	222,253	2.70

columns comprise the number of FCIs and the execution time of *BergeOpt* (given in seconds). The CPU time does not include the cost of computing FCIs and FGs since they are assumed as given.

As can be seen, *BergeOpt* is able to calculate the transversal hypergraph of the generators in the equivalence classes very efficiently in both sparse and dense datasets. To find out why the algorithm *BergeOpt* performs so well, we investigated the size of its input data. Figure 2 shows the distribution of hypergraph sizes in the datasets T20I6D100K, MUSHROOMS, chess, and C20D10K.[6] Note that we obtained similar hypergraph-size distributions in the other four datasets too. Figure 2 indicates that most hypergraphs only have 1 edge, which is a trivial case, whereas large hypergraphs are relatively rare. As a consequence, *BergeOpt* can perform very efficiently.

We interpret the above results as an indication that the good performance of *BergeOpt* is independent of the density of the dataset. In other terms, provided

[6]For instance, the dataset T20I6D100K by *min_supp* = 0.25% contains 149,019 1-edged hypergraphs, 171 2-edged hypergraphs, 25 3-edged hypergraphs, 0 4-edged hypergraphs, 1 5-edged hypergraph, and 1 6-edged hypergraph.

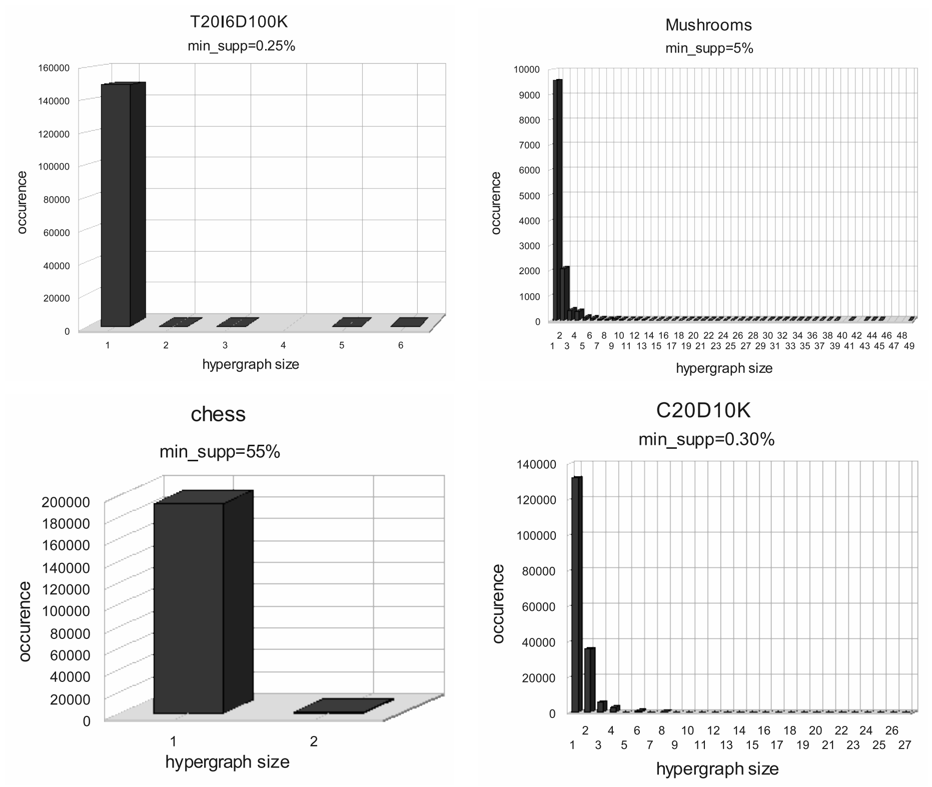


Figure 2. Distribution of hypergraph sizes.

that the input hypergraphs do not contain too many edges, i.e. there are only few FGs per FCIs, the computation is very fast. A natural question arises with this observation: does the modest number of FGs in each class hold for all realistic datasets in the literature? If not, could one profile those datasets which meet this condition?

5. Conclusion

In this paper we presented an optimization of Berge's original algorithm [2] called *BergeOpt* that can significantly reduce the number of expensive inclusion tests. Since Berge's algorithm several other, more efficient algorithms have been introduced. As pointed out in [5] for instance, the simple method of Berge needs exponential many steps to produce the whole output. It generates the first minimal transversal near the end of the procedure and its high memory requirements make it suitable only for small problem cases. However, as we pointed out in the previous section, our hypergraphs are usually very small, thus we did not have to face these efficiency problems. Experimental results show that *BergeOpt* provides a very efficient solution for the problem instance that we had to deal with, i.e. when we used *BergeOpt* as part of the *Snow* algorithm [10] to discover the precedence relation among FCIs.

References

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. I. Verkamo: *Fast discovery of association rules*, in: Advances in knowledge discovery and data mining, American Association for Artificial Intelligence, 1996, pp. 307–328, isbn: 0-262-56097-6.

[2] C. Berge: *Hypergraphs: Combinatorics of Finite Sets*, Amsterdam: North Holland, 1989.

[3] T. Eiter, G. Gottlob: *Identifying the Minimal Transversals of a Hypergraph and Related Problems*, SIAM Journal on Computing 24.6 (1995), pp. 1278–1304, issn: 0097-5397, doi: 10.1137/S0097539793250299.

[4] B. Ganter, R. Wille: *Formal concept analysis: mathematical foundations*, Berlin/Heidelberg: Springer, 1999, p. 284, isbn: 3540627715.

[5] D. J. Kavvadias, E. C. Stavropoulos: *An Efficient Algorithm for the Transversal Hypergraph Generation*, Journal of Graph Algorithms and Applications 9.2 (2005), pp. 239–264.

[6] M. Kryszkiewicz: *Concise Representations of Association Rules*, in: Proc. of the ESF Exploratory Workshop on Pattern Detection and Discovery, 2002, pp. 92–109.

[7] A. Le Floc'h, C. Fisette, R. Missaoui, P. Valtchev, R. Godin: *JEN : un algorithme efficace de construction de générateurs pour l'identification des règles d'association*, Spec. num. of Revue des Nouvelles Technologies de l'Information 1.1 (2003), pp. 135–146.

[8] J. L. Pfaltz, R. E. Jamison: *Closure Systems and their Structure*, Information Sciences 139.3–4 (2001), pp. 275–286.

[9] L. Szathmary: *Symbolic Data Mining Methods with the Coron Platform*, PhD Thesis in Computer Science, Univ. Henri Poincaré – Nancy 1, France, Nov. 2006.

[10] L. Szathmary, P. Valtchev, A. Napoli, R. Godin, A. Boc, V. Makarenkov: *A fast compound algorithm for mining generators, closed itemsets, and computing links between equivalence classes*, Annals of Mathematics and Artificial Intelligence (AMAI) 70.1–2 (2014), pp. 81–105, issn: 1012-2443, doi: 10.1007/s10472-013-9372-8.

Annales Mathematicae et Informaticae
58 (2023) pp. 160–169
DOI: 10.33039/ami.2023.08.015
URL: https://ami.uni-eszterhazy.hu

Analysis of retrial queueing systems with two-way communication and impatient customers using simulation

Ádám Tóth, János Sztrik

University of Debrecen, University Square 1, Debrecen H-4032, Hungary
{toth.adam|sztrik.janos}@inf.unideb.hu

Abstract. The aim of this research is to examine a finite-source retrial queueing system with two-way communication. The primary customers, who arrive from a finite-source following an exponential distribution, either receive service immediately if the service unit is available, or are redirected to the orbit and try again to reach the server after a random period. The system is unique in that when the server becomes idle, an outgoing call (secondary customer) is performed from the orbit or the source with varying parameters. Both primary and secondary customers have been serviced according to an exponential distribution but with different rates. Customers exhibit an impatience characteristic, which may lead to their departure before receiving service if they spend a certain amount of time waiting for the service unit. This investigation conduct a sensitivity analysis on the system's performance measures by utilizing different distributions of the customers' retrial time in two separate cases. The findings of the analysis have been presented graphically for comparison purposes.

Keywords: Finite-source queuing system, retrial queues, two-way communication, sensitivity analysis, simulation

AMS Subject Classification: 60K25

1. Introduction

Two-way communication is a popular research topic because it can be effectively modeled using retrial queueing systems in many real-life situations. Call centers are a prime example of this, where agents engage in various activities such as selling, advertising, and promoting products when not handling customer calls. Utilization

Submitted: July 25, 2023
Accepted: August 9, 2023
Published online: September 13, 2023

is one of the most crucial metrics in call center operations, and optimizing the efficiency of service units or agents is always a critical concern, see for example [1, 3, 7, 9, 13]. The distinctive feature of two-way communication is based on the occurrence of calls both inside and outside the system when the server is idle. Two types of outgoing calls can be identified:

- One type of outgoing call is when the server contacts a customer from the source for service, which is referred to as a primary outgoing call,

- Another type of outgoing call occurs when the server contacts a customer from the orbit, which is known as a secondary outgoing call.

In our model, we consider outgoing calls that can be made to either the source or the orbit. Existing literature on queueing systems reveals different schemes, where some models assume infinite queue size, causing incoming customers to wait until they receive service, while in others, customers leave the system immediately upon arrival if the service unit is fully occupied. However, in reality, there are scenarios where customers do not leave the system but wait in a virtual waiting room, known as an orbit, and attempt to connect with a server after some random time. Retrial queues are a suitable modeling tool for systems that involve an orbit. Queuing systems that utilize retrial queues are commonly used to model various problems arising in telecommunications systems, such as call centers, telephone switching systems, and computer networks like in [2, 6, 8]. Previously, scholars have explored examples of retrial queueing systems with two-way communication and infinite sources, some of which are listed below: [12, 14, 15].

Dragieva and Phung-Duc [5] examined a scenario in which secondary outgoing calls return to the source after service, while this study is a natural extension of [10], which considered a more realistic scenario. In this case, rather than returning secondary outgoing customers to the source, they are sent back to the orbit where they can retry their request for servicing the original incoming call. Investigating finite-source retrial models with two-way communication is motivated by real-life situations in which customers cannot receive immediate service upon arrival and must go to another location before attempting to check the system again or wait for the server to call for them when idle.

Some queuing models make the assumption that a consumer must wait in line indefinitely before being serviced. When a customer enters and discovers the service area is occupied, some additional models—known as loss models—have the customer leave and lose it forever. However, there are countless situations in real life where customers choose to give up the attempt to be served after an arbitrary amount of time rather than waiting. In this scenario, the client waits in a virtual waiting area called an orbit before making another attempt to contact the server. Retrial queues can be used to represent models that have an orbit.

The originality of this work lies in the sensitivity analysis that was conducted to examine how different retrial time distributions affect the key performance metrics. Our stochastic simulation program, which is based on SimPack produces the results.

To support discrete event simulation, continuous simulation, and combined (multi-model) simulation, this is a collection of C/C++ libraries and executable programs. Any sort of queueing system and simulation model can be freely modeled, and any performance metric can be calculated using any random number generator for the specified random variable. The comparison of the operating modes and various distributions will be shown through graphical representations.

The system model

This section introduces the finite-source retrial queueing paradigm with a single server under consideration (see Figure 1). The source contains a total of N re-

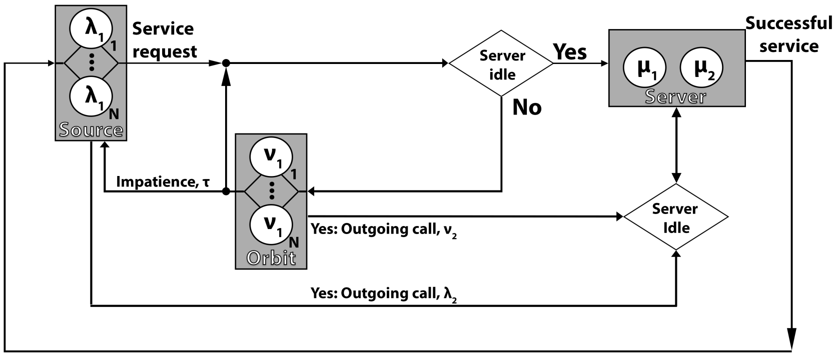


Figure 1. System model.

quests, each of which can produce a primary incoming call to the server. The inter-request times are determined by exponentially distributed random variables with the parameter λ_1. When the server is idle, an incoming customer's service starts immediately and follows an exponential distribution with parameter μ_1. After receiving satisfactory service, clients return to the original provider. Customers who arrive and find the service unit busy will not be lost; instead, they are transported to orbit. These are the secondary arriving jobs from the orbit that might make another attempt to contact the service unit following an arbitrary waiting period. Gamma, hyper-exponential, Pareto, and lognormal distributions are all used to describe this period's distribution, albeit they all have the same mean value. But the idle server may also request calls from the orbit and the source. We distinguish between two categories of outgoing calls:

- After an exponentially varying amount of time, the service unit may request a primary outgoing call from the source to be served with parameter λ_2,
- After an exponentially distributed period, the service unit may make a call (secondary outgoing call) from orbit with parameter ν_2.

The outgoing customers' service time is distributed exponentially with the parameter μ_2. When an incoming call is received from the orbit, there are two distinct scenarios:

- Operation mode number 1: After the outgoing service is complete, the call is returned back to the orbit to have its incoming call served because it has an unmet incoming request,
- Operation mode number 2: Here, the call also has an incoming request that hasn't been fulfilled, but as soon as the outgoing service is complete, the service unit fulfills the incoming request. A two-phase service will result from this, with the outgoing call being processed first and then the incoming one. When both service phases have been completed, the call goes back to the source.

Every primary customer has an impatience trait, and in our investigated model a primary customer eventually departs the system after waiting in the orbit for some time without obtaining the proper service, which is also an exponentially distributed random variable with rate τ. The arrivals of primary incoming calls, retrial intervals for secondary incoming calls, service times for incoming and outgoing calls, and the amount of time needed to make outgoing calls are all considered to be independent of one another.

Utilizing this model, our goal is to perform a sensitivity analysis on the main performance measures using several distributions of retrial times. A number of system properties are compared between various operating modes as well. We developed a simulation program to get the results, which will be shown in a series of graphs.

2. Simulation results

2.1. First scenario

We utilized SimPack as the foundation of our program and incorporated the necessary functionalities. To estimate the desired performance measures, we employed a statistical package that utilizes the popular batch means method. The simulation period is divided into a set of batches, with $s = R - M/T$ observations conducted in each batch, where M represents the discarded warm-up period observations and R is the simulation length. Once the initial phase is complete, the average of the entire simulation is computed. It is crucial for the batches to be of sufficient length and for each batch average to be independent for meaningful results. For additional information about the process used, please refer to the following papers: [4, 11].

The input parameters used in the simulations are presented in Table 1. A relative half-width of 0.00001 and a confidence level of 99.9% were employed to halt the simulation sequence. To ensure the accuracy of the results, the size of a batch during the initial transient period was set to 1000 and cannot be too small.

Table 1. Numerical values of model parameters.

N	μ_1	μ_2	λ_2	ν_2	τ
10	1	1	0.2	0.2	0.01;0.05;0.1

Table 2 lists the retrial time parameters of the customers, which were selected to have the same mean and variance value for a valid comparison. Various parameter values were tested in the simulation program, and the most significant results will be discussed in this paper. As demonstrated in the table, the squared coefficient of variation exceeds one in this case, enabling an investigation into the influence of specific random variables. Additionally, we will present outcomes with a distinct set of parameters when the squared coefficient of variation is less than one.

Table 2. Parameters of retrial time.

Distribution	**Gamma**	**Hyper-exponential**	**Pareto**	**Lognormal**
Parameters	$\alpha = 0.02$ $\beta = 0.2$	$p = 0.489$ $\lambda_1 = 9.798$ $\lambda_2 = 10.202$	$\alpha = 2.01$ $k = 0.05$	$m = -4.258$ $\sigma = 1.978$
Mean	0.1			
Variance	0.49			
Squared coefficient of variation	49			

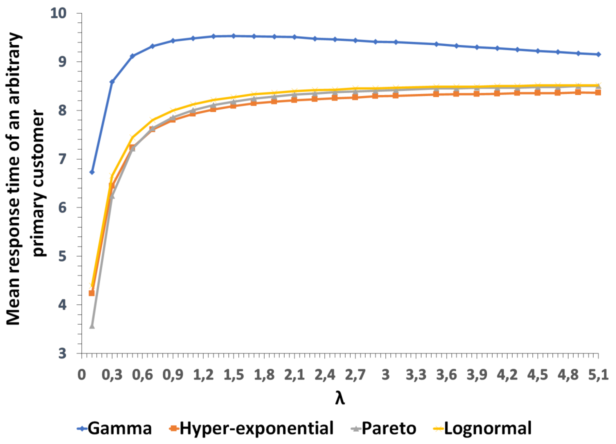


Figure 2. Mean response time of an arbitrary primary customer vs. arrival intensity.

The mean response time of an arbitrary primary customer is depicted in Figure 2 in the function of the arrival intensity. By comparing the impact of various distributions with the same first two moments, a noticeable discrepancy is observed.

Customers spend relatively more time in the orbit when gamma distribution is employed, but more or less the same amount of time when other distribution is utilized. Additionally, the system's intriguing maximum property is evident even as the arrival intensity increases, which is a characteristic of a finite-source retrial queueing system.

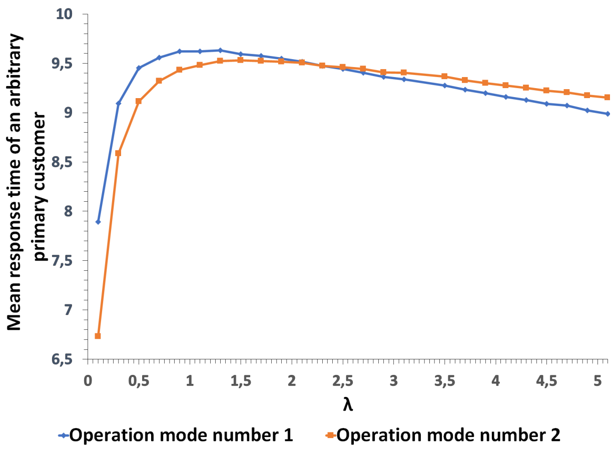


Figure 3. Comparison of the mean response times using different operation modes.

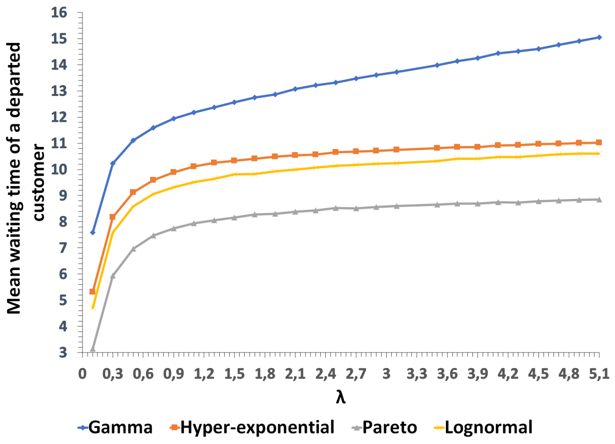


Figure 4. Mean waiting time of a departed customer using the different distributions of retrial times.

Figure 3 illustrates the impact of two operation modes on the mean response time of a customer under gamma distribution of retrial times, as the arrival intensity increases. It is interesting to observe that in case of lower λ values Operation mode number 2 performs better resulting in lower mean response time and this trend changes after λ is greater than 2. However, the differences among operation modes are not that significant in this parameter setting.

Figure 4 demonstrates the comparison of mean waiting time of a departed customer beside various scenarios. Under this performance measure we refer to those mean waiting times of customers who exit from the system due to impatience. Despite having the same mean and variance, there are significant differences between the applied distributions, with increasing gaps observed as the arrival intensity increases. The mean waiting time of an impatient customer also increases with the arrival intensity, and the Pareto distribution consistently results in lower mean waiting times compared to the other distributions, particularly in comparison to the gamma distribution.

2.2. Second scenario

After observing the results of the first scenario, we became curious about the effects of using different parameter values for each distribution while keeping the mean constant. For the second scenario, we reduced the squared coefficient of variation to less than 1 for each distribution, as shown in Table 3, while all other parameters remained the same as in Table 2. To conduct a sensitivity analysis, we replaced the hyper-exponential distribution with a hypo-exponential distribution.

Table 3. Parameters of retrial time.

Distribution	**Gamma**	**Hypo-exponential**	**Pareto**	**Lognormal**
Parameters	$\alpha = 1.6$ $\beta = 16$	$\mu_1 = 13.333$ $\mu_2 = 40$	$\alpha = 2.612$ $k = 0.062$	$m = -2.545$ $\sigma = 0.697$
Mean	0.1			
Variance	0.00625			
Squared coefficient of variation	0.625			

Figure 5 illustrates how the mean response time of an arbitrary primary customer changes with increasing arrival intensity in a scenario where the mean value remains constant but the variance value is significantly reduced. The difference in average response time among the distributions is not very significant, it can be stated that they overlap each other totally. This indicates that variance has a considerable impact on performance measures, as larger variance values can result in greater disparities in performance measures.

Figure 6 compares the mean waiting time of a departed customer across varying arrival intensities. As expected from the previous figure, the differences in the obtained values are relatively small, which is true for every utilized distribution. Therefore, it can be concluded that in this parameter setting, the performance measures do not exhibit significant differences among the distributions. Naturally, the mean waiting time of a departed customer increases with the increment of the arrival intensity of the primary customer.

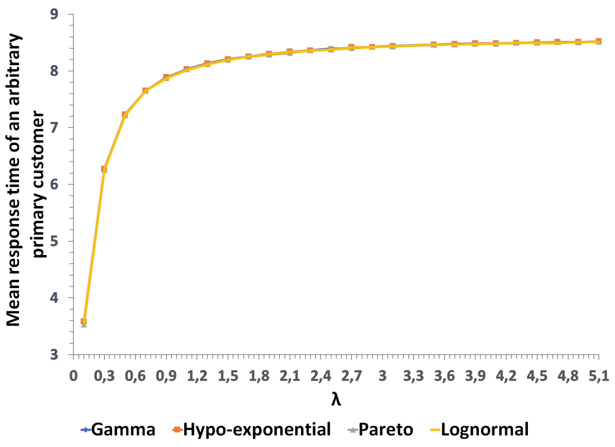


Figure 5. Mean response time of an arbitrary primary customer vs. arrival intensity.

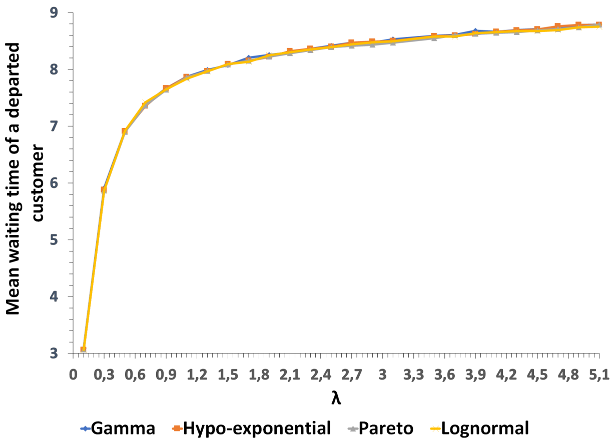


Figure 6. Mean waiting time of a departed customer using the different distributions of retrial times.

Finally, Figure 7 depicts the impact of increasing arrival intensity on the mean response times of the different operation modes. The values obtained in this scenario are very close to each other compared to the previous scenario. In this scenario, looking at the graphs, it can be said that there are no discrepancies in terms of comparing different distributions of retrial times or different operation modes.

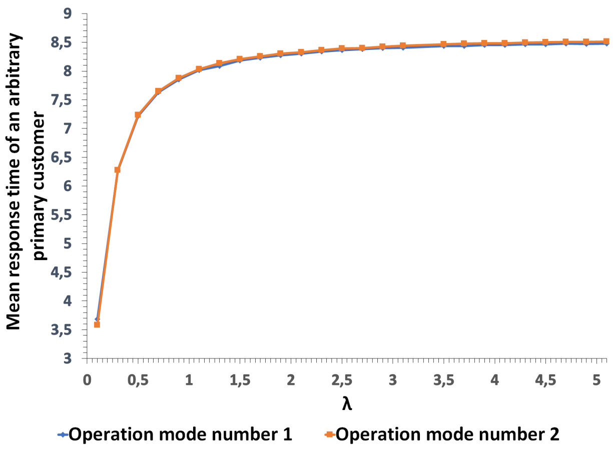


Figure 7. Comparison of the mean response times using different operation modes.

3. Conclusion

We present a study on a two-way communication finite-source retrial queueing system that utilizes different retrial time distributions. Our investigation includes various scenarios with different parameters, focusing on the mean response time of an arbitrary primary customer and the mean waiting time of a departed customer. Through simulations and graphical figures, we demonstrate that choosing an appropriate distribution is critical when the squared coefficient of variation is greater than one. The figures also display the impact of outgoing calls and suggest that Operation mode number 2 (keeping customers waiting inside the bank) may result in smaller waiting and response times than Operation mode number 1.

For instance, in a banking setting, outgoing calls may be used to allocate signatures both inside and outside the bank while customers wait for their transactions. It is more advantageous for the bank to keep the customer waiting inside (Operation mode number 2) rather than turning them away or serving their initial request after obtaining the signature (Operation mode number 1).

Future research may explore other types of two-way communication finite-source retrial queuing systems or consider adding a backup service unit.

References

[1] S. Aguir, F. Karaesmen, O. Z. Akşin, F. Chauvet: *The impact of retrials on call center performance*, OR Spectrum 26.3 (2004), pp. 353–376.

[2] Z. Aksin, M. Armony, V. Mehrotra: *The modern call center: A multi-disciplinary perspective on operations management research*, Production and operations management 16.6 (2007), pp. 665–688.

[3] J. Artalejo, A. G. Corral: *Retrial Queueing Systems: A Computational Approach*, Springer, 2008.

[4] E. J. Chen, W. D. Kelton: *A Procedure for Generating Batch-Means Confidence Intervals for Simulation: Checking Independence and Normality*, SIMULATION 83.10 (2007), pp. 683–694.

[5] V. Dragieva, T. Phung-Duc: *Two-Way Communication $M/M/1//N$ Retrial Queue*, in: International Conference on Analytical and Stochastic Modeling Techniques and Applications, Springer, 2017, pp. 81–94.

[6] G. Falin, J. Artalejo: *A finite source retrial queue*, European Journal of Operational Research 108 (1998), pp. 409–424.

[7] D. Fiems, T. Phung-Duc: *Light-traffic analysis of random access systems without collisions*, Annals of Operations Research 277.2 (2019), pp. 311–327, doi: 10.1007/s10479-017-2636-7.

[8] A. Gómez-Corral, T. Phung-Duc: *Retrial queues and related models*, Annals of Operations Research 247.1 (2016), pp. 1–2, issn: 1572-9338, doi: 10.1007/s10479-016-2305-2.

[9] J. Kim, B. Kim: *A survey of retrial queueing systems*, Annals of Operations Research 247.1 (2016), pp. 3–36, issn: 1572-9338, doi: 10.1007/s10479-015-2038-7.

[10] A. Kuki, J. Sztrik, Á. Tóth, T. Bérczes: *A Contribution to Modeling Two-Way Communication with Retrial Queueing Systems*, in: Information Technologies and Mathematical Modelling. Queueing Theory and Applications, Springer, 2018, pp. 236–247, doi: 10.1007/978-3-319-97595-5_19.

[11] A. M. Law, W. D. Kelton: *Simulation Modeling and Analysis*, McGraw-Hill Education, 1991, isbn: 0-07-100803-9.

[12] A. Nazarov, T. Phung-Duc, S. Paul: *Heavy outgoing call asymptotics for MMP $P/M/1/1$ retrial queue with two-way communication*, in: Information Technologies and Mathematical Modelling. Queueing Theory and Applications, ed. by A. Dudin, A. Nazarov, A. Kirpichnikov, vol. 800, Cham: Springer International Publishing, 2017, pp. 28–41, doi: 10.1007/978-3-319-68069-9_3.

[13] S. Pustova: *Investigation of call centers as retrial queuing systems*, Cybernetics and Systems Analysis 46.3 (2010), pp. 494–499.

[14] H. Sakurai, T. Phung-Duc: *Scaling limits for single server retrial queues with two-way communication*, Ann. Oper. Res. 247.1 (2016), pp. 229–256.

[15] H. Sakurai, T. Phung-Duc: *Two-way communication retrial queues with multiple types of outgoing calls*, Top 23.2 (2015), pp. 466–492.

Annales Mathematicae et Informaticae
58 (2023) pp. 170–181
DOI: 10.33039/ami.2023.11.001
URL: https://ami.uni-eszterhazy.hu

The educational challenges of ChatGPT*

Geoffrey Vaughan[a], **Ádám Kovács**[ab], **Zoltán Szűts**[a]

[a]Eszterházy Károly Catholic University
geoffrey.vaughan@uni-eszterhazy.hu
kovacs2.adam@uni-eszterhazy.hu
szuts.zoltan@uni-eszterhazy.hu

[b]University of Debrecen, Doctoral School of Informatics

Abstract. In 2023, ChatGPT exploded onto the educational scene and started fostering a myriad of research endeavors exploring the utilization of artificial intelligence in education [8, 38]. Many scholars argue that discerning between AI-generated and original academic work has become increasingly challenging, a testament to AI's evolving capability [16]. This paper focuses on the pivotal role of education professionals in mediating the integration of such digital technologies, a key aspect as AI becomes a standard in education. It provides a thoughtful analysis of the prospective advantages and challenges of employing ChatGPT from a techno-realist perspective and formulates precise research questions to evaluate its impact on learning. Our goal is to critically explore whether ChatGPT and related AI technologies serve as assets or disadvantages in education, shedding light on the unavoidable challenges encountered during their incorporation.

Keywords: ChatGPT, artificial intelligence, learning, digital pedagogy

1. Introduction

ChatGPT, an innovation by OpenAI, is a representation of generative artificial intelligence technology, responding to user queries by leveraging Large Language Models (LLM). The rapid assimilation of this technology since its debut has set records in the domain of technological advancements. As reported by Reuters in February 2023, ChatGPT achieved an estimated 100 million active monthly users in January, establishing itself as the fastest proliferating customer application to

*This paper was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

Submitted: July 22, 2023
Accepted: November 7, 2023
Published online: November 9, 2023

date [24, p. 1].

The advent of this technology, alongside comparable developments like Microsoft's Bing and Google's Bard, has ignited extensive discussions and concerns in the educational sector, given the profound implications of these AI-driven tools on learning methodologies and environments.

With the inception of LLMs, the landscape of global education at various levels has undergone significant transformations. Clark elucidates:

> AI is not just a learning technology, it is a technology that learns. AI is transforming the way we live and work, and as it continues to evolve, its impact on learning is only going to become more profound [10, p. 227].

Given the transformative nature of AI, it is crucial to align research focus on pragmatic and evidence-based evaluations of the potential merits and challenges posed by AI-driven technologies like ChatGPT in the learning domain. Our research philosophy adheres to a techno-realist perspective, emphasizing the necessity for an empirical approach to digital pedagogy practices beneficial for the education sector [38].

The fast-paced evolution in this field has prompted an influx of commentary, speculations, and discussions, predominantly in the form of media articles and opinion pieces. Several academic papers, including this one, are in the exploratory phase, deliberating the prospective benefits and challenges, such as [4, 8, 15, 25, 26, 35, 39, 42]. A handful of them [13, 22, 40] have adopted an empirical approach to study learner behavior using ChatGPT, while others have concentrated on the ethical considerations and the impacts on assessment and professional development in higher education [1, 2, 12, 19, 23, 28, 30–32, 35, 41].

This research provides insights and prompts further exploration into the potentials and implications of generative AI technology, emphasizing its significance as knowledge of such innovations becomes widespread.

2. Exploring the dimensions of AI in education

2.1. Advantages

Zhai introduces the positive benefits of using ChatGPT for learning:

> The potent functions of interaction, reasoning, questioning, and feedback showcased by ChatGPT offer novel opportunities for educational transformation [39, p. 1].

Zhou et al. in their investigation of ChatGPT 3.5 against previous versions (3 and InstructGPT) raise a number of potential advantages of ChatGPT, headlined as "Generalization", "Correction", "Safety" and "Creativity" [42, p. 2]. In their other conclusions, the authors emphasise one distinct advantage: "we believe that

ChatGPT is changing the usage of traditional search engines and causes a deep impact on this field" [42, p. 5].

Deng and Lin, whilst focusing on the business benefits and dangers of ChatGPT, also recognise similar advantages: "Increased efficiency", "Improved accuracy" and "Cost savings" [15, p. 82]. Rudolph et al. provide a wealth of evidence both for and against the use of such technology in higher education, especially concerning assessment. In reflecting on the general capacity of ChatGPT for higher education use the author's state, "in the rapidly expanding field of education technology, AIEd represents an opportunity to demonstrate a broad spectrum of tools and applications at an entirely new level" [35, p. 9]. Chen et al. come to a similar conclusion:

> AI systems are likely to be used more widely, which is expected to thrive on all aspects of students, i.e., personal skill, knowledge mastery, learning ability, and career development, instead of just assisting students in understanding specific knowledge [8, p. 14].

2.2. Challenges

The challenges facing education with regards to using ChatGPT are significant, so some schools and districts in the US have reacted with surprising speed and have banned the algorithm [3]. Banning ChatGPT may be because schools do not want students to use ChatGPT to circumvent learning or to share information from dubious sources with others. Indeed, ChatGPT can also write an essay or solve a maths problem on demand. In addition, schools may feel that using ChatGPT can affect students' writing skills and human communication.

Cheating and plagiarism stand out as the most distinctive educational challenges. Cotton et al., in outlining these challenges and providing suggestions for combating them in higher education, conclude that:

> These tools also raise a number of challenges and concerns, particularly in relation to academic honesty and plagiarism. ChatAPIs and GPT-3 can be used to facilitate cheating, and it can be difficult to distinguish between human and machine-generated writing [12, p. 6].

Neumann et al. in their analysis of pre-print (grey) academic literature highlight 5 challenges (C1–5) that ChatGPT poses for education. These are summarised as: "C1: Unknown handling by students, C2: Heterogeneous evaluation C3: Acceptable/unacceptable use of ChatGPT C4: More time-consuming assessments C5: Unknown potential" [31, p. 3]. By analysing from the "grey" literature the authors identify what could be seen as more valid, live concerns at this stage of ChatGPT use.

2.3. Ethical considerations

Ethical considerations of the use of AI generative applications are becoming increasingly significant. Already, many international educational establishments and organisations have drawn up ethical guidelines for the use of Artificial Intelligence in regions, countries, and organisations. The 2021 EU AI Act seeks to regulate the use of AI across Europe. The European Trade Union Committee for Education position paper states five "key demands of education trade unions on the impact of ChatGPT and other generative artificial intelligence on the teaching profession" [17]. The European Universities Association Learning and Teaching Steering Committee [18] released an initial position statement with regard to the use of ChatGPT and similar technologies.

Our research work is informed by local, national, and international ethical and legal developments pertaining to this field and the research team will also act as a partner in the development of organisational, local, national, and international AI policy. These considerations provide the initial focus domains for the research of generative AI technology such as ChatGPT for learning which are proposed below.

3. Research design and methods

Four initial domains frame our research stage with accompanying questions: Teaching, Learning, Assessment, and Evaluation.

Domain 1: Teaching Teachers across all education sectors are under increasing time pressure. Many of the administrative and resource production tasks that currently overburden teachers across all educational phases can be offloaded to AI generative tools. Clark recognises the advantages that teachers can possibly benefit from:

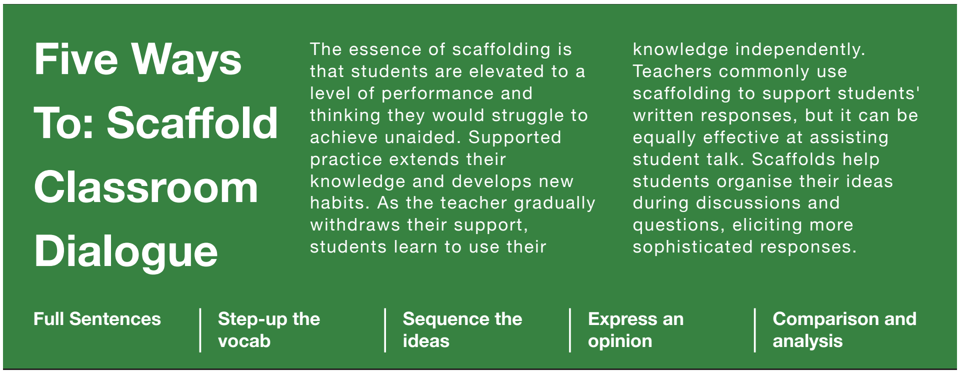
> It is not the case of dispensing with teachers, but reducing workload, giving them support, and raising their game. Teachers should welcome something that takes away the administration and pain, so let us embrace possible solutions [9, p. 75].

Academics and teachers need as many ways to reduce the ineffective administrative burdens to concentrate on the essential nature of pedagogy: helping students learn.

Learning curation is essential to learning. With careful prompting, sophisticated learning content and resources can be produced and curated in a fraction of the time that they would normally have taken teachers to create. Fitzpatrick [20] has called this the "PREP prompting method". Intelligent prompting produces output that can then be checked for accuracy by the expert teacher, curated through the learning management system, and delivered in the classroom saving hours of preparation time. Specific applications that combine text with image and video are already aligning with ChatGPT and other generative chat tools across the major platforms. We are seeing the real-life integration of tested e-learning theory with regards to "multimedia presentation" [11, p. 70].

In addition, where learning occurs is an important factor. With the proliferation of personal smartphones, the mobility of learning has elevated to an unprecedented level, allowing for continuous access to knowledge 24/7, both within and beyond traditional educational environments. In this evolving technological paradigm, parental expertise and guidance become paramount, especially in relation to the application of generative AI tools for educational purposes at home.

Teachers across all sectors need to be provided with effective training and ongoing professional development in using this new technology efficiently and effectively. Academic guides are beginning to be published that address the professional development needs of educators using this technology. Atlas [2] provides a thorough academic guide that would serve academia well in terms of initial professional development training. In England, there has been a flourishing of evidence-informed and research-aligned teacher professional development. Figure 1 shows Sherrington and Goodwin's [37] professional development "Five Ways To..." series, which has seen huge take up from the teaching profession. Training and ongoing professional development about ChatGPT and generative technology must develop teachers' ability to understand and use the technology in a practical, rational manner that supports their professional practice.



Five Ways To: Scaffold Classroom Dialogue

The essence of scaffolding is that students are elevated to a level of performance and thinking they would struggle to achieve unaided. Supported practice extends their knowledge and develops new habits. As the teacher gradually withdraws their support, students learn to use their knowledge independently. Teachers commonly use scaffolding to support students' written responses, but it can be equally effective at assisting student talk. Scaffolds help students organise their ideas during discussions and questions, eliciting more sophisticated responses.

Full Sentences | Step-up the vocab | Sequence the ideas | Express an opinion | Comparison and analysis

Figure 1. Five Ways to: Scaffold Classroom Dialogue.

Research focus 1: Teaching

- How can generative AI technology such as ChatGPT reduce teacher workload?
- How can generative AI technology such as ChatGPT improve teaching content quality and curation to ensure maximum impact for learning?

Domain 2: Learning At the heart of learning is the learner's ability to interact with content. This functionality is starting to appear across the major learning platforms. The new KhanAcademy adaptive technology, Khanmigo and the initial release of Microsoft's CoPilot, which introduces the intelligent tutor into everyday software are obvious examples. Adaptation through the use of AI generative tools and applications is likely to be at the forefront of AI for learning.

ChatGPT can engage each learner at the dialogic level that they are at, and, with the addition of voice activation, the dialogue will be increasingly conversational and realistic. Borkowsky stresses the importance of this dialogic approach to teaching and learning: "it's a very effective way to engage children and focus them on thinking and expressing themselves" [6, p. 59].

This return to a dialogic interaction with knowledge brings the learner far closer to their personal, individualised learning cognitive processes than has been able to be realised previously.

Generative AI technology such as ChatGPT also has the potential to deliver learning to those who have been economically the most difficult to reach, at potentially a minimal cost (a smartphone and an internet connection). Globally, the Organisation for Economic Co-operation and Development [33] reports that: "total public spending on education (from primary to tertiary level) averages 10.6% of total government expenditure across OECD countries". The implications for learning and for a drastically reduced economic investment for helping to support those who face more severe economic disadvantages are huge.

The importance of feedback in learning is crucial for student success. Black and Wiliam make it clear that "formative assessment can be a powerful weapon if it is communicated in the right way" [5, p. 8]. With ChatGPT learner feedback, in the form of formative assessment is immediate, personalised, and relevant, allowing for the learner to immediately see areas that they can improve upon. All these factors are hugely significant to the future of learning; promising to provide truly individual and adaptive learning content to all students.

Research focus 2: Learning

- How can generative AI technology such as ChatGPT be used to improve dialogic learning?
- How can access to generative and adaptive AI tools such as ChatGPT improve personalised learning for specific groups of learners (low socioeconomic status, special educational needs, high ability, EFL/ESL)?
- How can generative AI technology such as ChatGPT improve the quality and personalisation of learner feedback?

Domain 3: Assessment Changing approaches to assessment is a significant consideration in this new era of generative AI technology. Cheating and plagiarism are at the forefront of academic concerns with the adoption and adaptation of this new technology and they are going to remain so as the educational system develops its policies and practices to reflect this new reality. Dehouche details the concerns of plagiarism using the technology and in the conclusion of the opinion piece states:

"the advent of this powerful NLP technology calls for an urgent update of our concepts of plagiarism" [14, p. 22]. Cassidy [7] reports on the measures being taken by Australian universities to ban ChatGPT for assessment and Roose [34] reports on New York City public schools blocking access and Seattle schools restricting access to ChatGPT.

Several countries are now reconsidering a return to oral or pen/paper examinations to negate the possibility of plagiarised student submissions. Sankaranarayanan [36] poses the question, "Is there a place for oral exams in today's fast-paced, hi-tech world?" and refers to European models of oral assessment (the French baccalaureate and Norwegian oral assessment practices). Indeed, the European model of oral assessment and, more locally, the strong Hungarian tradition of oral examination assessment could be poised for an assessment renaissance. Initial research work will look at whether the benefits of more traditional forms of personal examination assessment that have been an important element of the Hungarian education system are a model that could influence international approaches to assessment in the new era of concerns about the use of ChatGPT and assessment.

Ultimately, there may need to be a fundamental reconfiguring of what skills are needed to be assessed, particularly in the domain of written subject content. As Zhai emphasises:

> To meet societal demands and evolving educational objectives, educators must consider innovative assessment tasks and evaluation forms that assess and improve these skills [39, p. 75].

Research focus 3: Assessment

- What are the issues, concerns, and possible solutions for assessment in the era of generative AI tools and applications?
- What, specifically, can the Hungarian education assessment model provide to international assessment research and development to influence assessment practice and policy?

Domain 4: Evaluation The field of education has often suffered from being unable to measure the effectiveness of a new technology or a new intervention. Indeed, it is often the cause of both scepticism and hostility within the profession to any new programme, strategy, or tool. These are valid concerns but they should be addressed through applied research that gives teachers increasing confidence about the use and effectiveness of new technological advances focused on learning outcomes.

Can evaluation of learning become more focused on learning and are there ways to capture learning data in the form of learning analytics that reflect this? A systematic review of empirical studies on learning analytics dashboards proposes a model, MULAS, with the core objective of optimizing learning. The MULAS model aims "to guide developers, researchers, evaluators, and practitioners in their endeavors that aim to understand and optimize learning and environments in which learning occurs" [29, p. 16]. This model to evaluate learning is one that may have real benefits when it comes to evaluating learning improvement through the use of generative AI tools and applications such as ChatGPT.

Education needs to improve its ability to collect realistic and worthwhile data that relates to learning. That data capture may actually be more effective, especially in the initial stage of evaluating new technology, by being collected and analyzed on a smaller scale. Clark states that:

> It is far better to focus on the use of data in adaptive learning or small-scale teaching and learning projects where relatively small amounts of data can be put to good use [9, p. 195].

Data analysis needs to show initial trends that can be analyzed, discussed, and shared by professionals in the education sector who will then be far more actively involved in the process of analysis to inform future pedagogical approaches.

Research focus 4: Evaluation

- How can the use of generative AI technology such as ChatGPT for learning be measured, disseminated, implemented, and evaluated?

This initial research overview has set out an outline of the specific pedagogic advantages and challenges of new generative AI technology such as ChatGPT and how these may be addressed through a research focus. The goal has to be to see if this new technology, which is with us here and now, truly has the ability to improve learning.

4. Results

In our theoretical exploration of the implications and effectiveness of AI generative tools like ChatGPT in education, we sought to answer several pertinent questions regarding their capability to transform educational practices and learning experiences. Our findings propose that tools like ChatGPT can significantly alleviate teacher workload by automating numerous administrative tasks and content curation, allowing educators more time to focus on instructional strategies and student interactions, thus potentially enhancing the efficacy and impact of teaching content.

Additionally, the investigation provided insights into the use of ChatGPT in enhancing dialogic learning, fostering environments where enriched and dynamic dialogues can occur, potentially aiding in the development of a more reflective and cooperative learning environment. By generating nuanced and diverse responses, it can facilitate meaningful interactions between students and between students and educators. Our scrutiny also highlighted the potential of such generative AI technology in creating more personalized and adaptable learning experiences for various groups of learners, such as those with special educational needs, high ability, and

EFL/ESL learners. By addressing individual learning needs and preferences, it offers tailored content and feedback, which is especially crucial for ensuring equitable access to quality education for learners from low socioeconomic backgrounds.

In response to concerns over assessment, our research suggests a critical examination of the validity and fairness of AI-mediated assessments is paramount, with the Hungarian educational assessment model emerging as a potential influencer in international assessment research, offering novel insights and solutions to shape assessment practices and policies in an AI-driven educational landscape.

Furthermore, the analysis underscored the need for a structured approach to measuring, disseminating, implementing, and evaluating the integration of AI tools in learning environments, with emphasis on continual assessment of their efficacy, reliability, and inclusivity. Comparing our theoretical insights with existing studies reveals a shared acknowledgment of the transformative potential and the inherent challenges of incorporating AI generative tools in education, and emphasizes the urgency for studies to unravel the complexities of AI integration in varied educational scenarios.

These assertions, grounded in theoretical analysis and literature, echo the necessity for nuanced discussions and rigorous empirical studies to validate the proposed benefits and to address the challenges of integrating AI tools in education, paving the way for informed implementations and ethical practices in AI-driven educational interventions.

5. Conclusion

The exploration and theoretical analysis of newly introduced AI technologies, particularly exemplified by ChatGPT, illuminate their transformative potential in the realm of education, whilst also bringing forth substantial challenges. The nuances of this technological evolution have sparked varied perspectives and dialogues within the academic and tech communities. Marcus [27] has vociferously critiqued the swift proliferation of generative AI chat applications by tech giants since the unveiling of ChatGPT in November 2022, emphasizing the urgent need for transparency and ethical considerations in their deployment. Conversely, Gates [21] envisions a future where AI-driven software significantly revolutionizes teaching and learning methodologies within the next decade.

The detailed theoretical exploration conducted in this paper has aimed to bridge the gap between optimistic projections and critical viewpoints, offering a balanced, techno-realist perspective on the opportunities and impediments presented by ChatGPT and similar innovations. It has underlined the imperative for the education sector to engage in informed, evidence-based discussions and strategies to navigate the intricate landscape of AInlightenment effectively. Our findings, grounded in theoretical conjectures and existing literature, serve as a precursor to more rigorous empirical research required to validate the implications and challenges of integrating AI in diverse educational settings. The highlighted potential benefits and challenges necessitate extensive, focused, and meticulous research endeavors to foster a comprehensive and nuanced understanding of the implications of intertwining AI with education.

In conclusion, this paper's theoretical discourse seeks to contribute to the ongoing dialogue on AI's role in education, emphasizing the need for empirical validations and robust academic discussions to refine the understanding of the multifaceted impacts of AI tools in educational environments.

References

[1] J. K. M. Ali, M. A. A. Shamsan, T. A. Hezam, A. A. Q. Mohammed: *Impact of ChatGPT on Learning Motivation: Teachers and Students' Voices*, Journal of English Studies in Arabia Felix 2.1 (Mar. 2023), pp. 41–49, doi: 10.56540/jesaf.v2i1.51.

[2] S. Atlas: *ChatGPT for Higher Education and Professional Development: A Guide to Conversational AI* (Jan. 2023), url: https://digitalcommons.uri.edu/cba_facpubs/548.

[3] D. Avery: *New York City Schools Ban ChatGPT Amid Cheating Worries*, Jan. 2023, url: https://www.cnet.com/tech/computing/new-york-city-schools-ban-chatgpt-amid-cheating-worries/?fbclid=IwAR1IoRK6yvLuD86cBylf8JHNjdJUM-5nhm_Bx0OHd2jLsolPdIUSqv2DTMw (visited on 03/31/2023).

[4] Ö. Aydin, E. Karaarslan: *Is ChatGPT Leading Generative AI? What is Beyond Expectations?* (Jan. 2023), doi: 10.2139/ssrn.4341500.

[5] P. Black, D. Wiliam: *Inside the Black Box: Raising Standards through Classroom Assessment*, Phi Delta Kappan 92 (Sept. 2010), pp. 81–90, doi: 10.1177/003172171009200119.

[6] F. Borkowsky: *"If Only I Would Have Known. . . ": What I wish the Pediatrician would have told me about Language, Literacy, and Dyslexia*, New York: High Five Literacy Publishing, 2019, isbn: 9781734068801.

[7] C. Cassidy: *Australian universities to return to 'pen and paper' exams after students caught using AI to write essays*, Jan. 2023, url: https://www.theguardian.com/australia-news/2023/jan/10/universities-to-return-to-pen-and-paper-exams-after-students-caught-using-ai-to-write-essays (visited on 03/31/2023).

[8] L. Chen, P. Chen, Z. Lin: *Artificial Intelligence in Education: A Review*, IEEE Access 8 (2020), pp. 75264–75278, doi: 10.1109/ACCESS.2020.2988510.

[9] D. Clark: *Artificial Intelligence for Learning: How to use AI to Support Employee Development*, London, New York, Daryaganj: Kogan Page, 2020, isbn: 9781789660821, url: https://books.google.hu/books?id=bhn0DwAAQBAJ.

[10] D. Clark: *Learning Technology (1st ed.)* London, New York, Daryaganj: Kogan Page, 2023, isbn: 9781398608764, url: https://books.google.hu/books?id=7IChEAAAQBAJ.

[11] R. C. Clark, R. E. Mayer: *e-Learning and the Science of Instruction (4th ed.)* Hoboken: John Wiley & Sons, Ltd, 2016, chap. 4, pp. 67–87, isbn: 9781119239086, doi: 10.1002/9781119239086.ch4.

[12] D. R. E. Cotton, P. A. C. Cotton, J. R. Shipway: *Chatting and cheating: Ensuring academic integrity in the era of ChatGPT*, Innovations in Education and Teaching International 0.0 (Mar. 2023), pp. 1–12, doi: 10.1080/14703297.2023.2190148.

[13] J. Crawford, M. Cowling, K.-A. Allen: *Leadership is needed for ethical ChatGPT: Character, assessment, and learning using artificial intelligence (AI)*, Journal of University Teaching and Learning Practice (JUTLP) 20 (2023), doi: 10.53761/1.20.3.02.

[14] N. Dehouche: *Plagiarism in the age of massive Generative Pre-trained Transformers (GPT-3)*, Ethics in Science and Environmental Politics 21 (Mar. 2021), pp. 17–23, doi: 10.3354/esep00195.

[15] J. Deng, Y. Lin: *The Benefits and Challenges of ChatGPT: An Overview*, Frontiers in Computing and Intelligent Systems 2.2 (Jan. 2023), pp. 81–83, doi: 10.54097/fcis.v2i2.4465.

[16] H. Else: *Abstracts written by ChatGPT fool scientists*, Nature 613.7944 (2023), p. 423, doi: 10.1038/d41586-023-00056-7.

[17] European Trade Union Committee: *The impact of ChatGPT on the teaching profession*, Mar. 2023, url: https://www.csee-etuce.org/en/news/etuce/5128-the-impact-of-chatgpt-on-the-teaching-profession (visited on 03/31/2023).

[18] European University Association's Learning and Teaching Steering Committee: *Artificial intelligence tools and their responsible use in higher education learning and teaching*, Feb. 2023, url: https://eua.eu/resources/publications/1059:artificial-intelligence-tools-and-their-responsible-use-in-higher-education-learning-and-teaching.html (visited on 03/31/2023).

[19] R. Firaina, D. Sulisworo: *Exploring the Usage of ChatGPT in Higher Education: Frequency and Impact on Productivity*, Buletin Edukasi Indonesia 2.1 (Mar. 2023), pp. 67–74, doi: 10.56741/bei.v2i01.310.

[20] D. Fitzpatrick: *AI Won't Replace Teachers, but it will replace teachers who don't use AI.* Feb. 2023, url: https://www.linkedin.com/pulse/ai-wont-replace-teachers-who-dont-use-dan-fitzpatrick/?trk=pulse-article (visited on 03/31/2023).

[21] B. Gates: *The Age of AI has begun*, Mar. 2023, url: https://www.gatesnotes.com/The-Age-of-AI-Has-Begun (visited on 03/31/2023).

[22] D. Gruda, J. A. Schermer: *C-H-A-T-G-P-T, Find Out What it Means To Me: ChatGPT, Artificial Intelligence, and the Study of Individual Differences* (Mar. 2023), doi: 10.31234/osf.io/5ctfq.

[23] M. Halaweh: *ChatGPT in education: Strategies for responsible implementation*, Contemporary Educational Technology 15 (Mar. 2023), doi: 10.30935/cedtech/13036.

[24] K. Hu: *ChatGPT sets record for fastest-growing user base - analyst note*, Feb. 2023, url: https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/#:~:text=Feb%5C%201%5C%20(Reuters)%5C%20%5C%2D%5C%20ChatGPT,a%5C%20UBS%5C%20study%5C%20on%5C%20Wednesday. (visited on 03/31/2023).

[25] C. Karthikeyan: *Literature Review on Pros and Cons of ChatGPT Implications in Education*, International Journal of Science and Research (IJSR) 12 (Mar. 2023), pp. 283–291, doi: 10.21275/SR23219122412.

[26] E. Kasneci et al.: *ChatGPT for good? On opportunities and challenges of large language models for education*, Learning and Individual Differences 103 (Jan. 2023), p. 102274, doi: 10.35542/osf.io/5er8f.

[27] G. Marcus: *The Sparks of AGI? Or the End of Science?*, Mar. 2023, url: https://garymarcus.substack.com/p/the-sparks-of-agi-or-the-end-of-science (visited on 03/31/2023).

[28] R. Marusenko: *New challenges in assessing students' knowledge: chatbot ChatGPT and real-time deepfakes* (Feb. 2023), doi: 10.13140/RG.2.2.21715.25120.

[29] W. Matcha, N. A. Uzir, D. Gašević, A. Pardo: *A Systematic Review of Empirical Studies on Learning Analytics Dashboards: A Self-Regulated Learning Perspective*, IEEE Transactions on Learning Technologies 13.2 (2020), pp. 226–245, doi: 10.1109/TLT.2019.2916802.

[30] D. Mhlanga: *Open AI in Education, the Responsible and Ethical Use of ChatGPT Towards Lifelong Learning*, SSRN Electronic Journal (Feb. 2023), doi: 10.2139/ssrn.4354422.

[31] M. Neumann, M. Rauschenberger, E.-M. Schön: *"We Need To Talk About ChatGPT": The Future of AI and Higher Education* (Mar. 2023), p. 4, doi: 10.25968/opus-2467.

[32] E. C. Opara, A. M.-E. Theresa, C. A. Tolorunleke: *ChatGPT for Teaching, Learning and Research: Prospects and Challenges*, Glob Acad J Humanit Soc Sci 5 (Mar. 2023), pp. 33–40, doi: 10.36348/gajhss.2023.v05i02.001.

[33] Organisation for Economic Co-operation and Development: *Education at a Glance 2022: OECD Indicators*, Paris: OECD Publishing, 2022, isbn: 9789264341647, doi: 10.1787/3197152b-en.

[34] K. Roose: *Don't Ban ChatGPT in Schools. Teach With It.* Jan. 2023, url: https://www.nytimes.com/2023/01/12/technology/chatgpt-schools-teachers.html (visited on 03/31/2023).

[35] J. Rudolph, S. Tan, S. Tan: *ChatGPT: Bullshit spewer or the end of traditional assessments in higher education?*, Frontiers of Information Technology & Electronic Engineering (Jan. 2023), doi: 10.37074/jalt.2023.6.1.9.

[36] A. Sankaranarayanan: *Should we have more oral exams?*, Feb. 2023, url: https://www.thehindu.com/education/should-we-have-more-oral-exams/article66461975.ece (visited on 03/31/2023).

[37] T. Sherrington, D. Goodwin: *Five Ways to: Scaffold Classroom Dialogue*, Dec. 2021, url: https://teacherhead.com/2021/12/01/five-ways-to-scaffold-classroom-dialogue/ (visited on 03/31/2023).

[38] Z. Szűts: *A digitális pedagógia elmélete*, Budapest: Akadémiai Kiadó, 2020.

[39] X. Zhai: *ChatGPT and AI: The Game Changer for Education* (Mar. 2023), doi: 10.13140/RG.2.2.31107.37923.

[40] X. Zhai: *ChatGPT User Experience: Implications for Education* (Dec. 2022), doi: 10.2139/ssrn.4312418.

[41] B. Zhang: *Preparing Educators and Students for ChatGPT and AI Technology in Higher Education:Benefits, Limitations, Strategies, and Implications of ChatGPT & AI Technologies* (Jan. 2023), doi: 10.13140/RG.2.2.32105.98404.

[42] J. Zhou, P. Ke, X. Qiu, M. Huang, J. Zhang: *ChatGPT: potential, prospects, and limitations*, Frontiers of Information Technology & Electronic Engineering (Feb. 2023), doi: 10.1631/FITEE.2300089.

Annales Mathematicae et Informaticae
58 (2023) pp. 182–190
DOI: 10.33039/ami.2023.08.008
URL: https://ami.uni-eszterhazy.hu

Enhancing machine translation with quality estimation and reinforcement learning

Zijian Győző Yang, László János Laki

Hungarian Research Centre for Linguistics
{yang.zijian.gyozo,laki.laszlo}@nytud.hu

Abstract. In recent times, our research has focused on training large language models and exploring their potential. With the emergence of ChatGPT, it has been demonstrated that it is possible to fine-tune language models in a task-agnostic way. The success of ChatGPT is attributed to the reinforcement learning method, which integrates human feedback into the language model fine-tuning process. As a part of our research, we initially adapted the method of reinforcement learning for a specific task, which is machine translation, respectively. In this paper, we propose a novel approach to enhance machine translation with reinforcement learning and quality estimation methods. Our proposed approach uses reinforcement learning to learn to adjust the machine translation output based on quality estimation feedback, with the goal of improving the overall translation quality. We evaluated our approach on the WMT09 dataset for English-Hungarian language pair. We conducted an analysis to show how our approach improves the quality of machine translation output. Our approach offers a promising avenue for enhancing the quality of machine translation and demonstrates the potential of utilizing reinforcement learning to improve other natural language processing tasks.

Keywords: machine translation, reinforcement learning, quality estimation, mT5

AMS Subject Classification: 68T07, 68T50

1. Introduction

In recent years, significant progress in artificial intelligence and deep learning has resulted in notable enhancements in the quality of machine translation. Quality

Submitted: July 27, 2023
Accepted: August 7, 2023
Published online: August 19, 2023

estimation has become an important task in the field, involving predicting the quality of machine-translated text without having access to a reference translation. Incorporating a real-time quality estimation system is a crucial step in the machine translation pipeline, as it enables the system to determine the most accurate translation and select the best one to present to the user. In the past month, reinforcement learning has been adopted into natural language processing tasks, marking a significant advancement in this field. In the context of machine translation, reinforcement learning has been applied to fine-tune machine translation models and integrate human feedback into the training process. By combining reinforcement learning and quality estimation, machine translation systems can deliver higher-quality translations.

The success of ChatGPT[1] has demonstrated that reinforcement learning can be effectively adopted in human language technology. ChatGPT suggests that language models can be fine-tuned in a task-agnostic manner. This not only stabilizes the non-deterministic behavior of the models but also brings to light their vast knowledge about the world. The ChatGPT system is fine-tuned from a model in the GPT-3.5 series[2]. The GPT-3.5 series belongs to Large Language Models (LLM) [1], which has garnered significant attention and popularity in the field of research in recent times. The enormous success of ChatGPT can be attributed to the utilization of reinforcement learning (RL), a technique that incorporates human feedback into the language modeling process.

Following the popular trend, we have also started training the Hungarian ChatGPT, although this process is extremely time-consuming. Therefore, as a preliminary step, we experimented with the RL in the field of machine translation (MT). In our current research, we have successfully incorporated a neural quality estimation (QE) model and the RL method into the MT training process to enhance its quality.

2. Related works

The OpenAI was the first, who successfully integrated the RL approach to the natural language processing training process [14, 22]. The first experiments were done with the summarization task. Thereafter, the RL was adapted to InstructGPT [11], which is the basis of ChatGPT. There are many algorithms in modern RL, but in natural language processing currently the Proximal Policy Optimization (PPO) [13] algorithm became decisive.

Reinforcement learning experiments is still in its early stages in machine translation task. There are studies [18] that show RL is an effective approach for improving the performance of neural machine translation. There have been some studies with skepticism in the field as well [2, 7]. For English-Hungarian language pair, Laki and Yang [8] conducted comprehensive research on machine translation, however reinforcement learning method has not been applied yet.

[1]https://chat.openai.com
[2]https://openai.com/blog/chatgpt

The reward model in the reinforcement learning method can be trained as a QE model in the MT task. Quality estimation is a prediction task, where different quality indicators are extracted from the source and the machine translated segments. The QE model is built with machine learning methods based on these quality indicators. Then the QE model is used to predict the quality of unseen translations [20]. In the recent years, instead of human feature extraction, neural based deep learning methods are used for this task. Since the QE compares two texts from different languages, pretrained multilingual neural language models [12, 15] or dual encoders [6] are used for this task. The pretrained language models can be combined with Multitask Learning architectures [5, 9], or additional custom extracted features can be added to the model [17, 21].

3. Methods and experiments

OpenAI showed in its research [10] that using reinforcement learning, we can enhance the performance of a neural language model. Based on research (see Figure 1) by OpenAI, our implementation steps for using reinforcement learning in fine-tuning language models are as follows:

1. *Fine-tuning language model with supervised learning*: In our experiment, we used a mT5 small model[3] [8], that fine-tuned for English-Hungarian translation task (supervised fine-tuned model - SFT).

2. *Collect human feedback and training a reward model*: For this task, we trained a QE model as reward model for English-Hungarian translation.

3. *Fine-tuning language model with reward model and reinforcement learning method*: We further fine-tuned the SFT-mT5 model with reward model and reinforcement learning method (RL).

In the first step, we utilized an already fine-tuned language model, hence we did not train a new model specifically for this task.

In the second step, we conducted experiments using five different models to train QE models:

- mT5 models: Following the research conducted by OpenAI, we initially performed fine-tuned our SFT-mT5 model. Then, conducted experiments using the original mT5-small and mT5-base [19] models.

- mBERT: the BERT multilingual base [4] model was fine-tuned.

- XLM-R: the XLM-RoBERTa-base [3] model was fine-tuned.

To train the QE models we used the HuQ [20] corpus that contains 1500 manual evaluated English-Hungarian sentence pairs. All the 1500 sentences were evaluated

[3]https://huggingface.co/NYTK/translation-mt5-small-128-en-hu

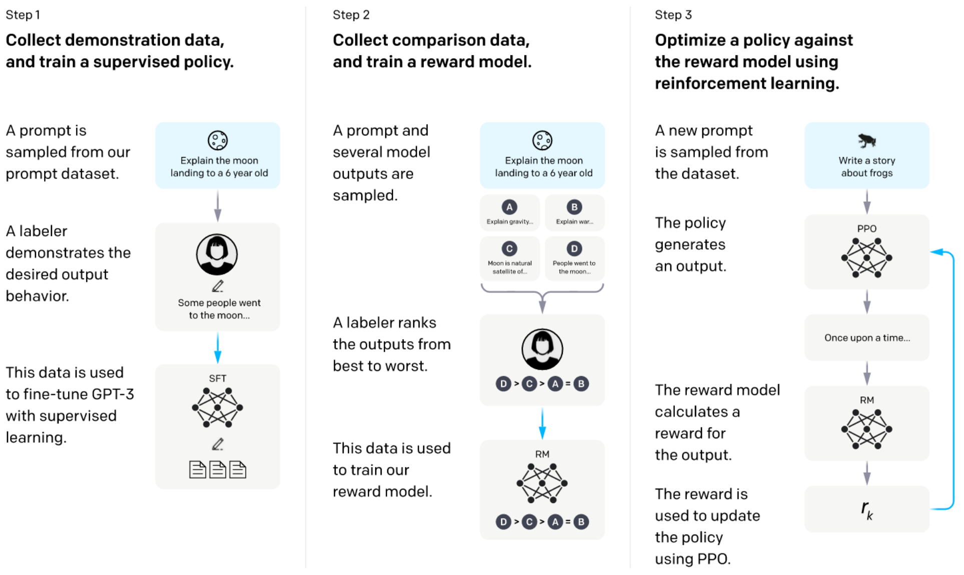

Step 1
Collect demonstration data, and train a supervised policy.
A prompt is sampled from our prompt dataset.
Explain the moon landing to a 6 year old
A labeler demonstrates the desired output behavior.
Some people went to the moon...
This data is used to fine-tune GPT-3 with supervised learning.
SFT
Step 2
Collect comparison data, and train a reward model.
A prompt and several model outputs are sampled.
Explain the moon landing to a 6 year old
A Explain gravity...
B Explain war...
C Moon is natural satellite of...
D People went to the moon...
A labeler ranks the outputs from best to worst.
D > C > A = B
This data is used to train our reward model.
RM
D > C > A = B
Step 3
Optimize a policy against the reward model using reinforcement learning.
A new prompt is sampled from the dataset.
Write a story about frogs
The policy generates an output.
PPO
Once upon a time...
The reward model calculates a reward for the output.
RM
The reward is used to update the policy using PPO.
r_k

Figure 1. Using reinforcement learning in fine-tuning language models. [10]

by 3 human annotators. provided quality scores ranging from 1 to 5, considering adequacy and fluency aspects. For the experiments, we randomly shuffled the segments and divided them into 80% for the train sub-corpus and 20% for the test sub-corpus.

In the third step, we employed the CarperAI implementation[4] to fine-tune our SFT model using reinforcement learning (see Figure 2).

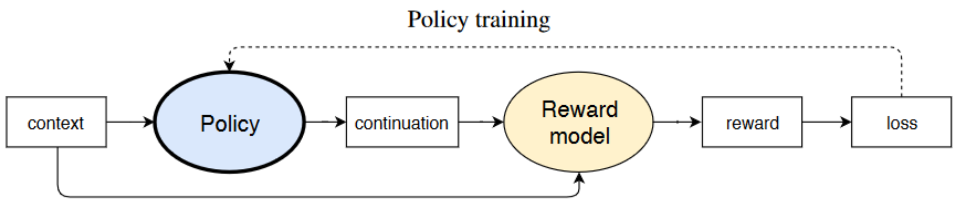


Figure 2. Training process for policy. [23]

We used our fine-tuned QE model as the reward model. For training and testing purposes, we used the official sub-corpora of Hunglish [16] corpus from Shared Task of WMT 2009[5]. In the case of reinforcement learning, a smaller amount of training data is sufficient, thus we used the development set as the training

[4]https://github.com/CarperAI/trlx
[5]https://www.statmt.org/wmt09/translation-task.html

corpus. To ensure a fair comparison, we also conducted a fine-tuning experiment where we further fine-tuned the SFT model (SFT-mT5 FFT) using traditional methods without reinforcement learning on the same development set with the same hyperparameters. The main hyperparameters used in our fine-tuning experiments (both RL and FFT) are as follows: learning rate: 2e-5; sequence length: 256; epoch: 10;

4. Results

In Table 1, you can see the results of our QE experiments. The mT5 models were unable to effectively perform the regression task, which means that a regression task may not be well-suited for a sequence-to-sequence approach. However the encoder-only multilingual models could solve this task with high performance. The XLM-R model could gain the highest correlation result. To provide a better comparison with previous research in this field, we conducted a 10-fold cross-validation with the XLM-R model and compared it with the baseline model (as shown in Table 2) from the research of Yang et. al [20]. Our XLM-R model achieved state-of-the-art results in the English-Hungarian QE task. In our test set, XLM-R achieved an 83.8% correlation. Refer to Figure 3 (left side) for the correlation diagram.

Table 1. Results of the quality estimation task.

	Correlation	MAE	RMSE
ChatGPT	-0.0150	1.3072	1.5698
mT5-small	0.3422	1.0794	1.5059
SFT-mT5-small	0.3809	1.0156	1.4339
mT5-base	0.4579	0.9294	1.3016
mBERT	0.7358	0.64836	0.8950
XLM-R	**0.8382**	**0.5785**	**0.8184**

We conducted an experiment to test the ChatGPT (gpt-3.5-turbo) model [1]. The prompt template we used in this experiment is as follows:

- role: system
- content: You are a quality estimator system, which rate a given translation how good it is based on the original source sentence. Rate the translation quality between 0 to 5, where 5 is a perfect translation.
- role: user
- content: Source sentence: {src} \n Translation: {trans} \n Score:

In the prompt template above, {src} represents the source sentence, and {trans} represents the translated sentence.

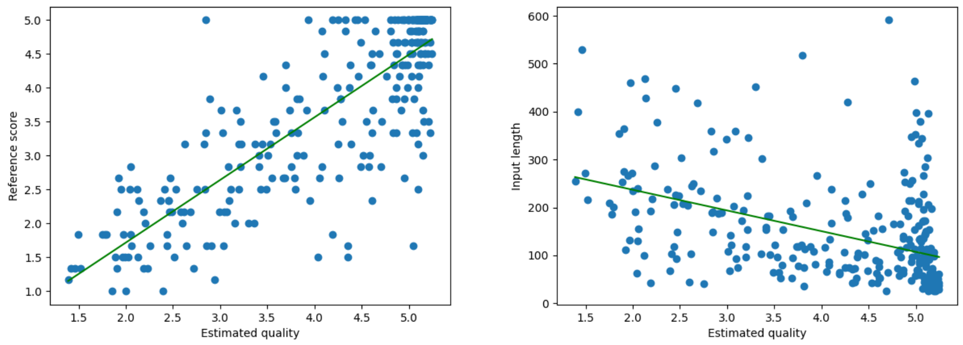


Figure 3. Correlation diagrams.

As we can see from the results, the text-davinci-003 model could not solve this problem. However, more prompt experiments could have been conducted.

Table 2. Results of the quality estimation task with 10-fold cross validation.

	Correlation	MAE	RMSE
baseline	0.6100	0.7459	0.9775
XLM-R	**0.7948**	**0.6451**	**0.8898**

In Figure 3 (right side), you can see the correlation between the estimated quality score and the input text length. The correlation is -0.4478, which means they are not correlated. The length of input does not affect the quality.

In Table 3, the results of SFT-mT5 and 'SFT-mT5 FFT' and 'SFT-mT5 RL' MT models are presented. Our 'SFT-mT5 RL' model could significantly outperform the SFT-mT5 model (>5 BLEU score). To provide a better comparision, we also fine-tuned the SFT-mT5 model using the traditional method. Fine-tuning a model on an out-of-domain corpus result in decreased performance on the original corpus. In Table 4, you can see the expected lower performance of the fine-tuned (FFT and RL) models. As you can see in Table 3 and Table 4, the RL model outperformed the FFT model in both the original corpus and the WMT09 corpus. It means that the RL model was able to adapt the new WMT09 corpus with higher performance, achieving the highest results on the new WMT09 corpus while while only slightly decreasing in performance on the original corpus.

The original 'SFT-mT5' model faced challenges such as generating outputs that were longer than the source text and containing incorrect repeated phrases (as demonstrated in Table 5): 'a hétvégén, a hétvégén' (this weekend, this weekend)). This led to high recall results but low precision. However by utilizing a human-based reward model and reinforcement learning method, we were able to correct these issues (as evidenced in Table 5) with the 'SFT-mT5 RL' model. Additionally, the 'SFT-mT5 FFT' model suffered from information loss ('She has good instincts nonetheles’) during translation.

Table 3. Results of FFT and RL models on WMT09.

	BLEU	chrF-3	chrF-6
SFT-mT5	7.34	45.60	38.62
SFT-mT5 FFT	12.59	46.17	39.65
SFT-mT5 RL	**12.91**	**47.02**	**40.39**

Table 4. Results of FFT and RL models on the original test set.

	BLEU	chrF-3	chrF-6
SFT-mT5	27.69	53.73	48.57
SFT-mT5 FFT	25.85	52.61	47.42
SFT-mT5 RL	26.72	53.32	48.20

Table 5. A translation sample of the different models.

Source	She has good instincts nonetheless, warned Bill Clinton this weekend.
Reference	Bill Clinton azonban így figyelmeztetett a hétvégén: Hiba lenne Palint alulbecsülni.
SFT-mT5	Ennek ellenére jó ösztönei vannak – figyelmeztette Bill Clinton a hétvégén, a hétvégén.
SFT-mT5 FFT	A hétvégén Bill Clinton is figyelmeztette.
SFT-mT5 RL	**Azonban jó ösztönei vannak – figyelmeztette Bill Clinton a hétvégén.**

5. Conclusion

In our research, we have successfully adapted the reinforcement learning method to the machine translation task. We trained a neural quality estimation model as a reward model. Using the XLM-RoBERTa multilingual model, we achieved state-of-the-art results in Hungarian quality estimation task. For fine-tuning a language model with reinforcement learning approach, we have used an already fine-tuned mT5 model that trained for English-Hungarian machine translation task. In our experiments, we have demonstrated that reinforcement learning method can effectively enhance the performance of machine translation task by correcting the subtle errors and mistakes.

For future work, we would like to explore machine translation experiments using multilingual large language models, further extending our research in this area.

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei: *Language Models are Few-Shot Learners*, in: Advances in Neural Information Processing Systems, ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin, vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901.

[2] L. Choshen, L. Fox, Z. Aizenbud, O. Abend: *On the Weaknesses of Reinforcement Learning for Neural Machine Translation*, in: International Conference on Learning Representations, 2020.

[3] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, V. Stoyanov: *Unsupervised Cross-lingual Representation Learning at Scale*, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online: Association for Computational Linguistics, July 2020, pp. 8440–8451, doi: 10.18653/v1/2020.acl-main.747.

[4] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova: *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186, doi: 10.18653/v1/N19-1423, url: https://aclanthology.org/N19-1423.

[5] X. Geng, Y. Zhang, S. Huang, S. Tao, H. Yang, J. Chen: *NJUNLP's Participation for the WMT2022 Quality Estimation Shared Task*, in: Proceedings of the Seventh Conference on Machine Translation (WMT), Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, Dec. 2022, pp. 615–620.

[6] D. Heo, W. Lee, B. Jung, J.-H. Lee: *Quality Estimation Using Dual Encoders with Transfer Learning*, in: Proceedings of the Sixth Conference on Machine Translation, Online: Association for Computational Linguistics, Nov. 2021, pp. 920–927.

[7] S. Kiegeland, J. Kreutzer: *Revisiting the Weaknesses of Reinforcement Learning for Neural Machine Translation*, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online: Association for Computational Linguistics, June 2021, pp. 1673–1681, doi: 10.18653/v1/2021.naacl-main.133.

[8] L. J. Laki, Z. G. Yang: *Neural machine translation for Hungarian*, Acta Linguistica Academica 69.4 (2022), pp. 501–520, doi: 10.1556/2062.2022.00576.

[9] S. Lim, H. Kim, H. Kim: *Papago's Submission for the WMT21 Quality Estimation Shared Task*, in: Proceedings of the Sixth Conference on Machine Translation, Online: Association for Computational Linguistics, Nov. 2021, pp. 935–940.

[10] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Gray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, R. Lowe: *Training language models to follow instructions with human feedback*, in: Advances in Neural Information Processing Systems, ed. by A. H. Oh, A. Agarwal, D. Belgrave, K. Cho, 2022.

[11] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, R. Lowe: *Training language models to follow instructions with human feedback*, in: Advances in Neural Information Processing Systems, ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh, vol. 35, Curran Associates, Inc., 2022, pp. 27730–27744.

[12] R. Rei, M. Treviso, N. M. Guerreiro, C. Zerva, A. C. Farinha, C. Maroti, J. G. C. de Souza, T. Glushkova, D. Alves, L. Coheur, A. Lavie, A. F. T. Martins: *CometKiwi: IST-Unbabel 2022 Submission for the Quality Estimation Shared Task*, in: Proceedings of the Seventh Conference on Machine Translation (WMT), Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, Dec. 2022, pp. 634–645.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov: *Proximal Policy Optimization Algorithms*, CoRR abs/1707.06347 (2017), arXiv: 1707.06347.

[14] N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, P. F. Christiano: *Learning to summarize with human feedback*, in: Advances in Neural Information Processing Systems, ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin, vol. 33, Curran Associates, Inc., 2020, pp. 3008–3021.

[15] S. Tao, S. Chang, M. Miaomiao, H. Yang, X. Geng, S. Huang, M. Zhang, J. Guo, M. Wang, Y. Li: *CrossQE: HW-TSC 2022 Submission for the Quality Estimation Shared Task*, in: Proceedings of the Seventh Conference on Machine Translation (WMT), Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, Dec. 2022, pp. 646–652.

[16] D. Varga, P. Halacsy, A. Kornai, V. Nagy, L. Nemeth, V. Tron: *Parallel corpora for medium density languages*, in: Recent Advances in Natural Language Processing IV. Selected papers from RANLP-05, ed. by N. Nicolov, K. Bontcheva, G. Angelova, R. Mitkov, Amsterdam: Benjamins, 2007, pp. 247–258.

[17] J. Wang, K. Wang, B. Chen, Y. Zhao, W. Luo, Y. Zhang: *QEMind: Alibaba's Submission to the WMT21 Quality Estimation Shared Task*, in: Proceedings of the Sixth Conference on Machine Translation, Online: Association for Computational Linguistics, Nov. 2021, pp. 948–954.

[18] L. Wu, F. Tian, T. Qin, J. Lai, T.-Y. Liu: *A Study of Reinforcement Learning for Neural Machine Translation*, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 3612–3621, doi: 10.18653/v1/D18-1397.

[19] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, C. Raffel: *mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer*, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online: Association for Computational Linguistics, June 2021, pp. 483–498, doi: 10.18653/v1/2021.naacl-main.41, url: https://aclanthology.org/2021.naacl-main.41.

[20] Z. G. Yang, J. L. Laki, B. Siklósi: *HuQ: An English-Hungarian Corpus for Quality Estimation*, in: Proceedings of the LREC 2016 Workshop - Translation Evaluation: From Fragmented Tools and Data Sets to an Integrated Ecosystem (Portorož, Slovenia, May 24, 2016), 2016.

[21] C. Zerva, D. van Stigt, R. Rei, A. C. Farinha, P. Ramos, J. G. C. de Souza, T. Glushkova, M. Vera, F. Kepler, A. F. T. Martins: *IST-Unbabel 2021 Submission for the Quality Estimation Shared Task*, in: Proceedings of the Sixth Conference on Machine Translation, Online: Association for Computational Linguistics, Nov. 2021, pp. 961–972.

[22] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, G. Irving: *Fine-Tuning Language Models from Human Preferences*, arXiv preprint arXiv:1909.08593 (2019).

[23] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, G. Irving: *Fine-Tuning Language Models from Human Preferences*, 2020, arXiv: 1909.08593 [cs.CL].