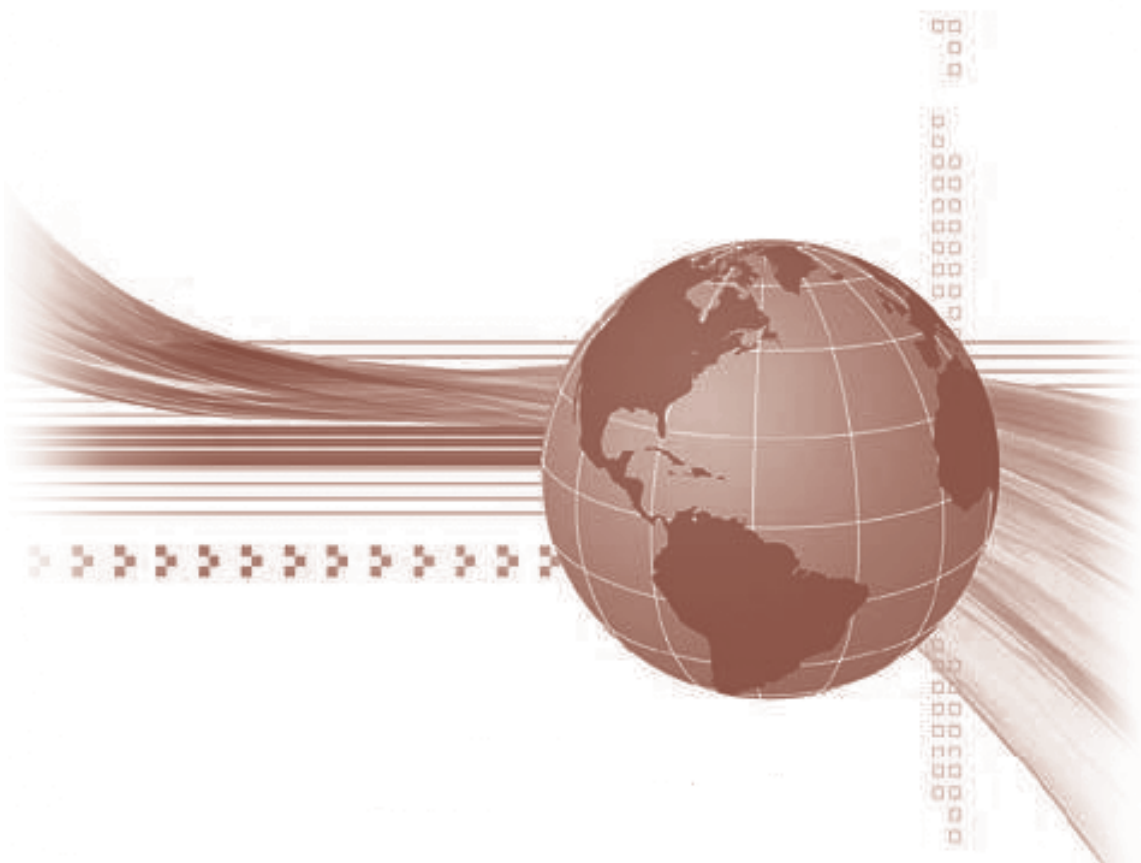
STUDIA UNIVERSITATIS
BABEŞ-BOLYAI



INFORMATICA

1/2020

STUDIA

UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

No. 1/2020

January - June

EDITORIAL BOARD

YEAR **VOLUME 65 (LXV) 2020**
MONTH **JUNE**
ISSUE **1**

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

1

EDITORIAL OFFICE: M. Kogălniceanu 1 • 400084 Cluj-Napoca • Tel: 0264.405300

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXV**, Number 1, 2020
DOI: 10.24193/subbi.2020.1.01

EMPIRICAL VALIDATION OF OO METRICS IMPACT ON CHANGEABILITY

DANA CIUBĂNCAN AND PAUL ŢIRBAN

Abstract. In the context of current software development, where changes to an application are made from one sprint to another, it's more and more necessary for developers to be able to easily change the existing code. Starting from the existing literature and using two commercial projects with multiple versions and different architecture we are trying to confirm a correlation between a possible changeability indicator and variations of OO metrics. In our research we managed to provide empirical evidence of the relation between certain OO metrics that can be computed for a system and the changeability quality attribute.

1. Introduction

Nowadays in a software industry, where agile methodologies and fast increments of the software are written, the changes of the existing code are more and more frequent and time consuming. The challenge that developers face more and more is how to change easier and faster existing code of the projects. The challenge for the project managers is to estimate and, if possible, reduce the cost of system changes.

Changeability, sometimes referred as modifiability, had been considered as a software quality characteristics or subcharacteristics in almost all software quality models, proving the importance and impact it has on the overall quality of a software system. Several such models (as Boehm [5], ISO 9126 [3], and the current standard ISO 25010 [12]), consider changeability as a subcharacteristic of maintainability. However, our investigation has identified that despite its importance, these models do not offer ways to improve this factor or numerical values, respectively ranges, to evaluate changeability.

Received by the editors: 13 November 2019.

2010 *Mathematics Subject Classification.* 68N30.

1998 *CR Categories and Descriptors.* D.2.8. **Software Engineering**: Metric — *Product Metrics*.

Key words and phrases. Software Quality, Changeability, Object Oriented Metrics.

Many studies [8, 9, 7, 15] are focusing on the topic of changeability that would help developers work with existing code easier and further develop it without making it more error prone.

The purpose of this work is to study the evolution of Changeability in the context of large software systems, that have been developed by teams (several developers), over a significant period of time, through the relation to software metrics. This objective has been split into the following research objectives:

- *RO1*: provide empirical evidence of the relation between software metrics and changeability;
- *RO2*: examine if the software architecture has an impact on changeability.
- *RO3*: investigate the relation between metrics, changeability over the development period (namely, analyze several versions of the application)

The rest of the paper is organized as follows: the next section details changeability as a software quality factor and the software metrics used in the study, and presents similar research approaches. Section 3 describes the methodology used, the case study and analyze the obtained results, while the last section states some conclusions and suggests future investigations.

2. Preliminaries and Related Work

2.1. **Changeability as Software Quality Factor.** Changeability, also referred to as modifiability in some contexts, has been given a lot of definitions, amongst which are also the following:

- "how well software can be changed" [18]
- "the cost of change and refers to the ease with which a software system can accommodate changes"[20]
- "the ease with which it can be modified to changes in the environment, requirements or functional specification"[9]

Different quality models specify changeability as part of their classification. Both Boehm [5] and ISO 9126 [3] quality models mention changeability as part of maintainability, while in the FURPS and McCall [13] models, it is mentioned as related to flexibility. Also, in the redesigned ISO 9126 model, called the SQuaRE's Model [3], the changeability is still part of maintainability, but under the name of modifiabilty.

Considering the classifications from the quality models, when talking about changeability there are two aspects to consider: *maintainability* and *flexibility*. In a lot of cases, the definitions of these two quality attributes are mainly about what encompasses changeability. When we are thinking about changes that impact the maintainability of a system, we need to think about changes that

correct existing faults, expand or implement a functionality or refactorization, done to improve the overall quality of the system. Whereas, when we think about flexibility, we should have in mind the changes that are done to the system as a full or to separate components in order to make them available for use in other applications and environments in which they were not originally designed.

In our research we focus on changes that have repercussions on the maintainability of a system, focusing on functional changes and code readability, reliability and understandability improvements.

2.2. **Object Oriented Metrics.** Software metrics have become useful in software engineering, playing an important role in understanding, controlling and improving the code, but also the development process. Earlier metrics include LOC (lines of code), lines of comments, or number of functions or modules. With the introduction of object oriented (OO) metrics, the usefulness of these metrics has significantly increased in project management, software quality or refactoring. Several metrics have been proposed [10, 11], and have been classified according to the characteristics of object oriented paradigm, namely coupling, inheritance, cohesion and structural complexity [19]. We selected the metrics such that each of this characteristic is captured.

- *LCOM5*: defined by Henderson-Sellers in 1996 [11], it computes the lack of cohesion of a class and decides into how many classes that class should be split (this metrics represents the latest update of the initially defined LCOM metric)
- *WMC*: part of the Chidamber & Kemerer metric suite [10], it represents the sum of the McCabe Cyclomatic Complexity values of the local methods and initialization blocks
- *CBO*: also defined in the Chidamber & Kemerer metric suite, it computed the number of classes that are directly used by a class
- *RFC*: part of the CK metric suite, computes the number of methods of a class to each is added the number of methods called from within them
- *DIT*: represents the length of the longest path from a class to a root class in the inheritance tree of that system; defined, as well, in the CK metric suite
- *NOC*: defined by Chidamber & Kermerer, it is opposite to DIT, it computes the number of subordinated classes a class has, by going down the class hierarchy tree
- *LOC*: as the name states, it computes the number of lines of code from a specific scope; in our research the metric will always have the *class* scope

- *SIZE2*: gives information about the size of a class, by simply counting the number of attributes and methods that are part of that class

The last two metrics are capturing the size of the software system and the size of classes, and included them in our study besides well known OO metrics.

2.3. **Related work.** From the existing literature we have found that there are some factors that can be used to determine the level of changeability of a system, as follows: the complexity of components, its size and the scope of the changes [8]. The more complex a component is the harder will be for changes to be made to it and the same applies for the size of components. The scope of the changes will decide if the change will be contained and it will only impact a couple of components or if it will have a larger impact across the full application or even other applications.

Other studies concluded that the architecture of a system is a major influencer of its changeability [9]. When designing the architecture of a system it is easier to incorporate changes that have been thought of, changes that have been foreseen than it is to incorporate unpredictable changes. Their approach is based on five steps (selecting a goal, describe the software architecture, elicitation of possible scenarios, evaluation of the scenarios, interpretation) in order to analyze software systems architectures for changeability. The result can be used to predict future maintenance cost and identify possible inflexibilities of the system, as well as giving the possibility of comparing different architectures.

The importance of changebility can also be argued by the number of other software quality attributes or activities that are influenced [8]. Extensibility, refactorization, scalability and functional flexibility are a few of the impacted concerns; all of them imply changes to the system. What does it mean that it has an impact on these concerns? It means that having a system with an increased changeability will positively impact the above-mentioned concerns or, respectively, negatively impact them if we have a decreased changeability. Being able to easily modify a system we will also be able to more easily refactor it, work on its understandability, and also reduce the effort needed for a developer to make changes. We will be able to extend its functionality as well with a reduced effort and cost in order to remain relevant and we will be able to more easily scale the system in order to meet the needs of nowadays' market.

There have been also some researches on the topic of changeability, trying to fill the gap by providing techniques and tactics for achieving changeability [7] or by analyzing different architectures in order to determine if changeability is achieved more by one or another [9].

Researchers from the Montreal University, Canada [15] tried connecting changeability to a more measurable concept: metrics. While trying to prove cohesion as a reliable indicator of changeability, they have investigated the connection between the design of the software system, its architecture, and changeability. To do this they have developed a change impact model. They identified thirteen types of changes that could happen on an object-oriented system and the types of links that connect classes with one another, then computed the impact of a change based on the change's types and the links between the classes that are involved in the change, that are an important factor on how easy will be for that change to happen. The impact is defined as the set of classes that require modifications as the result of the change [15]. The research has concluded that cohesion is not an indicator for changeability, by using the impact of change, but the model they have defined could be furtherly used to assess connection between the quality attribute and other software system quality concepts. An important conlcusion of this study is that cohesion by itself, computed using the LCC (Loose Class Cohesion) and LCOM (Lack of Cohesion of Methods), is not a reliable indicator for changeability [15].

A very comprehensive survey of the relations between ISO9126 quality categories [3] and OO metrics is presented in [17], in the form of a compendium in which relations for all quality characteristics and subcharacteristics with software metrics are stated and graded "related" or "highly related" . We should remark that all the metrics we have selected for our study are considered highly related to changeability.

3. Estimating Changeability changes using Object Oriented Metrics

3.1. **Approach.** The purpose of the initial study was to determine how can OO metrics for different software versions of the same software system provide relevant information for changeability, and if the collected data can be used to improve changeability of the system.

The chosen metrics reflect on a systems complexity and size, cohesion and coupling, factors that were studied before in relation with changeability. Our initial study has computed the metrics for all versions of the considered study, and our first finding was that some metrics display little variation between versions, and in terms of software system evolution they cannot provide useful values. We decided to eliminate them from our study, such that the final list of metrics contained only LCOM5, WMC, CBO, LOC and SIZE2 (values for DIT, RFC and NOC were eliminated)

We used SourceMeter [22], a free tool for computing different metrics that also supports projects written in Java. Due to the fact that different tools

compute metrics in different ways, we considered an important aspect that all metrics are computed with the same tool.

The next step was to determine a way to count the changes performed on a system. Computing this value has two purposes: 1) to find a relation between changes and metrics in different versions of the source code and 2) to use it in order to predict further possible changes. As suggested in similar studies [4], we restricted our count to simple/atomic changes, and took into consideration changes applied to: classes, methods and fields (variables). The type of changes that we considered are: add, delete, change type, change signature. Starting from this, we assessed the possible changes in changeability using the formula:

$$Changeability\_Possible\_Change = DIFF\_LOC/CHANGE\_COUNT$$

where:

$DIFF\_LOC$ = difference of LOC between two versions

$CHANGE\_COUNT$ = number of changes between two versions

After we computed the OO metrics for multiple versions of projects and the Changeability_Possible_Change (ChangeCoef) we tried to see if there exists any correlation that could indicate us what classes we should further investigate. The first step was to run the Shapiro-Wilk test [21], for each of the variables to see if they are normally distributed. Because the Shapiro-Wilk test is limited to a number of maximum 5000 observations, we also tested normal distribution with Anderson-Darling normality test [23]. After this, the next step was to use either Pearson's correlation (if both variables are normally distributed) or Spearman's correlation (if one or both variables are not normally distributed) [1]. Given the fact that our data was not normally distributed, we used Spearman's correlation. The last step was to interpret the results from the correlations.

3.2. **Case Study and Results.** For our case study we chose two implementations with different architectures of the same health care system (considering almost the same functionalities). For privacy reasons we shall call these projects Project A and Project B. Both have been developed as commercial applications, over significant period of time, and in teams of developers with different levels of experience.

Project A was started 7 to 8 years ago and is still being used in production today. The system was built on a Liferay [16] platform; it used Java up to version 7 and jQuery [14] when needed for the UI. As for the architecture, when working with Liferay, a lot of functionality comes bundled in the core Liferay plugin and you can create new plugins for your specific functionality that you need to add than to Liferay. Liferay also generates a lot of code on

behalf of the developer. For example, when creating a new model, a couple of service classes will be generated, wrappers, exception classes and in the end also the model itself. This creates a lot of coupling and redundant code that deeply affects understandability. In our study, we only analyzed the main plugin that holds a big part of the functionality of the application.

The development phase of project B started in June 2018. It uses a microservice architecture that exposes REST APIs for the frontend microservices to call when displaying the UIs. Here the technology stack is up to date, using Java with Spring Boot [2] for the backend microservices and Angular 7 [6] for the frontend. We can say that this project represents a rewriting of project A with newer technologies. Since the project is only one year old, not all the functionality of project A is included in project B and neither is that the goal for project B, but to mainly recreate the main functionalities of project A. For our analysis, the classes from one microservice were taken into consideration, the microservice that contains the most similar functionality with the plugin chosen from project A.

The versions of the system were selected so that they are scattered across the implementation thus far of the systems, having variable time passed between versions, but long enough time so that the versions are not too similar.

Table 1 and Table 2 represents an excerpt of the computed data, and almost all differences between versions gave the same results. The tables show the results of Spearman correlation (*rho* and *p-value*) and the conclusion based on these values for *ChangeCoef* and the selected set of metrics. For both projects we can see that there is a correlation between the number of changes and the differences of the metrics in values.

Some remarks regarding the obtained results can be summarized in:

- the correlation for *ChangeCoef* and *LCOM5* is weaker than the correlations corresponding to the other metrics;
- the correlation for *ChangeCoef* and different metrics can be influenced by the architecture of the system - some correlations are "strong", others are "very strong".

So, regarding our first research objective, we can say that except LCOM, the conclusions stated by [17] are confirmed by our data. In case of LCOM: [17] considered that changeability and LCOM are highly related, while [15] concluded that cohesion (given by LCOM metric) is not an good indicator for changeability. Due to the variations in our results, and obtaining only "moderate" correlation for some cases, the conclusion will be to use LCOM or LCOM5 with care and together with other metrics when analyzing changeability of the system.

Table 1. Project A v1-3-3 and v1-3-14

	rho	p-value	correlation interpretation
ChangeCoef-LOC	0.7748249	< 2.2e-16	Strong
ChangeCoef-SIZE2	0.6923647	< 2.2e-16	Strong
ChangeCoef-WMC	0.8149562	< 2.2e-16	Very strong
ChangeCoef-CBO	0.6357793	< 2.2e-16	Strong
ChangeCoef-LCOM5	0.5867332	< 2.2e-16	Moderate

Table 2. Project B v1 and v2

	rho	p-value	correlation interpretation
ChangeCoef-LOC	0.9761374	< 2.2e-16	Very strong
ChangeCoef-SIZE2	0.8960441	< 2.2e-16	Very strong
ChangeCoef-WMC	0.8087428	< 2.2e-16	Very strong
ChangeCoef-CBO	0.826574	< 2.2e-16	Very strong
ChangeCoef-LCOM5	0.7780307	< 2.2e-16	Strong

Considering the second research objective, our computations showed that the architecture of the system can influence the relation between OO metrics and changeability, and newer technologies, such as microservices expose a higher correlation between OO metrics and changeability. The two tables were chosen as representatives of the performed computations (namely, most of the compared versions produced on average the same results), and we can notice higher values for *rho* in project B, leading to "very strong" correlation.

In order to achieve or third research objective, we developed a tool that based on the analysis described above can identify the situations in which changeability has recorded a significant increase, that can lead to costly situations in further development of the software system.

Figure 1 shows how metrics can be compared for different versions of the same application, giving the developer an indication where a certain metric has exposed a significant difference. For example, NM (Number of Methods) represents an increase between the two versions, but this is due to adding new functionalities to the application, and as a consequence WMC has changed, but not in a disturbing way.

In order to identify the "hotspots" regarding chageability, the values for each metric are split equally into four quartiles. The diagnostics are done on different level: danger, medium and simple threat. The levels represent how much of a threat do the class pose. The predefined threat levels that can be chosen are:

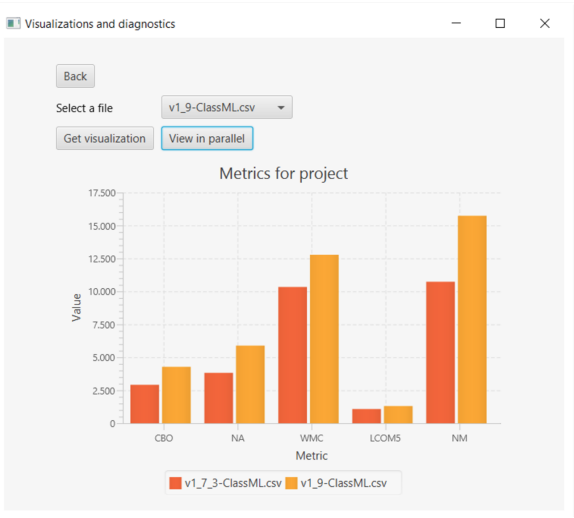


FIGURE 1. Parallel visualization for metrics from 2 versions (NA (Number of Attributes) + NM(Number of Methods) = SIZE2

- *Danger*: classes having all metric values in each metric's forth quartile
- *Medium*: classes having all metric values in each metric's third quartile or above
- *Threat*: classes having all metric values in each metric's second quartile or above.

The classes that are displayed as part of the diagnosis can also be viewed as the classes that need adjustments if you are concerned with the threat level that you have chosen. This means that if you chose the medium threat level, as shown in Figure 2, you receive both the classes having metric values in the third quartile but also the ones having all the metric values in the fourth quartile since those are the ones that are more important to be changed.

The tool can be a useful assistant to developers in early detection of situations that can become out of control, especially in large software systems, with a significant number of versions, developed by big teams over large period of time. However, this tool need to be correlated with a through inspection of the code in order to avoid false results (exemplified in Figure 1).

The biggest threat to validity is given by the small amount of data used in the analysis. Finding real project from the industry was hard and because of this the study should be extended on other projects or on open source projects.

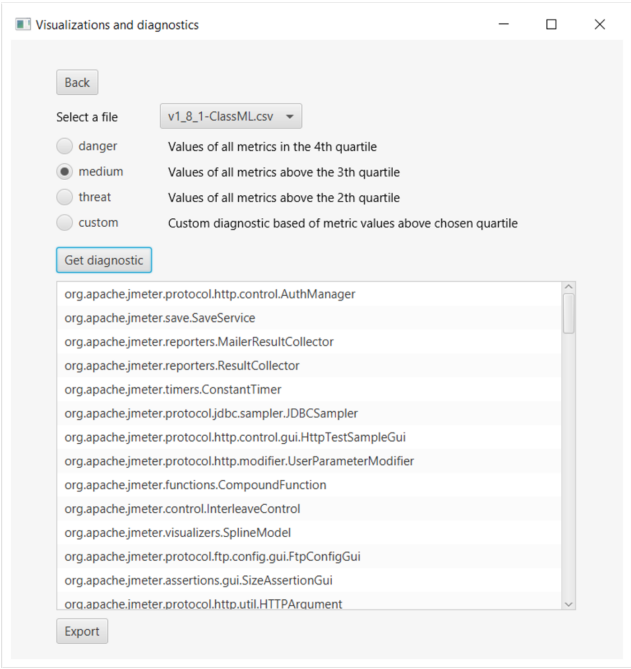


FIGURE 2. Classes exposing a medium threat to changeability

Even if we selected the metrics so that key characteristics of software were covered, a further extension of them is required.

4. CONCLUSIONS AND FUTURE WORK

Changeability is a quality attribute that has an impact on multiple disciplines that are involved in developing and launching a product. Right now, its improvement has been targeted indirectly by enhancements done on other quality attributes that are more easily assessed and controlled. There is still a lot to be learned about changeability and with this paper we have provided a better understanding of what influences changeability and what is its relation with OO metrics. In case of large software systems, controlling the changeability during the development phase can improve future maintenance and refactoring.

With our research we believe that we are a step closer in providing empirical evidence of the relation between certain OO metrics that can be computed for a system and the changeability quality attribute. We also showed one possibility in which the collected data about metrics can be applied to detect situations

in which changeability expose a significant increase between versions of an application.

The positive results motivate us in further extending the research by applying the research on more projects but also to add additional OO metrics. Also another direction that we will try to explore is to take in consideration also the effort needed to make a change.

References

[1] R correlations. https://www.statmethods.net/stats/correlations.html, 2019. Online; accessed 10 June 2019.

[2] Spring boot. https://spring.io/projects/spring-boot, 2020. Online; accessed 10 January 2020.

[3] 9126-1:2001, I. Software engineering - product quality. http://www.iso.org, 2019. Online; accessed 10 June 2019.

[4] Ajrnal Chaumun, M., Kabaili, H., Keller, R. K., Lustman, F., and Saint-Denis, G. Design properties and object-oriented software changeability. In *Proceedings of the Fourth European Conference on Software Maintenance and Reengineering* (2000), pp. 45–54.

[5] Al-Qutaish. R. Quality Models in Software Engineering Litrature: An Analytical and Comparative Study. *Journal of American Science 6*, 3 (2010), 166–175.

[6] Angular. Angular. https://angular.io, 2020. Online; accessed 10 January 2020.

[7] Bachmann, F., Bass, L., and Nord, R. Modifiability tactics. 63.

[8] Barbacci, M. R. Software quality attributes: Modifiability and usability. http://www.ieee.org.ar/downloads/Barbacci-05-notas1.pdf, 2003.

[9] Bengtsson, P., Lassing, N., Bosch, J., and Vliet, H. Analyzing software architectures for modifiability. https://www.researchgate.net/ publication/30499164_Analyzing_Software_Architectures_for_Modifiability, 2009.

[10] Chidamber, S., and Kemerer, C. A Metric Suite for Object- Oriented Design. *IEEE Transactions on Software Engineering 20*, 6 (1994), 476–493.

[11] Henderson-Sellers, B. Software metrics. *Hernel Hempstead: Prentice Hall* (1996).

[12] ISO25010. Iso25010 description information. http://iso25000.com/index.php/en/iso-25000-standards/iso-25010, 2019. Online; accessed 10 June 2019.

[13] J. McCall, P. R., and Walters, G. Factors in software quality. *Nat Tech.Information Service 1* (1977).

[14] jQuery. jquery. https://jquery.com/, 2020. Online; accessed January 2020.

[15] Kabaili, H., Keller, R. K., and Lustman, F. Cohesion as changeability indicator in object-oriented systems. In *Proceedings Fifth European Conference on Software Maintenance and Reengineering* (March 2001), pp. 39–46.

[16] Liferay. Digital experience software tailored to your needs. https://www.liferay.com/, 2020. Online; accessed 10 January 2020.

[17] Lincke, R., and Lowe, W. Foundations for defining software metrics, 2006.

[18] Lincke, R., and Lowe, W. Compendium of software quality standards and metrics. http://www.arisa.se/compendium/quality-metrics-compendium.html, 2019. Online; accessed 10 June 2019.

[19] Marinescu, R. *Measurement and Quality in Object Oriented Design*. PhD thesis, Faculty of Automatics and Computer Science, University of Timisoara, 2002.

[20] Northrop, L. Achieving product qualities through software architecture practices. https://apps.dtic.mil/dtic/tr/fulltext/u2/a631504.pdf, 2004.
[21] Royston, J. P. An extension of shapiro and wilk's w test for normality to large samples. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 31*, 2 (1982), 115–124.
[22] SourceMeter. www.sourcemeter.com, "sourcemeter 8.2.0". https://www.sourcemeter.com/resources/java, 2019.
[23] Stephens, M. A. Edf statistics for goodness of fit and some comparisons. *Journal of the American Statistical Association 69*, 347 (1974), 730–737.

Babeş-Bolyai University, Department of Computer Science, 1 M. Kogălniceanu Street, 400084 Cluj-Napoca, Romania
Email address: cdis1553@scs.ubbcluj.ro
Email address: tirban@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXV**, Number 1, 2020
DOI: 10.24193/subbi.2020.1.02

REVIEW OF DIGITAL BADGES IN COMPUTER SCIENCE LITERATURE AND APPLICATIONS

IMRE ZSIGMOND, BEÁTA LŐRINCZ, ANDREA-ÉVA MOLNÁR

Abstract. In the increasingly researched field of gamification, that is the application of the game design elements in no game contexts, one of the most used techniques is the use of digital badges. Through this paper, we conducted a literature review on the use and effectiveness of badges. We include other literature reviews, empirical case studies, and application reports. This analysis shows the widespread use, the different types and specialized nature of digital badges throughout the private and public sectors.

1. Introduction

Gamification increasingly gains interest in various e-learning environments. By the most accepted definition, it is the use of game design elements in non-game contexts [12]. This definition, being generic, allows for a wide range of implementations to be classified under it. In practice, game mechanics is the most common implementation method. There are several definitions of game mechanics that have been used in different contexts (e.g. game design [30], analysis [35]). One definition that use concepts from programming is: game mechanics are methods invoked by agents designed for interaction with the game state, thus providing gameplay [43]. The most frequently used game mechanics are: points, badges, leaderboards, and quests.

The earliest usage of gamification elements in the private sector was introduced by airline companies through frequent flyer programs to encourage customers to use the same airline for traveling [38]. Kotlet et al. emphasizes the human desire of achieving goals and being rewarded for achieving these goals. This desire can ultimately be used as a motivational and engagement tool. Based on this principle gamification elements started gaining popularity

Received by the editors: 2 November 2019.

2010 *Mathematics Subject Classification.* 68-02, 68Q32.

1998 *CR Categories and Descriptors.* H.5.2 [**Information Systems**]: User Interfaces – *Gamification*; A.m [**General Literature**]: MISCELLANEOUS – *Literature review*.

Key words and phrases. literature review, achievements, badges, gamification, public sector, private sector.

FIGURE 1. Digital badge example

in different industrial areas where customers were rewarded for transactions, buying products and using the same services repeatedly. In the last decade, enterprises are paying more attention to build and develop tools that use gamification elements in order to enhance the benefits of social software. According to the [47] report, "global gamification market was valued at 6.8 billion USD in 2018 and is projected to grow [...] on account of growing demand for customer experience enrichment and improved engagement of employees."

In this study we analyze digital badges, their usage in solutions for the public and private sectors. By definition, a digital badge - or it's equally used synonym achievement (badge from now on) - is a "representation of an accomplishment, interest or affiliation" and a digital badge in modern context means it's available online and contains information about the process and result of the activity that it is attached to [22]. Aside from the clear description of what it takes to achieve it, badges tend to have a representative image, as well as a title, see Figure 1 for an example. In [27] Hamari states that badges are optional goals and rewards where their fulfillment is outside of the scope of the activity.

One of the main theories behind the persuasiveness of badges is the phenomenon called social comparison [18]. It emerges from people comparing their points and badges as social status or benchmark for themselves. Social comparison theory forms a basis for more specific theories related to comparisons between individuals such as social influence and the theory of planned behavior [3].

Badges by its origins marked subsets of coats of arms and were used to identify outstanding skills or merits [25]. In historical times other symbols of outstanding skills, ranks or accomplishments were medals or trophies granted to individuals considered worthy for the award [22]. Yet another historical use is boy scout badges.

Badges started to appear in video games as these became more advanced [29]. Current mainstream video games tend to all have badges in some form, the mechanic has become ubiquitous in all genres. An inherent advantage of video games is the precision their measurements have when it comes to player behavior. Awarding a badge can be instantaneous and precise. A side effect of

video game consumption is that audiences get familiar with badges and need less to no onboarding when the same mechanic is used outside of video games. Though, awarding of the badges tend to be slower and less precise unless we have full control over the experience.

Gamification and badges have gained even more interest in education as it is a constant challenge to increase the motivation of learners and enhance their engagement. Several studies focus on the effect of applying gamified elements in education [22, 24, 39] and discuss the positive and negative effects of their usage. Badges are examined in environments with or without effect on grading emphasizing that the goal is not turning the learning process into a game, but improving the users' interaction, behavior in their education [24] and allowing them to compete with themselves and others. As behavioral outcomes, [39] lists quantifiable goals of education, such as grades, participation and speed in solving tasks.

1.1. **Related work.** Previous literature reviews tend to focus on the greater concept of gamification, and thus mention badges as a part of the concept or report on exceptional cases of badge use. That being said there are a few articles specializing in badges. [17] focuses on categorization namely in 3 ways: by their function, by their structure, and by design and interaction features. By function, the following subcategories can be distinguished: motivation and engagement, awareness and behavior change, and recognition of achievement. Taking into consideration the design and interaction features, we can identify the following subcategories: visual appeal, badge leader board/dashboard, badge socialization, notification of badges earned, tracking progress and personalization. In terms of their structure, the subcategories are: progressive/milestone, outcomes-based, levelled, categorized, hidden/locked, complex connection structures, starter, and negative. Our own categorization used this paper as a starting point.

To illustrate the importance of badges in gamification, let us consider [39]. In this article, the authors reviewed 128 papers using gamification in an educational context and report that 47 of these use some form of digital badge. In the same paper the authors looked at a subset of 91 papers and report 71.43% positive, 25.27% mixed, and a mere 3.3% negative results. Out of the 91 papers, 39 reported using badges with 64.10% positive, 30.76% mixed and 5.12% negative results. On the result side, [36] found 19 reports of gamified systems pertaining to health and well-being. 59% reported positive, 41% mixed effects, with mostly moderate or lower quality of evidence provided. Physical health related behaviors were positive, while the cognitive behaviors were mixed. In [28] the authors analyzed 24 peer-reviewed empirical studies

on gamification, reporting 13.33% positive and 86.66% with mixed results. We notice that out of the 24 studies 9 used badges.

With regards to reporting findings, the authors in [48] chose 14 papers to review, 9 of them used badges. The authors use low level summaries without reporting success rates. Paper [37] provides a high level summary of gamification elements. The author mentions 13 applications that use gamification elements, but does not mention how many of those use badges and specifies that in some implementations, achievements must be awarded manually, e.g. *Moodle*. In [46], the authors report on a small range of gamification mechanics used. The use of physical badges is also described here (being the only mention of this kind of badges in the reviewed literature) that are awarded on the spot, which are digitized later.

Regarding the discussion of theories, an in depth study on the theories behind gamification is given in [5]. The authors also reports on some case studies of the use of gamification, mentioning badges along the way. In [15], game theory models and formulas were proposed, detailing incentives created by absolute and relative standards.

2. Review procedure and research questions

2.1. **Review procedure.** We conducted a search through Google Scholar, Scopus, IEEE and Web of Science databases and selected literature review articles that summarize studies performed on the applications of badges in educational and other learning environments. The keywords used in the search were: "gamification", "badges", "digital badges", "achievements", "achievement badges" and "game mechanics". We chose a number of 32 conference papers, articles or book chapters. This number includes all the reviewed studies, the ones originally chosen based on the search results and papers referenced by these. While this selection is nowhere complete (the Scopus, IEEE and Web of Science databases returned a few hundred results with Google Scholar reaching a number of 150.000 results), it is indicative of the literature on our queries. In addition to the reviewed studies, our procedure included an analysis of applications that use badges. The main focus in the selection was to find frequently used apps that use badges, and to cover a wide range of application fields that these are utilized in. While there are a high number of applications that implement badges we only selected applications that we found referenced in the academic literature. We selected 11 applications mentioned in conference papers or articles covering 4 different types of application fields that deal with the role and effectiveness of badges.

2.2. **Research questions.** When we set out to conduct this literature review, we were interested in 3 main aspects of badges:

(1) What types of badges are used or defined?
(2) What were the results of implementing badges in various contexts?
(3) What applications used badges and how?

Our findings are detailed in Sections 3.1, 3.2, and 3.3 respectively.

3. Literature review

3.1. **Badge types.** Badges can be categorized based on different criteria such as function, structure or design elements [17]. Other criteria is the application area or customized features that are unique to the conducted experiment. Badges can be assigned to multiple categories within the same criteria. In this section we list the categorization criteria and types reported in the reviewed literature with mentioning that some studies do not detail the badge specifics, therefore some of the categorizations were inferred.

The badge function that is emphasized the most is: motivation, the engagement of users in the activity that uses gamified elements [16, 27, 39]. Other functions mentioned by studies are behavioral changes, awareness and the recognition of achievements [17].

To achieve these functions several structural features were conceptualized and implemented for badge systems. Table 1 summarizes the badge types based on their structure, including the types that were found in the reviewed literature.

In addition to the badge types listed in Table 1, [1] considered two distinct models for educational badges: merit badges and video game achievements. [9] study presents permanent and temporary badges, temporary badges being refreshed fortnightly. A somewhat unique mention of a badge system appears in [46], where the author reports on physical badges in the form of cards handed out on the spot which could be digitized later. [41] analyzes topic badges, that can be also considered simple badges as they can be earned by solving a set of exercises related to a specific topic. The same study reports on repetitive badges that can be seen as progressive, respectively challenge badges at the same time as several exercises need to be solved correctly in a row, but also rapidly. In [16], participation, attendance and flowchart badges were applied in educational environments for programming courses.

Badges are categorized based on their design features, the implementation of the system. The design of the badges, the used colors and symbols have an impact on users engagement. The interaction features, such as notification types, visual presentation, tracking tools, also influence the user experience [17].

Badges differ in terms of the fields of their applications. The majority of reviewed articles concentrate on badge usage in educational environments,

Type	**Description**	**Example**	**Used in**
Simple badges	Easily obtainable, outcome based.	Complete first assignment.	[2, 8, 11, 14, 19, 24, 26, 27, 41]
Progressive (milestone, streak) badges	Assigned to users based on a metric that defines milestones representing the progress. Users are granted these badges if correct solutions are given to tasks consecutively.	Complete first five assignments.	[6, 8, 19, 26, 27]
Hidden badges	Users are unaware of their existence until the badge is granted to them.	Easter egg	[11, 24]
Locked badges	Badges are visible, but locked for users until the conditions for obtaining these are not fulfilled.	Khan Academy: Black hole	[23, 27]
Challenge badges	Users can achieve these badges if the tasks are solved with features that have an increased difficulty.	Complete assignment within 10 minutes.	[8, 14, 26]
Negative badges	These badges can be assigned to users if an activity was not completed within the specified criteria.	Did not deliver activity.	[42]
Not specified (Custom)	Badge specifications not provided.		[16, 46]

TABLE 1. Badge types

that can be considered as part of the public sector. In addition, badge systems available online for any users can be seen as part of the public sector as well. Badge systems used in industrial settings (where the employees of the industrial organization are the users) can be considered as part of the private sector.

Effect	Studies
Positive	[2, 7, 8, 10, 16, 19, 20, 27, 32]
Mixed	[1, 6, 11, 14, 41]
Negative	[24, 26, 33, 40]

TABLE 2. List of papers by reported outcomes

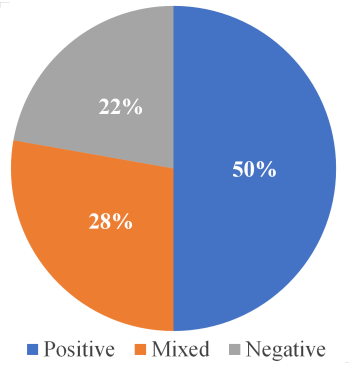


FIGURE 2. Overview of reported results in the papers

Some of the categorizations discussed fall squarely in domain specific categories. From the perspective of empirical research, we find that the more specific types detailed in Table 1 are to be considered.

3.2. **Reported results.** For the empirical case studies that we analyzed, 3 categories of success were chosen (see Table 2 & Figure 2). The threshold was based on the author's self-reporting on a given paper. Studies with a clear majority of positive findings would be classified as such, the same for negative result, while all studies with no clear positive or negative reported results, or a similar number of positive and negative reported results would be classified as mixed results. What follows are short summaries of the case studies.

The following studies reported positive results. In [2], high school students were the subjects of an online vocabulary study group. The authors report an increase of the study time and number of memorized words, but we need to take into consideration the lack of control groups. In [8] a computer engineering master's level college course was gamified. In comparison to the previous year, authors have reported 511% to 845% increase in forum engagement, while the increment in the case of faculty's forum activity was 373%. Grades did not increase compared to the previous year, only student engagement. Orientation passport app [19] used badges to direct attention to various events and locations useful for a first-year college student. Activities involved going to places/events taking pictures of QR codes or adding people from the peer group. Survey suggested positive student attitude towards the activities. The findings of [16] state that the participation and flowchart badges motivated users, while the attendance badges were not highly valued

by students. In [10], the authors highlighted that badges are valuable in stimulating voluntary participation. In [7], the authors establish from empirical data that users indeed value badges and modify their site activities to earn badges, and propose a model of how a user splits his/her time across various possible actions on a site (such as asking, answering, voting), in response to badges that are awarded for these actions. Paper [20] presents a gamified data structures course, with the use of badges. Although, badges did not have a direct impact on the final grade nevertheless it did increase the student's course grade. Positive effects were also found on task submission and trial and error submissions. The attractiveness of badges in *Duolingo* was analyzed in [32]. Concentrating on the platform's milestones, the authors found a positive statistically significant but lower than expected correlation between badges and progress.

The following studies reported mixed results. In [6], the authors tried two tracks of the same course: one started with badges and gamification, while the other without. At the halfway point, they switched the settings for both groups. They reported a slight increase in activity for "no gamification to gamification" group, while in the case of the activity for the "gamification to no gamification" group a moderate decrease was observable. They had no control groups of either gamification or no gamification only, to compare against. Although, overall grades appeared to be similar for both groups. In [11], a competition focused gamification experiment was set against a cooperation based social network, and a control group in a basic computer knowledge course. While performance was similar for all 3 groups, in *Word* and *Excel* gamification trailed behind both other groups. In the case of [14], the authors created an achievement add-on to an existing e-learning platform with an opt in system and a control group. Achievements were awarded manually and results were mixed with the authors reporting no theory as to why. Paper [41] described a specific badge usage metric and the usage of this metric on a group of engineering students. The authors divided students into 3 groups, based on an unsupervised clustering method: the first two groups included students who put the most effort, respectively students who made low effort, while the students who invested a decent amount of work but without many results were placed into the third group. In their study, Hamari reports mostly positive results and state that, depending on the type badges, badges had a positive effect on the practical assignments, but negative effects on written assignments [27]. The authors mention that a potential problem is the excessive usage of badges. In [1], the authors discovered that there is a connection between students' prior knowledge and how they value the obtained badges.

Context/field	**Applications**	**Papers concerning the applications**
Education and learning	Khan Academy, Codeacademy, Duolingo	[13, 31–34, 50]
Sustainability	Joule Bug	[4]
Health/exercise	FitBit[2], Samsung Health, Fitocracy	[45, 49]
Community/ sharing	Google Maps, Tripadvisor, Foursquare[3], Stack Overflow	[10, 21, 38, 44]

TABLE 3. Reviewed applications using badges

The following studies reported no or negative effects. In [26], the authors implemented a badge system on a purely utilitarian trading service, checking whether badge visibility affected the results. Their results showed that badges had no effect on how their application was used, concluding that likely more hedonistic contexts favor gamification. The study [24] on an online learning environment reported no significant difference between the experimental and control groups, and mentioned that some of the badges encouraged unwanted behavior. The authors of [40] discuss the contribution of badges to grades as negative effects as students reported frustrations because part of their grades depended on the collected badges. In [33], the authors analyzed two game elements - Badge and Winning Streak - with respect to the foreign language learning application, Duolingo[1]. The authors show that advanced users find badges less interesting.

3.3. **Applications.** Nowadays more and more companies incorporate the notion of gamification into their mobile apps and web pages. We reviewed several popular and interesting applications that use gamification techniques, focusing on the applications that award the users with badges. In Table 3, we included a summary of the selected applications and their associated literature, grouped by the field where they are used.

Rewarding the completion of learning tasks in educational environments is considered a game action. Therefore, learning applications award users with badges mostly to encourage them to complete the given tasks and to use the application regularly.

[1]https://www.duolingo.com
[2]https://www.fitbit.com
[3]https://foursquare.com/

Khan Academy[4] is a learning website, which uses video lessons to teach different topics for users. Users receive rewards for watching instructional videos, answering questions and resolving quizzes in the form of badges[5]. In Khan Academy hundreds of different badges are available, separated into 5 different tiers: *Meteorite* (including the most common, i.e. the most easily obtainable badges), *Moon*, *Earth*, *Sun*, *Black Hole* (including the rarest, i.e. the most hardly obtainable badges). *Five is Alive!* is a meteorite badge that can be earned by finishing five practice tasks, while *Incredible Question* is an Earth Badge and it can be achieved by asking a question that earns 50+ votes. Khan Academy also provides streak badges, which can be achieved by consecutively answering correctly to a certain number of questions (e.g. *Ludicrous Streak* is a Moon badge, which can be earned if 100 correct answers are submitted consecutively for any type of exercise). This application also includes some subject-specific badges (e.g. *Computer Science badges* that can be achieved by completing tasks in the Computer Science subject). As a user finishes certain course units, tutorials, or special activities, he/she is rewarded with *Challenge Patches* (e.g. *Intro to SQL: Querying and managing data is a challenge patches* badge, which can be collected if a user completes all the 'Intro to SQL' challenges).

Codeacademy[6] is also a learning website that teaches computer science. Codeacademy uses badges in order to track the progress of learners and to encourage perseverance. Codeacademy uses 4 types of badges. A learner can get an *exercise badge* when completes a specific number of exercises (e.g. 10 exercises). *Course badges* can be collected completing lessons that do not have a specific badge (e.g. *Rails Achievement* badge can be achieved by completing Ruby on Rails lessons), while *course-specific badges* are connected to particular lessons or courses (e.g. *Functions in JavaScript* badge). There are also *promotion badges* which are connected to an event or a partnership (e.g. *404 badge*).

In recent years foreign language learning applications have become widespread. *Duolingo* is one of the most popular educational gamified language-learning platforms, which uses several gamification elements in order to help users enhance their regular learning activity. Duolingo includes several badges (e.g. *Champion*, *Wildfire*), each of them comes in tiers (e.g. in the case of *Champion badge* the first, second, respectively the third tier are finished if the user completes 10, 20 skills, respectively a language tree).

[4]https://www.khanacademy.org
[5]https://www.khanacademy.org/badges
[6]https://www.codecademy.com

The number of applications and companies which try to draw attention to several important aspects of life, for example the importance of reading, environmental awareness or healthy lifestyle, has increased dramatically. Badges are widely used in health and fitness applications, trying to influence positive changes in users' health behaviors. We analyzed two fitness applications, which use gamification elements. *Fitocracy*[7] offers streak badges (e.g. *Get Low*, *Monster Squat*) and *achievement badges*. These badges are earned for various activities (e.g. *Let's get outta here* is given after 100km cycling). *Samsung Health*[8] includes *reward badges*, which are earned if a user completes a goal or he/she breaks his/her own record in an activity (e.g. *Best Pace* - breaking our best pace in walking).

Nowadays, environmental awareness and sustainable development are playing an increasingly important role in our daily lives. *Joule bug*[9] is an application which uses achievements in order to encourage people to make small changes in their behaviors in all aspects of sustainability, therefore to pursue an environmentally conscious lifestyle.

The use of gamification for loyalty programs and customer communities is also gaining popularity. In loyalty programs, customers can collect reward points that can be redeemed, while in any type of community customers can be motivated with reputation points, also called badges. Depending on the type of contribution (e.g. sharing photos of places, reviewing a place), in the web mapping service *Google Maps*[10] (developed by Google) users can collect different types of badges: *Reviewer*, *Photographer*, *Director*, *Trailblazer*, *Fact Finder*. There are 3 levels available for each type: *novice*, *expert* and *master*. By sharing and contributing more users can upgrade their badges to a higher level.

Tripadvisor[11] is the world's largest travel portal, where travelers can plan and book almost everything for their next trip (accommodation, guiding tours etc.), and share their experiences (by reviews). Tripadvisor provides *TripCollective badges*[12] for appreciating the users' contributions to the travel community and for illustrating their expertise and assistance to other users (e.g. *Explorer badge* can be collected for writing the first review for a restaurant, hotel or attraction in a given language).

[7]https://www.fitocracy.com/

[8]https://www.samsung.com/us/samsung-health/

[9]https://joulebug.com

[10]https://www.google.com/maps

[11]https://www.tripadvisor.com/

[12]https://www.tripadvisor.com/TripCollectiveBadges

The number of websites and applications that incorporated badges in order to reflect the real skills of their users has increased in recent years. This is due to the fact that these companies have recognized that these skills are exactly the ones that employers and schools are looking for. *Stackoverflow*[13], the online developer community, realized that the individual responses indicate the skill level of the users, so they created a badge system. There are 6 types of badges (over 80 different badges): *question*, *answer*, *participation*, *tag*, *moderation*, *other badges*. These badges are organized into 3 categories (*bronze*, *silver* and *gold*), depending on how easily they can be obtained. For example, the *Scholar question badge* can be earned very easily: if someone asks a question and accepts an answer. However, the *Mortarboard badge* (*gold*, *participation badge*) can be collected only if a user earns 200 daily reputation 150 times.[14]

We can notice that application badges are most often used in learning environments, respectively in crowd-sourcing systems.

4. Discussions and Conclusions

Digital badges have emerged in various tools and learning environments, both in the public and private sectors. They aim to increase user engagement and to motivate users to change their behavior and enhance their skills. In our paper we have proposed to answer 3 research questions. The first refers to the types of badges used. Out of the 32 studies reviewed, 17 mentioned the type of badge used. The categorization criteria varies in these studies, where all of these principles are associated with the function, structure, application domain or design elements of badges. The most frequently used criteria is concerned with the structure of the badges, in which we have discussed 7 different badge types. The second question is to evaluate the results of implementing badges in various contexts. Out of the 32 articles 18 reported positive, negative or mixed results, with 9 mentioning positive and only 5 negative results. These were evaluated by analyzing the functions and effects of badges in relation to the user experience. While using badges, generally positive results may be expected, depending on the circumstances and badges used, mixed or negative results may be assumed, although rarely. Research is so far inconclusive on the why and when, mixed or negative results are expected, current theories seem untested. The third question was focused on the badge usage in applications. We have tested 11 applications that use badges and are mentioned in academic literature. The primary fields of applications are: education, health

[13]https://stackoverflow.com/

[14]https://stackoverflow.com/help/badges

and community. Badge systems are used in a wide range of applications, and it is gaining more and more interest as it aims to improve user engagement.

More research is required for analyzing the circumstances and features of badges that lead to different results. Most of the specifics of the badge design is not reported in various studies making research on the topic less precise than desired.

References

[1] S. Abramovich, C. Schunn, and R. M. Higashi. Are badges useful in education?: It depends upon the type of badge and expertise of learner. *Educational Technology Research and Development*, 61(2):217–232, 2013.

[2] S. S. Abrams and S. Walsh. Gamified vocabulary. *Journal of Adolescent & Adult Literacy*, 58(1):49–58, 2014.

[3] I. Ajzen. The theory of planned behavior. *Organizational behavior and human decision processes*, 50(2):179–211, 1991.

[4] D. Akbulut and G. Yıldırım. The usage of green gamification for public interest function of public relations. *CTC 2019*, 2019.

[5] R. S. Alsawaier. The effect of gamification on motivation and engagement. *The International Journal of Information and Learning Technology*, 35(1):56 – 79, 2018.

[6] A. Amriani, A. F. Aji, A. Y. Utomo, and K. M. Junus. An empirical study of gamification impact on e-learning environment. In *Proceedings of 2013 3rd International Conference on Computer Science and Network Technology*, pages 265 – 269. IEEE, 2013.

[7] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec. Steering user behavior with badges. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 95–106. ACM, 2013.

[8] G. Barata, S. Gama, J. Jorge, and D. Gonçalves. Engaging engineering students with gamification. In *2013 5th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES)*, pages 1–8. IEEE, 2013.

[9] S. K. Bista, S. Nepal, N. Colineau, and C. Paris. Using gamification in an online community. In *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pages 611–618. IEEE, 2012.

[10] H. Cavusoglu, Z. Li, and K.-W. Huang. Can gamification motivate volun-tary contributions?: the case of Stackoverflow Q&A community. In *Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing*, pages 171–174. ACM, 2015.

[11] L. De-Marcos, A. Domínguez, J. Saenz-de Navarrete, and C. Pagés. An empirical study comparing gamification and social networking on e-learning. *Computers & Education*, 75:82–91, 2014.

[12] S. Deterding, D. Dixon, R. Khaled, and L. Nacke. From game design elements to gamefulness: Defining gamification. In *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*, volume 11, pages 9 – 15. ACM, 2011.

[13] B. J. DiSalvo and B. B. Morison. Khan Academy gamifies computer science. In *Proc. 45th ACM Techn. Symp. Comput. Sci. Educ.*, pages 39 – 44, 2014.

[14] A. Domínguez, J. Saenz-De-Navarrete, L. De-Marcos, L. Fernández-Sanz, C. Pagés, and J.-J. Martínez-Herráiz. Gamifying learning experiences: Practical implications and outcomes. *Computers & Education*, 63:380–392, 2013.

[15] D. Easley and A. Ghosh. Incentives, gamification, and game theory: An economic approach to badge design. *ACM Trans. Econ. Comput.*, 4(3):16:1 – 16:26, 2016.

[16] L. Facey-Shaw, M. Specht, and J. Bartley-Bryan. Digital badges for motivating introductory programmers: Qualitative findings from focus groups. In *2018 IEEE Frontiers in Education Conference (FIE)*, pages 1–7, Oct 2018.

[17] L. Facey-Shaw, M. Specht, P. van Rosmalen, D. Brner, and J. Bartley-Bryan. Educational functions and design of badge systems: A conceptual literature review. *IEEE Transactions on Learning Technologies*, 11(4):536 – 544, 2018.

[18] L. Festinger. A theory of social comparison processes. *Human relations*, 7(2):117–140, 1954.

[19] Z. Fitz-Walter, D. Tjondronegoro, and P. Wyeth. Orientation passport: Using gamification to engage university students. In *Proceedings of the 23rd Australian Computer-Human Interaction Conference*, OzCHI '11, pages 122 – 125, New York, NY, USA, 2011. ACM.

[20] J. Fresno, H. Ortega-Arranz, A. Ortega-Arranz, A. Gonzalez-Escribano, and D. R. Llanos. Applying gamification in a parallel programming course. In *Gamification in Education: Breakthroughs in Research and Practice*, pages 278–302. IGI Global, 2018.

[21] J. Frith. Turning life into a game: Foursquare, gamification, and personal mobility. *Mobile Media & Communication*, 1(2):248–262, 2013.

[22] D. Gibson, N. Ostashewski, K. Flintoff, S. Grant, and E. Knight. Digital badges in education. *Education and Information Technologies*, 20(2):403 – 410, 2015.

[23] L. Haaranen, L. Hakulinen, P. Ihantola, and A. Korhonen. Software architectures for implementing achievement badges - practical experiences. In *2014 International Conference on Teaching and Learning in Computing and Engineering*, pages 41–46, April 2014.

[24] L. Hakulinen, T. Auvinen, and A. Korhonen. Empirical study on the effect of achievement badges in TRAKLA2 online learning environment. In *2013 Learning and teaching in computing and engineering*, pages 47–54. IEEE, 2013.

[25] A. M. Halavais. A genealogy of badges: Inherited meaning and monstrous moral hybrids. *Information, Communication & Society*, 15(3):354 – 373, 2012.

[26] J. Hamari. Transforming homo economicus into homo ludens: A field experiment on gamification in a utilitarian peer-to-peer trading service. *Electronic Commerce Research and Applications*, 12(4):236 – 245, 2013. Social Commerce- Part 2.

[27] J. Hamari. Do badges increase user activity? A field experiment on the effects of gamification. *Computers in human behavior*, 71:469 – 478, 2017.

[28] J. Hamari, J. Koivisto, H. Sarsa, et al. Does gamification work? - A literature review of empirical studies on gamification. In *HICSS*, volume 14, pages 3025–3034, 2014.

[29] K. Hilliard. Activision badges - the original gaming achievement (2013, october 23). Retrieved October, 2019, from Game Informer: https://www.gameinformer.com/b/features/archive/2013/10/26/activision-badges-the-original-gaming-achievement.aspx.

[30] R. Hunicke, M. LeBlanc, and R. Zubek. Mda: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*, volume 4, page 1722, 2004.

[31] D. Huynh and H. Iida. An analysis of winning streak's effects in language course of "Duolingo". *Asia-Pacific Journal of Information Technology and Multimedia*, 6(2), 2017.

[32] D. Huynh, L. Zuo, and H. Iida. Analyzing gamification of "Duolingo" with focus on its course structure. In *International Conference on Games and Learning Alliance*, pages 268–277. Springer, 2016.

[33] D. Huynh, L. Zuo, and H. Iida. An Assessment of Game Elements in Language-Learning Platform Duolingo. In *2018 4th International Conference on Computer and Information Sciences (ICCOINS)*, pages 1–4. IEEE, 2018.

[34] M.-B. Ibanez, A. Di-Serio, and C. Delgado-Kloos. Gamification for engaging computer science students in learning activities: A case study. *IEEE Transactions on learning technologies*, 7(3):291–301, 2014.

[35] A. Järvinen. *Games without frontiers: Theories and methods for game studies and design*. Tampere University Press, 2008.

[36] D. Johnson, S. Deterding, K.-A. Kuhn, A. Staneva, S. Stoyanov, and L. Hides. Gamification for health and wellbeing: A systematic review of the literature. *Internet interventions*, 6:89–106, 2016.

[37] G. Kiryakova, N. Angelova, and L. Yordanova. Gamification in education. In *Proceedings of 9th International Balkan Education and Science Conference*, 2014.

[38] P. Kotler, H. Kartajaya, and I. Setiawan. *Marketing 4.0: Moving from traditional to digital*. John Wiley & Sons, 2016.

[39] K.-J. Majuri, Jenni and J. Hamari. Gamification of education and learning: A review of empirical literature. In *Proceedings of the 2nd International GamiFIN Conference, GamiFIN 2018*. CEUR-WS, 2018.

[40] R. McDaniel, R. Lindgren, and J. Friskics. Using badges for shaping interactions in online learning environments. In *2012 IEEE International Professional Communication Conference*, pages 1–4, Oct 2012.

[41] J. A. Ruipérez-Valiente, P. J. Muñoz-Merino, and C. Delgado Kloos. Detecting and clustering students by their gamification behavior with badges: A case study in engineering education. *International Journal of Engineering Education*, 33(2-B):816 – 830, 2017.

[42] L. Seixas, A. Gomes, and I. Melo Filho. Effectiveness of gamification in the engagement of students. *Computers in Human Behavior*, 58:48–63, 2016.

[43] M. Sicart. Defining game mechanics. *Game Studies*, 8(2):n, 2008.

[44] M. Sigala. The application and impact of gamification funware on trip planning and experiences: The case of tripadvisorś funware. *Electronic Markets*, 25(3):189–209, 2015.

[45] R. Stålnacke Larsson. Motivations in sports and fitness gamification: A study to understand what motivates the users of sports and fitness gamification services, 2013.

[46] A. Stott and C. Neustaedter. Analysis of gamification in education. *Surrey, BC, Canada*, 8:36, 2013.

[47] TechSci Research. Global gamification market by solution, by deployment, by organization size, by application, by end-user vertical, by region, competition, forecast & opportunities, 2024. *TechSci Research*, 2019.

[48] M. M. Tenório, F. A. F. Reinaldo, L. A. Góis, R. P. Lopes, and G. dos Santos Junior. Elements of gamification in virtual learning environments. In *International Conference on Interactive Collaborative Learning*, pages 86–96. Springer, 2017.

[49] Á. Tóth and E. Lógó. The effect of gamification in sport applications. In *2018 9th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 69–74. IEEE, 2018.

[50] R. van Roy, S. Deterding, and B. Zaman. Collecting Pokémon or receiving rewards? How people functionalise badges in gamified online learning environments in the wild. *International Journal of Human-Computer Studies*, 127:62–80, 2019.

All authors are with the Faculty of Mathematics and Computer Science, Babeș-Bolyai University, 1 Mihail Kogălniceanu, RO-400084 Cluj-Napoca, Romania

Email address: imre@cs.ubbcluj.ro and lorinczb@cs.ubbcluj.ro and andrea.molnar@math.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXV**, Number 1, 2020
DOI: 10.24193/subbi.2020.1.03

INTUITIVE ESTIMATION OF SPEED USING MOTION AND MONOCULAR DEPTH INFORMATION

RÓBERT ADRIAN RILL[1,2]

Abstract. Advances in deep learning make monocular vision approaches attractive for the autonomous driving domain. This work investigates a method for estimating the speed of the ego-vehicle using state-of-the-art deep neural network based optical flow and single-view depth prediction models. Adopting a straightforward intuitive approach and approximating a single scale factor, several application schemes of the deep networks are evaluated and meaningful conclusions are formulated, such as: combining depth information with optical flow improves speed estimation accuracy as opposed to using optical flow alone; the quality of the deep neural network results influences speed estimation performance; using the depth and optical flow data from smaller crops of wide images degrades performance. With these observations in mind, a RMSE of less than 1 m/s for ego-speed estimation was achieved on the KITTI benchmark using monocular images as input. Limitations and possible future directions are discussed as well.

1. Introduction

The significant progress in recent years makes deep learning solutions attractive in automotive industry applications [24]. In intelligent transportation systems, self-driving cars, advanced driver-assistance systems (ADAS) vehicle speed is one of the most important parameters for vehicle control and safety considerations. It can be measured using wheel sensor, Inertial Navigation System (INS) or Global Positioning System (GPS). Although these methods achieve high accuracy, they have limitations. Speed sensors involve a trade-off

Received by the editors: 10 January 2020.

2010 *Mathematics Subject Classification.* 68T45, 97R40.

1998 *CR Categories and Descriptors.* I.5.4 [**Computing Methodologies**]: Applications – *Computer Vision*; I.4.8 [**Computing Methodologies**]: Scene Analysis – *Motion, Depth cues*.

Key words and phrases. monocular vision, speed estimation, deep learning, optical flow, single-view depth.

This paper is based on the preprint: R.A. Rill. Speed estimation evaluation on the KITTI benchmark based on motion and monocular depth information. arXiv e-prints arXiv:1907.06989 [cs.CV] (2019).

between accuracy and cost. INS suffers from integration drift, i.e. small errors accumulate over time. GPS is prone to signal interference and loss in blocked areas, and may provide unreliable data when accelerating or decelerating. Some researchers also propose model-based approaches (see, e.g., [22]), which, however, may be significantly affected by incorrect parameter estimates.

Other methods for automotive sensing technologies, including speed estimation, are Radio Detection and Ranging (RADAR) or Light Detection and Ranging (LiDAR) systems that use radio frequency or laser signals, respectively [1]. Although both systems are popular in the industry, they have deficiencies. LiDAR systems can identify fine details of the 3D environment, but are greatly affected by unfavorable weather conditions and more importantly they are costly to produce and maintain. In contrast, RADAR systems are more robust, lightweight and cheap, but have lower accuracy and resolution.

To address the limitations of conventional speed estimation methods, computer vision based approaches have become attractive alternatives in recent years. Powerful hardware is available for autonomous vehicles to run deep learning algorithms in real-time [16]. In this work a simple monocular vision-based speed estimation approach is presented that exploits the recent advances in deep learning-based optical flow and monocular depth prediction methods. Optical flow is the pattern of apparent motion of objects in a visual scene caused by the relative motion between an observer and the scene. Monocular depth estimation aims to obtain a representation of the spatial structure of a scene by determining the distance of objects from a single image. The two problems are fundamental in computer vision and represent highly correlated tasks (see, e.g., [25]). The proposed method relies on the intuition that the magnitude of optical flow is positively correlated with the moving speed of the observer and that objects closer to the camera appear to move faster than the more distant ones. Different schemes of the proposed approach are investigated on a representative subset of the KITTI dataset [6, 5], and a RMSE of less than 1 m/s is achieved.

The rest of the paper is organized as follows. Section 2 presents related works providing a background and motivation for this study. Section 3 introduces the KITTI dataset and the deep learning methods used, and details the speed estimation pipeline. The quantitative and qualitative results are presented in Section 4. Section 5 discusses the results and limitations, highlights future directions and finally Section 6 concludes the paper.

2. Related work

Vision based approaches represent a promising direction for vehicle speed estimation that may replace or complement traditional methods. Most of the

works are concerned with estimating the speed of the vehicles in traffic using a camera mounted for traffic surveillance. These methods involve different image processing techniques: background extraction [3, 20, 23, 14], image rectification [3, 20, 23, 12], detecting and tracking reference points [3, 20, 14] or centroids [23] over successive frames, converting the displacement vectors from the image to the real-world coordinate system. The state-of-the-art results of deep learning in vision tasks makes object detection and tracking [12], locating license plates on vehicles [14], 3D convolutional networks [4] other promising directions in the task of speed estimation. Disadvantages of these approaches and of the traffic enforcement solutions already in use, such as speed or point-to-point cameras, include the need for calibration processes, meticulous positioning of the devices at predefined locations, investment in infrastructure and maintenance.

As opposed to traffic surveillance purposes, few studies address the problem from an ADAS perspective, namely estimating speed using monocular images from a camera mounted on the car. Some works estimate the relative speed of other participants in traffic (see, e.g., [18] or [11]). The present study is concerned with estimating the absolute speed of the car the camera is mounted on, also called the ego/forward/longitudinal speed.

In [17] the authors used sparse optical flow to track feature points on images from a downward-looking camera mounted on the rear axle of the car and achieved a mean error relative to GPS measurement of 0.121 m/s. However, the method works only in restricted conditions and was evaluated on self-collected data at low speed values. Han [8] used projective geometry concepts to estimate relative and absolute speed in different case studies. Using black box footages, a maximum of 3% difference was reported for higher ego-speed values when compared to GPS measurements. The major limitation of this study is the assumption of known distances between stationary objects such as lane markings. Banerjee et al. [2] used a rather complicated neural network architecture trained on self-collected data and reported an RMSE of 10 mph on the KITTI benchmark [5].

In this work a simple intuitive approach for ego-speed estimation is investigated using state-of-the-art deep neural network-based optical flow and monocular depth prediction methods. The proposed method achieves a RMSE of less than 1 m/s on recordings from the KITTI dataset.

3. Methods

3.1. **Dataset and deep neural networks.** The KITTI Vision Benchmark Suite[1] [6, 5] is a popular real-world dataset consisting of 6 hours of traffic

[1] `http://www.cvlibs.net/datasets/kitti`

scenario recordings captured while driving in and around a mid-size city. The traffic situations range from highways over rural areas to inner-city scenes with many static and dynamic objects. To evaluate speed estimation 15 recordings of rectified images were manually selected from the left input color camera of the KITTI dataset. The list of drive numbers are: 1, 2, 5, 9, 14, 19, 27, 48, 56, 59, 84, 91, 95, 96, 104, all from 2011.09.26. These are representative videos where the car is moving almost always.

In the experiments presented in this paper two optical flow estimation methods are compared: FlowNet2[2] [9] and PWC-Net[3] [19]. FlowNet2 is a consolidation of the original FlowNet idea that proposed to use convolutional neural networks to learn optical flow and poses the problem as an end-to-end supervised learning task. FlowNet2, compared to its initial versions, shows large improvements in quality and speed. While it achieves impressive performance by stacking basic models into a large capacity model, the much smaller and easier to train PWC-Net obtains similar or better results by embedding classical and well-established principles into the network itself.

Similarly, two single-view depth estimation methods are examined: MonoDepth[4] [7] and MegaDepth[5] [13]. MonoDepth innovates beyond existing learning based single image depth estimation methods by replacing the use of large quantities and difficult to obtain quality training data with easier to obtain binocular stereo footage. It poses the task as an image reconstruction problem. On the other hand, MegaDepth refers to a large depth dataset generated via modern structure-from-motion and multi-view stereo methods from Internet photo collections. The models trained on MegaDepth exhibit high accuracy and strong generalization ability to novel scenes. One important difference is that while MonoDepth predicts disparity values, MegaDepth models predict ordinal depth defined up to a scale factor.

For more details about the four deep learning algorithms investigated in this work and related methods please see the cited works and the references therein. Before applying the methods for speed estimation, their performance was evaluated quantitatively on test data from KITTI. Note, however, that in order to obtain dense optical flow and depth information the 1242×375 resolution input images in the KITTI dataset need to be resized accordingly. The output of the deep neural networks was resized back to the original resolution as summarized in Table 1. To run the neural networks pre-trained weights were used, provided in the corresponding Github repositories. The quantitative results obtained are reported in Section 4.

[2] `https://github.com/lmb-freiburg/flownet2`

[3] `https://github.com/sniklaus/pytorch-pwc`

[4] `https://github.com/mrharicot/monodepth`

[5] `https://github.com/lixx2938/MegaDepth`

TABLE 1. **Deep neural network technicalities.** *For details see the references and/or the Github repositories.

Method	Input resolution	Input resize	Output resize	Model used*
FlowNet2	divisible by 64	pad with zeros	trim zeros	FlowNet2
PWC-Net	1024×436	bilinear interpolation	anti-aliasing	default network
MonoDepth	512×256	anti-aliasing	anti-aliasing	city2kitti
MegaDepth	512×384	anti-aliasing	anti-aliasing	best generalization

3.2. **Speed estimation pipeline.** In order to estimate speed from a moving camera, two observations should be made: (i) optical flow is expected to highly correlate with moving speed, and (ii) the apparent motion of objects closer to the camera is faster than those of more distant ones. Therefore, to obtain the speed of a given object on the image – that is the moving speed of the camera – it is reasonable to multiply optical flow magnitude by depth.

After extensively considering variations of the above intuitive idea with respect to deep learning based optical flow and single-view depth estimation methods, the following base multistep process is proposed, and some modifications will be inspected as well. In a first step the results of one optical flow and one depth estimation algorithm are retrieved for a given image frame. The magnitude of optical flow vectors will be denoted by OF, and the disparity by DISP in the following. In the second step, the OF and DISP values are considered at valid pixels from a predefined crop of the original image, and the mean OF is divided by the mean DISP to get a scaled speed estimate. The valid pixels are obtained by imposing thresholds: $\text{OF} > 0.1$ and $\text{DISP} > 0.02$, which were selected after extensive experiments. Note, however, that selecting other close values gave similar results and does not affect the conclusions of the paper. The next step is the concatenation of the scaled speed estimates over the temporal dimension, i.e. over frames of a video. The aggregated vectors are temporally smoothed using a 1D convolution of size 25 with equal weights. Finally, the resulting smoothed lists are taken for multiple recordings and a scaling factor is approximated that minimizes the ratio between the ground truth and predicted speed. This scaling factor is used to convert speed from the image domain to real-world units.

To summarize, the steps of the base speed estimation algorithm (denoted by A) are as follows:

(1) Run optical flow and depth estimation methods on a given image.
(2) Compute the scaled speed for the given frame: consider the OF and DISP values at valid pixels from a predefined image crop, and compute the quotient between their means.

(3) Repeat the previous steps for all the frames from a video and apply temporal smoothing.
(4) Repeat previous step for multiple videos and determine scaling factor.

In Section 4 several modifications of the above base pipeline are experimented with. These are summarized below.

(A_1) Neglect depth information completely, and use optical flow alone.
(A_2) Replace OF by the magnitude of horizontal optical flow only (which is expected to highly correlate with the moving speed especially towards the edges of the image).
(A_3) Apply temporal smoothing at the pixel-level separately for OF and DISP, before computing their means.
(A_4) Run the neural networks directly on the image crop, as opposed to using the full frame as input first and extracting OF and DISP for the crop after.
(A_5) Investigate different image crops in the base algorithm, including the full wide frame.

In experiment A_5 defined above the modification of the base speed estimation pipeline is evaluated on three different image crops and the full frame. They are defined in Table 2. Reasons for using crops only include the possible unavailability of wide images, or memory and run-time considerations.

TABLE 2. **Definition of image crops used in the experiments.** (x, y) defines the upper left corner and w, h the width and height of the bounding boxes in pixels.

Image crop	bounding box			
	x	y	w	h
crop_1	720	180	200	120
crop_2	700	100	400	240
crop_3	640	20	580	340

4. Experiments and results

Figure 1 shows visualizations of the deep neural network methods on one sample frame from the KITTI dataset. Both optical flow estimation methods produce smooth flow fields with sharp motion boundaries. PWC-Net seems to be more robust against shadow effects. The depth prediction methods also show good visual quality, with MonoDepth being able to capture object boundaries and thin structures more reliably.

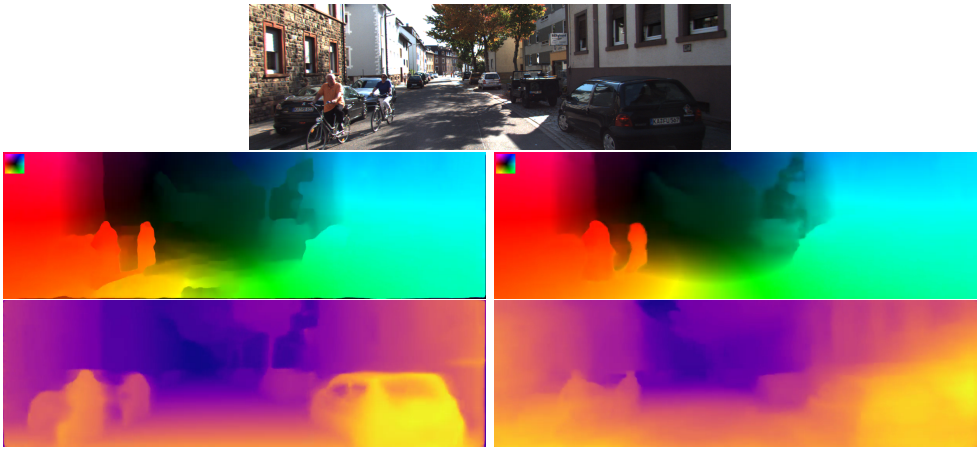


FIGURE 1. **Sample visualisation of network results.** From left to right and top to bottom: frame 71 of *drive_0095*, FlowNet2, PWC-Net, MonoDepth, MegaDepth. The colored square represents the color coding of optical flow.

The methods were evaluated quantitatively as well. The optical flow estimation algorithms are evaluated on the 200 training images from the KITTI 2015 benchmark [15]. The results displayed in Table 3 show that PWC-Net outperforms FlowNet2. The depth prediction algorithms are evaluated on the given 1000 manually selected images from the full validation split of the derived depth prediction and completion KITTI dataset [21]. According to Table 4, MonoDepth achieves better performance in several metrics compared to MegaDepth. In all four cases the results are in correspondence with those reported in the references presenting the methods.

TABLE 3. **Evaluation of optical flow estimation methods.** The KITTI 2015 benchmark [15] was used. AEPE: average endpoint error; Fl-all: Ratio of pixels where flow estimate is wrong by both ≥ 3 pixels and $\geq 5\%$.

Method	AEPE	Fl-all
FlowNet2	11.686	32.183%
PWC-Net	2.705	9.187%

Table 5 shows the speed estimation results for the experimental algorithms A_1 – A_4. Inspecting the values in detail allows to draw the following conclusions. According to A_1, using depth information as well improves speed

TABLE 4. **Evaluation of depth estimation methods.** The manual selection of the validation split of the derived depth prediction and completion KITTI 2017 dataset [21] was used.

Method	RMSE	RMSE(log)	Abs Rel	Sq Rel	log10	Scale-inv.
MonoDepth	4.532	0.150	0.090	0.749	0.040	0.142
MegaDepth	6.719	0.336	0.322	1.994	0.124	0.289

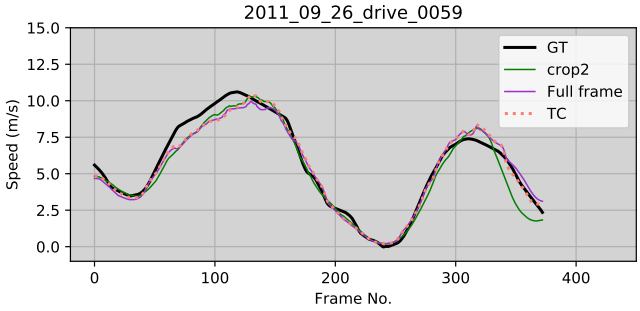
estimation performance, as opposed to considering optical flow alone. A_2 shows that replacing OF with horizontal optical flow results in slightly higher RMSE values. As demonstrated by A_3, applying temporal smoothing at the pixel level increases performance in some cases but only marginally. Finally, according to experiment A_4, running the neural networks on smaller image crops degrades performance considerably.

TABLE 5. **Results of speed estimation experiments.** RMSE values are shown using crop_2 defined in Table 2. A_i, $i \in \{1, 2, 3, 4\}$ refers to the algorithms from Section 3.

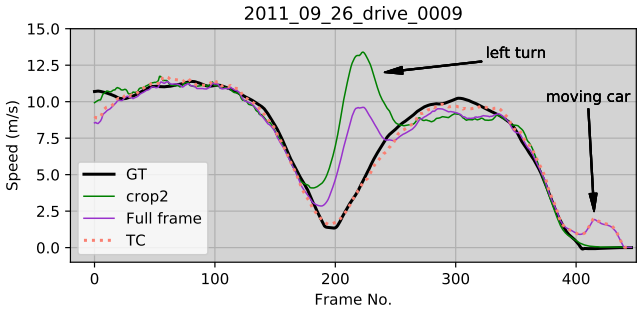
Method	Base algorithm A	Horizontal optical flow (A_2)	Pixel-level smoothing (A_3)	Methods only on crop (A_4)
FlowNet2 (A_1)	2.921	3.005	2.919	3.093
PWC-Net (A_1)	2.472	2.621	2.472	3.170
FlowNet2 & MonoDepth	2.305	2.448	2.399	2.618
PWC-Net & MegaDepth	1.915	2.059	1.908	3.180
FlowNet2 & MegaDepth	2.485	2.526	2.475	2.901
PWC-Net & MonoDepth	**1.467**	1.707	1.865	2.967

Furthermore, from Table 5 it can be seen that the best results are obtained when PWC-Net is combined with MonoDepth. Figure 2 shows speed estimation results for two KITTI recordings using the base pipeline. This simple method captures speed changes with low error in straight travel scenarios (Figure 2a), but having difficulty in cases when the car is taking a turn (around frame 200 on Figure 2b speed decreases as the car is turning left, yet optical flow increases on the right side – and in crop_2 too – of the wide KITTI images).

Modification A_5 of the base algorithm is evaluated on three image crops and the full frame. Table 6 illustrates that speed estimation accuracy improves in general as the size of the image increases. The best results are obtained again by the PWC-Net – MonoDepth combination. When the full frame is used errors moderately decrease at car turning events (around frame 200 on



(a)



(b)

FIGURE 2. **Speed estimation on sample KITTI videos.** Results are shown for the base pipeline with PWC-Net and MonoDepth; crop_2: defined in Table 2; *Full frame*: full wide image; GT: ground truth speed; TC: compensating for car turning events. For details see text.

Figure 2b), but overestimations are still present in dynamic scenes due to the motion of other cars for instance (after frame 400 on Figure 2b).

TABLE 6. **Results using different image crops.** RMSE values are shown for A_5. Image crops are defined in Table 2. TC: compensating for car turning events (for details see text).

Method	crop_1	crop_2	crop_3	Full frame	TC
FlowNet2 & MonoDepth	2.170	2.305	2.558	2.370	2.138
PWC-Net & MegaDepth	2.363	1.915	2.015	1.786	1.445
FlowNet2 & MegaDepth	2.671	2.485	2.583	2.544	2.125
PWC-Net & MonoDepth	1.735	1.467	1.505	**1.178**	**0.977**

In order to decrease speed overestimations at car turning events an additional modification of the base pipeline was experimented with. In such cases the average horizontal optical flow from the left part of the image has the same direction as the average from the right side. Whenever this condition is true, instead of the mean optical flow magnitude from the full wide frame, the absolute value of the difference between the means of horizontal optical flow of the left and right sides is computed, and divided by the mean disparity corresponding to the whole frame. Applying this heuristic compensation for turning events (TC) decreases the RMSE to under 1 m/s, as shown in Table 6. The performance improvement is illustrated by Figure 2b as well.

5. Discussion

Two optical flow estimation (FlowNet2 [9] and PWC-Net [19]) and two depth estimation (MonoDepth [7] and MegaDepth [13]) algorithms were investigated. Evaluating them on ground truth data from the KITTI dataset showed that PWC-Net and MonoDepth achieved better performance in several error metrics. The reason for this is presumably some combination of the following: the MonoDepth model used was fine-tuned on data from KITTI, MegaDepth does not predict metric depth but ordinal depth, it seems that during training of the PWC-Net model KITTI data was used as well. Nonetheless, the conclusion is that better performance optical flow and depth estimation methods result in better speed estimates. Besides, continuous efforts are made to improve these two fundamental computer vision algorithms, including their joint training in an unsupervised manner (see, e.g., [25]). Accordingly, fine-tuning to arbitrary images becomes accessible without the need for difficult to obtain ground truth labels.

After evaluating several modifications of the intuitive approach presented in this paper, meaningful conclusions were formulated: combining optical flow with depth information improves accuracy, using only the horizontal component of optical flow is not beneficial, applying temporal smoothing helps since it reduces the noise present in optical flow and depth estimation methods, using the full wide image frames as input to the deep neural networks and these results for speed estimation provides better approximations as opposed to using smaller image crops. It should be noted that other experiments were carried out as well, the results of which are not presented in the current paper.

There are two major limitations of the proposed method. Firstly, speed is erroneously estimated when the car is turning. However, in such cases estimation errors can be corrected by taking into account that horizontal optical flow on the left and right side of the wide images has the same direction (see the last column of Table 6 and Figure 2b). Secondly, the proposed method is

most reliable when the background is static. For example in heavy traffic scenarios when the surrounding cars are moving as well, the correlation of optical flow with ego-speed might be small and speed can be over- (see Figure 2b) or underestimated. In such scenarios combining monocular depth estimation with semantic segmentation [10] represents one promising direction; and the estimation of relative speed can help, which is another problem where recent advances are being made (see, e.g., [18] or [11]). One might also argue that the deep neural network methods providing the best speed estimates were fine-tuned on ground truth data from KITTI. But, as explained above, efforts are being made to train such methods on unlabelled data [25].

To improve the presented method, future works can treat the task as a regression problem and adopt for example a lightweight multilayer perceptron using as input the aggregated optical flow and depth results from different smaller regions of the original image. Another possibility is the exploitation of the more sophisticated convolutional neural network, which, however would possibly require a larger amount of training data [4].

6. Conclusion

In this work a simple algorithm was investigated for ego-speed estimation from images of a camera mounted on the moving car, using state-of-the-art deep neural network based optical flow estimation and monocular depth prediction. The method relies on the intuition that optical flow magnitude is highly correlated with the moving speed of the observer and that the closer objects are to the observer the faster they appear to be moving. Extensive evaluations of the intuitive algorithm lead to a RMSE of less than 1 m/s on a representative subset of the widely exploited KITTI dataset. As a closing remark, it is noteworthy that due to the recent and ongoing advancements in deep learning, monocular vision-based approaches are a promising direction for ego-speed estimation, and autonomous driving in general.

Acknowledgements

The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013 and EFOP-3.6.3-VEKOP-16-2017-00002). The author acknowledges his PhD supervisors, Horia F. Pop, for his assistance, and András Lőrincz, for his helpful ideas that promoted the prosperous progression of this project. The author would also like to thank Kinga B. Faragó for her cooperation in supervising the progress of two students, and Szilvia Szeier and Kevin A. Hartyányi for their efforts related to their scientific student conference, which contributed to this work as well.

References

[1] Abuella, H., Miramirkhani, F., Ekin, S., Uysal, M., and Ahmed, S. ViLDAR - visible light sensing based speed estimation using vehicle's headlamps. *arXiv e-prints* (2018), arXiv:1807.05412.

[2] Banerjee, K., Van Dinh, T., and Levkova, L. Velocity estimation from monocular video for automotive applications using convolutional neural networks. In *IEEE IV Symposium* (2017), pp. 373–378.

[3] Doğan, S., Temiz, M. S., and Külür, S. Real time speed estimation of moving vehicles from side view images from an uncalibrated video camera. In *Sensors* (2010).

[4] Dong, H., Wen, M., and Yang, Z. Vehicle speed estimation based on 3d convnets and non-local blocks. *Future Internet 11*, 6 (2019).

[5] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. Vision meets robotics: The KITTI dataset. *IJRR* (2013).

[6] Geiger, A., Lenz, P., and Urtasun, R. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR* (2012).

[7] Godard, C., Aodha, O. M., and Brostow, G. J. Unsupervised monocular depth estimation with left-right consistency. In *CVPR* (2017), pp. 6602–6611.

[8] Han, I. Car speed estimation based on cross-ratio using video data of car-mounted camera (black box). *Forensic Science International 269* (2016), 89–96.

[9] Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR* (2017), pp. 1647–1655.

[10] Jiang, H., Larsson, G., Maire, M., Shakhnarovich, G., and Learned-Miller, E. Self-supervised relative depth learning for urban scene understanding. In *ECCV* (2018), Springer, pp. 20–37.

[11] Kampelmühler, M., Müller, M., and Feichtenhofer, C. Camera-based vehicle velocity estimation from monocular video. In *CVWW* (2018).

[12] Kumar, A., Khorramshahi, P., Lin, W.-A., Dhar, P., Chen, J.-C., and Chellappa, R. A semi-automatic 2d solution for vehicle speed estimation from monocular videos. In *CVPR Workshops* (2018).

[13] Li, Z., and Snavely, N. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR* (2018), pp. 2041–2050.

[14] Luvizon, D. C., Nassu, B. T., and Minetto, R. A video-based system for vehicle speed measurement in urban roadways. *IEEE Transactions on Intelligent Transportation Systems 18*, 6 (2017), 1393–1404.

[15] Menze, M., and Geiger, A. Object scene flow for autonomous vehicles. In *CVPR* (2015).

[16] NVIDIA. Nvidia drive AGX, 2019. https://www.nvidia.com/en-us/self-driving-cars/drive-platform/hardware/.

[17] Qimin, X., Xu, L., Mingming, W., Bin, L., and Xianghui, S. A methodology of vehicle speed estimation based on optical flow. In *Proceedings of 2014 IEEE International Conference on Service Operations and Logistics, and Informatics* (2014), pp. 33–37.

[18] Salahat, S., Al-Janahi, A., Weruaga, L., and Bentiba, A. Speed estimation from smart phone in-motion camera for the next generation of self-driven intelligent vehicles. In *IEEE 85th VTC* (2017), pp. 1–5.

[19] Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR* (2018), pp. 8934–8943.

[20] Temiz, M. S., Kulur, S., and Doğan, S. Real time speed estimation from monocular video. *ISPRS Archives XXXIX-B3* (2012), 427–432.

[21] Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., and Geiger, A. Sparsity invariant CNNs. In *3DV* (2017).

[22] Xu, Q., Li, X., and Chan, C.-Y. A cost-effective vehicle localization solution using an interacting multiple model-unscented kalman filters (IMM-UKF) algorithm and grey neural network. *Sensors 17*, 6 (2017).

[23] Y. G. Anil Rao, N. Sujith Kumar, H. S. Amaresh, and H. V. Chirag. Real-time speed estimation of vehicles from uncalibrated view-independent traffic cameras. In *TENCON 2015 - IEEE Region 10 Conference* (2015), pp. 1–6.

[24] Yao, B., and Feng, T. Machine learning in automotive industry. *Advances in Mechanical Engineering* (2018).

[25] Zou, Y., Luo, Z., and Huang, J.-B. DF-Net: Unsupervised joint learning of depth and flow using cross-task consistency. In *ECCV* (2018), Springer, pp. 38–55.

[(1)]Faculty of Informatics, Eötvös Loránd University. H-1117 Budapest, Pázmány P. stny 1/C, Hungary.

[(2)]Faculty of Mathematics and Computer Science, Babeş-Bolyai University. No. 1 Mihail Kogalniceanu St., RO-400084 Cluj-Napoca, Romania.

Email address: rillrobert@cs.ubbcluj.ro, rillroberto88@yahoo.com
(ORCID iD: http://orcid.org/0000-0002-3004-7294)

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXV**, Number 1, 2020
DOI: 10.24193/subbi.2020.1.04

LOCATION PREDICTION IN MOBILE APPLICATIONS

ALEX CIPCIGAN-RAŢIU

Abstract. The vast developments of mobile technologies and applications in recent years produced a lot of issues to address, as mobile devices have surpassed the usage of classical computers. Predicting the future location of a user who utilizes a mobile application caught the eye of both academia and the industry. Most existing results either use excessive computing, most often relying on a server, or neglect battery usage. We propose a new method that takes these major points into consideration, which gives good results by only relying on the end user's mobile device, not draining the battery, respecting the privacy of the users and that achieves an accuracy of 80%.

1. Introduction

Given the growth and vast developments in mobile communication technology in recent years, mobile phones and tablets likewise have become the most used devices by people all around the globe, surpassing classical computers.

One of the most used services and possibly one of the main reasons why mobile took over the world is geolocation. Thus, location-based services (LBS) are nowadays used almost everywhere, in social networks, navigation, advertisement industry, recommendations and so on.

In the current environment, predicting the next location of a user is highly interesting and studied subject [17], which has caught the eyes of academia, as well as the ones of the industry. As research shows, human activities are likely to repeat themselves, having a strong regularity, showing that the potential predictability in user mobility is 93% [14].

There are multiple ways and patterns of predicting the location of a mobile user, but all have in common an important factor: prediction is done based on previously gathered locations that the user has visited. Gathering

Received by the editors: 10 March 2020.

2010 *Mathematics Subject Classification.* 68T05, 91E40.

1998 *CR Categories and Descriptors.* I.2.6 [**Artificial Intelligence**]: Learning – *Analogies*; I.5.2: [**Pattern Recognition**]: Design Methodology – *Pattern analysis*.

Key words and phrases. location prediction, mobile, individual next location prediction.

the geographical locations visited by an individual is done through Global Positioning System (GPS) satellites, Wi-Fi and cell towers, the former being the most accurate but also the most battery consuming [8].

Thus, predicting the next location is an interesting topic and new ways of achieving it can emerge, which constitutes a challenge in terms of efficiency and battery consumption. In the following sections, a unique approach that tackles this subject will be presented, which only uses the individual's mobile device, is efficient battery-wise, and gives good results in terms of accuracy.

Based on the stated information, in what follows, this paper will address the issue of predicting the next location of an individual using a mobile application, by illustrating various techniques and explain why opting for a simple solution gives remarkable results.

2. Related Work

Existing papers and solutions have been analyzed and compared in order to choose the most suitable one for our purpose. This section aims at taking a closer look at predicting the user location using Markov models [6, 4], k-means clustering mixed with Markov models [1], path matching [7], and decision trees [15] (see Table 1 for a quick comparison of the related works).

Most papers rely on Markov models for predicting the future location of an individual, being one of the best alternatives based on the results it delivers [6]. But, Markov models do require prior computation of collecting the locations (states). This is done most of the time by using a clustering algorithm, such as k-means clustering. Path matching is also an innovative solution, as described in [7]. A rather diverse alternative for tackling this problem is by using decision trees [15]. Using prior gathered points of interest and sequences of visited ones, the next most likely place to be visited can be predicted.

2.1. **Markov Models.**

One of the widely used ways of predicting the next geographical location of an individual is by using a Markov model or by extending it. A Markov model is a stochastic model used to represent a randomly changing system [10]. The base idea is that any future state depends only on the current state, not on events that occurred before. This can be extended in order to keep track of more than just the current state, taking into account the n previously visited states and building a n-Markov chain ($n \geq 1$).

Certain variants of the Markov model were implemented to tackle the problem in need. Some interesting ones are based on a variant of mobility Markov chains [6] and on a continuous-time series Markov model [4].

Basic mobility Markov chains have no memory, meaning they do not keep track of the previously visited states, as in the idea behind a basic Markov model. As presented in [6], by keeping track of previous states and predicting

Method Name	Input	Clustering Algorithm	Description
Mobility Markov chain	GPS coordinates	DBSCAN	Mobility Markov chain that keeps track of previous visited states
Continuous-time series Markov Model	GPS coordinates and time	DJ-Cluster	Time interval based location prediction using a continuous time series Markov model
K-means mixed with Markov model	GPS coordinates	K-means	Markov model
Path matching using cellular data	GSM data	Incremental clustering of cells and routes	Previously observed routes by matching their similarities with the current one
Decision trees	GPS coordinates	DBSCAN	A decision tree

TABLE 1. A comparison of the highlighted related works

using this in mind, the efficiency of the prediction is between 70% and 95% as soon as two previous states are taken into account.

But, even a mobility Markov Chain that keeps in mind the previously visited locations is unable to predict the next location based on a time interval [4]. Thus, [4] proposes a solution in which not only the geographical paths that an individual takes are considered, but time as well. Using a recursive algorithm, it manages to give an average accuracy of 43% [4] when predicting the next location, which is quite high compared to other time-based prediction solutions.

In both solutions presented above, a clustering algorithm is used to gather the set of states, that represent the locations that the individual visited and will visit next. In the former, density-based spatial clustering of applications with noise (DBSCAN) algorithm is used as the clustering algorithm [5]. It groups geographical points that are close to each other, but unlike K-means, it does not need to set the number of clusters in advance. It requires only 2 parameters: the neighborhood radius and the minimum number of points to treat the point as a core point. The latter on the other hand uses a variant

of the k-means algorithm to discover the points of interest, called Density-Joinable cluster (DJ-cluster) [18]. This was chosen to the detriment of k-means for the same reason, being that k-means needs to set the number of clusters beforehand. DJ-cluster is adaptive, the points of interest are discovered based on the mobility behavior of the individual.

2.2. **K-Means Clustering with Markov Models.** Another approach is by combining k-means clustering with Markov models. K-means clustering is an unsupervised algorithm that groups similar data in groups called clusters [9]. This is a well-known technique for discovering points of interest, such as locations, hence its popularity among academia and industry.

Identifying locations an individual visits can be done by using this algorithm and clustering places alongside a radius, using a variation of k-means [1]. All the geographical locations within the radius are treated as a common one, which is computed using the mean of all the geographical locations recorded within the radius. Setting a small radius will result in more identified places. But, setting one that is too small will end up identifying only one geographical location for each place. Hence, an optimal radius must be found and set. Having this set up, a Markov model [10] or one of its variants can be used to predict the next location of the user, each node in the model representing a place from the cluster.

2.3. **Path Matching Using Cellular Data.** Another proposed approach in order to identify points of interest is by using Global System for Mobile Communications (GSM) cell-based location data [7]. The data saved consists of a cell, which is identified by a string, and a timestamp associated with the cell. This data is then clustered location-based, each cluster representing a place of interest to the user. A cell cluster is a group of nearby cells. Further clustering the routes using incremental clustering, a route being a string representing cell identifiers, locations that the individual visits can be identified. Prediction is done by taking into account the recently visited cells and the current place visited by the user. The next place that the user will visit is decided by finding all the observed routes having as the starting point the current one. Then, the recently visited cells are compared against cells of those identified routes. The destination of the route that shares the most similarities with the current one will be the result of the prediction. Also, several routes may share the same number of similarities. So, the time of day, day of the week or route frequency are taken into account as well when choosing between two or more routes with the same number of commonalities.

2.4. **Decision Trees.** An interesting proposed solution in predicting the next user location is using decision trees. A decision tree is an algorithm that

contains only conditional control statements [2]. It uses a tree-like model of decisions and their possible consequences. Thus, it can be used for predicting the most likely future location to be visited by an individual.

One such solution is presented in [15]. A preliminary step is gathering the points of interest, using the DBSCAN clustering algorithm [5]. Then, a decision tree is built, taking into account the temporal and sequential features. The former denotes the day of the week and the time of the day (split into 24 parts), while the latter refers to the sequence of places the individual visits. Thus, by taking as input the current location of the user and using the built decision tree, the most probable next location the user will visit can be predicted.

As seen in the same paper, the results of the experiments indicate a higher prediction accuracy when using such a decision tree in favor of the classical Markov model, especially in the case of locations with a low-frequency visit.

3. Contextual Next Location Prediction

This section details the proposed approach in solving the problem of predicting the next location of an individual in a mobile environment. Using a variant of a k-nearest neighbor algorithm and data collected for a particular individual, we can predict the next location he/she will visit.

3.1. **Collecting Data.** Before being able to predict the future we need to look in the past. By using prior fetched GPS data for an individual we can gather some geographical locations which he/she has been to, giving us a sense of a routine (daily, weekly, etc.)[1]. These GPS coordinates are saved to a database, alongside their recorded time, giving us access to the day, week, month and year. Tying the geographical feature to the temporal one gives us a better model to work with, by associating the presence of the individual in that specific spot with a specific timestamp.

We chose to use GPS data because it is the most precise option in terms of geolocation. Even though it is the most intensive on the battery, we can minimize its effects on the devices' battery by collecting data when the app is in the background or closed, using a background service, scheduled once every couple of tenths of minutes (see Figure 1). Besides this, the background service effects on the battery are limited by the two major operating systems, Android and iOS. They both provide similar power-saving features, such that the effects of background services and unused apps on the battery are minimal [12], Doze, and App Standby for Android, the counterpart on iOS being Standby. Also,

[1]https://github.com/Lexcrd1337/Location-Prediction-in-Mobile-Applications/blob/master/all.gpx

the battery is not that affected because the collecting data process is done periodically and the phone screen is not always on.

With reference to the devices' memory space issues, the data collected can be saved in a cloud-hosted database. For this particular implementation, the Firebase Realtime Database [11] was used. By doing so, the devices do not need to keep the data on them, which would have led to a decrease in the available memory space.

Regarding privacy, only the collected data for each individual is taken into account when predicting one's next location and all the other users are unable to access this information. No other third party data is collected at all and collecting data about an individual cannot be done without the consent of the user. Also, most cloud-hosted database services offer the possibility of assigning security and access rules, as Firebase Realtime Database does.

3.2. **Algorithm.** The algorithm proposed for predicting the location is a variant of the k-nearest neighbor algorithm. K-NN is a supervised classification algorithm that gives the k closest data points for a new point [3]. The points in our model are the GPS coordinates collected for an individual.

The proposed variation of the k-NN (see Algorithm 1) takes as input the current timestamp from the mobile device and the GPS coordinates collected for the individual by using the application. It then extracts the day of the week from it and searches existing data collected for the same day of the week. Thus, there are 7 prediction problems, one for each day of the week. Next, it finds the k nearest GPS based locations in which timestamps are the closest in terms of hours, minutes and seconds. For the identified locations, a mean is computed based on latitude and longitude, and the result represents the prediction output of the algorithm, consisting of a GPS coordinate.

Using this k-NN variation offers unique advantages as opposed to other solutions. Most existing work must complete a preliminary step before even starting to try and predict the next location. The step consists of gathering the data, the locations. Also, some of them need to train the data fetched, in order to identify common places or routes. But by using k-NN, there is no need for prior fetching the locations or training the data to recognize certain places of interest or routes. Additionally, it is an active learning algorithm. Each new location collected in the system is used in future predictions as well, thus enlarging its model and accuracy, respectively.

4. Experiments and Evaluations

This section presents the evaluation of the prediction algorithm based on collected data for various users. The dataset consists of GPS coordinates in the area of Cluj-Napoca, throughout December 2019 - January 2020 (see Figure

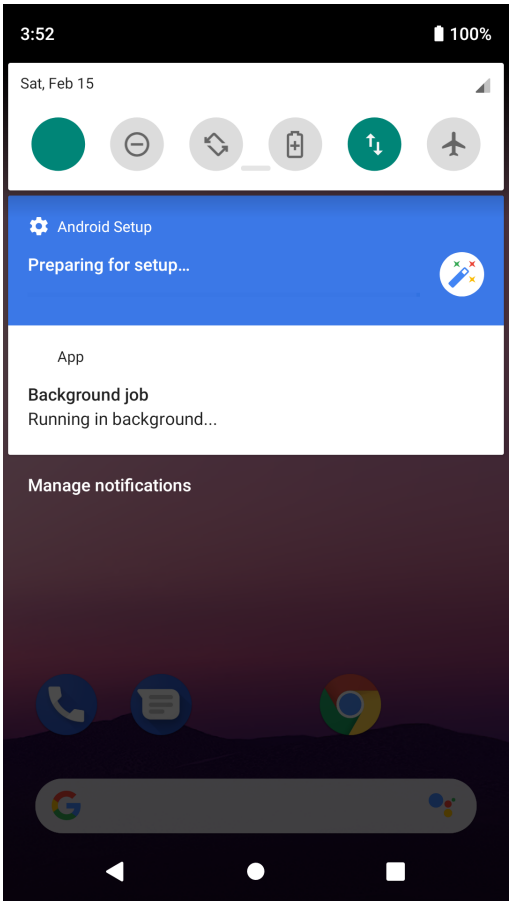


FIGURE 1. Background service (i.e. Background job) for collecting GPS locations when the application is in background or closed

2). In what regards the time feature, the geographical locations were gathered in the time interval between 8:30 AM and 7 PM, with a variation of around 15 minutes between them [1]. More data has been collected during weekdays in comparison with weekends. There are around 80 collected data points for each weekday and 25 for each day of the weekend.

4.1. **Date and Evaluation Metrics.** We have collected GPS data and then started testing it against the algorithm. The locations gathered were saved to a database alongside their timestamp, as previously described.

Algorithm 1 Next Location Prediction Algorithm
Where *gpsCoordinates* - array of the collected GPS coordinates alongside their timestamps,
currentDateTime - timestamp of the current date and time,
NUMBER OF NEIGHBORS - how many neighbors to find

```
1: function predictNextLocation(gpsCoordinates, currentDateTime)
2:     closestLatitudes = [];
3:     closestLongitudes = [];
4:
5:     for neighborNumber in NUMBER OF NEIGHBORS do
6:         closestNeighbor = findClosestNeighbor(gpsCoordinates,
7:                                                 currentDateTime);
8:         closestLatitudes.push(closestNeighbor.latitude);
9:         closestLongitudes.push(closestNeighbor.longitude);
10:        gpsCoordinates.remove(closestNeighbor);
11:    end for
12:
13:    averageLatitude = sum(closestLatitude);
14:    averageLatitude /= NUMBER OF NEIGHBORS;
15:    averageLongitude = sum(closestLongitudes);
16:    averageLongitude /= NUMBER OF NEIGHBORS;
17:
18:    return (averageLatitude, averageLongitude);
19: end function
```

In order to asses the results of the proposed algorithm, we computed the difference in meters between the predicted location and the actual location of the user. We compared the output of our algorithm that predicts the next geographical location with a precise value in meters.

As seen in [16], the simple relationship between distance and qualitative proximity between objects is sensitive to complex spatial structure effects, such as the relative location of objects and the geographic extent of the area being considered. Proximity perception is influenced by the size of the frame of reference in a perceiver's mental map. As a result, what is considered near at one scale may be perceived far at another.

Thus, we considered a prediction as being accurate if the computed difference did not surpass the threshold of 50 meters. We considered this value as being an appropriate one because it is an appreciable distance in a real-case scenario, considered close in walkable distance.

FIGURE 2. Collected data points in the course of one month for a particular user

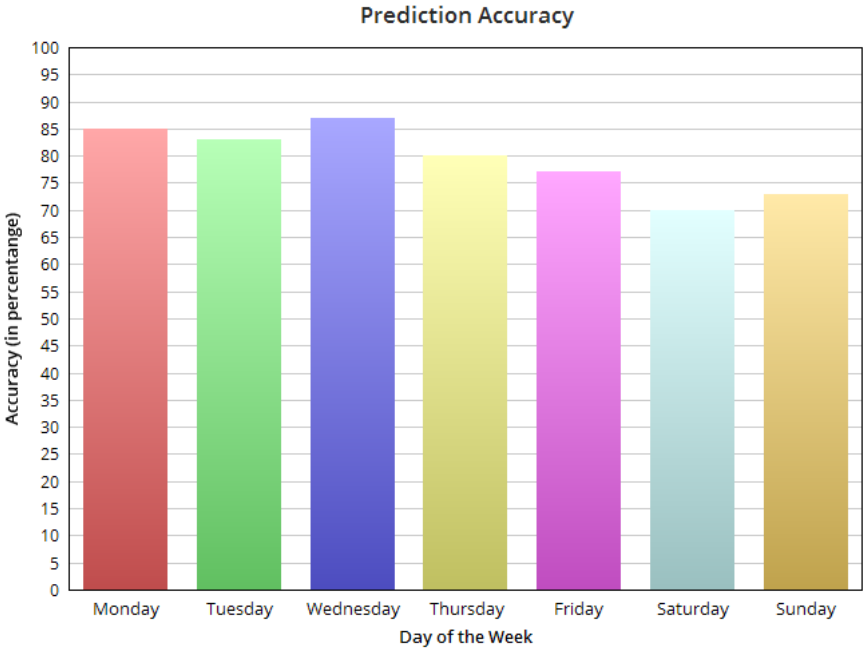


FIGURE 3. Accuracy of the algorithm tested against the collected data

Algorithm 2 Finding the Closest Neighbor Algorithm
Finds the closest GPS coordinate based on the current day of the week and time,
currentDateTime - timestamp of the current date and time

```
1: function findClosestNeighbor(gpsCoordinates, currentDateTime)
2:     hourAndMinuteArray = [];
3:
4:     for gpsCoordinate in gpsCoordinates do
5:         hourAndMinutes = gpsCoordinate.hour + gpsCoordinate.minutes
   / 100;
6:         hourAndMinuteArray.push(hourAndMinute);
7:     end for
8:
9:     hourAndMinuteNow = currentDateTime.hours;
10:    hourAndMinuteNow += currentDateTime.minutes / 100;
11:    firstItem = hoursAndMinutesArray[0];
12:    maxDifference = abs(firstItem − hourAndMinuteNow);
13:    closestNeighborIndex = 0;
14:
15:    for index, hourAndMinute in hourAndMinuteArray do
16:        difference = abs(hourAndMinute − hourAndMinuteNow);
17:        if difference > maxDifference then
18:            closestNeighborIndex = index;
19:            maxDifference = difference;
20:        end if
21:    end for
22:
23:    return gpsCoordinates[closestNeighborIndex];
24: end function
```

4.2. **Numerical Results and Analysis.** Based on the chart (see Figure 3), we can see higher accuracy during weekdays, since there was more data collected during them, as opposed to weekends. Moreover, the collected data has a huge role in influencing the accuracy outcome. Weekdays tended to be more repetitive, mostly following the same course of locations. On the other hand, the data gathered on weekends differed from one weekend to another, only Sundays being much alike. Another factor would be the amount of data on which the test was conducted. The data collected consists of only one season, winter. The season greatly influences the places an individual visits [13]. Only the workplace and home do not vary significantly from one season to

another. During spring and summer, people tend to go out more and practice more activities in the open, such as riding the bike, hiking, going to the beach, playing outdoor sports and so on. But even so, the test shows a high accuracy nonetheless.

In addition, the results of the experiment may fluctuate if the threshold of accepted predictions is changed. For example, if the threshold would be changed to a value between 30 and 70 meters, the accuracy would remain roughly the same throughout each day. But decreasing it to a value between 5 and 10 meters would result in a decrease of the algorithm's truthfulness with about 10%-15% for each day. Nonetheless, it would perform quite well, giving similar results with the other algorithms that have high accuracy, such as the one using a mobility Markov chain presented in [6].

Since the predictability in user mobility is 93% [14], the results of using k-NN are no surprise. It works best when the user is mostly at a commonly visited place for a given timestamp. It does not give so good results as the other presented solutions if the user is not at a commonplace, but given the fact that it is a fairly simple algorithm, not battery consuming and privacy-oriented, it is an acceptable trade-off to make. Furthermore, by collecting more and more data, its efficiency grows even in these cases, when previously it would perform more poorly than the rest.

4.3. **Threats of Validity.** No artificial intelligence-based algorithm works perfectly, with an accuracy of 100%. Our proposed solution may not behave so great when tested against more complex cases.

In our current scenario, only the day of the week and the current timestamp, including the date and the exact hour and minutes are taken into consideration. There are a lot of factors that can influence an individuals' behavior, hence changing the routes he/she takes. Thus, predicting the future location of an individual is a problematic process.

Moreover, depending only on the timestamp as the only input feature in our algorithm can decrease the accuracy of it in certain scenarios. For example, if we follow a common route, but delay the moment in which we start, the algorithm may not perform that well. It will need time to collect more data in order to accommodate the new timestamps for that specific geographical location. In order to avoid this, we may identify the common routes taken by an individual, and find the time and date intervals when they occur. Thus, by recognizing that the individual takes a common route, but at a slightly different hour or day, the algorithm would no longer drop its accuracy values in these scenarios.

Also, since we have only tested during the course of a month in a certain season, results may lose accuracy when the algorithm is tested for a full year

or several years. In that case, we would have to take into account multiple variables. Considering the year and month, besides the current day of the week and timestamp, can improve the results of our algorithm when predicting the location for the course of multiple following years. Hence, extending our model by encapsulating features that can greatly affect how the algorithm behaves, such as the year, month, season or weather, will surely improve its efficiency overall, as well as in those complex scenarios.

5. Conclusions and Future Work

This paper presents a simple and accurate solution for predicting the next location of an individual in a mobile application.

Most prior works regarding this particular problem offer an intensive computing approach, not relying solely on the end-users' mobile device. But, the presented solution manages to do all the processing on the individuals' mobile device, in the background, which is not demanding at all in what regards memory and battery efficiency. Furthermore, users' privacy is also respected by using this approach. The experiments that were carried also show the high accuracy of the proposed solution.

Future work will be comprised of taking into consideration multiple features, such as the weather or the month and year as well, not only the day of the week, extending the method to be capable of predicting the next location using the additional features as well.

References

[1] Ashbrook, D., and Starner, T. Using gps to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous computing 7*, 5 (2003), 275–286.

[2] Breiman, L. *Classification and regression trees*. Routledge, 2017.

[3] Cover, T., and Hart, P. Nearest neighbor pattern classification. *IEEE transactions on information theory 13*, 1 (1967), 21–27.

[4] Du, Y., Wang, C., Qiao, Y., Zhao, D., and Guo, W. A geographical location prediction method based on continuous time series markov model. *PloS one 13*, 11 (2018).

[5] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (1996), vol. 96, pp. 226–231.

[6] Gambs, S., Killijian, M.-O., and del Prado Cortez, M. N. Next place prediction using mobility markov chains. In *Proceedings of the First Workshop on Measurement, Privacy, and Mobility* (2012), pp. 1–6.

[7] Laasonen, K. Clustering and prediction of mobile user routes from cellular data. In *European Conference on Principles of Data Mining and Knowledge Discovery* (2005), Springer, pp. 569–576.

[8] Lin, K., Kansal, A., Lymberopoulos, D., and Zhao, F. Energy-accuracy trade-off for continuous mobile device location. In *Proceedings of the 8th international conference on Mobile systems, applications, and services* (2010), pp. 285–298.

[9] Lloyd, S. Least squares quantization in pcm. *IEEE transactions on information theory 28*, 2 (1982), 129–137.

[10] Markov, A. A. The theory of algorithms. *Trudy Matematicheskogo Instituta Imeni VA Steklova 42* (1954), 3–375.

[11] Moroney, L. The firebase realtime database. In *The Definitive Guide to Firebase*. Springer, 2017, pp. 51–71.

[12] Narzt, W. Context-based energy saving strategies for continuous determination of position on ios devices. In *2014 47th Hawaii International Conference on System Sciences* (2014), IEEE, pp. 945–954.

[13] Smith, K. The influence of weather and climate on recreation and tourism. *Weather 48*, 12 (1993), 398–404.

[14] Song, C., Qu, Z., Blumm, N., and Barabási, A.-L. Limits of predictability in human mobility. *Science 327*, 5968 (2010), 1018–1021.

[15] Xia, L., Huang, Q., and Wu, D. Decision tree-based contextual location prediction from mobile device logs. *Mobile Information Systems 2018* (2018).

[16] Yao, X., and Thill, J.-C. How far is too far?–a statistical approach to context-contingent proximity modeling. *Transactions in GIS 9*, 2 (2005), 157–178.

[17] Ye, J., Zhu, Z., and Cheng, H. What's your next move: User activity prediction in location-based social networks. In *Proceedings of the 2013 SIAM International Conference on Data Mining* (2013), SIAM, pp. 171–179.

[18] Zhou, C., Frankowski, D., Ludford, P., Shekhar, S., and Terveen, L. Discovering personal gazetteers: an interactive clustering approach. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems* (2004), pp. 266–273.

Babeş-Bolyai University, Department of Computer Science, 1 M. Kogălniceanu Street, 400084 Cluj-Napoca, Romania

Email address: caie2278@scs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXV**, Number 1, 2020
DOI: 10.24193/subbi.2020.1.05

PERFORMANCE EVALUATION OF BETWEENNESS CENTRALITY USING CLUSTERING METHODS

BENCE SZABARI AND ATTILA KISS

Abstract. Betweenness centrality measure is used as a general measure of centrality, which can be applied in many scientific fields like social networks, biological networks, telecommunication networks or even in any area that can be well modelled using complex networks where it is important to identify more influential nodes. In this paper, we propose using different clustering algorithms to improve the computation of betweenness centrality over large networks. The experiments show how to achieve faster evaluation without altering the overall computational complexity.

1. Introduction

Nowadays, large networks play a vital role in many scientific areas like computer science [17], social engineering [11], biology [13], chemistry [8], [9], building sensor networks [1]. Over the years many centrality measures were defined such as closeness centrality which is based on the average length of the shortest path between a selected node and all other nodes in the graph. [3] The eigenvector centrality where the score of a node is influenced by the scores of its adjacents nodes. [19] Furthermore, betweenness centrality which quantifies the number of times when a node behaves like a connecting bridge along the shortest path between two vertices. [7].

All of these are used to identify the most important nodes within a graph, based on different conditions and criteria. In this paper, we will focus on betweenness centrality. This centrality measure is still quite popular since it can be applied in many fields. A common example is, in a telecommunications network we can find those nodes which have the most influence over the network since a node with higher betweenness centrality has more control and

Received by the editors: 17 May 2020.

2010 *Mathematics Subject Classification.* 68U99, 94C15.

1998 *CR Categories and Descriptors.* C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *Network communications*; G.2.2 [**Discrete mathematics**]: Graph Theory – *Graph algorithms*.

Key words and phrases. clustering, complex networks, centrality measures, community detection, parallel computation.

more information can pass through. However, calculation of these values for each node is an extremely time-consuming procedure even though, in the past few years several more efficient algorithms have been developed. [6], [2], [14].

1.1. **Related work.** In this paper, we propose a novel cluster-based algorithm which addresses the issue of identifying influential nodes in complex networks using betweenness centrality. For clustering, we use the Louvain, Markov and Paris algorithm, these algorithms were introduced in [4] [18] [5]. Afterwards, we assign a cluster to each of the nodes of the network, we can calculate the centrality measure on these smaller sub-graphs thereby reducing the calculation time. We also use a modified version of this algorithm, where we introduce a parallel computation model.

The paper is organized as follows. Section 2.1 gives a brief overview of the definition of a graph, essential graph theory concepts and the definition of betweenness centrality. In this section, the examined clustering algorithms are also explained and in 2.3 the proposed algorithm was described. In Section 3, the implemented experiments are described in detail which is based on real-life networks to highlight the difference in performance between the proposed and the other methods. The conclusion and the possible future work about the issue in question can be seen in Section 4.

2. Overview of clusterization algorithms

2.1. **Preliminary and concepts.**

Definition 2.1 Adjacency matrix
An adjacency matrix is a $n \times n$ square matrix A, in which:

$$a_{ij} = \begin{cases} 1 & \text{if there exists an edge from } v_i \text{ to } v_j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Definition 2.2 Betweenness centrality
Betweenness centrality is defined as

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} is the total number of shortest paths from node s to node t and $\sigma_{st}(v)$ is the number of those paths that pass through v.

2.2. **Graph clustering.** The goal of graph clustering is to divide nodes into different clusters in a large graph based on specific criteria such as node connectivity or neighbourhood similarity. Clustering techniques are useful for the

detection of densely connected groups in a large graph. In the following, we examine the following clustering algorithms: Markov, Louvain and Paris.

2.2.1. *Markov algorithm.*

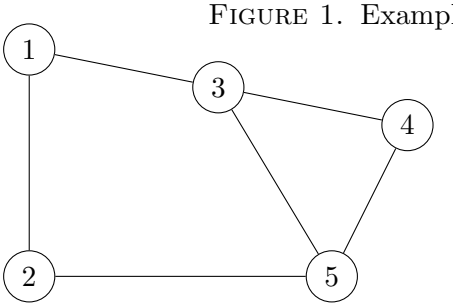
Definition 2.3 Markov matrix
A matrix A is a Markov matrix if its entries are greater than or equal to zero and each column's entries sum to one. Furthermore, each entry represents transition probabilities from one state to another.

Definition 2.4 Random walk
Let G be a graph or a digraph (directed graph) with the additional assumption that if G is a digraph, then $deg^+(v) > 0$ for every vertex v. Now let's have an object placed at vertex v_j. At each stage, the object must move to an adjacent vertex. The probability that it moves to vertex v_i is denoted as

(2)
$$m_{ij} = \begin{cases} \frac{1}{deg(v_j)} & \text{if } (v_j, v_i) \text{ is an edge in } G, \text{ in case of digraph it is } \frac{1}{deg^+(v_j)} \\ 0 & \text{otherwise} \end{cases}$$

where m_{ij} represents the probability that a random walk of length k starting at vertex v_j, ends at vertex v_i, where the length is the number of edges.

FIGURE 1. Example of Markov matrix



$$M = \begin{bmatrix} 0 & 0.5 & 0.33 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0.33 \\ 0.5 & 0 & 0 & 0.5 & 0.33 \\ 0 & 0 & 0.33 & 0 & 0.33 \\ 0 & 0.5 & 0.33 & 0.5 & 0 \end{bmatrix}$$

Random walk is a special case of the Markov chain, using the transition probability matrices. By walking randomly on a graph, we can find out where the flow tends to gather, and therefore where the clusters are located. This is the concept what Markov clustering and other clustering algorithms are based on.

Definition 2.5 Inflation
Given a matrix $M \in R^{k \times l}, M \geq 0$, and a real non-negative number r, the matrix resulting from re-scaling each of the columns of M with the power

coefficient r is called $\tau_r M$ and τ_r is called the inflation operator with power coefficient r.

$$\tau_r : \mathbb{R}^{k\times l} \longrightarrow \mathbb{R}^{k\times l}$$
$$(\tau_r M)_{pq} = (M_{pq})^r / \sum_{i=1}^{k} (M_{iq})^r$$

In this case, the inflation operator is responsible for both strengthening and weakening of current while the parameter r controls the extent of this strengthening or weakening. Expansion is when we take the powers of the Markov Chain transition matrix, and it is responsible for allowing flow to connect different regions of the graph.

The algorithm converges to a "doubly idempotent" matrix which is at steady-state and there is only one value in a single column. To be able to identify clusters, the nodes are divided into two groups: attractors that attract other nodes, and nodes influenced by the previous group. Attractors always have at least one positive value within their row in the steady state-matrix and they draw those nodes that having a positive value in the same row. Nodes that have such a relationship can be classified in the same group.

FIGURE 2. Example of steady state matrix

$$\begin{bmatrix} 1 & - & 1 & - & - & - \\ - & - & - & - & - & - \\ - & 1 & - & 1 & - & - \\ - & - & - & - & 0.5 & 0.5 \end{bmatrix}$$

clusters: (1, 3), (2, 4), (5, 6)

The Markov clustering algorithm can be summarized as follows. Given an undirected graph, power parameter denoted as e, and inflation parameter denoted as r. First create the adjacency matrix, then add self loop to each node. After we have to normalize the matrix and expand it by taking the e^{th} power of it then inflate by taking inflation of the result with parameter r. Repeat steps until we reach the steady-state, finally we can obtain clusters.

2.2.2. *Louvain algorithm.*

Definition 2.6 Modularity
The modularity of a partition is a scalar value between -1/2 and 1 that measures the density of links inside communities as compared to links between communities. In case of weighted networks, it is defined by Newman and Girvan [15]

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

where

- A_{ij} represents the edge weight between nodes i and j
- k_i and k_j are the sum of the weights of the edges attached to nodes i and j, respectively
- m is the sum of all of edge weights in the graph
- c_i and c_j are the communities of the nodes
- δ is Kronecker delta function ($\delta_{x,y} = 1$ if $x = y$, 0 otherwise)

Modularity can be used to compare the nature of the partitions collected by different methods. Louvain clustering can find high modularity partitions of large graphs and also unfold a complete hierarchical community structure for the network quite effectively.

To maximize modularity, the Louvain method uses two phases that are repeated iteratively. The algorithm's input is a weighted graph of n nodes. In the first step, we assign different community to each node of the graph which means we have n communities. After that, for each node i we calculate the gain of modularity what we can achieve by deleting i from its community and by putting into a community of j, where j is the neighbour of i. The gain in modularity ΔQ, when node i is moved into community C, can be calculated as:

$$\Delta Q = \left[\frac{\sum_{in} + k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right]$$

where

- $\sum_{in}$ is sum of the weights of the links inside C
- $\sum_{tot}$ is the sum of the weights of the links incident to nodes in C
- k_i is the sum of the weights of the links incident to node i
- $k_{i,in}$ is the sum of the weights of the links from i to nodes in C and m is the sum of the weights of all the links in the network

After this value is calculated for all communities i is connected to, i is placed into the community that resulted in the largest modularity increase, if there is no such increase then node i stays its original community. This process is applied to all nodes until no modularity increase can occur.

In the second step, it groups all nodes in the same community and creates a new network where the nodes are the groups of the previous step. Links

between nodes of the same group are represented as self-loops while links between different communities defined as weighted edges between the community nodes.

2.2.3. *Paris algorithm.*

Definition 2.7 Weighted adjacency matrix
A weighted adjacency matrix is an A symmetric, non-negative matrix that for each pair of $(i, j) \in V, a_{ij} > 0$ iff there is an edge between node i and node j, and in this case a_{ij} is the weight of the edge $i, j \in E$.

Definition 2.8 Weight of a node
The weight of a node i is the sum of the weights of its incident edges

$$w_i = \sum_{j \in V} A_{ij}.$$

In the case of unit weights, w_i is the degree of node i. The cumulative weight of the nodes is

$$w = \sum_{i \in V} w_i = \sum_{i,j \in V} A_{ij}.$$

In the following, we will refer to C as a cluster which is a subset of V. The weights introduce a probability distribution on node pairs

$$\forall i, j \in V, \quad p(i, j) = \frac{A_{ij}}{w},$$

and also a probability distribution on nodes,

$$\forall i \in V, \quad p(i) = \sum_{j \in V} p(i, j) = \frac{w_i}{w}.$$

The distance between node i, j is defined as the node pair sampling ratio.

$$d(i, j) = \frac{p(i)p(j)}{p(i, j)}$$

The node distance can be also defined in a different way, which comes from the following conditional probability.

$$\forall i, j \in V, \quad p(i, j) = \frac{p(i, j)}{p(j)} = \frac{A_{ij}}{w_j}$$

Then the distance between i and j can be be written as

$$d(i,j) = \frac{p(i)}{p(i|j)} = \frac{p(j)}{p(j|i)}$$

Accordingly, consider a cluster C of the graph G. The weights induce a probability distribution on cluster pairs

$$\forall a, b \in C, \quad p(a,b) = \sum_{i \in a, j \in b} p(i,j)$$

and also a probability distribution on clusters,

$$\forall a \in C, \quad p(a) = \sum_{i \in a} p(i) = \sum_{b \in C} p(a,b)$$

and then the cluster pair sampling ratio is the distance between two different clusters a, b:

$$d(a,b) = \frac{p(a)p(b)}{p(a,b)}$$

Defines a conditional probability

$$\forall a, b \in C, \quad p(a|b) = \frac{p(a,b)}{p(b)}$$

which is the conditional probability of sampling a given that b is sampled, we have

$$d(a,b) = \frac{p(a)}{p(a|b)} = \frac{p(b)}{p(b|a)}$$

this distance will be used in Paris clustering algorithm to merge the closes clusters.

As in the Louvain algorithmn it starts with unique cluster therefore, every node is a cluster. In all phases, the two closest cluster will be merged. In section 2.2.2 we defined cluster modularity which can be rewritten in terms of probability distributions,

$$Q(C) = \sum_{i,j \in V} p(i,j) - p(i)p(j)\delta_C(i,j).$$

It can also be expressed from the probability distributions at the cluster level,

$$Q(C) = \sum_{a \in C} (p(a,a) - p(a)^2).$$

As Fortunato et al presented, maximizing the modularity has a resolution limit. Therefore Bonald and Charpentier [5] introduced a multiplicative factor γ, called the resolution. Then the modularity becomes:

$$Q_\gamma(C) = \sum_{i,j \in V} (p(i,j) - \gamma p(i)p(j))\delta_C(i,j)$$

Thus, the Paris algorithm can be understood as the deeply modified version of the Louvain algorithm, where the first phase is replaced by a simple merge.

2.3. **Proposed algorithm.** We now introduce our algorithm that can obtain betweenness centrality of large networks in a short time. We assume that out network G is undirected and consists of N nodes. The algorithm is divided into two-phase. First of all, we have to create clusters from the given network by using the introduced algorithms. Once the clusters are generated, we receive a mapping

$$\begin{aligned} c_1 &\longrightarrow [node\ ids] \\ &\dots \\ c_n &\longrightarrow [node\ ids] \end{aligned} \tag{3}$$

where the keys are the cluster labels and the values are a list of nodes that belong to that cluster. These partitions form the basis of the method, therefore the mappings are saved in JSON format for later use. It is important to note that in the case of Markov clustering, the final result is greatly influenced by the value of the inflation parameter, therefore, the choice of this parameter requires a pre-calculation for each graph. To optimize this parameter, we selected numbers between $[1.1, 2, 6]$ intervals and then determined the value that produced the best modularity value.

In the second phase, we apply a concept similar to the divide and conquer programming strategy to graphs, as follows: first we have our large network G, then we load the previously generated cluster mapping and we create a list of sub-graphs based on the mapping. Thus, we obtain significantly smaller slices from the graph, on which the calculation of the centrality value is a much less expensive subproblem. Thereafter, we have to iterate through this list of sub-graphs defined by the clusters and execute the algorithm for calculating the betweenness centrality as described in [6].

To determine the betweenness centrality value for node v, we have to sum the pair-dependencies of all pairs on that node, where the pair-dependency was introduced as the dependency of a node $s \in V$ on a single node $v \in V$:

$$\delta_s(v) = \sum_{t \in V} \delta_{st}(v).$$

The calculation of the betweenness centrality can be split in two steps. First, we compute the length and the number of the shortest paths between all pairs then finally sum all pair-dependencies.

After the algorithm has performed on a single sub-graph, the sub-results are aggregated to obtain the centrality values for the whole graph.

Algorithm 1: Clusterized betweenness centrality

```
Input  : graph
Output: centrality-dict
centrality_dict ← {};
cluster_types ← ['Markov', 'Louvain', 'Paris'];
for type in cluster_types do
    cluster_mapping ← create_clusters(type);
    sub_graphs ← graph.subgraph(cluster_mapping);
    for sub_graph in sub − graphs do
        subresult ← calculate_bc();
        update(centrality_dict, subresult);
    end
end
return centrality_dict;
```

We also introduce a parallel version of the previous method which takes advantage of the multi-core computing capacity of today's modern processors. It contains the following changes: when we pass a sub-graph of G denoted as $G^{'}$ to calculate the betweenness centrality, the function that calculates centrality will accept a bunch of nodes and computes the contribution of those nodes to the centrality of the whole network.

Also, it divides the network into chunks of nodes, therefore those centrality measures can be calculated on separate different CPU cores as described in. [10] To determine the node of chunks, we take the particular sub-graph and divide it into $N/number\ of\ cpu\ cores$ pieces where N is the number of nodes in the sub-graph.

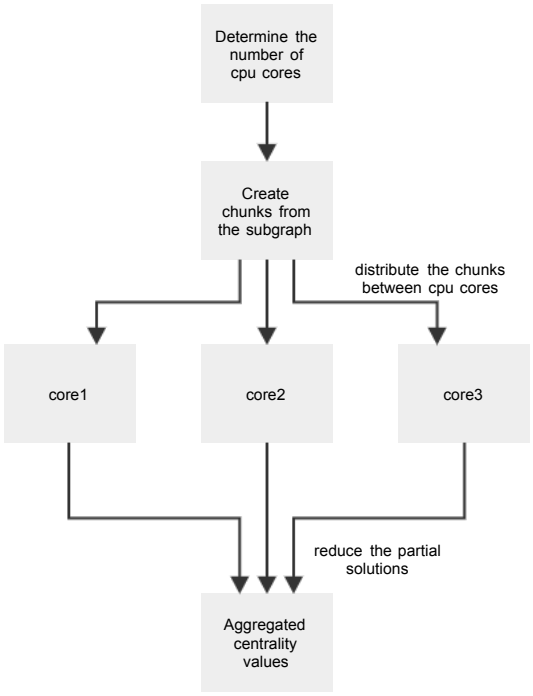


FIGURE 3. Parallelized betweenness centrality

3. EXPERIMENTS

For our experiments, we used six actual networks to evaluate the performance of the examined method in case of big networks. The datasets are mostly from social sites, which served as an excellent basis for our observation, as they have many vertices and edges, and we also took into account road networks, web-networks. These networks are fb-pages-food, which consist of 620 nodes and 2100 edges, facebook-combined that has 4039 nodes and 88234 and power-bcspwr10 with 5300 nodes and 8300 edges. The fb-pages-politician includes 5900 nodes and 41700 edges. Also web-spam with 4800 nodes and 37400 edges, road-euroroad with 1200 nodes and 1400 edges. These networks can be gathered from [16] [12].

To demonstrate the efficiency of the proposed method, we made several measurements with different scenarios. As we mentioned earlier we chose betweenness centrality at the heart of our research therefore we will compare these calculated values with the values presented by the investigated method. In the proposed method we use Louvain, Markov and Parov clustering to calculate these centrality values. The experiments are divided into the following three parts and explained below.

From the clustering algorithms, only Markov clustering is not a parameter-free method, in which case the following inflation parameters were used: fb-pages-food (1.23), fb-pages-politician (1.17), while in the other networks inflation parameter was set to 2.

TABLE 1. Clusterization of the network

network	louvain	markov	paris
facebook-combined	16 (0.37)	10 (0.83)	6 (0.19)
fb-pages-food	19 $(-12 \cdot 10^{-4})$	10 (0.65)	6 $(-6 \cdot 10^{-4})$
fb-pages-politician	30 $(-25 \cdot 10^{-4})$	23 (0.87)	8 $(-17 \cdot 10^{-4})$
power-bcspwr10	50 (0.01)	30 (0.95)	4 (0.28)
road-euroroad	48 (0.41)	40 (0.88)	26 (0.09)
web-spam	2 $(-13 \cdot 10^{-4})$	29 (0.50)	2 $(-4 \cdot 10^{-6})$

Table 1 summarizes how the networks were split into clusters, the first numbers are the number of clusters while the values in brackets are the clusters' modularity. From this, it can be seen that the Louvain and Markov methods created order of magnitude a similar number of clusters while the Paris algorithm was able to detect quite a few groups within the same graph. In terms of modularity, it can be seen that the Louvain algorithm has performed best, while the other two algorithms have detected the lower density of links inside communities.

3.1. **Experiment 1.** In this experiment, we compared the centrality values for each node (referred to as simple) with values extracted from the proposed method.

TABLE 2. facebook-combined top 5 nodes

rank	simple	louvain	markov	paris
1.	107 (0.48)	437 (0.86)	0 (0.82)	0 (0.76)
2.	1684 (0.34)	0 (0.83)	107 (0.72)	107 (0.68)
3.	3437 (0.24)	3980 (0.80)	698 (0.32)	1684 (0.31)
4.	1912 (0.23)	1684 (0.66)	1085 (0.32)	1912 (0.29)
5.	1085 (0.15)	1912 (0.54)	862 (0.32)	1577 (0.22)

TABLE 3. fb-pages-food top 5 nodes

rank	simple	louvain	markov	paris
1.	265 (0.35)	483 (1.00)	265 (0.28)	265 (0.28)
2.	31 (0.16)	141 (0.90)	518 (0.16)	518 (0.16)
3.	518 (0.14)	96 (0.83)	31 (0.12)	31 (0.12)
4.	618 (0.09)	265 (0.80)	220 (0.09)	220 (0.09)
5.	35 (0.09)	49 (0.74)	217 (0.08)	340 (0.09)

TABLE 4. power-bcspwr10 top 5 nodes

rank	simple	louvain	markov	paris
1.	5268 (0.27)	5144 (0.58)	4774 (0.00)	5040 ($9.8 \cdot 10^{-6}$)
2.	3160 (0.26)	2645 (0.52)	2941 (0.00015)	4146 ($15 \cdot 10^{-5}$)
3.	5298 (0.26)	4074 (0.47)	1519 (0.00)	4898 ($14 \cdot 10^{-5}$)
4.	5040 (0.25)	4891 (0.47)	4671 (0.00)	5074 ($13.8 \cdot 10^{-5}$)
5.	4752 (0.23)	5232 (0.47)	5223 (0.00)	4870 ($13.5 \cdot 10^{-5}$)

TABLE 5. fb-pages-politician top 5 nodes

rank	simple	louvain	markov	paris
1.	5800 (0.27)	3008 (0.80)	5800 (0.16)	5800 (0.28)
2.	1864 (0.06)	5646 (0.74)	2900 (0.14)	3576 (0.07)
3.	3576 (0.05)	5699 (0.65)	4032 (0.09)	1864 (0.07)
4.	2900 (0.05)	5800 (0.64)	3094 (0.06)	1965 (0.06)
5.	1324 (0.05)	96 (0.56)	158 (0.05)	4032 (0.04)

TABLE 6. road-euroroad top 5 nodes

rank	simple	louvain	markov	paris
1.	401 (0.21)	58 (1.00)	111 (1.00)	111 (1.00)
2.	283 (0.21)	61 (1.00)	968 (0.17)	401 (0.20)
3.	276 (0.20)	646 (1.00)	1132 (0.17)	400 (0.20)
4.	452 (0.18)	18 (0.72)	1133 (0.17)	283 (0.18)
5.	451 (0.17)	917 (0.68)	1103 (0.13)	431 (0.17)

TABLE 7. web-spam top 5 nodes

rank	simple	louvain	markov	paris
1.	4044 (0.09)	3956 (1.00)	4044 (0.09)	4044 (0.09)
2.	981 (0.06)	327 (1.00)	981 (0.06)	981 (0.06)
3.	2561 (0.04)	709 (1.00)	2561 (0.05)	2561 (0.04)
4.	3114 (0.03)	1548 (1.00)	2916 (0.04)	3114 (0.03)
5.	3070 (0.03)	813 (0.95)	3070 (0.04)	3070 (0.03)

In the case of all networks, it can be seen that the investigated method mostly retained the nodes that were ranked among the top 5 peaks in the original calculation, only a few cases added new nodes to the top of the ranking.

3.2. **Experiment 2.** In this experiment, the efficiency and the performance were considered, resulting in a comparison of the runtime of the original betweenness centrality method and the proposed method.

It can be seen that the investigated method's improved the performance of the calculation of betweenness centrality. In this aspect, the Markov method

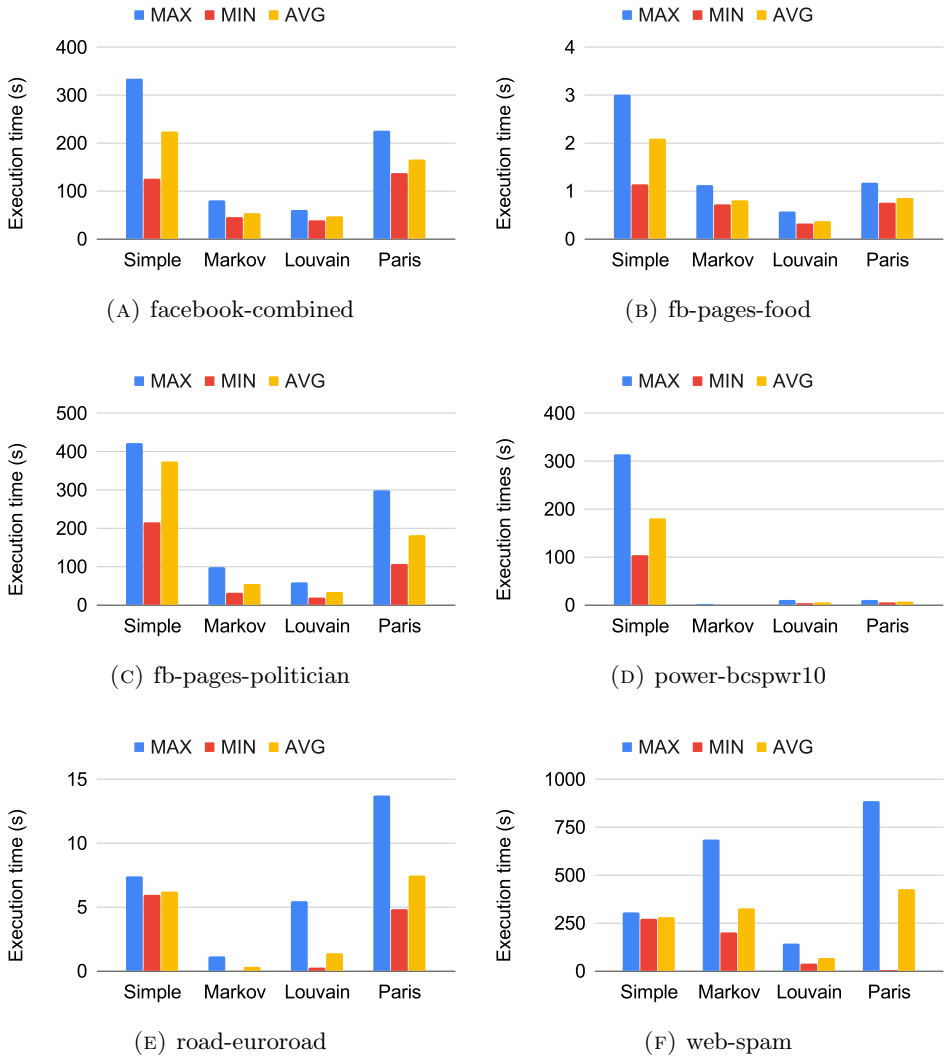
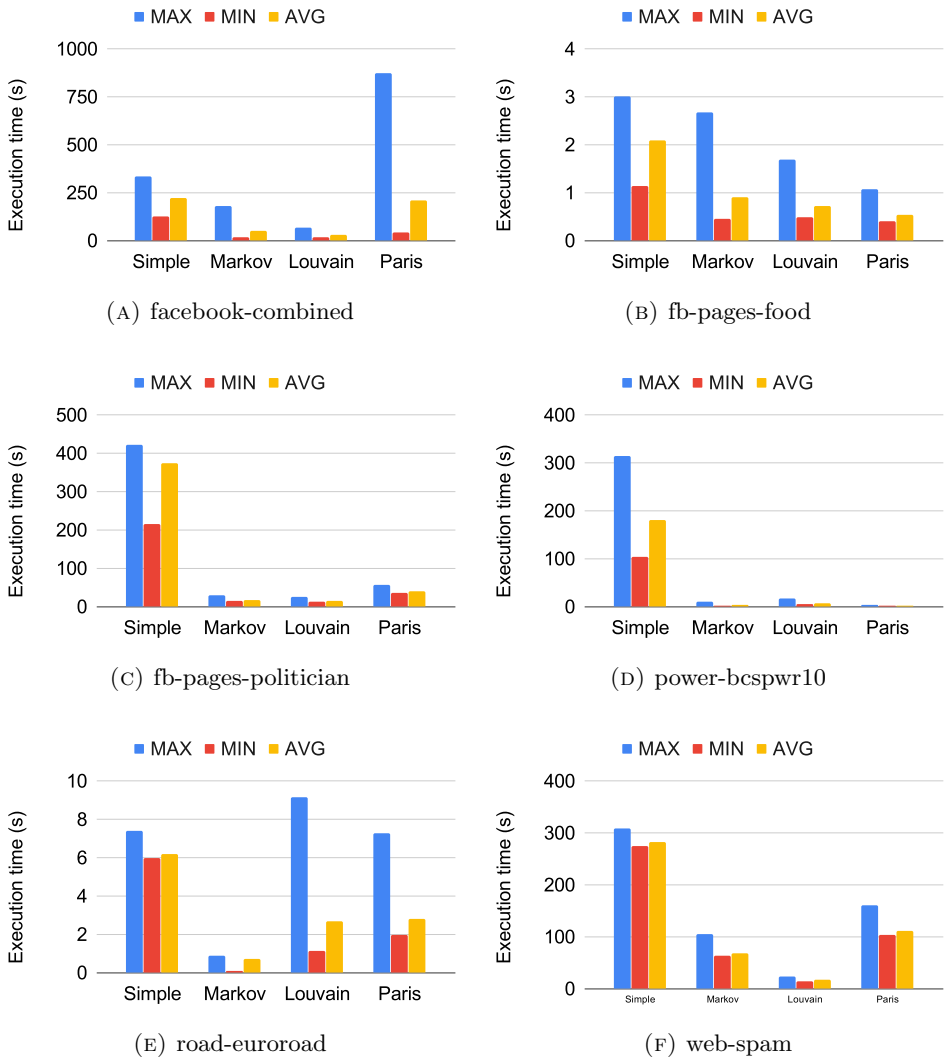


FIGURE 4. Performance with clustering

falls behind Louvain, while the Paris clustering outstandingly worse than the others. The reason behind this is, that clustering algorithm, as shown in the table above, has formed a fairly small number of groups, so it is not worthwhile to perform the division with such a low cluster number.

3.3. **Experiments 3.** In this experiment also the efficiency and the performance were considered as in the previous but in this case, we used parallel

computation of betweenness centrality which was compared with the original betweenness centrality method.



(A) facebook-combined

(B) fb-pages-food

(C) fb-pages-politician

(D) power-bcspwr10

(E) road-euroroad

(F) web-spam

FIGURE 5. Performance with clustering and parallel computation

From these results, it can be seen that the computation time can be significantly reduced if we use clustering with parallel bc calculation. However,

in some cases, spikes are visible in terms of maximum runtime. We have observed that if the graph fits in the memory the traditional calculation method is more profitable, and this method only worth for larger graphs.

4. Conclusion and future work

In this paper, we examined the performance of the proposed method to calculate betweenness centrality based on network clustering. This method calculates the centrality value for each node in the sub-graphs determined by different clustering methods. Our proposed solution is based on Louvain, Markov and Paris clustering algorithms. We investigated the method in large networks, to get a better picture of its performance. To determine the correctness of the method, we compared the values of the five most influential nodes obtained by the basic bc method with the values and nodes obtained by the method we proposed. The experimental results showed that the Louvain's performance was the best of all the investigated clustering methods since in most of the cases it was able to create an optimal number of clusters.

We only investigated the proposed method with betweenness centrality however, it is possible to use the same procedure with different centrality measures, which we intend to do in the future.

5. Acknowledgements

The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

References

[1] Aberer, K., Hauswirth, M., and Salehi, A. Infrastructure for data processing in large-scale interconnected sensor networks. In *2007 International Conference on Mobile Data Management* (2007), IEEE, pp. 198–205.

[2] Bader, D. A., Kintali, S., Madduri, K., and Mihail, M. Approximating betweenness centrality. In *International Workshop on Algorithms and Models for the Web-Graph* (2007), Springer, pp. 124–137.

[3] Bavelas, A. Communication patterns in task-oriented groups. *The journal of the acoustical society of America 22*, 6 (1950), 725–730.

[4] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment 2008*, 10 (2008), P10008.

[5] Bonald, T., Charpentier, B., Galland, A., and Hollocou, A. Hierarchical graph clustering using node pair sampling. *arXiv preprint arXiv:1806.01664* (2018).

[6] Brandes, U. A faster algorithm for betweenness centrality. *Journal of mathematical sociology 25*, 2 (2001), 163–177.

[7] Freeman, L. C. A set of measures of centrality based on betweenness. *Sociometry* (1977), 35–41.

[8] Gasteiger, J., and Zupan, J. Neural networks in chemistry. *Angewandte Chemie International Edition in English 32*, 4 (1993), 503–527.

[9] Gonzalez-Diaz, H., Vilar, S., Santana, L., and Uriarte, E. Medicinal chemistry and bioinformatics-current trends in drugs discovery with networks topological indices. *Current Topics in Medicinal Chemistry 7*, 10 (2007), 1015–1029.

[10] Hagberg, A., Swart, P., and S Chult, D. Exploring network structure, dynamics, and function using networkx. Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[11] Irani, D., Balduzzi, M., Balzarotti, D., Kirda, E., and Pu, C. Reverse social engineering attacks in online social networks. In *International conference on detection of intrusions and malware, and vulnerability assessment* (2011), Springer, pp. 55–74.

[12] Leskovec, J., and Krevl, A. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[13] Mason, O., and Verwoerd, M. Graph theory and networks in biology. *IET systems biology 1*, 2 (2007), 89–119.

[14] Newman, M. E. A measure of betweenness centrality based on random walks. *Social networks 27*, 1 (2005), 39–54.

[15] Newman, M. E., and Girvan, M. Finding and evaluating community structure in networks. *Physical review E 69*, 2 (2004), 026113.

[16] Rossi, R. A., and Ahmed, N. K. The network data repository with interactive graph analytics and visualization. In *AAAI* (2015).

[17] Sen, S., and Wang, J. Analyzing peer-to-peer traffic across large networks. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment* (2002), pp. 137–150.

[18] van Dongen, S. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, 2000.

[19] Zaki, M. J., and Meira, W. *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.

Eötvös Loránd University, Budapest, Hungary
Email address: n0qsdc@inf.elte.hu

J. Selye University, Komárno, Slovakia
Email address: kissae@ujs.sk

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXV**, Number 1, 2020
DOI: 10.24193/subbi.2020.1.06

GENERALIZED CELLULAR AUTOMATA FOR EDGE DETECTION

ALINA ENESCU

Abstract. Cellular Automata (CA) are simple, easily parallelizable models that have been used extensively for various computational tasks. Such models are especially useful for image processing, as mapping automaton cells to image pixels is straightforward and intuitive. This paper proposes a novel optimization framework for CA rules based on evolutionary algorithms and used in edge detection. This approach addresses the problem of optimizing an individual CA rule for one image and extends it to the optimization of a generic CA rule for a set of similar images. In order to maximize the transferability of the optimized rule, the algorithm is trained on sets of images using a curriculum learning approach. A study of the supervised fitness function as well as batch optimization experiments show that the algorithm is robust and competitive with the state-of-the-art methods.

1. Introduction

Cellular Automata (CA) are simple models of parallel computation easy to be mapped on images which makes them suitable to be used in image processing. The image processing task that will be tackled in the paper is edge detection in greyscale images. The scope is to find the most suitable edge detector for sets of images. The proposed approach uses a Genetic Algorithm (GA) to evolve the CA's rule to detect edges both on images individually and on sets of images using a curriculum learning setting.

The proposed approach is an improved version of a previous approach (7). Compared to this approach where only two parameters of the CA's rule are

Received by the editors: 22 May 2020.

2010 *Mathematics Subject Classification.* 65K10, 90C70.

1998 *CR Categories and Descriptors.* F.1.1 [**Computation by abstract devices**]: Models of Computation – *Unbounded-action devices (cellular automata)*; I.2.8 [**Artificial intelligence**]: Problem Solving, Control Methods, and Search – *Heuristic methods*; I.4.6 [**Image processing and computer vision**]: Segmentation – *Edge and feature detection.*

Key words and phrases. Cellular Automata, Genetic Algorithm, Edge Detection, Curriculum Learning, Optimization.

optimized, in the proposed approach all three parameters of the CA's rule are optimized by the means of a GA.

Many edge detectors were proposed in the literature, among which several based on gradient and also several based on CA. As gradient based methods, the ones that are the most known are Canny detetctor (1), Sobel operator (22), while as CA based methods, there are two categories: simple CA (15; 5; 3) and evolved CA (26; 13; 8). The main difference between the proposed approach and the gradient based methods is that the proposed approach adapts to each image or sets of images, while the gradient based methods do not. Adaptability to images individually or to sets of images represents an advantage in image processing, leading to shorter time spent on new batches of images. The same property of adaptability may be seen at the evolved CA compared to simple CA where the rule is usually fixed and directly applied to the images. Compared to a similar evolved CA method (26) where one parameter of the CA's rule, the linear rule, is fixed and only the other two parameters are evolved by the means of a Particle Swarm Optimization (PSO) algorithm, in the proposed approach all three parameters are evolved by the means of a GA. Moreover, there are differences in the methodologies chosen to search for the optimal CA's rule compared to similar evolved CA approaches, GA instead of Learning Automata (LA) or PSO.

The experiments along with the results shown in this paper are meant to assess the capabilities of the proposed edge detector based on evolved CA.

The rest of the paper is structured as follows: Section 2 is a theoretical overview on the edge detection problem, cellular automata and the quality metric; related work in the field is briefly presented in section 3; section 4 details the proposed edge detection approach and the results are presented in Section 5; a summary of the experiments together with future improvements are discussed in Section 6.

2. Theoretical background

Before describing the methods proposed in the current literature, the problem of edge detection is formulated, followed by a brief introduction to Cellular Automata (CA) and some theoretical aspects regarding the performance measure of the methods.

2.1. **Edge detection problem.**

A given point P_0 of a greyscale image having values within $\{0, 1, \ldots, 255\}$ and eight neighbours $P_i, i \in \{1, 2, \ldots, 8\}$ as in Figure 1a, may be classified as an edge (11) if a significant local change in the intensity of the neighbours $P_i, i \in \{1, 2, \ldots, 8\}$ is observed. The curve that contains the points classified as edges is called a contour.

The system meant to extract the set of edge points for a given image is an algorithm called edge detector. Given a set of images, finding an edge detector for them is what the edge detection problem refers to.

(A) (B)

FIGURE 1. Visual representations of Moore neighbourhood. (A) A visual representation of a point P_0 and its eight surrounding neighbors $P_i, i \in \{1, 2, \ldots, 8\}$. (B) A visual representation of the nine fundamental rules for a 2D cellular automaton with a Moore neighbourhood, based on which a linear rule is obtained.

2.2. **Cellular Automata.** Cellular Automata (CA) are simple models of parallel computation (16). A CA is usually represented by a five tuple

$$CA = \{C, N, S, s_0, F\},$$

where C is the set of cells, N is the set of neighbors of size n: $n =\mid N \mid$, S is the set of states, s_0 is the initial state of the CA and $F : S^n \to S$ is the transition rule. The state of the cells are synchronously (or asynchronously) updated by the transition rule according to the state of the neighbor cells and the state of the cells themselves. The equation of the transition rule is:

$$s_i(t+1) = F(s_i(t), S_i(t)),$$

where $s_i(t)$ and $s_i(t+1)$ are the states of the i^{th} cell at time t and time $t+1$, respectively, and $S_i(t)$ is the state set of the i^{th} cell's neighbors at time t.

2.3. **Performance measure.** The Baddeley's Delta Metric (BDM), presented in (19) measures the dissimilarity between two binary images, but besides other metrics, it considers the spatial distribution of the pixels that differ.

Given I_1 and I_2, two binary images with the same sizes $M \times N$ and $\Omega = \{1, .., M\} \times \{1, .., N\}$ the set of each pixel's position in the images, one may define the l-BDM between these two images for a $0 < l < \infty$ as Equation 1 (denoted $\Pi^l(I_1, I_2)$):

$$\Pi^l(I_1, I_2) = \left[\frac{1}{|\Omega|} \sum_{t \in \Omega} |w(d(t, b(I_1))) - w(d(t, b(I_2)))| \right]^l \tag{1}$$

where $b(I_i)$ is the set of feature (edge) points in I_i, $d(t, b(I_i))$ represents the distance between a position t and the closest feature point and $w : [0, \infty] \to [0, \infty]$ is a concave function. In the experiments the Euclidean distance was used, while $w(x) = x$ and $l = 2$.

3. Related work

There are multiple edge detection approaches in the literature which involve the use of Cellular Automata (CA). For greyscale edge detection, there are approaches such as (18) which applies on only one image the 512 linear rules and split them in three groups: no edge detection rules, strong edge detection rules and weak edge detection rules. In (14), the authors identifies only 4 rules out of the 512 linear rules as being the best for edge detection.

EAs have also been proposed for optimizing CA rules for edge detection. An edge detection approach based on CA, fuzzy logic and Particle Swarm Optimization (PSO) is presented in (26). Similar to (21), the authors uses a heuristic membership function to decide the cell's state according to its neighbours. Then a threshold is used to classify the cell as an edge or a non–edge by comparing it with the value obtained by the heuristic membership function.

Apart from these methods, we mention the classic well-known methods based on computing the gradient: Canny edge detector (1), Sobel operator and Prewitt operator (17). The Canny edge detector presented in (1) is a method characterized by three principles: detection, localization and single edge response. The first step of the Canny method is to smooth the greyscale image by Gaussian filtering, then to compute the first derivative in both vertical and horizontal directions. As the second step, it finds the magnitude and direction of each pixel. As a final step, it uses a non-maximal suppression to ensure that the edges have one-pixel width and it uses two thresholds to select the final edge points and to trace them.

4. Proposed approach

4.1. **Cellular Automata Model.** As described in the introduction, the proposed approach is an edge detector based on evolved CA. This method uses a 2D-CA for which the cells are arranged as a grid, the states values are $S = \{0, 1, ..., 255\}$, the neighborhood N is the Moore neighborhood and the initial state s_0 is extracted from the input image. To compute the next state,

first an *edge membership function* (24; 25) is applied. The *edge membership* for a pixel $P_0 \in \{0, 1, ..., 255\}$ and its eight neighbours $P_i \in \{0, 1, ..., 255\}$ is defined as:

$$\mu(P_0) = \frac{\sum_i |P_0 - P_i|}{\Delta + \sum_i |P_0 - P_i|}, \tag{2}$$

where $i \in \{1, 2, ..., 8\}$ is selected according to a linear rule. The parameter $\Delta \in \{0, ..., 255\}$ (one of the possible gray values for a pixel represented on 8 bits) is inversely proportional to the number of detected edges. The higher its value, the fewer edges will be detected. The transition rule is a function $F : P \mapsto \{0, 1\}$ given by Equation 3 (26) which gives the final edge map. The value of the threshold $\tau \in [0, 1)$ controls the number of pixels that pass as edges, therefore the higher the threshold the fewer values will pass.

$$F(P_0) = \begin{cases} 1, & \text{if } \mu(P_0) > \tau \\ 0, & \text{if } \mu(P_0) \leq \tau \end{cases} \tag{3}$$

The quality of the edge detection is influenced by the Δ and τ parameters as well as the linear rule. Choosing these values, however, is a non-trivial task, therefore automatizing the search for the best values is necessary.

4.2. **Evolved Cellular Automata.** The GA used to evolve the CA's rule to detect edge points is presented in the following.

The chromosome encodes the three parameters that need to be optimized: the gray encoding (9; 2; 20) of the two parameters, Δ and τ, and the binary encoding of a linear rule. In order to use the gray encoding for the real parameter τ, it was encoded an integer value within $\{0, 1, \ldots, 127\}$ and divided it by 128 to obtain the real value when the Equation 3 is applied. The choice of using the gray encoding of the two parameters was to keep the genetic operators, crossover and mutation, used in the binary encoding and to overcome the Hamming Cliff (20) problem when using the binary encoding of an integer. Figure 2 shows an example of a chromosome encoding the binary representation of linear rule 35, highlighted in blue, the grey code of threshold, highlighted in purple and the grey code of parameter Δ, highlighted in red.

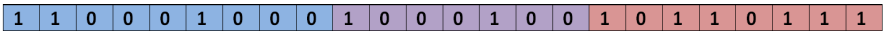


FIGURE 2. An example of a chromosome encoding linear rule 35, the threshold $\tau = 120$ (for Equation 3 will be $120/128 = 0.9375$) and the parameter $\Delta = 218$.

In order to assess the quality of the individuals a fitness function is needed. The fitness function is computed as the Dice Similarity Coefficient (DSC)

(4; 23) from Equation 4. Because DSC quantifies the correctly detected edge points more precisely, it becomes more effective.

$$DSC = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}, \tag{4}$$

where TP is the number of true positives, FP is the number of false positives and FN is the number of false negatives.

For crossover, the binary tournament selection is used to select two chromosomes and three random cross–point are chosen such as all three components to be crossed. The three–points crossover applied to the chromosomes is expressed in Figure 3. Then the bit flip mutation is applied to each offspring: for each component of the offspring, a random bit of the gene is selected and inverted by a probability. Both offspring are added to the new population. These genetic operators are applied to the population until the new population reaches the same size as the old one and replaces it completely.

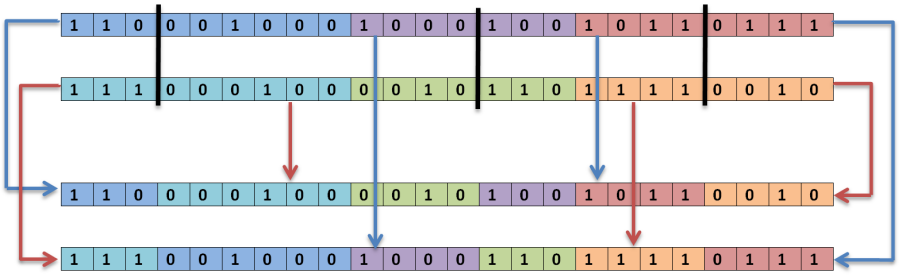


FIGURE 3. Three-points crossover between two chromosomes encoding linear rule 35 with threshold $\tau = 120$ and $\Delta = 218$, and linear rule 452 with threshold $\tau = 27$ and $\Delta = 163$, respectively. The two offspring that are obtained encode the linear rule 388 with threshold $\tau = 24$ and $\Delta = 220$, and linear rule 456 with threshold $\tau = 123$ and $\Delta = 165$, respectively.

4.3. **Framework.** The input of the proposed approach is a greyscale image on which the evolving process is performed.

Firstly, the initial configuration of the CA, called the input state, is extracted from the input image. The state of each CA's cell is given by the pixel value, while the neighbours of each CA's cell is a list of its eight surrounding pixels.

Further, the steps of the GA that optimizes the rules to detect edge points are described. The first step is generating the initial population: $PopSize$ chromosomes, each encoding the binary representation of a linear rule, the gray code of a threshold and the gray code of Δ, are generated. Each chromosome is evaluated by the means of fitness function, then selection, crossover and mutation are applied producing a new population to which the best chromosome is added. These steps are repeated for G generations. At the end of the execution of the GA, one best chromosome is obtained.

5. Experiments and comparisons

In the following experiments, 22 images of sizes 321×481 (or 481×321) from Berkely dataset (BSD) (12), along with the available ground truth, and a synthetic data set containing geometric shapes, were used. Only 22 images from Berkely dataset (12) were used in these experiments, the same subset used in *Uguz et al.* (26), so that the proposed approach could be compared with the one presented in (26).

In the first part of experiments, the 22 images of the BSD subset are given as input to the GA that evolves the CA's rule and several runs are performed for each image. The GA runs for 100 generations, having a population of 50 chromosomes.

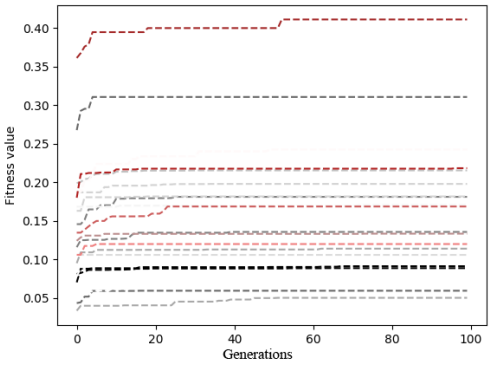


Figure 4. Best fitness values for the first 100 generations. Each line corresponds to a different image from the BSD subset.

A first analyse is aimed towards the convergence of the GA that evolves the CA's rule to detect edge points. The scope of this analysis is to emphasize that the GA proposed for this task keeps improving during the evolving process and reaches a maximum stable state. In the graph displayed in Figure 4, the best

values of the fitness were extracted for the first 100 generations of the GA run on each of the 22 images of the BSD subset.

According to the graph, the GA used in the optimization process of the CA's rule, converges due to the elitism that keeps at a high level the quality of the chromosomes. To ensure the diversity of the population, the mutation probability was set to 0.7 and the crossover operator was applied to each parameter encoded of the chromosome (see Figure 3).

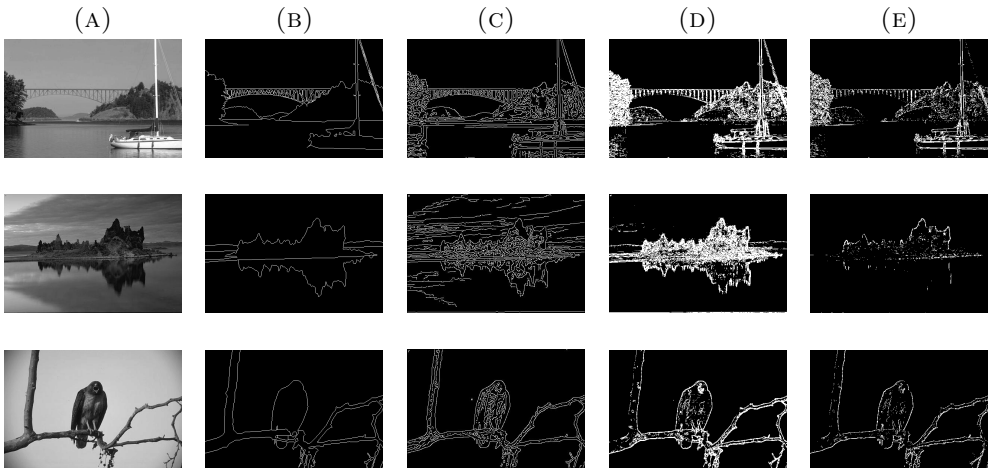


FIGURE 5. Images obtained by individual parameter optimization: (A) Input image. (B) Ground truth. (C) Canny (1) result. (D) *Uguz et al.* (26) result. (E) Proposed approach result.

A second analysis is performed on the performance of the proposed approach compared with the methods proposed in the current literature. The BDM values computed for the images obtained by applying the best parameters obtained by the proposed method, for the images obtained by applying the Canny edge detector (1) and for the images obtained by *Uguz et al.* (26) are compared. Best parameters found by this method along with the best parameters proposed by *Uguz et al.* (26) and BDM values obtained for the three methods are displayed in Table 1. The differences in the obtained values are explained by the differences in the proposed method compared to *Uguz et al.* (26) and *Canny* (1). Firstly, in the experiment conducted on the proposed approach, the Δ and τ parameters are optimized as well as the linear rule while *Uguz et al.* (26) only optimizes the first two. Secondly, the fitness

function used in this experiment is the Dice Similarity Coefficient (DSC). On average, the proposed method performs better than *Uguz et al.* (26) and *Canny* (1), however measuring the results on a larger data set would provide more significant insight. Three examples of the obtained images can be seen in Figure 5.

TABLE 1. All parameters values for *Uguz et al.* (26) and the ones found as best for the proposed approach, and BDM values for our proposed approach versus Canny (1) and *Uguz et al.* (26) on the 22 selected images in (26), with p-values computed based on Wilcoxon Signed-Ranks Test.

	Canny	Uguz et al.				Proposed approach			
Img.	BDM	Δ	τ	rule	BDM	Δ	τ	rule	BDM
Im 1	101.7416	110	0.62	47	74.9449	192	0.4687	436	74.5924
Im 2	36.2963	39	0.472	35	33.5381	181	0.0625	258	35.0743
Im 3	16.1734	59	0.414	139	5.1064	172	0.4140	156	4.4463
Im 4	41.01326	68	0.6078	154	24.2914	218	0.2578	132	31.9772
Im 5	28.7809	136	0.3889	60	41.8038	85	0.3437	285	27.911
Im 6	25.467	83	0.6642	194	72.0194	190	0.3515	150	19.7797
Im 7	15.956	104	0.194	50	14.6908	159	0.164	4	14.0289
Im 8	40.6465	107	0.4209	265	37.5357	113	0.414	4	35.4066
Im 9	26.6485	98	0.6854	309	20.9226	99	0.4218	134	19.0317
Im 10	19.8602	54	0.203	140	20.2130	174	0.3593	200	23.3961
Im 11	100.6888	87	0.45	66	86.2903	199	0.4296	468	81.0943
Im 12	29.0242	92	0.628	87	29.2202	110	0.4687	18	28.6511
Im 13	43.9345	29	0.3673	19	31.1133	251	0.3437	180	16.1749
Im 14	22.4603	76	0.3506	89	22.0495	218	0.125	204	21.4574
Im 15	13.892	103	0.5097	434	7.2010	106	0.3281	394	7.9008
Im 16	66.2486	70	0.55	75	63.2492	124	0.3984	286	60.9771
Im 17	78.2544	103	0.3818	104	3.5905	154	0.2656	30	13.6351
Im 18	21.0141	62	0.6186	46	17.8882	78	0.5546	157	16.7753
Im 19	42.8603	47	0.336	136	54.2173	79	0.164	6	37.0839
Im 20	22.1191	64	0.6141	195	19.2414	154	0.2656	30	21.4092
Im 21	33.3709	112	0.65	252	25.2466	80	0.375	6	25.9750
Im 22	24.0112	76	0.54	412	21.7873	187	0.2421	8	21.9363
Avg.	38.6574				33.0073				29.0325
p-value	0.1886				0.2234				

In this second part of the experiments, the focus is moved to evaluating the ability to generalize the behaviour of the proposed approach on multiple

images. An input dataset is selected to learn the parameters and a test dataset to evaluate the previously optimized parameters.

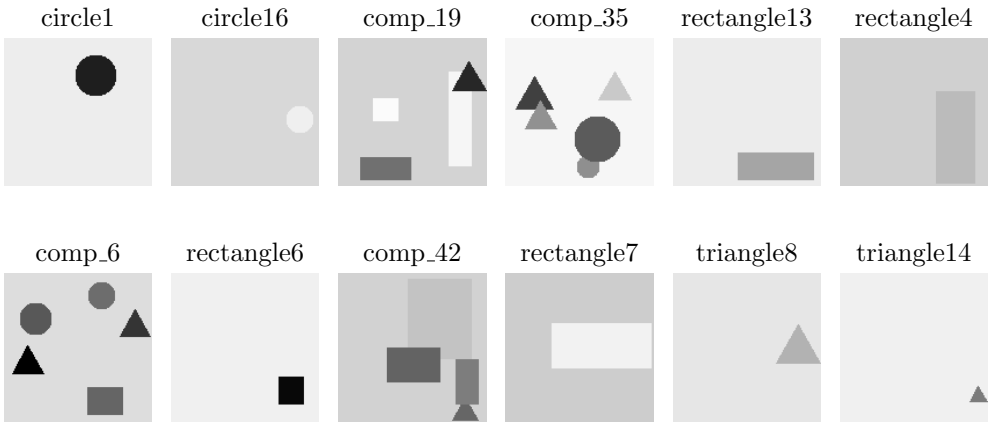


FIGURE 6. Subset of the synthetic training set.

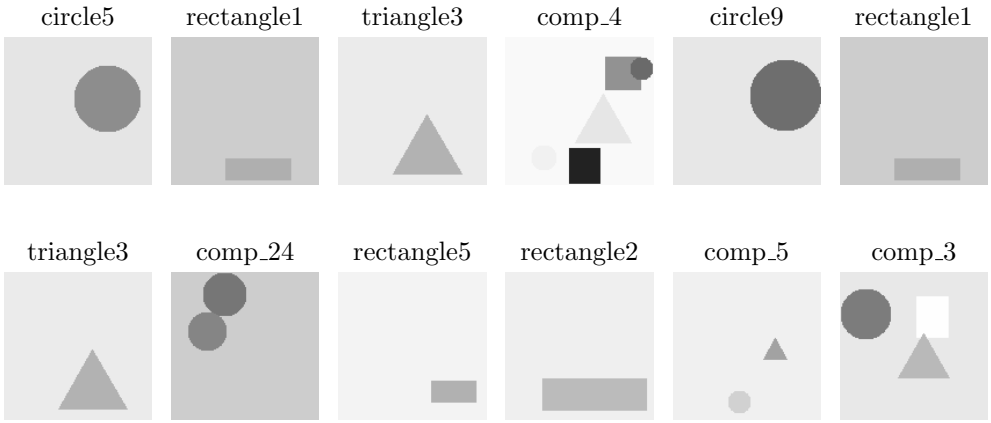
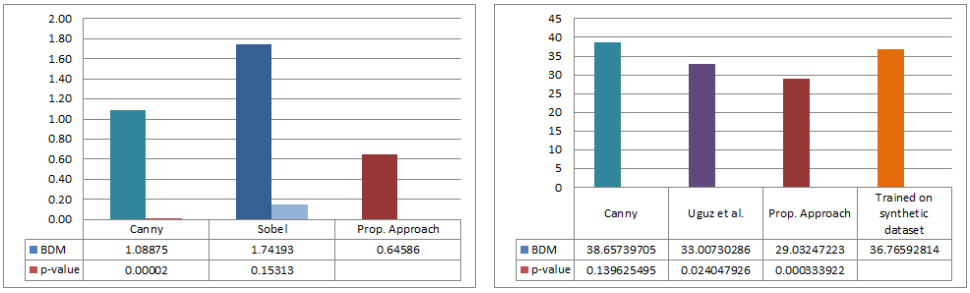


FIGURE 7. Subset of the synthetic test set.

For this experiment, a dataset of images containing different geometric shapes was generated. These images were then split into an input set (Figure 6) and a test set (Figure 7). The model was evolved using a curriculum learning strategy (6), meaning that the algorithm is fed new input examples in an

ascending order of difficulty. In this case, the difficulty was measured by the magnitude of the gradient between the shapes and the background. It was reported in the literature that feeding simpler examples first and then gradually increasing difficulty helps a learner generalize better (6; 10), which fits the aim to achieve transferability of the optimized parameters. For this reason, in the optimization process there were fed examples to the optimizer starting with high gradient images (edges easier to detect) and gradually progressing towards low gradient images (edges harder to detect).



(A) (B)

FIGURE 8. The proposed approach is compared with the other methods used in comparison in terms of a Wilcoxon Signed-Ranks Test. (A) BDM results on images used for the testing process. (B) BDM values for results obtained by applying the edge detector optimised on synthetic images versus Canny (1), Uguz et al.(26) and results obtained by individually training with the proposed approach on the 22 selected images in (26).

Figure 8a shows for the three compared methods, the average BDM values computed on the test set along with the p-values computed with the Wilcoxon Signed-Ranks Test. As may be seen, overall the proposed approach performs better than both Canny edge detector (1) and Sobel operator (17). Several examples of the obtained images can be seen in Figure 9. The main difference between the three methods may be seen in the last row, on images that contain more than one shape, where the proposed approach detects all edges, while the Canny edge detector (1) struggles to find the circle and Sobel operator (17) struggles to find both the circle and the triangle.

In order to assess the generalization capabilities of the model the parameters obtained on the synthetic dataset were also tested on the BSD subset used in *Uguz et al.* (26). One configuration used to evolve the parameters for the synthetic dataset had the best results on the BSD subset. For this

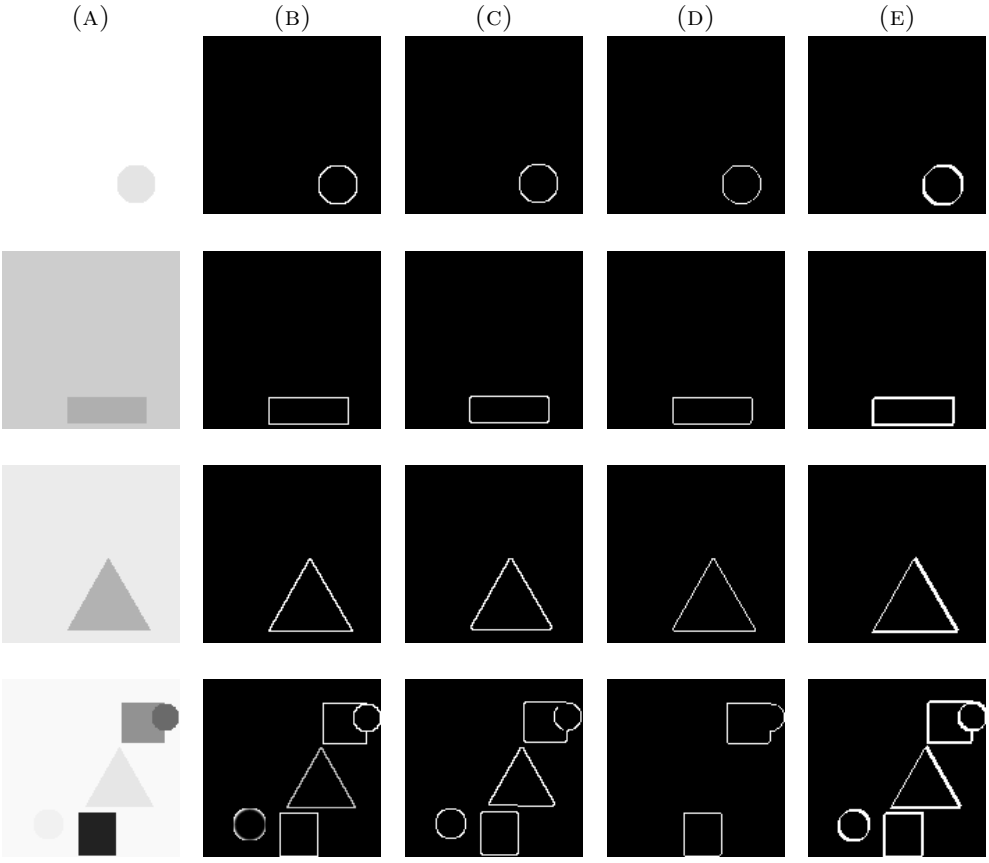


FIGURE 9. Batch optimization results on the synthetic test set. (A) Input image. (B) Ground truth. (C) Canny edge detector (1) result. (D) Sobel operator (17) result. (E) Proposed approach result.

configuration, 100 individuals were used. The graph from Figure 8b shows the average of the BDM values obtained for images in the BSD subset. It can be seen that, on average, the method proposed in *Uguz et al.* (26) and the proposed approach for individually evolving performs better than our model by a small margin, but our model performs better than Canny detector (1). However, the results from *Uguz et al.* (26) and of the proposed approach are obtained by supervised optimization on each individual image, whereas in the proposed model the rule is generalized for multiple images. This means that

the generalization capacity of the proposed model using a curriculum learning setting is comparable to individual optimization of the parameters on the test image. Sets of example images are presented in Figure 10. The main difference between these four methods may be seen in the first two rows, where both *Uguz et al.* (26) and proposed method on individual evolving detects the least edge points that makes them the most similar to the ground truth, while the proposed approach trained to generalize detects too many edge points in terms of BDM.

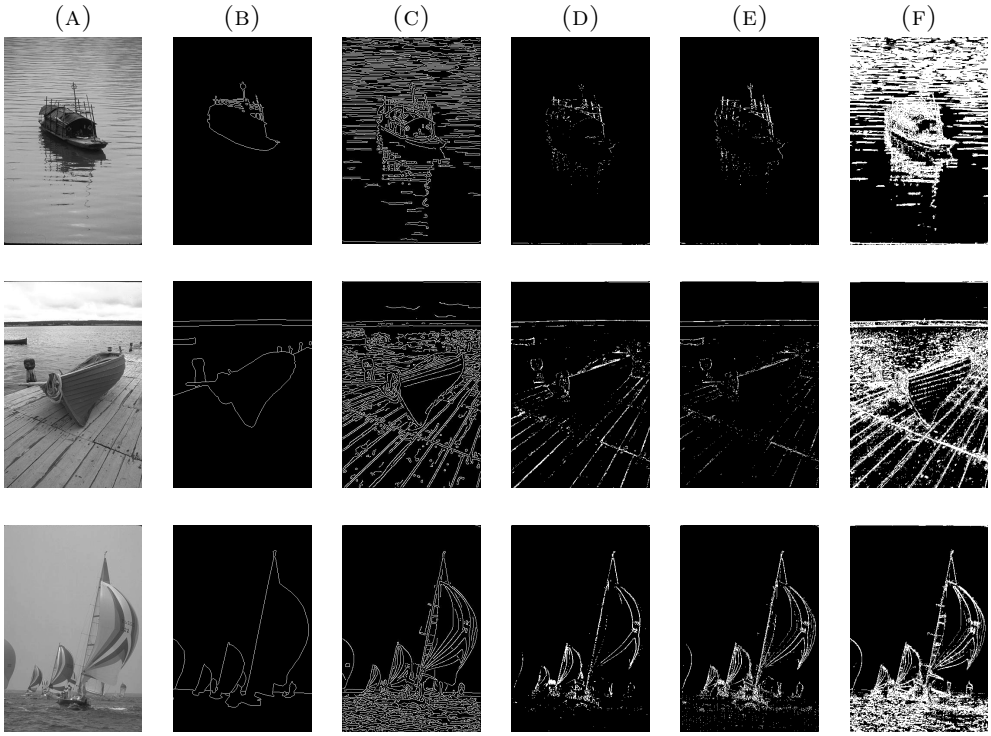


FIGURE 10. Images obtained by individual parameter optimization: (A) Input image. (B) Ground truth. (C) Canny (1) result. (D) *Uguz et al.* (26) result. (E) Proposed approach result. (F) Proposed approach results of training on the synthtic dataset.

6. Conclusions

In this paper it was presented an edge detector designed for greyscale images, based on evolved CA. The proposed method used one GA to optimize the three parameters of the CA's rule: the parameter Δ, the threshold τ and the linear rule r.

The proposed approach distinguishes from the methods presented in the literature by optimizing all the parameters by encoding all of them in a single chromosome, using both binary and grey encoding. The results obtained on both individual and batch optimization experiments indicate that the algorithm is robust and competitive with the best related method, such as Canny edge detector (1) and *Uguz et al.* (26). Moreover, the generalization capacity using a curriculum learning setting reached competitive performance related to individual optimization, which supports the idea of using transferable parameters.

As future work, more improvements can be done to both individual and batch optimization, such as using larger and more complex datasets.

References

[1] Canny, J. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, 6 (1986), 679–698.

[2] Caruana, R. A., and Schaffer, J. D. Representation and hidden bias: Gray vs. binary coding for genetic algorithms. In *Machine Learning Proceedings 1988*. Elsevier, 1988, pp. 153–161.

[3] Christiyana, C. C., Rajamani, V., and Devi, A. U. Ultra sound kidney image retrieval using time efficient one dimensional glcm texture feature. *IJCA Special Issue on Advanced Computing and Communication Technologies for HPC Applications*, 4 (2012), 12–17.

[4] Dice, L. R. Measures of the amount of ecologic association between species. *Ecology 26*, 3 (1945), 297–302.

[5] Diwakar, M., Patel, P. K., and Gupta, K. Cellular automata based edge-detection for brain tumor. In *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (2013), IEEE, pp. 53–59.

[6] Elman, J. L. Learning and development in neural networks: The importance of starting small. *Cognition 48*, 1 (1993), 71–99.

[7] Enescu, A., Andreica, A., and Diosan, L. Evolved cellular automata for edge detection in grayscale images. In *2019 21st International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)* (2019), IEEE, pp. 326–332.

[8] Farbod, M., Akbarizadeh, G., Kosarian, A., and Rangzan, K. Optimized fuzzy cellular automata for synthetic aperture radar image edge detection. *Journal of Electronic Imaging 27*, 1 (2018), 013030.

[9] Frank, G. Pulse code communication, Mar. 17 1953. US Patent 2,632,058.

[10] Hacohen, G., and Weinshall, D. On the power of curriculum learning in training deep networks. *arXiv preprint arXiv:1904.03626* (2019).

[11] Jain, R., Kasturi, R., and Schunck, B. G. Edge detection. In *Machine vision* (1995), vol. 5, McGraw-Hill New York, pp. 140–185.

[12] Martin, D., Fowlkes, C., Tal, D., and Malik, J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on* (2001), vol. 2, IEEE, pp. 416–423.

[13] Mofrad, M. H., Sadeghi, S., Rezvanian, A., and Meybodi, M. R. Cellular edge detection: Combining cellular automata and cellular learning automata. *AEU-International Journal of Electronics and Communications 69*, 9 (2015), 1282–1290.

[14] Mohammed, J., and Nayak, D. R. An efficient edge detection technique by two dimensional rectangular cellular automata. *CoRR abs/1312.6370* (2013).

[15] Nayak, D. R., Patra, P. K., and Mahapatra, A. A survey on two dimensional cellular automata and its application in image processing. *IJCA Proceedings on International Conference on Emergent Trends in Computing and Communication (ETCC-2014) ETCC*, 1 (2014), 78–87.

[16] Neumann, J., Burks, A. W., et al. *Theory of self-reproducing automata*, vol. 1102024. University of Illinois press Urbana, 1966.

[17] Poobathy, D., and Chezian, R. M. Edge detection operators: Peak signal to noise ratio based comparison. *International Journal of Image, Graphics and signal processing 6*, 10 (2014), 55.

[18] Qadir, F., Peer, M. A., and Khan, K. A. Efficient edge detection methods for diagnosis of lung cancer based on two dimensional cellular automata. *Advances in Applied Science Research 4*, 3 (2012), 2050–2058.

[19] Rue, H. Baddeley's delta metric. In *Banks (eds), Encyclopedia of Statistical Sciences, Vol. Update Volume* (1996), Citeseer.

[20] Schaffer, J. A study of control parameters affecting online performance of genetic algorithms for function optimization. *San Meteo, California* (1989).

[21] Sinaie, S., Ghanizadeh, A., Majd, E. M., and Shamsuddin, S. M. A hybrid edge detection method based on fuzzy set theory and cellular

learning automata. In *2009 International Conference on Computational Science and Its Applications* (2009), IEEE, pp. 208–214.

[22] Sobel, I., and Feldman, G. A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in* (1968), 271–272.

[23] Sørensen, T. *A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species Content and Its Application to Analyses of the Vegetation on Danish Commons.* Biologiske skrifter. I kommission hos E. Munksgaard, 1948.

[24] Tizhoosh, H. R. Fast fuzzy edge detection. In *2002 Annual Meeting of the North American Fuzzy Information Processing Society Proceedings. NAFIPS-FLINT 2002 (Cat. No. 02TH8622)* (2002), IEEE, pp. 239–242.

[25] Tizhoosh, H. R., Haussecker, H., Jähne, B., Haussecker, H., and Geissler, P. Fuzzy image processing: an overview. *Handbook on computer vision and applications, Academic Press, Boston* (1998).

[26] Uguz, S., Sahin, U., and Sahin, F. Edge detection with fuzzy cellular automata transition function optimized by pso. *Computers & Electrical Engineering 43* (2015), 180–192.

Babeş-Bolyai University, Department of Computer Science, 1 M. Kogălniceanu Street, 400084 Cluj-Napoca, Romania

Email address: aenescu@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXV**, Number 1, 2020
DOI: 10.24193/subbi.2020.1.07

EVOLUTION OF SOFTWARE QUALITY MODELS FROM AN NLP PERSPECTIVE

D. LUPŞA, S. MOTOGNA, AND I. CIUCIU

Abstract. The research question addressed in this study is to analyze the evolution of software quality models, and to investigate the similarities between the most known quality models. The models are decomposed according to their inherent structure, then are analyzed based on their subcharacteristics, at several levels. The focus of the study is on the lexical analysis of software quality models based on natural language processing, with the purpose to highlight the similarities in software quality models. In the end, the current software quality model ISO25010 is compared with all previous ones.

1. Introduction

As software systems become every day, more and more complex and spread in different application areas, the interest for software quality (SQ) has increased significantly.

From a business perspective, quality aspects of software applications, such as reliability, usability, interoperability can bring significant advantages over the competition. From an industrial perspective, different software quality characteristics, such as security, safety, maintainability, are essential in some types of applications: real-time, safety-critical, respectively health support systems. Even more, there are standardized software quality models for software products (ISO25010[9]) and processes. From the research perspective, the last decade has seen an increased interest of the research community for software quality aspects, with different conferences and journals focusing on SQ in general, but also on specific SQ characteristics such as: reliability, security, safety a.s.o.

A software quality model represents a set of characteristics that completely characterize a software system. They are classified as external factors (such

Received by the editors: 1 June 2020.

2010 *Mathematics Subject Classification.* 68N99.

1998 *CR Categories and Descriptors.* D.2.0 [**Software Engineering**]: General – *Standards*; D.2.9 [**Software Engineering**]: Management – *Software Quality Assurance*.

Key words and phrases. Software Quality, quality factors, word similarity.

as usability, reliability, efficiency a.s.o.) and internal (including maintainability, testability, reusability a.s.o.). The characteristics are then divided into subcharacteristics, with the intent to cover all aspects of software products.

Software development evolution has a direct impact on software models, with several SQ models proposed, and lately, standardized.

The aim of this article is to study the evolution of software quality models using an NLP approach. Our aim is to be able to explain the similarities and variations between existing SQ models by looking at both their syntactical and semantical structure. More specifically, our approach decomposes a model according to its inherent structure (i.e. characteristics and subcharacteristics) and tries to interpret (analyze) it at lexical level, by applying various NLP metrics. This study is a continuation of our previous work [15], in terms of the quantitative analysis that is employed and also regarding the software quality models under investigation.

The rest of the paper is structured as follows: we begin with a short presentation of the SQ models under review, followed by the Natural Language Processing (NLP) measurements used in our investigation. The next section discusses our finding related to models evolution, and presents the similarity study performed for two models: McCall and ISO 25010. We end with some concluding remarks and future ideas.

2. SQ Models

The McCall model[12] introduced in 1977 is consider the first software quality model, defining 11 factors characterizing software products, that are further decomposed in criteria, as an initiative from aeronautical industry. Several notable models followed, proposing new factors or modifying the ones from McCall model. A synthesis of the most important SQ models, including their component factors, is presented in Table 1.

Our study will include the following SQ models: McCall, Boehm, ISO 9126 and ISO 25010. The decision to exclude FURPS model is based on the fact that it was initially developed as an "in–the–house" model at Hewlett Packard, and although it is widely used in industry the definition of characteristics and their evaluation varies from company to company.

Looking at the factors, we can notice that some of them, like maintainability and reliability, are quite "stable", namely have been included in all proposed quality models, while other factors appeared only in a subset of these models. This can be explained by the dynamics of software domain, which changes very fast. For example security appeared in the last SQ model, due to the spread of web and mobile applications. A deeper look also identifies similarities or differences between subcharacteristics.

TABLE 1. Software quality models [15]

SQ Model	Year	Factors
McCall [12]	1977	Correctness, Reliability, Efficiency, Integrity, Usability, Maintainability,Testability, Flexibility, Portability, Reusab., Interoperability
Boehm [2]	1977	Portability, As in Utility - containing Reliability, Efficiency, Human Engineering, Maintainability - including Testability, Understandab., Flexibility
FURPS [8]	1987	Functionality(F), Usability (U), Reliability (R), Performance (P) and Support (S)
ISO 9126 [1]	2001	Functionality, Reliability, Usability, Efficiency, Maintainability, Portability
ISO 25010 [9]	2011	Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability, Portability

Analyzing the SQ models, we can conclude the following:

- The domain of software quality uses several concepts and terms with different definitions and meaning;
- The domain of software quality is dynamic: software development rapid growth directly influences SQ domain.

2.1. **State of the art.** Regarding the evolution of SQ models, a study taking into account the hierarchy of characteristics-subcharacteristics, relations of subordination between them and special metrics is performed in [6] in the context of ISO 25010, where the need of a more detailed semantic analysis for the disambiguation of quality model components is recognized. The same study was applied in the context of "green software" in [7], and managed to predict the complexity for the next software quality model and to propose variants of green software attributes to be included in the model.

It is agreed within the software quality community that in order to apply semantic analysis on SQ models, it is necessary to have a commonly agreed-upon model, i.e. an ontology. However, such model is currently not completely available. Several contributions address the topic of constructing a common set of concepts that are incorporated in quality models. A study [11] that employs extraction of a condensed model for software product quality attributes is using the frequency and association of terminology and concepts that appear in different sources such as models, standards and related documents to software quality. Based on this, an ontology involving semantics of quality attributes is built. This ontology was the starting point for further studies targeting a common understanding and agreed models to be used by software

engineers, researchers, stakeholders and practitioners. A three-dimensional ontology model was proposed in [3], SQuAP-Ont (https://w3id.org/squap/), based on current standards in place for software quality, processes and architecture. For software quality, SQuAP is using the ISO25010 standard, and can be easily extendable. Kara et al. [10] proposed a generic software quality model based on existing software standards that can be instantiated using an algorithm based on fuzzy logic.

The novelty of our study is given by the fact that the analysis is considering all significant SQ models, in an attempt to identify similarities and differences from a NLP perspective, which will help unify the terminology used in this domain.

3. Introducing NLP measures

As discussed in [15], several NLP approaches can be considered for software quality domain, computing similarities between characteristics and subcharacteristics. These computations can be classified in:

- set–based similarities [5]: they are using mostly the number of common words. No semantic information is considered. The intuition behind them is that the more common words two texts share, the more similar they are;
- lexical similarities: they can be computed based on existing ontologies (such as WordNet [4]) or can be computed based on the word co-occurrences in texts, as in the case of word embedding techniques.

The experiments from [15] showed that set-based measures, based on keywords, give the most accurate results for set-based similarity. That is the argument why the rest of the study will consider characteristics and subcharacteristics as keywords and use only them to compute similarity between SQ models.

3.1. **Set-based measures.**

Overlap, Jaccard and Dice are set-based similarity measures that can be used.

Overlap coefficient is computed based on the number of common terms and uses a normalization factor, but it considers two sets a full match, having maximum similarity value, which is 1, if one is a subset of the other.

Jaccard similarity is computed as the number of shared (common) keywords over the number of all unique keywords (union) in two sets, and can be defined in terms of what is the same (common) and what is modified (added or removed) when having a SQ model SQM1 and creating a second model SQM2.

Dice′s coefficient is defined as twice the number of common terms in the compared sets divided by the total number of terms in both sets, which is also:

$$Dice(SQM1, SQM2) = \frac{2 \times common(SQM1,SQM2)}{2 \times common(SQM1,SQM2) + added(SQM1,SQM2) + removed(SQM1,SQM2)} \quad (1)$$

Both Dice and Jaccard depend on the number of items that are modified (added or removed) and they decrease when this number increases, but Dice has higher values than Jaccard. It is also proven that Jaccard and Dice are monotonic in one another, meaning that it not exists 3 sets A, B and C such that: $Jaccard(A; B) > Jaccard(A; C)$ and $Dice(A; B) < Dice(A; C)$.

Dice and Jaccard give a similar kind of information and we decided to use one of them, namely Dice.

3.2. **Lexical information.** One of the things we are aiming at is to identify renaming of the same concept in SQ models, for example, to identify words like *maintainability* and *changeability* as being synonyms (that is lexically similar).

There are a lot of measures to determine the similarity between words. Some of them use existing ontologies (such as WordNet) to compute the semantic distance between words. Others are based on the word co-occurrences in texts, such as word embedding techniques.

Word embedding techniques were introduced around 2013 by a team led by Mikolov that proposed the word2vec algorithm [13], [14]. It was followed by the apparition of other unsupervised learning algorithms for obtaining vector representations for words, such as GloVe [16]. These methods obtain good results, but they rely on having a very large corpus. But *How large the corpus should be?* in order to cover the semantic information that is present in human-build ontologies (such as WordNet). For example, we would expect to get a high similarity value between words like *change* and *modify*, but let us look to the data. By using one of Standford GloVe word embeddings vectors (Wikipedia 2014 + Gigaword 5) the similarity between *change* and *modify* is 0.461, while the most 5 similar words with modify and their similarity are: (*modifying*, 0.791), (*amend*, 0.751), (*revise*, 0.728), (*alter*, 0.717), (*rework*, 0.672) and the most 5 similar words with change are: (*changes*, 0.904), (*changing*, 0.809), (*this*, 0.787), (*shift*, 0.787), (*move*, 0.785).

The amount of data corresponding to this study is limited, so we decided that such an approach may lead to false results, and as a consequence we decied to use WordNet.

Since we aim at identifying the renaming of the same concept in SQ models and we need information about synonymy between words, and we choose to use WordNet. WordNet has the advantage that is human build ontology (and

it means that we can trust the data), it covers our needs and it is ready-to-use. If two words appear in the same WordNet synset, they are considered synonyms and describe the same concept.

We cannot identify all the synonymy relations that we need directly by searching WordNet, because there are a lot of missing words. But we expect that most of the missing words in WordNet to be derivations of others, which we are going to call base word. For example, we can't find *modifiability* in WordNet, but we can find its base word: *modify* and also can find that *change* and *modify* are synonyms. With this approach, we were able to find in WordNet all the (base) words that are important for us.

So, in this study, in order to find and use lexical information, we first took all the base words for all the names of characteristics and subcharacteristics and then verify if they are synonyms or not.

Another problem we had to face is the use of multi-word expression for characteristics and subcharacteristics names. As they could be considered as containing only key-names for a concept, our choice was to consider as being lexically similar, expressions that have at least one (base) word in common or, at least, lexically similar.

By associating all these key-names that are similar from a lexical perspective to a given key-name, many times there will be more than one association to a name. Consider that we have a number na of names that are in set A, and for which there is at least a similar name in a set B. Denote with nb all the words from B that are lexical associations to A. We have to remark that, many times, $nb \neq na$.

When we compare a model SQM1 to SQM2, we consider the number of key-expressions from the SQM1 for which we find a lexically similar key-expression in SQM2. For example, in Table 7, column Boehm vs. 9126, second half of the table, named: use lexical information, we found lexical association for 5 characteristics from Boehm into 9126 and we didn't find any for 4 characteristics of Boehm. So, there are 5 lexical matches and 4 removed ones. On the line *lexical match* there is also a number put in paranthesis: (vs. 6). That number 6 means that for all the 6 characteristics from 9126 there are lexical associations in Boehm and that is nothing left in 9126 that has no association in Boehm. This also means that there is no new characteristic added to Boehm.

We cannot directly use Dice as a metric. We are modifying it as follows: instead of counting the common names, we are going to number all the names from the two sets for which there are lexically similar names in the other set. For example, Table 7, Boehm vs. 916, modified Dice is $(5+6)/(5+6+0+4)$.

4. Evaluation data. Comparative analysis of SQ models

We considered an investigation of transformations between the models. A first aspect of the analysis takes into consideration questions like: how many properties remain the same, how many are added and how many are removed. Another question to be answered is: how do we define the properties of a model? In this approach we are going to look at the characteristics and subcharacteristics names and relations among names from a lexical perspective.

A second aspect of the analysis is the perspective from which we look at the data. We consider the SQ models: McCall, Boehm, ISO 9126, ISO 25010 taken in historical order, but also analyze how close models are relative to the latest model among them, which is ISO 25010. The data has been collected using naming of characteristics and subcharacteristics.

Table 2. Naming of the characteristics and subcharacteristics. Evolution

		McCall to Boehm	Boehm to 9126	9126 to 25010
names	common	11	7	18
	added	12	20	21
	removed	23	16	9
	Dice	0.39	0.28	0.55
use lexical information	lexical match	20	9	23
	added	7	15	11
	removed	14	14	4
	modified Dice	0.63	0.42	0.77

4.1. **Quantitative analysis based on all names of characteristics and subcharacteristics.** We can see in Table 2 that, from McCall to Boehm, 35 naming were changed (added or removed), which are a little less than from Boehm to ISO 9126 (36), and only 30 are changed between ISO 9126 and ISO 25010, which seemed to be the most similar software quality models among those considered here. This behaviour of the evolution is even more pregnant if we look at the values obtained by using lexical information.

When compared with ISO 25010, the biggest differences are found for Boehm model. We can also notice that ISO25010 is the model that preserved the most of characteristics (Tables 2 and 3) during all models history.

If we look to the common names (words) chosen for characteristics and subcharacteristics in all four models, we see that there are only four names that appear in all of them, and these are: maintainability, portability, reliability, testability, and only two of them remained as characteristics in all models.

TABLE 3. Naming of characteristics and subcharacteristics compared to ISO 25010

		McCall vs. 25010	Boehm vs. 25010	9126 vs. 25010
names	common	10	7	18
	added	29	32	21
	removed	24	16	9
	Dice	0.27	0.23	0.55
use lexical information	lexical match	19	11	23
	added	18	21	11
	removed	15	12	4
	modified Dice	0.55	0.47	0.77

TABLE 4. Common characteristics and subcharacteristics in SQ Models

	McCall	Boehm	9126	25010
maintainability	factor	superfactor	factor	characteristic
portability	factor	factor	factor	characteristic
reliability	factor	factor	factor	characteristic
testability	factor	factor	subfactor	subcharacteristic

4.2. **Quantitative analysis based on all names of subcharacteristics.** In this section we consider only subcharacteristics names, as a characteristics can be considered satisfied when all its subcharacteristics are satisfied. An interesting remark: no subcharacteristics appears in all models as subcharacteristics. Only Testability, which is a subcharacteristic of Maintainability in both ISO 9126 and in ISO 25010, appears in McCall and in Boehm models, but as a subcharacteristics.

More than that, only two subcharacteristics appear (by the same name) in three of the four models: accuracy (in McCall, Boehm, ISO 9126) and operability (McCall, ISO 9126, ISO 25010). An interesting fact is that 43 subcharacteristics names (from 65) are used only in one model, and 20 are used in two.

4.3. **Quantitative analysis based on all names of characteristics.** The list of characteristics of a SQ model can give an overview of the aspects on which that model focus on. A comparison of what is added, removed or kept along time and how it differs from ISO 25010, in the same style as before, is presented in what follows.

This analysis shows that if we look only at the level of characteristics the SQ models may seem more similar than they really are. As presented in Table 7,

TABLE 5. Naming of the subcharacteristics. Evolution

		McCall to Boehm	Boehm to 9126	9126 to 25010
names	common	6	1	13
	added	8	20	18
	removed	17	13	8
	Dice	0.32	0.06	0.50
use lexical information	lexical match	20	9	18
	added	7	15	10
	removed	14	14	9
	modified Dice	0.63	0.42	0.67

TABLE 6. Naming of the subcharacteristics versus ISO 25010

		McCall vs. 25010	Boehm vs. 25010	9126 vs. 25010
names	common	2	2	13
	added	29	29	18
	removed	21	12	8
	Dice	0.07	0.09	0.50
use lexical information	lexical match	14	8	18
	added	16	19	10
	removed	20	15	9
	modified Dice	0.45	0.37	0.67

TABLE 7. Naming of the characteristics. Evolution

		McCall to Boehm	Boehm to 9126	9126 to 25010
names	common	5	4	4
	added	4	2	4
	removed	6	5	2
	Dice	0.50	0.53	0.57
use lexical information	lexical match	6 (vs. 6)	5 (vs. 6)	6 (vs. 6)
	added	3	0	2
	removed	5	4	0
	modified Dice	0.6	0.73	0.86

the Dice similarity values are very close to each other and, in case of comparing McCall with Boehm, respectively Boehm with ISO 9126 models, much greater than the values from the previous comparisons.

The same increase in similarity for characteristics can be observed when comparing previous SQ models with ISO 25010, as shown in Table 8.

TABLE 8. Naming of the characteristics versus ISO 25010

		McCall vs. 25010	Boehm vs. 25010	9126 vs. 25010
names	common	4	3	4
	added	4	5	4
	removed	7	6	2
	Dice	0.42	0.35	0.57
use lexical information	lexical match	5 (vs. 6)	5 (vs. 6)	6 (vs.6)
	added	2	2	2
	removed	6	4	0
	modified Dice	0.58	0.65	0.86

5. Comparative analysis of ISO25010

A conclusion of the evaluations from the last section, when comparing models with ISO 25010, can be summarized in:

The most similar model is ISO 9126, the direct predecessor in terms of evolution, as shown in Tables 3, 6, 8. This is something to be expected, but however if we make the comparison with the modifications between the other consecutive models, we notice that in almost all the cases (the exception is the total number of characteristics and subcharacteristics added, as shown in Table 2) the modifications are fewer and the common elements are the most. Thus, we can conclude that we tend to see a stability in the evolution of SQ models.

Another interesting perspective is to analyze the influence of characteristics from all previous models into the ISO 25010 model. Based on characteristic naming, we put in correspondence the factors of the models. More precisely: considering a given SQ model with characteristics and their subcharacteristics, we associate characteristics of a second model to the characteristics of the first model. We do this on two steps: first consider the naming of the characteristics, then a set-based similarity is computed over the set of all keywords of the characteristics (that is the set consisting of characteristic name and subcharacteristics) . The higher similarity is considered in order to associate the characteristics of the two models. The evaluation presented here uses Dice similarity measure. The final score for a characteristic is the sum of all the association score. This evaluation has highlighted the following interesting remarks: the only SQ model that contains similarities for all 8 characteristics of ISO 25010 is the McCall model. Also the highest similarity score is obtained for Maintainability characteristics from this model (significantly higher than the rest). This is explained due to the fact that similarities has been found not only performing lexical comparison between characteristics, but also computing similarities between subcharacteristics of maintainability with other

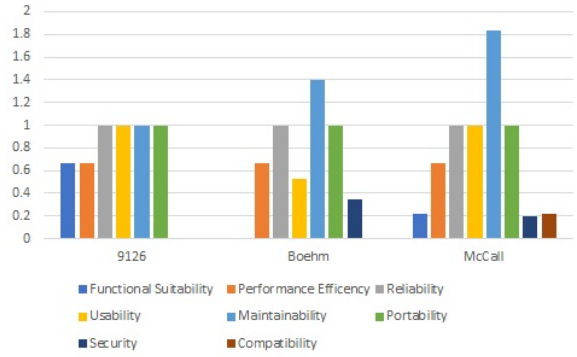


FIGURE 1. Analysis of characteristics in ISO 25010 based on similarity score

subcharacteristics of ISO 25010. The results prove the accuracy of the McCall model and also its completeness, even if it was proposed over 40 years ago and considering the dynamics of evolution of software systems.

6. Conclusions and future work

The terminology and concepts corresponding to software quality models and standards can be characterized by a significant degree of variation. The software quality community cannot refer to a unique set of concepts and associated meanings when assessing the software quality domain or when introducing a new software quality model. As predicted by [7], since in the near future we expect a new SQ model to be proposed and standardized, an evaluation of the evolution of these models so far is an important assessment of the domain. This paper has used NLP techniques in order to determine similarity measures, which are then used to compare the evolution of consecutive SQ models and to compare the current standard, ISO 25010, with the previous ones.

The conclusions of our study can be summarized in: there are still significant modifications between models (added and removed terms), so a special care should be taken in order to avoid overloading the domain lexical set: when adding a characteristics, a survey of the previous models should be carried in order to identify if it is significantly different from the previous terms.

A second conclusion will give credits to the first SQ model introduced by McCall [12]. The similarity displayed with the current standard shows that McCall model was a very solid one, and the aspects considered in evaluating different characteristics are still valid nowadays.

As a continuation of this study, we intend to automate the process of SQ standard compliance. When a new standard will be proposed (considering that the last one, ISO 25010 was introduced in 2011, this is to be expected), software products should comply with the new standard. Instead of fulfilling all characteristics from scratch, our approach can be used to determine the set of characteristics, respectively subcharacteristics that need to be changed/added/moved or removed in order to comply the new standard.

It is also our intent to extend the SQ model evaluation to construct and ontology of SQ models, such that the evaluation can be enhanced at the graph (ontology) level using the semantic meaning of its constituents.

References

[1] 9126-1, I. Software engineering – product quality. `https://www.iso.org/standard/22749.html`, 2001. Accessed: 2015.

[2] Boehm, B. W., Brown, J. R., and Lipow, M. Quantitative evaluation of software quality. In *Proceedings of the 2nd International Conference on Software Engineering* (1976), ICSE '76, IEEE Computer Society Press, p. 592–605.

[3] Ciancarini, P., Nuzzolese, A. G., Presutti, V., and Russo, D. Squap-ont: an ontology of software quality relational factors from financial systems. *CoRR abs/1909.01602* (2019).

[4] Fellbaum, C. *WordNet – An Electronical Lexical Database*, vol. 25. 01 1998.

[5] Gomaa, W. H., and Fahmy, A. A. Article: A survey of text similarity approaches. *International Journal of Computer Applications 68*, 13 (2013), 13–18. Full text available.

[6] Gordieiev, O., Kharchenko, V., Fominykh, N., and Sklyar, V. Evolution of software quality models in context of the standard iso 25010. In *Proceedings of the Ninth International Conference DepCoS-RELCOMEX, Advances in Intelligent Systems and Computing* (2014), pp. 223–232.

[7] Gordieiev, O., Kharchenko, V., and Fusani, M. Software quality standards and models evolution: Greenness and reliability issues. In *Information and Communication Technologies in Education, Research, and Industrial Applications* (2016), pp. 38–55.

[8] Grady, R. B. *Practical Software Metrics for Project Management and Process Improvement*. Prentice-Hall, Inc., USA, 1992.

[9] ISO/IEC 25010:2011. Systems and software engineering. `http://www.iso.org`, 2011. Accessed: 2015.

[10] Kara, M., Lamouchi, O., and Ramdane-Cherif, A. Ontology software quality model for fuzzy logic evaluation approach. *Procedia Computer Science 83* (2016), 637 – 641. The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016) / The 6th International Conference on Sustainable Energy Information Technology (SEIT-2016) / Affiliated Workshops.

[11] Kayed, A., Hirzalla, N., Samhan, A. A., and Alfayoumi, M. Towards an ontology for software product quality attributes. In *Proceedings of the 2009 Fourth International Conference on Internet and Web Applications and Services* (Washington, DC, USA, 2009), ICIW '09, IEEE Computer Society, pp. 200–204.

[12] McCall, J., Richards, P., and Walters, G. Factors in software quality. *Nat Tech.Information Service 1* (1977), 0–0.

[13] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013* (2013), Y. Bengio and Y. LeCun, Eds.

[14] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*. 2013, pp. 3111–3119.

[15] Motogna, S., Lupsa, D., and Ciuciu, I. A NLP approach to software quality models evaluation. In *On the Move to Meaningful Internet Systems: OTM 2018 Workshops - Confederated International Workshops: EI2N, FBM, ICSP, and Meta4eS 2018, Valletta, Malta, October 22-26, 2018, Revised Selected Papers* (2018), C. Debruyne, H. Panetto, W. Guédria, P. Bollen, I. Ciuciu, and R. Meersman, Eds., vol. 11231 of *Lecture Notes in Computer Science*, Springer, pp. 207–217.

[16] Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *EMNLP* (2014), vol. 14, pp. 1532–1543.

Babeş-Bolyai University, Department of Computer Science, 1 M. Kogălniceanu Street, 400084 Cluj-Napoca, Romania

Email address: dana@cs.ubbcluj.ro, motogna@cs.ubbcluj.ro, oana@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXV**, Number 1, 2020
DOI: 10.24193/subbi.2020.1.08

A TABU SEARCH APPROACH FOR PERMUTATION FLOW SHOP SCHEDULING

CRISTINA ELENA DODU AND MIRCEA ANCĂU

Abstract. The adaptive distance between the neighbourhood's makespans influences the local search to explore the non-investigated areas of the solutions space. A Tabu Search with the intensive concentric exploration over non-explored areas is proposed as an alternative solution to the simplest Tabu Search with the random shifting of two jobs indexes operation for Permutation Flow Shop Problem (PFSP) with the makespan minimization criteria.

1. INTRODUCTION

The Permutation Flow Shop Problem(PFSP) is a production problem where a set of n jobs have to be processed on the same order on m machines. Every job has a running time or processing time on each machine, no machine processes more than one job at a time, the preemption of jobs is not allowed and it is considered that machines never breakdown during the scheduling process. The goal is to find the right sequence among the possible $n!$ sequence to minimize the production time - the time at which the last job is completed on machine m, called the makespan. If there are 2 machines, then the problem can be solved in $O(nlog_n)$ time by Johnson's algorithm[15]-the most classical algorithm in the scheduling area. Minimum of three active machines condition in the PFSP environment's setup causes a migration of the problem-solving approach (Garey[15]) to the "NP-complete problem" standards because the optimization algorithms with polynomial time have not yet been found. It had a positive spillover effect, a plenitude of heuristic and meta-heuristics has concentrated on offering a viable global solution. The heuristic's life-cycle is defined by the self-governing steps, independent of each

Received by the editors: 24 February 2020.

2010 *Mathematics Subject Classification.* 68T20, 68P10.

1998 *CR Categories and Descriptors.* I.2.8 [**Computing methodologies**]: Artificial Intelligence – *Problem Solving, Control Methods, and Search*; D.2.8 [**Software engineering**]: Metrics – *Complexity measures*.

Key words and phrases. Metaheuristics, PFSP, Permutation Flow Shop Problem, Scheduling Flow shop with concentric exporation, Local search, Tabu Search.

other: index's establishing, solution's building and solution's improvement. Palmer[5], Campbell, Dudek, Smith[11], Gupta[12], Dannenbring[4] obtained widely known and esteemed results for the constructive algorithms and Nawaz, Enscore and Ham[14] for the insertion heuristic algorithms known as NEH algorithm. Heuristics do not have the knowledge of self-sustaining to alter the direction of the search approach when a local optimum is seeable and meta-heuristics are designed to fix this impediment. Meta-heuristics may accept a temporary deterioration of the solution which allows them to explore more thoroughly the solution space and thus to get a hopefully better solution (that sometimes will coincide with the global optimum). From the meta-heuristics typologies, the single-point approach transforms the current solution by analyzing its neighbourhood.

In section 2 were introduced the formal definition of PFSP and the Tabu Search methodology. Section 3 explaines in detail **TSA** algorithm -the proposed methodology and the results obtained on the Taillard's benchmark sets[7] are presented in the next section. In section 5 is analyzed the **TSA**' performance on solving PPSP and finally, section 6 takes up the conclusions.

2. METHODOLOGY REVIEW

The problem is formally defined in the following: n, jobs $j_1, j_2, ..., j_n$ have to be processed on a series of m machines: $m_1, m_2, ..., m_m$ and the processing order of the jobs on the machines is the same for every machine. For each job j on each machine i the processing time that is defined before the beginning of the process. A complete list of these assumptions is detailed by Baker[2]:

- All jobs are independent and available for processing at time 0.
- All machines are continuously available.
- Each machine can process at most one job at a time and each job can be processed only on one machine at a time.
- The processing of a given job at a machine cannot be interrupted once started, i.e, no preemption is allowed.
- Setup times are sequence independent and are included in the processing times or are otherwise ignored.
- All jobs are independent and available for processing at time 0.
- An infinite in-process storage buffer is assumed. If a given job needs an unavailable machine then it joins a queue of unlimited size waiting for that machine. The objective is to find the sequence of n jobs, which achieves the minimal makespan when all jobs are processed on the m machines. The total number of feasible solutions to this problem is derived from the possible job's permutations on machines.

Let be a jobs processing permutation $J = (j_1, j_2, ..., j_n)$, where j_k denotes the job which is in position k of J. Let $p(i, j)$ be the processing time ($j = 1, ..n$ and $i = 1..m$). The completion time $c(j, i)$ is calculated by the formula:

$$c(j,i) = \max(c(j-1,i), c(j,i-1) + p(i,j)) \tag{1}$$

where $c(0, i) = 0$ and $c(j, 0) = 0$

The makespan or the completion time is the difference between the time of completion of the last job and the starting time of the first job:

$$Cmax(J) = c(m,n) \tag{2}$$

Let β denote the set of all job permutations, J. The makespan minimization criterion or $Cmax$ means finding J^*, the optimal sequence of jobs that will minimize the completion time:

$$Cmax(J^*) = \min(J, J \in \beta) \tag{3}$$

For more than two machines, PFSP is one of the well-known NP-hard combinatorial optimization problems. The mainly used metaheuristics are based on an improvement of an initial solution by research in its neighbourhood by one of the disruption procedures of the current solution.

Tabu Search(TS)as a single-point meta-heuristic emissary localizes the best candidate from the neighbourhood(NH) of a proposed solution. TS avoids to re-visit the old solutions. memorizing in the Tabu List (TL) as is described by the Tabu Search methodology (Glover[9]]). The neighbourhood's population is feeded from random exchange on the current solution's attributes.

Taillard[7] spotlighted the starting solution of TS approach selecting the improved heuristic proposed by Nawaz, Enscore and Ham[14] - known as NEH and shifted a single position to obtain a new neighbor. Nowicki and Smutnicki[6] concluded that the interrelated units of jobs legitimatizes specific insertions more proficients than other combinations. Reeves et al.[3] have been monitored the vicinity of the global optimum solution because not far distance are concentrated the local optimum solutions and spotted the effect as "big valley phenomenon" (when going along trajectory between two local optimal solutions, it is possible to find a new optimum local or even an optimum global). Ochoa and Veerapen[10] decompose "big valley phenomenon" into sub-valleys or funnels and identified a possible issue in discovering high-quality solutions if the global optimum is not positioned in the largest

valley. Drezner[18] proposed the concentric tabu search for the Quadratic Assignment Problem and suggested different rules for the scanning around the center solution.

3. PROPOSED METHODOLOGY

First, Tabu Search approach (**TSA**) analyzes the nearest solutions and after that it searches in other non-explored areas that are registered into the buffer zone. **TSA** starts with the solution provided by the algorithm NEH[14] and randomly interchanges two jobs indexes in ordet to alter the current solution (SL) and the result becomes a neighbor if its makespan is filtered by the restrictions of the adaptive distance ($ADIST$).

The adaptive distance is increased only when the empty list of candidates in the neighbourhood forces the algorithm to uses the best candidate from the buffered list. The adaptive distance together with the selection in the buffer list conduct the exploration far distance from the last current solution. The following restrictions are applied on each candidate:

$$Cmax(SL) \neq Cmax(X) \tag{4}$$

$$Cmax(X) \leq Cmax(SL) + ADIST \tag{5}$$

TSA proposed a buffer zone that acts as a waiting list for the non-explored regions because here are collected all the "invisible" candidates to the neighbourhood. Considering that all-knowing entities prepared into an individual iteration are "unseeable" then the neighbourhood is empty and the buffer zone's repository provides the next current solution. Once the search for non-explored areas is started from the best solution from the buffer zone, the buffer zone is ready to collect other non-explored solutions from scratch and $ADIST$ is increased with a small value.

3.1. **Algorithm TSA.**

$k \leftarrow 1, s \leftarrow NEH, SL \leftarrow s, SLO \leftarrow s,$
$F \leftarrow Cmax(SL), ADIST \leftarrow 0, maxIterNum \leftarrow 5000, maxNum \leftarrow 100$
while $k \leq maxIterNum$ **do**
  $add(TL, SL)$
  $tBuffer \leftarrow empty$
  $num \leftarrow 0$
  repeat
    $G \leftarrow generateRandom$
    if $G \in TL$ **then**

repeat
$G \leftarrow generateRandom$
$num \leftarrow num + 1$
until $[G \notin TL]$**or**$[num = maxNum]$
end if
if $[Cmax(G) \neq Cmax(SL)]$**and**$[Cmax(G) \leq Cmax(SL) + ADIST]$ **then**
$add(NH(SL), G)$
else
$add(tBuffer, G)$
end if
until $num \leq maxNum$
$ordersByCmax(NH(SL))$
$ordersByCmax(tBuffer)$
if $NH(SL) = empty$ **then**
$BT \leftarrow extractFirst(tBuffer)$
if $BT \in TL$ **then**
repeat
$remove(tBuffer, BT)$
$BT \leftarrow extractFirst(tBuffer)$
until $BT \in TL$
$SL \leftarrow BT$
$ADIST \leftarrow ADIST + 1$
end if
else
$BS \leftarrow extractFirst(NH(SL))$
$SL \leftarrow BS$
end if
if $Cmax(SL) < F$ **then**
$SL \leftarrow SLO$
$ADIST \leftarrow 0$
end if
$k \leftarrow k + 1$
end while

Notations:

- s – the initial solution is the solution obtained by Nawaz, Enscore and Ham [14] for the insertion heuristic algorithm known as NEH
- k – the iteration number, $maxIterNum$ - the number of iterations
- $maxNum$ - the number of the neighbors
- SL - the current solution in the kth iteration

- $NH(SL)$ - the neighbourhood of SL
- BS - the best-evaluated entity from $NH(SL)$ obtained based on the following formula
- SLO - the optimal solution
- $F - Cmax(SLO)$
- G - neighbor and $G \in NH(SL)$. The distance between each neighbor and the current solutions restricted by the $ADIST$
- TL – the tabu list
- $tBuffer$ - the buffer zone
- TL – the tabu list
- BT- the best-evaluated entity from $tBuffer$:

$$Cmax(BT) = \min\{Cmax(T), T \in tBuffer\} \tag{6}$$

- $generateRandom$ generates randomly an entity by interchanging two jobs positions
- $add(list, entity)$ adds entity to the list
- $ordersByCmax(list)$ orders the list descending by Cmax value calculated for each element
- $extractFirst(list)$ returns first element from the list
- $remove(list, entity)$ removes entity element from the list

With the adaptive distance, the speed of the search process is improved due to increasing the local-minimum-found probability. If a total number of search rounds to locate a local minimum is M (M is a positive integer) without the adaptive distance, the time consumed is $M \times Time(x)$ where $Time(x)$ is the time consumed to visit any single solution x in the search space. Once the **TSA**'s restrictions are applied (forumulas 4 and 5), the total search round is reduced by:

$$\alpha \times M \times Time(x) < M \times Time(x) where 0 < \alpha \leq 1 \tag{7}$$

Let be X' the best solution of $NH(SL)$, $Cmax(SL) < Cmax(X')$, then

$$0 < \alpha \leq ADIST = 1, \alpha = Cmax(X') - Cmax(SL) \tag{8}$$

4. RESULTS

Taillard proposed 12 sets of the processing times, each set with 10 instances of n jobs and m machine. The first set is the small one, the number of jobs is 20 and the number of machines is 20. For the next sets Taillard increased gradually the number of jobs along with the number of the machines until 500 jobs and 200 machines. Each **TSA**'s solution represented by $Cmax$ is

compared with the Upper Bound denoted by $Mean$ provided by Taillard for each set, using for the gap the following formula:

$$GAP = \frac{Cmax - UB}{UB} * 100\% \tag{9}$$

and for all 12 instances from a benchmark set it is calculated the average of the gaps. Beside the gap, for each problem are calculated: the average ($Mean$), Standard Deviation(SD), Standard Score(S) - how many Standard Deviation from the Mean of Cmax and the Confidence Interval of the Mean with a 95% percent Level of Confidence:

$$Mean = \frac{1}{IterNum} \sum_{i=1}^{IterNum} Cmax_i \tag{10}$$

where $IterNum$ is the iterations number.

$$SD = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (Cmax_i - Mean)^2} \tag{11}$$

$$S = \frac{Cmax - Mean}{SD} \tag{12}$$

TSA runs using 5000 iterations and obtains results very closely or identical with the known upper bounds for Taillard's[8] data sets. After the successive running of **TSA**, the maximum number of the iterations was limited to 5000, $IterNum = 5000$, and the size of the neighbourhood to 100 for all Taillard's[8] benchmarks set over 120 instances.

TABLE 1. Results of **TSA** running over each problem from Taillard benchmark sets

Results of **TSA** running over each problem from Taillard's **20 jobs and 5 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	*[CI*	*CI]*	CI_{width}
1	1278	1278	0.00	1,279.44	1.64	1	1,279.40	1,279.49	0.09
2	1359	1365	0.44	1,366.16	1.49	1	1,366.12	1,366.20	0.08
3	1081	1081	0.00	1,096.69	7.15	3	1,096.49	1,096.89	0.4
4	1293	1293	0.00	1,308.69	5.45	3	1,308.54	1,308.85	0.30
5	1236	1235	**-0.08**	1,249.64	4.43	4	1,249.52	1,249.77	0.25
6	1195	1210	1.26	1,211.30	1.92	1	1,211.25	1,211.36	0.11
7	1239	1251	0.97	1,251.89	1.14	1	1,251.85	1,251.92	0.06
8	1206	1206	**0.00**	1,212.64	4.64	2	1,212.51	1,212.77	0.26
9	1230	1230	**0.00**	1,235.53	8.27	1	1,235.30	1,235.75	0.46
10	1108	1108	**0.00**	1,112.84	5.07	1	1,112.70	1,112.98	0.28
Average			**0.26**						0.23
Results of **TSA** running over each problem from Taillard's **20 jobs and 10 machines** benchmark set:									
Continued on next page									

Table 1 continued from previous page

Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	1582	1583	0.06	1,611.46	11.15	3	1,611.15	1,611.77	0.62
2	1659	1664	0.30	1,698.61	9.28	4	1,698.35	1,698.86	0.51
3	1496	1500	0.27	1,527.46	8.60	4	1,527.22	1,527.70	0.48
4	1378	1377	***-0.07***	1,400.43	7.96	3	1,400.20	1,400.65	0.44
5	1419	1419	**0.00**	1,447.01	10.27	3	1,446.72	1,447.29	0.57
6	1397	1401	0.29	1,426.70	6.51	4	1,426.52	1,426.88	0.36
7	1484	1484	**0.00**	1,503.46	9.22	3	1,503.21	1,503.72	0.51
8	1538	1538	**0.00**	1,576.34	11.43	4	1,576.03	1,576.66	0.63
9	1593	1593	**0.00**	1,611.42	6.94	3	1,611.23	1,611.61	0.39
10	1591	1598	0.44	1,625.43	9.21	3	1,625.17	1,625.68	0.51
Average			**0.13**						0.50

Results of **TSA** running over each problem from Taillard's **20 jobs and 20 machines** benchmark set:

Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	2297	2298	0.04	2,338.04	12.11	4	2,337.70	2,338.37	0.67
2	2100	2111	0.52	2,139.21	9.01	4	2,138.96	2,139.46	0.50
3	2326	2328	0.09	2,370.37	13.07	4	2,370.01	2,370.73	0.72
4	2223	2233	0.45	2,261.82	9.92	3	2,261.55	2,262.10	0.55
5	2291	2298	0.31	2,338.72	11.84	4	2,338.39	2,339.05	0.66
6	2226	2229	0.13	2,263.12	12.86	3	2,262.76	2,263.48	0.71
7	2273	2281	0.35	2,319.05	12.69	3	2,318.70	2,319.40	0.70
8	2200	2207	0.32	2,241.08	11.07	4	2,240.78	2,241.39	0.61
9	2237	2242	0.22	2,275.21	12.58	3	2,274.86	2,275.55	0.70
10	2178	2179	0.05	2,219.79	12.85	4	2,219.44	2,220.15	0.71
Average			**0.25**						0.65

Results of **TSA** running over each problem from Taillard's **50 jobs and 5 machines** benchmark set:

Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	2724	2724	**0.00**	2,724.60	0.69	1	2,724.58	2,724.61	0.04
2	2834	2838	0.14	2,838.67	0.78	1	2,838.65	2,838.69	0.04
3	2621	2621	**0.00**	2,621.73	0.90	1	2,621.70	2,621.75	0.05
4	2751	2762	0.40	2,766.94	6.88	1	2,766.75	2,767.13	0.38
5	2863	2864	0.03	2,864.61	0.71	1	2,864.59	2,864.63	0.04
6	2829	2835	0.21	2,835.81	0.98	1	2,835.78	2,835.83	0.05
7	2725	2725	**0.00**	2,732.18	3.21	3	2,732.09	2,732.27	0.18
8	2683	2683	**0.00**	2,684.37	2.69	1	2,684.29	2,684.44	0.15
9	2552	2561	0.35	2,561.76	0.90	1	2,561.73	2,561.78	0.05
10	2782	2782	**0.00**	2,783.83	1.51	2	2,783.78	2,783.87	0.08
Average			**0.11**						0.11

Results of **TSA** running over each problem from Taillard's **50 jobs and 10 machines** benchmark set:

Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	3025	3075	1.65	3,099.26	13.10	2	3,098.90	3,099.62	0.73
2	2892	2911	0.66	2,947.72	15.70	3	2,947.28	2,948.15	0.87
3	2864	2905	1.43	2,928.51	11.26	3	2,928.19	2,928.82	0.62
4	3064	3071	0.23	3,081.95	10.45	2	3,081.66	3,082.24	0.58
5	2986	3024	1.27	3,040.77	17.38	1	3,040.29	3,041.26	0.96
6	3006	3026	0.67	3,055.78	24.54	2	3,055.10	3,056.46	1.36
7	3107	3165	1.87	3,171.32	10.70	1	3,171.02	3,171.62	0.59
8	3039	3060	0.69	3,068.82	5.52	2	3,068.67	3,068.98	0.31
9	2902	2932	1.03	2,940.12	9.11	1	2,939.87	2,940.38	0.51
10	3091	3120	0.94	3,147.51	15.73	2	3,147.07	3,147.94	0.87
Average			**1.04**						0.74

Results of **TSA** running over each problem from Taillard's **50 jobs and 20 machines** benchmark set:

Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	3875	3926	1.32	3,977.79	20.68	3	3,977.22	3,978.36	1.15
2	3715	3786	1.91	3,837.90	16.19	4	3,837.46	3,838.35	0.90
3	3668	3730	1.69	3,783.79	19.55	3	3,783.25	3,784.34	1.08
4	3752	3796	1.17	3,859.72	24.04	3	3,859.06	3,860.39	1.33
5	3635	3731	2.64	3,776.80	21.24	3	3,776.21	3,777.39	1.18
6	3698	3761	1.70	3,808.23	27.11	2	3,807.48	3,808.98	1.50
7	3716	3776	1.61	3,823.12	19.28	3	3,822.59	3,823.66	1.07

Continued on next page

Table 1 continued from previous page									
8	3709	3794	2.29	3,850.41	22.45	3	3,849.79	3,851.04	1.24
9	3765	3815	1.33	3,873.25	18.54	4	3,872.73	3,873.76	1.03
10	3777	3827	1.32	3,875.01	22.00	3	3,874.40	3,875.62	1.22
Average			**1.70**						1.17
Results of **TSA** running over each problem from Taillard's **100 jobs and 5 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	5493	5495	0.04	5,496.31	4.10	1	5,496.19	5,496.42	0.23
2	5268	5284	0.30	5,284.76	2.27	1	5,284.70	5,284.82	0.13
3	5175	5179	0.08	5,181.07	6.05	1	5,180.90	5,181.24	0.34
4	5014	5023	0.18	5,023.49	0.55	1	5,023.48	5,023.51	0.03
5	5250	5255	0.10	5,255.90	1.54	1	5,255.86	5,255.95	0.09
6	5135	5139	0.08	5,139.47	0.56	1	5,139.46	5,139.49	0.03
7	5246	5251	0.10	5,252.73	1.03	2	5,252.70	5,252.76	0.06
8	5106	5114	0.16	5,114.47	0.62	1	5,114.45	5,114.48	0.03
9	5454	5454	0.00	5,474.94	11.42	2	5,474.62	5,475.26	0.63
10	5328	5339	0.21	5,342.30	0.89	4	5,342.28	5,342.33	0.05
Average			**0.12**						0.16
Results of **TSA** running over each problem from Taillard's **100 jobs and 10 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	5770	5790	0.35	5,797.52	9.75	1	5,797.25	5,797.79	0.54
2	5349	5365	0.30	5,384.17	16.15	2	5,383.72	5,384.61	0.90
3	5677	5719	0.74	5,730.49	17.14	1	5,730.01	5,730.96	0.95
4	5791	5812	0.36	5,836.99	17.94	2	5,836.49	5,837.48	0.99
5	5468	5510	0.77	5,525.32	15.86	1	5,524.88	5,525.76	0.88
6	5303	5312	0.17	5,322.65	10.41	2	5,322.36	5,322.94	0.58
7	5599	5675	1.36	5,679.87	6.87	1	5,679.68	5,680.06	0.38
8	5623	5695	1.28	5,697.68	5.15	1	5,697.54	5,697.82	0.29
9	5875	5940	1.11	5,950.02	11.10	1	5,949.71	5,950.33	0.62
10	5845	5903	0.99	5,903.61	0.82	1	5,903.59	5,903.63	0.05
Average			**0.74**						0.62
Results of **TSA** running over each problem from Taillard's **100 jobs and 20 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	6286	6367	1.29	6,434.09	24.63	3	6,433.41	6,434.78	1.37
2	6241	6351	1.76	6,396.25	29.75	2	6,395.43	6,397.08	1.65
3	6329	6461	2.09	6,500.94	23.87	2	6,500.28	6,501.60	1.32
4	6306	6408	1.62	6,431.68	22.09	2	6,431.07	6,432.30	1.22
5	6377	6475	1.54	6,532.58	31.03	2	6,531.72	6,533.44	1.72
6	6437	6512	1.17	6,563.04	29.20	2	6,562.23	6,563.85	1.62
7	6346	6422	1.20	6,490.71	32.86	3	6,489.80	6,491.62	1.82
8	6481	6552	1.10	6,620.51	35.37	2	6,619.53	6,621.49	1.96
9	6358	6440	1.29	6,496.68	40.01	2	6,495.57	6,497.79	2.22
10	6465	6599	2.07	6,609.70	11.09	1	6,609.39	6,610.01	0.61
Average			**1.51**						1.55
Results of **TSA** running over each problem from Taillard's **200 jobs and 10 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	10868	10892	0.22	10,930.37	19.98	2	10,929.82	10,930.93	1.11
2	10494	10555	0.58	10,577.32	20.13	2	10,576.76	10,577.88	1.12
3	10922	11017	0.87	11,019.35	3.35	1	11,019.26	11,019.45	0.19
4	10889	11010	1.11	11,013.59	11.71	1	11,013.26	11,013.91	0.65
5	10524	10575	0.48	10,579.12	9.48	1	10,578.85	10,579.38	0.53
6	10331	10378	0.45	10,384.23	12.98	1	10,383.87	10,384.59	0.72
7	10857	10936	0.73	10,941.75	15.44	1	10,941.32	10,942.18	0.86
8	10731	10828	0.90	10,828.57	0.77	1	10,828.55	10,828.59	0.04
9	10438	10478	0.38	10,485.67	12.07	1	10,485.33	10,486.00	0.67
10	10676	10728	0.49	10,746.67	16.70	2	10,746.21	10,747.14	0.93
Average			**0.62**						0.68
Results of **TSA** running over each problem from Taillard's **200 jobs and 20 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	[*CI*	*CI*]	CI_{width}
1	11294	11406	0.99	11,457.89	54.00	1	11,456.39	11,459.39	2.99
2	11420	11472	0.46	11,530.50	32.46	2	11,529.60	11,531.40	1.80

Continued on next page

Table 1 continued from previous page									
3	11446	11555	0.95	11,635.79	63.65	2	11,634.03	11,637.56	3.53
4	11347	11566	1.93	11,632.65	35.95	2	11,631.65	11,633.64	1.99
5	11311	11455	1.27	11,497.29	43.05	1	11,496.10	11,498.48	2.39
6	11282	11488	1.83	11,496.73	17.52	1	11,496.24	11,497.21	0.97
7	11456	11591	1.18	11,650.84	39.73	2	11,649.74	11,651.94	2.20
8	11415	11599	1.61	11,636.32	46.84	1	11,635.02	11,637.62	2.60
9	11343	11457	1.01	11,518.17	49.46	2	11,516.80	11,519.54	2.74
10	11422	11590	1.47	11,676.66	61.09	2	11,674.97	11,678.35	3.39
Average			**1.27**						2.46
Results of **TSA** running over each problem from Taillard's **500 jobs and 20 machines** benchmark set:									
Problem	*UB*	*TSA*	*AD*	*Mean*	*SD*	*S*	*[CI*	*CI]*	CI_{width}
1	26189	26429	0.92	26,480.42	35.83	2	26,479.42	26,481.41	1.99
2	26629	26907	1.04	27,023.21	68.29	2	27,021.31	27,025.10	3.79
3	26458	26721	0.99	26,745.47	32.12	1	26,744.58	26,746.36	1.78
4	26549	26799	0.94	26,843.07	63.34	1	26,841.31	26,844.82	3.51
5	26404	26588	0.70	26,619.27	31.30	1	26,618.41	26,620.14	1.74
6	26581	26784	0.76	26,833.32	51.03	1	26,831.91	26,834.74	2.83
7	26461	26600	0.53	26,649.42	53.46	1	26,647.94	26,650.91	2.96
8	26615	26926	1.17	26,961.00	42.15	1	26,959.83	26,962.17	2.34
9	26083	26430	1.33	26,459.02	24.88	2	26,458.33	26,459.71	1.38
10	26527	26741	0.81	26,802.75	42.96	2	26,801.56	26,803.95	2.38
Average			**0.92**						2.47

5. DISCUSSION

TSA, coded in Java, ran on a PC INTEL™Core-i5 CPU @ 2.30 GHz processor 16 GB. **TSA**'s CPU times vary from 5 seconds (on the 20 benchmark set) to 20 minutes (on the 500 benchmark set).

Standard Deviation is a measure of central tendency. As Standard Deviation is the measure of the central tendency of Cmax set, a small value means that the Cmax-s are in the vicinity of the Mean. A large standard deviation value means that the Cmax values are farther away from the Mean. On the 200 and 500 benchmark sets, high standard deviations indicates that **TSA**'s exploration has conducted far distant from NEH[14] solution. For 20 jobs and 10 machines, 20 jobs and 20 machines and 50 jobs and 20 machines benchmark sets the Score is 3 or 4 and for many other sets Score is 2 (95% of all Cmax are within two standard distributions).

When Score=1, the global's makespan (Cmax) is located at a short distance to Mean. In order to see if these results are statistically significant it is provided the 95% confidence interval. The width of the confidence interval depends on the large sample size of the Cmax set. In some situations, the large sample size of Cmax and small standard deviation have combined to give very small intervals (TABLE 1).

TSA performs well with different heuristics, especially for the larger benchmark set. In TABLE 2 the proposed algorithm, **TSA** is compared with TS – a simplistic approach of the tabu search and with the best results of ALA algorithm extracted from the paper of Agarwal[1]. ALA is an improvement

heuristic based on adaptive learning approach using a constructive heuristic and improving the solution by perturbing the data based on a weight factor and allowing a non-deterministic local neighbourhood search. TS is a version of **TSA** starting with NEH[14] where random interchange of two jobs indexes alters the current solution and without the formulas 4 and 5 and the buffer zone. **TSA** is compared with the other heuristics proposed by Gupta[16] – called here H and by Eskenasi[13] marked here as HGA. HGA is a hybridization of the genetic algorithm with the iterated greedy search algorithm. H is similar to CDS heuristic[11] and converts the original m-machines problem into m-1 artificial 2-machines problems. Johnson's rule[17] is then applied to first artificial 2-machine problem to determine the sequence of jobs and the process is repeated by reducing the weight parameter until m-1 sequences are found.

TABLE 2. The average of the GAPS for Taillard's set for HGA,TSA,H

Taillard sets	**TSA**	**TS**	**ALA**[1]	**Taillard's set**	**HGA**[13]	**TSA**	**H**[16]
20 jobs sets	0.21	1.07	0.26	20 jobs and 5 machines	0.04	0.26	7.7
				20 jobs and 10 machines	0.03	0.13	10.61
				20 jobs and 20 machines	0.03	0.25	8.76
50 jobs sets	0.95	1.15	2.62	50 jobs and 5 machines	0.01	0.11	4.09
				50 jobs and 10 machines	0.73	1.04	10.96
				50 jobs and 20 machines	1.18	1.70	12
100 jobs sets	0.79	2.84	2.04	100 jobs and 5 machines	0.01	0.12	2.88
				100 jobs and 10 machines	0.26	0.74	7.64
				100 jobs and 20 machines	1.63	**1.51**	10.53
200 / 500 jobs sets	0.93	1.13	2.07	200 jobs and 10 machines	0.23	0.62	5.32
				200 jobs and 20 machines	1.54	1.27	9.4
				500 jobs and 20 machines	-	**0.92**	6.29

6. CONCLUSION

Since the PFSP is NP-hard for more than two machines, the advantage of this approach compared to other heuristics and meta-heuristics is that the medium and large problems can be solved optimally in this way. In general, since the number of jobs and the number of machines can be high, it is difficult to find the right solution with an exact method. In the proposed approach, tabu search starts with NEH[14] and marches through the non-explored areas. The adaptive distance that is used on filtering the candidate's makespans is combined with the buffer zone that collects all the "invisible" neighbors. With the adaptive distance, the speed of the search process is improved. This strategy can be extended on future researches on PFSP with a set of jobs constituting a production's batch.

References

[1] A.Agarwal, S.Colak, and E.Eryarsoy. Improvement heuristic for the flow-shop scheduling problem: An adaptive-learning approach. *European Journal of Operational Research 169* (2006), 801–815.

[2] Baker, K. R. Introduction to sequencing and scheduling.

[3] C.R.Reeves, and T.Yamada. Genetic algorithms, path relinking and the flowshop sequencing problem. *Evol Comput 16* (1998), 230–234.

[4] D.G.Dannenbring. An evaluation of flow-shop sequence heuristics. *Management Science 23* (1977), 1174–1182.

[5] D.S.Palmer. Sequencing jobs through a multistage process in the minimum total time: a quick method of obtaining a near-optimum. *Operational Research Quarterly 16* (1965), 101–107.

[6] E.Nowicki, and C.Smutnicki. A fast tabu search algorithm for the permutation flow-shop problem. *European Journal of Operational Research 91* (1996), 160–175.

[7] E.Taillard. Some efficient heuristic methods for the flow-shop sequencing problem. *European Journal of Operational Research 6* (1990), 65–74.

[8] E.Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research 64* (1993), 278–285.

[9] F.Glover. Tabu search-part i. *ORSA Journal on Computing 1*, 3 (1989), 190–206.

[10] G.Ochoa, and N.Veerapen. Deconstructing the big valley search space hypothesis. In *Evolutionary Computation in Combinatorial Optimization* (2016), F. Chicano, B. Hu, and P. García-Sánchez, Eds., Springer International Publishing, pp. 58–73.

[11] H.G.Campbell, R.A.Dudek, and M.L.Smith. A heuristic algorithm for the n-job, m-machine, sequencing problem. *Management Science 6* (1970), 630–637.

[12] J.N.D.Gupta. A functional heuristic for the flow-shop scheduling problem. *Operational Research Quarterly 22* (1971), 39–47.

[13] M.Eskenasi, and M.Mehrandezh. The permutation flow-shop scheduling using a genetic algorithm-based iterative method. *Industrial Engineering and Management* (2016).

[14] M.Nawaz, E.Enscore, and I.Ham. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *OMEGA 11* (1983), 91–95.

[15] M.R.Garey, D.S.Johnson, and R.Sethi. The complexity of flowshop and job shop scheduling. *Mathematics of Operations Research 1* (2016), 117–129.

[16] R.A.Gupta, and S.Chauhan. A heuristic algorithm for scheduling in a flow shop environment to minimize makespan. *International Journal of Industrial Engineering Computations 6* (2015), 173–184.

[17] S.M.Johnson. Optimal two-and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly 1* (1954), 61–68.

[18] Z.A.Drezner. New heuristic for the quadratic assignment problem. *Journal of Applied Mathematics and Decision Sciences 6* (2002), 163–173.

Technical University of Cluj-Napoca, Department of Manufacturing Engineering, B-dul Muncii 103–105, 400641 Cluj-Napoca, Romania

Email address: cristina.dodu@tcm.utcluj.ro, mircea.ancau@tcm.utcluj.ro