Volume 45 Number 2 June 2021 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Special Issue:

Middle-European Conference on Applied Theoretical Computer Science (MATCOS-19)

Guest Editors:

Andrej Brodnik and Gábor Galambos

1977

Editorial Boards

Informatica is a journal primarily covering intelligent systems in the European computer science, informatics and cognitive community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the list of referees. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Higher Education, Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Special issue on "Middle-European Conference on Applied Theoretical Computer Science – MATCOS-19"

The 5th conference MATCOS was held in Koper on October 10-11, 2019. We were lucky to come together just before months of the COVID pandemic of the last year. Although the MATCOS-series started as a collaboration between the University of Primorska and the University of Szeged, for this 5th conference researchers as speakers or co-authors came from 7 countries (15 chairs or institutes) including Austria, Germany, Israel, Romania, USA, Slovenia and Hungary.

In response to call of papers we received 39 submissions. Each submission was reviewed by at least two referees. Based on the reviews, the Program Committee first selected 22 papers for presentation. Having calculated the reviewer's opinion in a second phase we accepted further papers for short presentation on the closing day. In addition to the selected papers, Thomas Pock from the TU Graz gave an invited talk on "Learning better models for imaging".

The presented talks covered a wide spectrum of topics from both of theoretical computer science and the applied mathematics, and all the way to a practical application and evaluation of theoretical results. Papers were presented in the fields of influence maximization, formal languages and automata, scheduling, bin packing, bioinformatics, and graph theory as well.

The intention to involve PhD students into "conference-life" was declared during our earlier MATCOS conferences and was carried on to this conference as well. Consequently we are very happy that the students accepted this invitation, and they took part in MATCOS-19, so they could get their first impression about the flavour of a conference.

The MATCOS-19 offered a further chance to the speakers: after a regular reviewing process they can publish their results in the Informatica. We want to thank the editors and the whole board to give us this opportunity. After the reviewing process finally 8 papers were accepted for publication. Unfortunately, the reviewing process took a bit longer due to pandemic time. We hope that the presented papers will give a correct transverse section of the presentations of MATCOS-19.

At last but not least we want to thank Balázs Dávid, Branko Kavšek, Matjaž Krnc and Rok Požar for their contribution as a local Organizing Committee.

See you in 2022 on the next MATCOS conference!

Koper, July 2021

Andrej Brodnik
Gábor Galambos

A Full Cycle Length Pseudorandom Number Generator Based on Compositions of Automata

Pál Dömösi
Institute of Mathematics and Informatics, University of Nyíregyháza, H-4400 Nyíregyháza, Sóstói út 31/B, Hungary
Faculty of Informatics, University of Debrecen, H-4028 Debrecen, Kassai út 26, Hungary
E-mail: domosi@unideb.hu, domosi.pal@nye.hu

József Gáll and Géza Horváth
Faculty of Informatics, University of Debrecen, H-4028 Debrecen, Kassai út 26, Hungary
E-mail: gall.jozsef@inf.unideb.hu, horvath.geza@inf.unideb.hu

Bertalan Borsos
Faculty of Informatics, Eötvös Loránd University, H-1117 Budapest, Pázmány Péter sétány 1/C, Hungary
E-mail: bertalanborsos@gmail.com

Norbert Tihanyi
Digital14 LLC, xen1thLabs, Cryptography Laboratory, Abu Dhabi, United Arab Emirates
E-mail: norbert.tihanyi@digital14.com

Yousef Al Hammadi
College of Information Technology, United Arab Emirates University
P.O. Box 15551, Al Ain, Abu Dhabi, United Arab Emirates
E-mail: yousef-A@uaeu.ac.ae

Keywords: automata network, NIST test, block cipher, pseudo random number generator, composition of automata, Gluškov product of automata, temporal product of automata

Received: April 3, 2020

In this paper a new Pseudorandom Number Generator based on compositions of abstract automata is presented. We show that it has full cycle with length of 2^{128}. *It is also shown that the output satisfies the statistical requirements of the NIST randomness test suite.*

Povzetek: V prispevku je predstavljen nov psevdonaključni generator števil.

1 Introduction

In this paper, the authors continue their joint research of cryptographic tools based on compositions of abstract finite automata started in [6].

Random number generators have been used for a wide variety of fields and purposes, such as cryptography, pattern recognition, gambling and VLSI testing. [18]. In this paper a Pseudorandom Number Generator (PRNG) based on automata theory will be introduced and studied. The most frequent type of automata theory based pseudorandom generators are implemented on the basis of cellular automata. The first pseudorandom number generator based on cellular automata was proposed by S. Wolfram [29, 30] and there are many pseudorandom generators based on cellular automata today. (See [2]-[5], [8]-[16], [19, 20],[23]-[28]).

A common problem of some well-known pseudorandom generators based on cellular automata is that they have serious application difficulties: some of them can be broken [1],[17], while in case of others the selection of the key automaton poses difficulties [10].

These reasons, among others, justify the use of new automata theory-based pseudorandom number generators based on principles other than cellular automata.

Counter-based random number generation [22] is a relatively new technique for creating a pseudorandom number generator using only an integer counter as the internal state of the generator. The state transition function is an increment by one modulo the size n of the finite state set $S = \{0, \ldots, n-1\}$ and the complexity comes in the map from the state to the random sample. Formally, a counter based random number generator (CBRNG) is a structure $\mathcal{CBRNG} = (K, Z_J, S, f, U, g)$, where K is the key space; $Z_J = \{0, 1, ..., J-1\}$, where J is a positive integer called output multiplicity; S is the state space; U is the output space; $f : S \to S$ is the state transition function, $s_i = f(s_{i-1})$; $g : K \times Z_J \times S \to U$ is the output function. If Z_J is a singleton (i.e., $Z_J = \{0\}$) then we will write g in the form $g : K \times S \to U$ and then we say that $\mathcal{CBRNG}$ has a *simple output multiplicity*. Given an output function $g : K \times S \to U$ having a simple output multiplicity and assume that $U \subseteq S$. We say that the output function $g' : K \times S \to U$ is a double round of the output function $g : K \times S \to U$ if for every $k \in K, s \in S$, $g'(k, s) = g(k, g(k, s))$. In general, we say that $g' : K \times S \to U$ is a k-times round of $g : K \times S \to U$ for some $k > 2$ if for every $k \in K, s \in S$, $g'(k, s) = g(k, h(k, s))$ such that $h : K \times S$ is a $k-1$-times round of $g : K \times S$. Finally, the single round of $g : K \times S \to U$ is the function $g : K \times S \to U$ itself.

The $\mathcal{CBRNG} = (K, Z_J, S, f, U, g)$ works in discrete time scale. It starts from a fixed state $s \in S$, called *initial state* and a fixed key $k \in K$. Then the generated random number sequence is $g(k, 0, f^1(s)), \dots, g(k, J-1, f^1(s))$, $g(k, 0, f^2(s)), \dots, g(k, J-1, f^2(s)), g(k, 0, f^n(s))$, $, \dots, g(k, J-1, f^n(s))$, where $f^1(s) = f(s), f^2(s) = f(f(s))$ and $f^n(s) = f(f^{n-1}(s))$ for every further $n > 2$. In this case, the vector $(g(k, 0, f(s)), \dots g(k, J-1, f(s))$ is called the output vector of initial state.

Given a $\mathcal{CBRNG} = (K, Z_J, S, f, U, g)$ we say that its state transition function $f : S \to S$ *has a full cycle* if for every $s \in S, S = \{f^n(s) | n = 1, \dots, |S|\}$, where, by definition, $|S|$ denotes the cardinality of S. Moreover, a $\mathcal{CBRNG}$ is said to have a *full cycle* or *full period* if for any key and initial state $s \in S$, the $\mathcal{CBRNG}$ traverses every output vector $(u_0, \dots, u_{J-1}) \in U^J$ before returning to the output vector of the initial state.

The following statement is clear.

***Proposition* 1** A $\mathcal{CBRNG} = (K, Z_J, S, f, U, g)$ has a full cycle if and only if its state transition function $f : S \to S$ has a full cycle and for every key $k \in K$, the function $g_k : S \to U^J$ with $g_k(s) = (g(k, 0, s), \dots, g(J-1, s)), s \in S$ is bijective.

In this paper we consider CBRNGs having a simple output multiplicity. For CBRNGs, g is complex, f is a simple counter with $f(s) = (s+1) \ mod \ 2^p$, where p is the state size in bits and $S = \{0, \dots, p-1\}$. Applying the ideas of this construction, in this paper we consider CBRNGs, where f is a counter, and g is defined by composition of abstract finite automata.

2 Preliminaries

We start with some standard concepts and notation. For all notions and notation not defined here we refer to the monograph [7]. By an *automaton* we mean a deterministic finite automaton without outputs. In more detail, an automaton is an algebraic structure $\mathcal{A} = (A, \Sigma, \delta)$ consisting of the nonempty and finite *state set* A, the nonempty and finite *input set* Σ, and a *transition function* $\delta : A \times \Sigma \to A$. The transition matrix of an automaton is a matrix with rows corresponding to each input and columns corresponding to each state; at the entry of any row indicated by an input $x \in \Sigma$ sign and any column indicated by a state $a \in A$ the state $\delta(a, x)$ is put. If all rows of the transition matrix are permutations of the state set then we speak about *permutation automaton*.

A Latin square of order n is an $n \times n$ matrix (with n rows and n columns) in which the elements of an n-state set $\{a_0, a_1, \dots, a_{n-1}\}$ are entered so that each element occurs exactly once in each fixed (row, column) pair.

In this paper we will consider special compositions of automata consisting of component automata such that all components have the same transition matrix of the Latin square form. We will show that these compositions of automata are permutation automata, moreover for every state of these automata compositions it has a very low likelihood that two randomly chosen distinct input signs take the automaton into the same state. By these properties, we would like to avoid vulnerability to statistical attacks. Finally we note that, apart from the trivial cases, the transition matrices of the considered automata compositions are not quadratic. Therefore, their transition matrix can not form Latin squares.

3 Construction

We start with some standard definitions. (See, for example [6, 7]).

Let $\mathcal{A}_i = (A_i, \Sigma_i, \delta_i)$ be automata where $i \in \{1, \dots, n\}$, $n \geq 1$. Take a finite nonvoid set Σ and a *feedback function* $\varphi_i : A_1 \times \dots \times A_n \times \Sigma \to \Sigma_i$ for every $i \in \{1, \dots, n\}$. A *Gluškov-type product* of the automata $\mathcal{A}_i$ with respect to the feedback functions φ_i $(i \in \{1, \dots, n\})$ is defined to be the automaton $\mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_n(\Sigma, (\varphi_1, \dots, \varphi_n))$ with state set $A = A_1 \times \dots \times A_n$, input set Σ, transition function δ given by $\delta((a_1, \dots, a_n), x) = (\delta_1(a_1, \varphi_1(a_1, \dots, a_n, x)), \dots, \delta_n(a_n, \varphi_n(a_1, \dots, a_n, x)))$ for all $(a_1, \dots, a_n) \in A$ and $x \in \Sigma$. In particular, if $\mathcal{A}_1 = \ldots = \mathcal{A}_n$ then we say that $\mathcal{A}$ is a *Gluškov-type power.*

Given a function $f : X_1 \times \dots \times X_n \to Y$, we say that f is *really independent of its i-th variable* if for every pair $(x_1, \dots, x_n), (x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n) \in X_1 \times \dots \times X_n$, $f(x_1, \dots, x_n) = f(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n)$. Otherwise we say that f *really depends on its i-th variable*. A (finite) *directed graph* (or, in short, a *digraph*) $\mathcal{D} = (V, E)$ (of order $n > 0$) is a pair consisting of sets of *vertices* $V = \{v_1, \dots, v_n\}$ and *edges* $E \subseteq V \times V$. Elements of V are sometimes called *nodes*. If $|V| = n$ then we also say that $\mathcal{D}$ is a digraph of order n. $\mathcal{D}$ is called *bipartite* if its vertices can be partioned into two sets A, B such that every (direct) edge connects a vertex in A to a vertex in B or vica versa. Further on, we will assume that V is an ordered set of integers $1, \dots n$ for some positive integer n. Given a digraph $\mathcal{D} = (V, E)$, we say that the above defined Gluškov product is a $\mathcal{D}$-product if for every pair $i, j \in \{1, \dots, n\}$, $(i, j) \notin E$ implies that the feedback function φ_i is really independent of its j-th variable. Let Σ be the set of all ℓ (preferably 4 or 8) length binary strings for a given length $\ell > 0$. Moreover, let $\mathcal{A}_i = (\Sigma, \Sigma, \delta_{\mathcal{A}_i}), i = 2, \dots, n$ be copies of $\mathcal{A}_1$, and let n be a positive integer power of 2. Consider the following simple bipartite digraphs:

$\mathcal{D}_1 = (\{1,\dots,n\},\{(n/2+1,1),(n/2+2,2),\dots,(n,n/2)\})$,

$\mathcal{D}_2 = (\{1,\dots,n\},\{(n/4+1,1),(n/4+2,2),\dots,(n/2,n/4),$
$(3n/4+1,n/2+1),(3n/4+2,n/2+2),\dots,(n,3n/4)\})$,

$\dots$,

$\mathcal{D}_{\log_2 n-1} = (\{1,\dots,n\},\{(3,1),(4,2),(7,5),(8,6),\dots,$
$(n-1,n-3),(n,n-2)\})$,

$\mathcal{D}_{\log_2 n} = (\{1,\dots,n\},\{(2,1),(4,3),\dots,(n,n-1)\})$,

$\mathcal{D}_{\log_2 n+1} = \mathcal{D}_1$.

For every digraph $\mathcal{D} = (V,E)$ with $\mathcal{D} \in \{\mathcal{D}_1,\dots,\mathcal{D}_{\log_2 n}\}$, let us define the Gluškov-type power, called *two-phase $\mathcal{D}$-power*, $\mathcal{A}_\mathcal{D} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n(\Sigma^n,(\varphi_1,\dots,\varphi_n))$ of $\mathcal{A}_1$ (with $\mathcal{A}_1 = \mathcal{A}_2 \dots = \mathcal{A}_n$) so that for every $(a_1,\dots,a_n),(x_1,\dots,x_n) \in \Sigma^n, (i,j) \in E$, $\varphi_i(a_1,\dots,a_n,(x_1,\dots,x_n)) = a'_j \oplus x_j$, and $\varphi_j(a_1,\dots,a_n,(x_1,\dots,x_n)) = a_i \oplus x_i$, where $a_i \oplus x_i$ is the bitwise addition modulo 2 of a_i and x_i, a'_j denotes the state into which $\varphi_j(a_1,\dots,a_n,(x_1,\dots,x_n))$ takes the automaton $\mathcal{A}_i$ from its state a_j, and $a'_j \oplus x_j$ is the bitwise addition modulo 2 of a'_j and x_j. [1]

Next we define the concept of *temporal product* of automata. Let $\mathcal{A}_t = (A,\Sigma_t,\delta_t), t = 1,2$ be automata having a common state set A. Take a finite nonvoid set Σ and a mapping φ of Σ into $\Sigma_1 \times \Sigma_2$. Then the automaton $\mathcal{A} = (A,\Sigma,\delta)$ is a *temporal product* (t-product) of $\mathcal{A}_1$ by $\mathcal{A}_2$ with respect to Σ and φ if for any $a \in A$ and $x \in \Sigma$, $\delta(a,x) = \delta_2(\delta_1(a,x_1),x_2)$, where $(x_1,x_2) = \varphi(x)$. The concept of temporal product is generalized in the natural way to an arbitrary finite family of $n > 0$ automata $\mathcal{A}_t$ ($t = 1,\dots,n$), all with the same state set A.

***Proposition* 2** Suppose that $\mathcal{A}_1 = (\Sigma,\Sigma,\delta)$ is a permutation automaton. Then for every $i = 1,\dots,\log_2 n$, the $\mathcal{D}_i$-power of $\mathcal{A}_1$ also forms a permutation automaton.

Proof. Assume that $\mathcal{A}_1$ is a permutation automaton. Then, by definition, all rows of its transition matrix are permutations of the state set. Therefore, none of these rows contain repetition. Consequently, for any states $a,b \in A_1$ and input $x \in \Sigma$, $\delta_1(a,x) = \delta_1(b,x)$ implies $a = b$.

By our above observation, if the $\mathcal{D}_i$-power $\mathcal{A}_{\mathcal{D}_i} = (\Sigma^n,\Sigma^n,\delta_{\mathcal{D}_i})$ is a permutation automaton, then for every pair of states $(a_1,\dots,a_n),(b_1,\dots,b_n)$ and input sign $(x_1,\dots,x_n)$, we have $\delta_{\mathcal{D}_i}((a_1,\dots,a_n),(x_1,\dots,x_n)) = \delta_{\mathcal{D}_i}((b_1,\dots,b_n),(x_1,\dots,x_n))$ that implies $(a_1,\dots,a_n) = (b_1,\dots,b_n)$.

Suppose that $\mathcal{A}_{\mathcal{D}_i}$ is not a permutation automaton. Then, by our observations, there are a pair of distinct states $(a_1,\dots,a_n),(b_1,\dots,b_n)$ and an input sign $(x_1,\dots,x_n)$ for which $\delta_{\mathcal{D}_i}((a_1,\dots,a_n),(x_1,\dots,x_n)) = \delta_{\mathcal{D}_i}((b_1,\dots,b_n),(x_1,\dots,x_n))$.

[1]We remark that there are $V_1,V_2 \subset V$ with $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \emptyset$ so that for every $j \in V_2$ there exists exactly one $i \in V_1$ with $(j,i) \in E$. Therefore all the functions $\varphi_1,\dots,\varphi_n$ are well-defined.

If $(a_1,\dots,a_n)$ and $(b_1,\dots,b_n)$ are distinct then there exists an index $j \in \{1,\dots,n\}$ with $a_j \neq b_j$. Put $(a'_1,\dots,a'_n) = \delta_{\mathcal{D}_i}((a_1,\dots,a_n),(x_1,\dots,x_n))$ and $(b'_1,\dots,b'_n) = \delta_{\mathcal{D}_i}((b_1,\dots,b_n),(x_1,\dots,x_n))$. It is enough to show that, in this case, $(a'_1,\dots,a'_n) \neq (b'_1,\dots,b'_n)$.

Let E denote the set of edges in $\mathcal{D}_i$.

First we suppose $(j,k) \in E$ for some $k \in \{1,\dots,n\} \setminus \{j\}$. Then, by definition, $a'_j = \delta_{\mathcal{A}_1}(a_j, a'_k \oplus x_k)$ and $b'_j = \delta_{\mathcal{A}_1}(b_j, b'_k \oplus x_k)$. Because $\mathcal{A}_1$ is a permutation automaton, by the assumption $a'_k = b'_k$, we get $a_j = b_j$, a contradiction. Therefore $a'_k \neq b'_k$. Hence, $(a'_1,\dots,a'_n) \neq (b'_1,\dots,b'_n)$.

Next we assume $(k,j) \in E$ for some $j \in \{1,\dots,n\} \setminus \{k\}$. Obviously, by $a'_j \neq b'_j$ we have $(a'_1,\dots,a'_n) \neq (b'_1,\dots,b'_n)$ and then we are done once again. Therefore, suppose $a'_j = b'_j$. Then, again by definition, $a'_j = \delta_{\mathcal{A}_1}(a_j, a_k \oplus x_k)$ and $b'_j = \delta_{\mathcal{A}_1}(b_j, b_k \oplus x_k)$. Because $\mathcal{A}_1$ is a permutation automaton and $a_k \neq b_k$, $a'_j = b'_j$ is possible only if $a_j = b_j$, a contradiction. Therefore, $a'_j \neq b'_j$ and thus we obtain $(a'_1,\dots,a'_n) \neq (b'_1,\dots,b'_n)$. The proof is complete. QED

Now we give an alternative proof of Lemma 2 in [6].

***Proposition* 3** Temporal products of permutation automata are also permutation automata.

Proof. Obviously, it is enough to prove our statement for temporal products of two components. Thus, let $\mathcal{M} = (M,\Sigma,\delta_\mathcal{M})$ be a temporal product of permutation automata $\mathcal{M}_1 = (M,\Sigma_1,\delta_{\mathcal{M}_1})$ and $\mathcal{M}_2 = (M,\Sigma_2,\delta_{\mathcal{M}_2})$ (having the same state set) with respect to Σ and $\varphi : \Sigma \to \Sigma_1 \times \Sigma_2$. Let $m_1,m_2 \in M$ be distinct states and $x \in \Sigma$ be an arbitrary input sign of $\mathcal{M}$. Moreover, let $\varphi(x) = (x_1,x_2)$ for some $x_1 \in \Sigma_1$ and $x_2 \in \Sigma_2$. Then for every distinct pair $m_1,m_2 \in M$ of states and $x_1 \in \Sigma_1$ we have $\delta_{\mathcal{M}_1}(m_1,x_1) \neq \delta_{\mathcal{M}_2}(m_2,x_2)$. On the other hand, $\mathcal{M}_2$ is also a permutation automaton. Therefore, because of $\delta_{\mathcal{M}_1}(m_1,x) \neq \delta_{\mathcal{M}_2}(m_2,x)$, for every $x_2 \in \Sigma_2$, $\delta_{\mathcal{M}_1}(\delta_{\mathcal{M}_1}(m_1,x_1),x_2) \neq \delta_{\mathcal{M}_2}(\delta_{\mathcal{M}_2}(m_2,x_1),x_2)$. Thus, using $\varphi(x) = (x_1,x_2), \delta_\mathcal{M}(m_1,x) \neq \delta_\mathcal{M}(m_2,x)$. In other words, for every distinct pair $m_1,m_2 \in M$ of states and input sign $x \in \Sigma, \delta_\mathcal{M}(m_1,x) \neq \delta_\mathcal{M}(m_2,x)$. But then all rows of the transition matrix of $\mathcal{M}$ are permutations of the state set. This completes the proof. QED

Let $\mathcal{B} = (\Sigma^n,(\Sigma^n)^{\log_2 n},\delta_\mathcal{B})$ be the temporal product of $\mathcal{A}_{\mathcal{D}_1},\dots,\mathcal{A}_{\mathcal{D}_{\log_2 n}}$ with respect to $(\Sigma^n)^{\log_2 n}$ and the identity map $\varphi : (\Sigma^n)^{\log_2 n} \to (\Sigma^n)^{\log_2 n}$, where $\mathcal{A}_{\mathcal{D}_i}, i = 1,\dots,\log_2 n$ is a $\mathcal{D}_i$ - power of the automaton $\mathcal{A}_1 = (\Sigma,\Sigma,\delta_{\mathcal{A}_1})$.

From now on we assume that $\mathcal{A}_1$ is a permutation automaton having $\delta_{\mathcal{A}_1}(a,x) \neq \delta_{\mathcal{A}_1}(a,x')$ for every $a,x,x' \in \Sigma, x \neq x'$, and we say that $\mathcal{B}$ is a *key-automaton* with respect to the permutation automaton $\mathcal{A}_1$ called the basic automaton of $\mathcal{B}$.[2]

Theorem 1 in [6] concerns key automata consisting of basic automata having a transition table forming a Latin

[2]Recall that n should be a positive integer power of 2.

cube. The next statement is formally the same but, of course, it concerns key automata consisting of basic automata having a transition table forming a Latin square. By this fact, the next statement could also be derived from Theorem [6] using some simplification regarding its proof.

We note that the following statement is formally the same as Theorem 1 [6] which is concerning key automata consisting of basic automata having a little bit different structure as basic automata of the present paper. (The transition table of basic automata in [6] forms a Latin cube while the transition table of basic automata of the present paper forms a Latin square.) This fact implies that the automata compositions discussed in the present paper are more or less similar to the ones in [6].

For the sake of simplicity, we give a direct proof of the next statement which, using some simplifications, can also be derived from the proof of Theorem 1 in [6] .

***Theorem* 4** Every key automaton is a permutation automaton.

Proof. Consider a key automaton $\mathcal{B} = (\Sigma^n, (\Sigma^n)^{\log_2 n}, \delta_{\mathcal{B}})$ and its basic automaton $\mathcal{A}_1 = (A_1, \Sigma_1, \delta 1)$. By the definition of key automaton, $\mathcal{A}_1$ is a permutation automaton.

Therefore, using Proposition 2, for every permutation automaton $\mathcal{A}_1$ and $i = 1, 2, \dots, \log_2 n$, the $\mathcal{D}_i$-power $\mathcal{A}_{\mathcal{D}_i} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n(\Sigma^n, (\varphi_1, \dots, \varphi_n))$ of $\mathcal{A}_1$ is a permutation automaton.

Recall that $\mathcal{B}$ is a temporal product of $\mathcal{A}_{\mathcal{D}_1}, \dots, \mathcal{A}_{\mathcal{D}_{\log_2 n}}$. Therefore, by Proposition 3, our proof is done. QED

***Proposition* 5** Suppose that the transition matrix of an automaton $\mathcal{A}_1 = (\Sigma, \Sigma, \delta)$ forms a Latin square. Then for every $i = 1, \dots, \log_2 n$, the transition matrix of the $\mathcal{D}_i$-power of $\mathcal{A}_1$ also forms a Latin square.

Proof. Obviously, if the transition matrix of an automaton $\mathcal{A}_i = (Ai, \Sigma, \delta)$ forms a Latin square then $\mathcal{A}$ is a permutation automaton. But then, by Proposition 2, all of $\mathcal{D}_i$-powers are permutation automata. In other words, the rows of their transition matrices form a permutation of their state set. All that is left is to show that the columns of their transition matrices also have this property.

Consider a $\mathcal{D}_i$-power $\mathcal{A}_{\mathcal{D}_i} = (\Sigma^n, \Sigma^n, \delta_{\mathcal{D}_i})$ of $\mathcal{A}_1$ having $\mathcal{A}_{\mathcal{D}_i} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n(\Sigma^n, (\varphi_1, \dots, \varphi_n))$ for some $i = 1, 2, \dots, \log_2 n$. We should prove that for every state $(a_1, \dots, a_n) \in \Sigma^n$ and distinct words $(x_1, \dots, x_n), (y_1, \dots, y_n) \in \Sigma^n$, $\delta_{\mathcal{D}_i}((a_1, \dots, a_n), (x_1, \dots, x_n)) \neq \delta_{\mathcal{D}_i}((a_1, \dots, a_n), (y_1, \dots, y_n))$. Let $j \in \{1, \dots, n\}$ be an index with $x_j \neq y_j$. Moreover, let E be the set of edges in $\mathcal{D}_i$ and let us assume that $(i, j) \in E$ (with $i \in \{1, \dots, n\} \setminus \{j\}$). Then $x_i \neq y_i$ implies $\delta(a_j, a_i \oplus x_i) \neq \delta(a_j, a_i \oplus y_i)$. Therefore, by our construction, the j-th components of $\delta_{\mathcal{D}_i}((a_1, \dots, a_n), (x_1, \dots, x_n))$ and $\delta_{\mathcal{D}_i}((a_1, \dots, a_n), (y_1, \dots, y_n))$ are distinct as we stated. Now we assume $(i, j) \in E$ (with $i \in \{1, \dots, n\} \setminus \{j\}$). If $\delta(a_i, a_j \oplus x_j) \neq \delta(a_i, a_j \oplus y_j)$ then the i-th components of $\delta_{\mathcal{D}_i}((a_1, \dots, a_n), (x_1, \dots, x_n))$ and $\delta_{\mathcal{D}_i}((a_1, \dots, a_n), (y_1, \dots, y_n))$ are distinct again. Therefore, let $\delta(a_i, a_j \oplus x_j) = \delta(a_i, a_j \oplus y_j)$. But then, by $x_i \neq y_i$, we receive $\delta(a_i, a_j \oplus x_j) \oplus x_i \neq \delta(a_i, a_j \oplus y_j) \oplus y_i$. Obviously, this leads to $\delta(a_j, \delta(a_i, a_j \oplus x_j) \oplus x_i) \neq \delta(a_j, \delta(a_i, a_j \oplus y_j) \oplus y_i)$. Thus we get again that the j-th component of $\delta_{\mathcal{D}_i}((a_1, \dots, a_n), (x_1, \dots, x_n))$ and $\delta_{\mathcal{D}_i}((a_1, \dots, a_n), (y_1, \dots, y_n))$ are distinct. This finishes the proof. QED

Obviously, if the automaton $\mathcal{A}_1$ has a transition table forming a Latin square then it is a permutation automaton. Therefore, it can be a basic automaton of an appropriate key automaton.

Observation 6 Consider a key automaton $\mathcal{B}$ for which its basic automaton $\mathcal{A}_1$ has a transition table forming a Latin square. Then for every state a of $\mathcal{B}$, the probability that its two random signs take $\mathcal{B}$ from a into the same state is $((2|\Sigma| - 1)/|\Sigma|^3)^{n/2}$, where Σ is the set of states of $\mathcal{A}_1$ and n is the number of (identical) component automata in each $\mathcal{D}_i$ power ($i = 1, \dots, \log_2 n$) for which $\mathcal{B}$ is a temporal product of these $\mathcal{D}_i$ powers.

Proof. Denote by $\mathcal{M} = (\Sigma^n, (\Sigma^n)^{\log_2 n - 1}, \delta_{\mathcal{M}})$ the temporal product consisting of the first $\log_2 n - 1$ components $\mathcal{A}_{\mathcal{D}_i}, i = 1, \dots, \log_2 n - 1$ of the key automaton $\mathcal{B} = (\Sigma^n, (\Sigma^n)^{\log_2 n}, \delta_{\mathcal{B}})$ and consider the $\log_2 n$-th component $\mathcal{A}_{\mathcal{D}_{\log_2 n}} = (\Sigma^n, \Sigma^n, \delta_{\mathcal{D}_{\log_2 n}})$ of $\mathcal{B}$. By Proposition 2 and Proposition 3, $\mathcal{M}$ is a permutation automaton. Therefore, for every state $(a_1, \dots, a_n) \in \Sigma^n$ of $\mathcal{M}$ its random input signs $w \in (\Sigma^n)^{\log_2 n - 1}$ take $\mathcal{M}$ into each state with the same probabbiliy $1/|\Sigma^n|$.

Consider a fixed state $(c_1, \dots, c_n) \in \Sigma^n$ and a randomly chosen pair $w_1, w_2 \in (\Sigma^n)^{\log_2 n - 1}$ of $\mathcal{M}$ and denote by $(a_1, \dots, a_n), (a_1, \dots, a_n) \in M$ the pair of states in $\mathcal{M}$ such that $\delta_{\mathcal{M}}((c_1, \dots, c_n), w_1) = (a_1, \dots, a_n)$ and $\delta_{\mathcal{M}}((c_1, \dots, c_n), w_2) = (a_1, \dots, a_n)$.

Moreover consider a randomly chosen pair $(x_1, \dots, x_n), (y_1, \dots, y_n) \in \Sigma^n$ of input signs of $\mathcal{A}_{\mathcal{D}_{\log_2 n}}$, and assume that $\delta_{\mathcal{D}_{\log_2 n}}((a_1, \dots, a_n), (x_1, \dots, x_n)) = \delta_{\mathcal{D}_{\log_2 n}}(b_1, \dots, b_n), (y_1, \dots, y_n))$. For every $i = 1, \dots, n$, there exists a single $i \in \{1, \dots, n\}$ such that either $(i, j) \in E$ or $(j, i) \in E$, where E denotes the set of edges in the digraph $\mathcal{D}_{\log_2 n}$. There are the following cases. If $(i, j) \in E$ then $\delta(a_i, \delta(a_j, a_i \oplus x_i) \oplus x_j)$ and $\delta(b_i, \delta(b_j, b_i \oplus y_i) \oplus y_j))$ will be the i-th and $\delta(a_j, a_i \oplus x_i)$ and $\delta(b_j, b_i \oplus y_i)$ will be the j-th component of $\delta_{\mathcal{D}_{\log_2 n}}((a_1, \dots, a_n), (x_1, \dots, x_n))$ and $\delta_{\mathcal{D}_{\log_2 n}}(b_1, \dots, b_n), (y_1, \dots, y_n))$, respectively. By the assumption $\delta_{\mathcal{D}_{\log_2 n}}((a_1, \dots, a_n), (x_1, \dots, x_n)) = \delta_{\mathcal{D}_{\log_2 n}}(b_1, \dots, b_n), (y_1, \dots, y_n))$, we have $\delta(a_i, \delta(a_j, a_i \oplus x_i) \oplus x_j) = \delta(b_i, \delta(b_j, b_i \oplus y_i) \oplus y_j)$ and $\delta(a_j, a_i \oplus x_i) = \delta(b_j, b_i \oplus y_i)$ By the second equality, we can write the first one in the form $\delta(a_i, \delta(a_j, a_i \oplus x_i) \oplus x_j) = \delta(b_i, \delta(a_j, a_i \oplus x_i) \oplus y_j)$. Recall that the transition matrix of $\mathcal{A}_1$ forms a Latin square. Therefore, by these considered equalities, $a_i = b_i$ if and only if $x_j = y_j$ and $a_j = b_j$ if and only if $x_i = y_i$.

On the other hand, for every $k = 1, \dots, n$, there are $|\Sigma|^2$

cases having $a_k = b_k$ and $x_k = y_k$, $a_k, b_k, x_k, y_k \in \Sigma$. Of course, all of these cases take the i-th and j-th components of $\mathcal{A}_{\mathcal{D}_{\log_2 n}}$ into the same state.

In addition, every element of Σ appears exactly $|\Sigma|$-times in the transition table of $\mathcal{A}_1$ because it forms a Latin square. Moreover, we can consider only nonequal pairs of quadruplets.

Hence there are $|\Sigma|(|\Sigma|-1)$ number of quadruple possibilities $a_i, b_i, x_i, y_i \in \Sigma, i = 1, \ldots, n$ having $a_i \neq b_i, x_i \neq y_i$ taking the i-th and j-th components of $\mathcal{A}_{\mathcal{D}_{\log_2 n}}$ into the same state.

In sum, we have that the probability that $\delta(a, x)$ and $\delta(b, y)$ coincide for a random quadruple $a, b, x, y \in \Sigma$ is $(2|\Sigma|^2 - |\Sigma|)/|\Sigma|^4 = (2|\Sigma| - 1)/|\Sigma|^3$.

By our constructions, the digraph $\mathcal{D}_{\log_2 n}$ has $n/2$ edges. Consequently, the probability that a random quadruple $(a_1, \ldots, a_n), (b_1, \ldots, b_n), (x_1, \ldots, x_n), (y_1, \ldots, y_n) \in \Sigma^n$ has the property $\delta_{\mathcal{D}_{\log_2 n}}((a_1, \ldots, a_n), (x_1, \ldots, x_n)) = \delta_{\mathcal{D}_{log_2 n}}(b_1, \ldots, b_n), (y_1, \ldots, y_n))$ is $((2|\Sigma| - 1)/|\Sigma|^3)^{n/2}$.

We remark that if we have an implementation with $|\Sigma| = 256$ and $n = 16$ then the considered probability is $((512 - 1)/256^3)^8 \approx 1/2^{120}$.

By our investigations, we receive that the probability of the equality $\delta_{\mathcal{B}}((c_1, \ldots, c_n), w_1(x_1, \ldots, x_n)) = \delta_{\mathcal{B}}((c_1, \ldots, c_n), w_2(y_1, \ldots, x_n))$ is $((2|\Sigma| - 1)/|\Sigma|^3)^{n/2}$, whenever $w_1(x_1, \ldots, x_n)$ and $w_2(y_1, \ldots, x_n)$ are randomly chosen input signs of $\mathcal{B}$. QED

***Theorem* 7** Every key automaton transition function can be applied as an output function of a counter-based pseudo random number generator.

Proof. As the proof of our statement, we give a construction of an appropriate counter based PRNG (CBRNG) $\mathcal{CBRNG} = (K, Z_J, S, f, U, g)$ having this property. First of all, consider a counter which realizes the state function as $f(n) = n + 1 \ mod\ m$, where m is a sufficiently large positive integer (preferably $m = 2^{128}$), and n is given as a fixed-length binary number (preferably with 128-bit length). For sake of simplicity, assume that $\mathcal{CBRNG}$ has a simple output multiplicity, i.e., let $Z_J = \{0\}$ be a singleton (although it may have more than one element). Thus the state space is $S = \{0, \ldots, m-1\}$. The elements of S may be considered binary strings of fixed length. Therefore, we may assume that S coincides with the state set Σ^n of the key automaton. We assume $K \subseteq S \times (\Sigma^n)^{2\log_2 n}$, where $S = \Sigma^n$ is the state set, and $(\Sigma^n)^{2\log_2 n}$ is the input set of the key automaton $\mathcal{B} = (\Sigma^n, (\Sigma^n)^{\log_2 n}, \delta_{\mathcal{B}})$. The first component of the elements of K are considered as possible seeds of the random number generator and the second one is an input element of $\mathcal{B}$. The output space U and the state space S coincide. The output function $g : K \times S \to U$ is given as $g(k, (a_1, \ldots, a_n)) = b_1||b_2|| \ldots ||b_n, k \in K, (a_1, \ldots, a_n) \in S(= \Sigma^n)$, where $b_1||b_2|| \ldots ||b_n$ is the concatenation of $b_1, \ldots, b_n$ as binary strings and $(b_1, \ldots, b_n)$ is a state of the key automaton $\mathcal{B}$ with $(b_1, \ldots, b_n) = \delta_{\mathcal{B}}((a_1, \ldots, a_n), (x_1, \ldots, x_n))$, where the concatenation $a_1|| \ldots ||a_n$ of $a_1, \ldots, a_n$ as binary strings is given by $a_1|| \ldots ||a_n = s + k \ mod\ m$, where $s + k \ mod\ m$ is the k-th state of the state space S starting from the state s, $s \in S$ is the first component of the key $(s, x_1|| \ldots ||x_n) \in K$ and $x_1|| \ldots ||x_n$ is the second one.

Finally, for every $w = a_1|| \ldots ||a_n, (a_1, \ldots, a_n) \in S$, put $\overline{w} = (a_1, \ldots, a_n)$. Then we can consider the double round $g' : K \times S \to U$ of $g : K \times S \to U$ (with $U = S$) such that for evey pair $k \in K, s \in S$, $g'(k, s) = g(k, \overline{g(k, s)})$. Similarly, for every $k > 2$, we can consider the k-times round $g'' : K \times S \to U$ of $g : K \times S \to U$ (with $U = S$) such that for evey pair $k \in K, s \in S$, $g''(k, s) = g(k, \overline{h(k, s)})$, where $h : K \times S \to U$ is a $(k-1)$-times round of $g : K \times S \to U$. This completes the proof. QED

By Proposition 1, Theorem 3, and Theorem 7, we can derive the following.

***Corollary* 8** Let $\mathcal{CBRNG} = (K, Z_J, S, f, U, g)$ be a counter based pseudorandom number generator with simple output multiplicity (i.e., $Z_J = \{0\}$) and assume that its output function is defined by the transition function of a given key automaton. Then $\mathcal{CBRNG}$ has a full cycle.

Proof. Of course, because the state transition f of $\mathcal{CBRNG}$ (having a simple output multiplicity) is a simple counter with $f(s) = (s + 1) \ mod\ 2^p$, where p is the state size in bits and $S = \{0, \ldots, p-1\}$, f has a full cycle. Moreover, by Theorem 4, the key automaton $\mathcal{B} = (\Sigma^n, (\Sigma^n)^{\log_2 n}, \delta_{\mathcal{B}}), n > 1$ is a permutation automaton, therefore, for every input sign $x \in (\Sigma^n)^{\log_2 n}$, $g_x : \Sigma^n \to \Sigma^n$ with $g_x(y) = \delta_{\mathcal{B}}(y, x)$ is a bijective mapping of Σ^n onto itself. By Proposition 1, that means that $\mathcal{CBRNG}$ (having a simple output multiplicity) has a full cycle. QED

Next we give an example and then we study the security of our $CBRNG$.

4 Example

In this section we show a simple example.

Consider the following transition table of an automaton $\mathcal{A} = (\{0,1\}, \{0,1\}^2, \delta)$:

δ	00	01	10	11
0	0	1	1	0
1	1	0	0	1

Let $n = 4$ and assume that all of $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ coincide with $\mathcal{A}$. Then $\log_2 n = \log_2 4 = 2$ and thus

$\mathcal{D}_1 = (\{1, \ldots, 4\}, \{(3,1), (4,2)\})$,

$\mathcal{D}_2 = (\{1, \ldots, 4\}, \{(2,1), (4,3)\})$.

Suppose that either a counter or a pseudorandom number generator sends an input $(1,0,1,0,1,0,1,0)$ to the key automaton $\mathcal{B}$ which is the temporal product of $\mathcal{A}_{\mathcal{D}_1}$ and $\mathcal{A}_{\mathcal{D}_2}$. Assume that $\mathcal{B}$ is in the state $(0,1,1,0)$.

Denote, in order, $\varphi_i, a_i, a'_i, x_i, i \in \{1,2,3,4\}$ the feedback functions, the state components, the next state components, and the input components of $\mathcal{A}_{\mathcal{D}_1}$. Then $\varphi_1((0,1,1,0),1,0,1,0) = (a_3 \oplus x_3, x_1) = (1 \oplus 1, 1) = (0,1)$, $\varphi_2((0,1,1,0),1,0,1,0) = (a_4 \oplus$

$x_4, x_2) = (0 \oplus 0, 0) = (0, 0)$, moreover $\delta(0, (0, 1)) = 1(= a'_1)$ and $\delta(1, (0, 0)) = 1(= a'_2)$, and thus $\varphi_3((0, 1, 1, 0), 1, 0, 1, 0) = (a'_1 \oplus x_1, x_3) = (1 \oplus 1, 1) = (0, 1)$, $\varphi_4((0, 1, 1, 0), 1, 0, 1, 0) = (a'_2 \oplus x_2, x_4) = (1 \oplus 0, 0) = (1, 0)$. Thus $\delta(1, (0, 1)) = 0(= a'_3)$ and $\delta(0, (1, 0)) = 1(= a'_4)$.

Next we denote by $\varphi_i, a_i, a'_i, x_i, i \in \{1, 2, 3, 4\}$ the feedback functions, the state components, the next state components, and the input components of $\mathcal{A}_{\mathcal{D}_2}$. Recall that (a_1, a_2, a_3, a_4) coincides with the new state of $\mathcal{A}_{\mathcal{D}_1}$. Then $(a_1, a_2, a_3, a_4) = (1, 1, 0, 1)$ and $(x_1, x_2, x_3, x_4) = (1, 0, 1, 0)$, where (x_1, x_2, x_3, x_4) consists of the last four components of the input $(1, 0, 1, 0, 1, 0, 1, 0)$ of the key automaton.

Then $\varphi_1((1, 1, 0, 1), 1, 0, 1, 0) = (a_2 \oplus x_2, x_1) = (1 \oplus 0, 1) = (1, 1)$, $\varphi_3((1, 1, 0, 1), 1, 0, 1, 0) = (a_4 \oplus x_4, x_3) = (1 \oplus 0, 1) = (1, 1)$, moreover $\delta(1, (1, 1)) = 1(= a'_1)$ and $\delta(0, (1, 1)) = 0(= a'_3))$, and thus $\varphi_2((1, 1, 0, 1), 1, 0, 1, 0) = (a'_1 \oplus x_1, x_2) = (1 \oplus 1, 0) = (0, 0)$, $\varphi_4((1, 1, 0, 1), 1, 0, 1, 0) = (a'_3 \oplus x_3, x_4) = (0 \oplus 1, 0) = (1, 0)$. Thus $\delta(1, (0, 0)) = 1(= a'_2)$ and $\delta(1, (1, 0)) = 0(= a'_4)$.

Hence the actual pseudorandom output is $(1, 1, 0, 0)$ which is also the next state.

5 Implementation

Next we give a detailed explanation of the enclosed pseudocode of our implementation. (See Algorithm ACBRNG.)

The procedure parameters are the number of random blocks ($SIZE$), the input word ($INPUT$) of the key automaton, the transition matrix of the basic automaton (AUT), and the initial (seed) state of the key automaton ($JSTATE$).

Each of the generated random blocks consists of 128 random strings and each of the random strings is 128 bits long. Thus, the size of the generated random blocks is 2048 Kbyte.

The key automaton is a four-component temporal product of automata which are (D_1, D_2, D_3, D_4)-powers of the basic automaton. The digraphs D_1, D_2, D_3, D_4 are defined by the matrix $P[4][16]$. Each of the D_1, D_2, D_3, D_4 powers consists of sixteen copies of the basic automaton which has 256 states and 256 input signs. Thus, the transition matrix AUT of the basic automaton is a 256×256-type matrix, where each state and input sign can be represented by a 8-bit binary string.

The connection digraphs D_1, D_2, D_3, D_4 are stored in the four consecutive row vectors of the 4×16-type connection matrix P.

We will consider $ROUND = 1, 2, 3$ rounds of the output function of $CBRNG$.

The four row vectors of the 4×16-type input matrix $INPUT$ represent four consecutive input signs of the four (D_1, D_2, D_3, D_4)-powers, the key automaton of which the temporal product consists.

Thus the matrix $INPUT$ represents a single input sign of the key automaton.

The main structure of the implementation is the following.

1. Read the number SIZE of the input block, the input word INPUT of the key automaton, the transition matrix AUT of the basic automaton, and the initial (seed) state JSTATE of the key automaton.
2. Store the initial (seed) state JSTATE in a working storage ISTATE.
3. Generate $SIZE$ number of random blocks as follows.
4. Consider the $ISTATE$ as a 128-bit length binary number and fput
$ISTATE = ISTATE + 1 \; mod \; 2^{128}$.
5. Repeat the following procedure $ROUND$-times. We could not pass the NIST test for $ROUND = 1$ and $ROUND = 2$ but we were successful for $ROUND = 3$.
6. Each of the $ROUND$ number of repetitions operates on the actual value of the actual key automaton state ($ISTATE$) by the consecutive element (the consecutive input sign) of the input word ($INPUT$) .
7. The operation of the states of (D_1, D_2, D_3, D_4) – products by the consecutive input sign (i.e., the consecutive column vector of the matrix $INPUT$) determined by the transition table (AUT) of the basic automaton and the digraphs D_1, D_2, D_3, D_4 defined by the matrix $P[4][16]$.
8. Collect the records of the pseudorandom block OARRAY.
9. Output the consecutive pseudorandom block OARRAY.

6 Experimental results

We implemented Algorithm ACBRNG in C++ in order to measure the actual running time and statistical properties of the generator. The test environment was a 2017 MacBook Pro equipped with 7th Generation Kaby Lake 2.9 GHz Intel Core i7 processor (7820HQ) using 16 GB RAM. We have generated 1 GB of random data and applied the NSIT SP-800-22 statistical randomness test.

6.1 NIST test

The National Institute of Standards and Technology (NIST) published a statistical package consisting of 15 statistical tests that were developed to test the randomness of arbitrarily long binary sequences produced by either hardware or software based cryptographic random or pseudorandom number generators [21]. In case of each statistical test a set of P-values was produced. Given a significance level α, if the P-value is less than or equal to α then the test suggests that the observed data is inconsistent with our null hypothesis, i.e. the 'hypothesis of randomness', so we reject it.

Table 1: Parameters used for NIST Test Suite

Test Name	**Block length**
Block Frequency	128
Non-overlapping Template	9
Overlapping Template	9
Approximate Entropy	10
Serial	16
Linear Complexity	500

We used $\alpha = 0.01$ as it is common in such problems in cryptography and PRNG testing. An α of 0.01 indicates that one would expect 1 sequence in 100 sequences to be rejected under the null hypothesis. Hence a P-value exceeding 0.01 would mean that the sequence would be considered to be random, and a P-value less than or equal to 0.01 would lead to the conclusion that the sequence is non-random. One of the most important characteristics of a PRNG is the indistinguishability from true random sources. That is, the evaluation of their output utilizing statistical tests should not provide any means by which to distinguish them computationally from a truly random source.

6.2 Minimum rounds to achieve randomness

ACBRNG has a cycle length of 2^{128}, however this does not yet mean that ACBRNG is really producing good quality random numbers. Consider the simple generator

$$k \mod 2^{128}, \quad (k \in 0 \dots 2^{128}) \tag{1}$$

If we start $k = 0$ and increment by 1 then the generator has a 2^{128} cycle length, however it is not random at all. ACBRNG has a more complex structure, but statistical tests were needed to check for possible weaknesses. In order to test the quality of ACBRNG the NIST statistical tests were performed using the same parameters as for the AES candidates in order to achieve the most reliable and comparable results. First the input parameters - such as the sequence length, sample size, and significance level - were fixed. Namely, these parameters were set to 2^{20} bits, 300 binary sequences, and $\alpha = 0.01$, respectively. Exact parameters can be seen in Table 1.

One Round ACBRNG We started the ACBRNG with a low entropy random seed. The running time was 4.5 sec using 8 parallel threads. Applying only one round ($ROUND = 1$ in line 25) ACBRNG did not pass the NIST requirements. More precisely, we have failed in almost every statistical test using one round. So one can conclude that only one round is not complex enough, and further investigation was needed. We would like to note, that surprisingly all Random Excursions tests from NIST were passed after one round.

Two Rounds ACBRNG Using $ROUND = 2$ surprisingly almost every statistical test was passed. The running time was 8.4 sec using 8 parallel threads. Only two non-overlapping templates were unsatisfied, which is quite a good achievement after two rounds. We did not expect such good quality random numbers and p-value distribution after two rounds. One can conclude that using only two rounds is enough to reach good quality random numbers and pass the NIST test.

Three Rounds ACBRNG After only three rounds ACBRNG did pass every requirement of the NIST statistical test suite. It has turned out that the output of the algorithm (using $ROUND = 3$) can not be distinguished in polynomial time from true random sources using the NIST statistical test. The running time was 12.25 sec using 8 parallel threads. The exact p-values of the evaluation of the ACBRNG for $ROUND = 3$ are shown in Table 2. We also tested the uniformity of the distribution of the p-values obtained by the statistical tests included in NIST. The uniformity of p-values provide no additional information about the PRNG. We have also shown that the proportions of binary sequences which passed the 0.01 level lie in the required confidence interval (see e.g. [21]).

7 Conclusion and further research

In this paper a full cycle length pseudorandom number generator (ACBRNG) based on compositions of automata was presented. We have seen that it produces promising results in terms of statistical randomness and passed all statistical tests included in the NIST test suite. We can see that the running time of the generator is efficient enough for practical use. In order to consider this generator cryptographically secure, formal verification of its security would be necessary which is a direction that might be worth investigating.

References

[1] Bao, F.: Cryptoanalysis of a partially known cellular automata cryptosystem. IEEE Trans. on Computers, 53 (2004), 1493-1497; `https://doi.org/10.1109/TC.2004.94`

[2] Bilan, S., Bilan, M., Bilan, S: Novel pseudorandom sequence of numbers generator based cellular automata. Information Technology and Security, Vol. 3(1), 2015, pp. 38-50. `https://doi.org/10.20535/2411-1031.2015.3.1.57710`

[3] Bhattacharjee, K., Das, S., Paul, D.: Pseudo-random Number Generation using a 3-state Cellu lar Automaton. Internat. J. of Modern Physics, vol 28, No 6,

Table 2: Results for the uniformity of p-values and the proportion of passing sequences

C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**P-value**	**Prop**	**Test name**
24	34	30	31	25	30	26	33	30	37	0.828458	297/300	`Frequency`
34	29	32	27	21	26	20	35	31	45	0.068287	296/300	`BlockFrequency`
29	32	36	31	31	24	31	27	29	30	0.964295	298/300	`CumulativeSums`
26	30	31	30	33	27	24	38	28	33	0.840081	295/300	`CumulativeSums`
32	22	25	35	36	25	32	30	41	22	0.198690	300/300	`Runs`
34	34	34	31	37	23	33	23	29	22	0.437274	298/300	`LongestRun`
22	26	23	35	35	32	34	39	29	25	0.334538	296/300	`Rank`
33	28	31	29	33	30	34	31	25	26	0.973936	296/300	`FFT`
30	30	21	35	40	29	29	28	25	33	0.514124	296/300	`NonOverLappingTemp`
43	30	24	29	34	27	32	22	31	28	0.339799	296/300	`NonOverLappingTemp`
.	.	.	.	.	.	.	.	.	.	.	.	`NonOverLappingTemp`
...	...	...	...	...	...	...	...	...	...	...	...	`NonOverLappingTemp`
22	21	30	44	38	32	23	31	35	24	0.043745	295/300	`OverlappingTemp`
32	44	35	29	28	26	20	23	34	29	0.132132	298/300	`Universal`
28	27	18	36	40	26	33	37	23	32	0.122325	297/300	`ApproximateEntropy`
17	15	23	20	20	21	16	27	17	18	0.707944	194/194	`RandomExcursions`
22	20	21	13	25	16	19	17	21	20	0.791218	193/194	`RandomExcursions`
.	.	.	.	.	.	.	.	.	.	.	.	`RandomExcursions`
...	...	...	...	...	...	...	...	...	...	...	...	`RandomExcursions`
22	17	15	16	23	23	20	21	23	14	0.729339	192/194	`RandomExcursionsV`
18	18	17	19	23	27	19	17	17	19	0.838872	193/194	`RandomExcursionsV`
.	.	.	.	.	.	.	.	.	.	.	.	`RandomExcursionsV`
...	...	...	...	...	...	...	...	...	...	...	...	`RandomExcursionsV`
31	28	27	22	27	39	26	36	35	29	0.514124	296/300	`Serial`
30	32	22	36	32	22	33	30	35	28	0.637119	294/300	`Serial`
32	28	31	26	34	24	32	37	36	20	0.449672	296/300	`LinearComplexity`

Algorithm ACBRNG

```
procedure ACBRNG(SIZE, INPUT, AUT, JSTATE)                 ▷ 1. Read the input parameters.
    for $i = 0 \to 15$ do                                   ▷ 2. Put the seed into working storage.
        $ISTATE[i] \leftarrow JSTATE[i]$                    ▷ 3. JSTATE is the initial (seed) state of the key automaton.
    end for
    for $kk = 0 \to SIZE$ do                                ▷ 4. SIZE number of pseudorandom blocks are generated
        for $m = 0 \to 127$ do
            $x \leftarrow 0$
            for $j = 15 \to 0$ do
                if $x = 0$ then
                    if $ISTATE[j] = 255$ then
                        $ISTATE[j] \leftarrow 0$
                    else
                        $ISTATE[j] \leftarrow ISTATE[j] + 1$
                        $x \leftarrow 1$
                    end if
                end if
                $STATE[j] \leftarrow ISTATE[j]$
            end for
            for $f = 0 \to ROUND$ do                        ▷ 5. Passes NIST test with $ROUND = 3$.
                for $i = 0 \to 3$ do                        ▷ 6. Key automaton state transition.
                    for $j = 0 \to 15$ by 2 do              ▷ 7. $D_1, D_2, D_3, D_4$-power state transitions.
                        $k \leftarrow P[i][j]$
                        $l \leftarrow P[i][j+1]$
                        $a_1 \leftarrow STATE[k]$
                        $a_2 \leftarrow STATE[l] \oplus INPUT[l][i]$
                        $STATE[k] \leftarrow AUT[a_1][a_2]$
                        $a_1 \leftarrow STATE[l]$
                        $a_2 \leftarrow STATE[k] \oplus INPUT[k][i]$
                        $STATE[l] \leftarrow AUT[a_1][a_2]$
                    end for
                end for
            end for
            for $i = 0 \to 15$ do                           ▷ 8. Collect the records of the pseudorandom block OARRAY
                $OARRAY[m][i] \leftarrow STATE[i]$
            end for
        end for
        PRINT($\&OARRAY$)                                   ▷ 9. Print the next random block.
    end for
end procedure
```

ppp. 1-23, 2017, https://doi.org/10.1142/S0129183117500784

[4] Chakraborty,K. and . Chowdhury, D. R. : CSHR: Selection of cryptographically suitable hybrid cellular automata rule. International Conference on Cellular Automata for Research and Industry, ACRI, Springer, pp. 591-600, 2012. https://doi.org/10.1007/978-3-642-33350-7_61

[5] Dogaru, R. and Dogaru,I.: FPGA implementation and evaluation of two cryptographically secure hybrid cellular automata. Proc. Communications (COMM) 2014, 10th International Conference on Communications, pp. 1-4, 2014. https://doi.org/10.1109/ICComm.2014.6866740

[6] Dömösi, P., Gáll, J., Horváth, G., Tihanyi, N. Some Remarks and Tests on the Dh1 Cryptosystem Based on Automata Compositions Informatica (Slovenia), vol 43, 2 (2019), pp. 199-207. https://doi.org/10.31449/inf.v43i2.2687

[7] Dömösi, P. and Nehaniv, C. L. Algebraic theory of automata networks. An introduction. SIAM Monographs on Discrete Mathematics and Applications, 11. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2005. https://doi.org/10.1137/1.9780898718492

[8] Guan, S. U. and Tan, S. K.: Pseudorandom number generation with self-programmable cellular automata. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 23, pp. 1095-1101, 2004. https://doi.org/10.1109/TCAD.2004.829808

[9] Guan, S.-U. and Zhang, S.: Pseudorandom number generation based on controllable cellular automata. Future Generation Computer Systems, vol. 20, pp. 627-641, 2004. https://doi.org/10.1016/S0167-739X(03)00128-6

[10] Guan, P.: Cellular automaton public key cryptosystem. Complex Systems, 1 (1987), 51-56].

[11] Hoe, D. H. K., Comer, J. M., Cerda, J. C., Martinez, C. D., Shirvaikar,M. V.: Cellular Automata-Based Parallel Random Number Generators Using FPGAs. International Journal of Reconfigurable Computing, Vol. 2012, 2012, pp. 1-13, Article ID 219028 https://doi.org/10.1155/2012/219028

[12] Hortensius, P. D., Mcleod, R. D., Pries, W.,Miller, D M., and Card, H C.: Cellular Automata- Based Pseudorandom Number Generators for Built-In self-Test, IEEE transactions on Computer-Aided Design, vol. 8, pp 842-859,1989. https://doi.org/10.1109/43.31545

[13] Kang, B., Lee, D., and Hong, C.: High-Performance Pseudorandom Number Generator Using Two-Dimensional Cellular Automata. 4th IEEE International Symposium on Electronic Design, Test and Applications (delta 2008), Hong Kong, 2008, pp. 597-602. https://doi.org/10.1109/DELTA.2008.46

[14] Kar, M., Rao, D. C., Rath, A. K.: Generating PNS for Secret Key Cryptography Using Cellular Automaton. International Journal of Advanced Computer Science and Applications, Vol. 2, No. 5, 2011, pp. 101-105. https://doi.org/10.14569/IJACSA.2011.020517

[15] Madghuri, A.: Hybrid Cellular Automata-Based Pseudo Random Sequence Generator for BIST Implementation. International Journal of Research Studies in Science, Engineering and Technology Volume 2, Issue 9, September 2015, pp. 72-76 ISSN 2349-4751 (Print) & ISSN 2349-476X (Online)

[16] Martin, B., Sole, P.: Pseudo-random Sequences Generated by Cellular Automata. International Conference on Ralations, Orders and Graphs: Interaction with Computer Scince, May 2008, Mandia, Tunisia, Nouha editions, 2008, pp. 401-410.

[17] Meier, W. and Staffelbach, O.: Analysis of pseudo random sequences generated by cellular automata. In: Davies, D. W. (ed.), Proc. Conf. Advances in Cryptology – EUROCRYPT '91, Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, LNCS 547 Springer-Verlag, Berlin, 1991, 186-199 https://doi.org/10.1007/3-540-46416-6

[18] Menezes, A., van Oorschot,P., and Vanstone, S.: Handbook of Applied Cryptography, CRC Press, 1996.

[19] Ping,P., Xu, F., and Wang, X.-J.: Generating high-quality random numbers by next nearest-neighbor cellular automata. Advanced material Research, vol. 765, pp. 1200-1204, 2013. https://doi.org/10.4028/www.scientific.net/AMR.765-767.1200

[20]] Ruboi, C. F., Encinas, L. H., White, S. H., del Rey, A. M., Sancher, R.: The use of Linear Hybrid Cellular Automata as Pseudorandom bit Generators in Cryptography. Neural Parallel & Scientific Comp. 12(2), 2004, pp. 175-192. http://hdl.handle.net/10261/21253

[21] Rukhin, A., Soto, J., Nechvatal, J.,Smid, M., Barker, E., Leigh, S., Levenson, M.,Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S.: NIST Special Publication 800-22 (2010).: A Statistical Test Suite for Random and Pseudo Random

Number Generators for Cryptographic Applications. *National Institute of Standards and Technology*, https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf, downloaded in March 2020. `https://doi.org/10.6028/NIST.SP.800-22r1a`

[22] Salmon, J., Moares, M., Dror, R., Shaw, D. Parallel random numbers: as easy as $1, 2, 3$. Proc. 2011 Intern. Conf. for High Performance Computing, Networking, Storage and Analysis, Article No. 16. doi:10.1145/2063384.2063405. `https://doi.org/10.1145/2063384.2063405`

[23] Seredynski, F., Bouvry, P., and Zomaya A. Y.: Cellular automata computations and secret key cryptography. Parallel Computing, vol. 30, pp. 753-766, 2004. `https://doi.org/10.1016/j.parco.2003.12.014`

[24] Shin,SH., Kim,DS., Yoo, KY. : A 2-Dimensional Cellular Automata Pseudorandom Number Generator with Non-linear Neighborhood Relationship. In: Benlamri R. (eds) Networked Digital Technologies. NDT 2012. Communications in Computer and Information Science, vol 293. Springer, Berlin, Heidelberg. `https://doi.org/10.1007/978-3-642-30507-8_31`

[25] Sipper, M., Tomassini, M. Generating parallel random number generators by cellular programming. International Journal of Modern Physics C. 7 (2), pp. 181–190, 1996. `https://doi.org/10.1142/S012918319600017X`

[26] Sukhinin, B.M.: High-speed pseudorandom sequence generators based on cellular automata. Applied discrete mathematics, No 2, 2010, pp. 34 – 41. `https://doi.org/10.17223/20710410/8/5`

[27] Sukhinin B.M.: Development of generators of pseudorandom binary sequences based on cellular automata. Science and education, No 9, 2010, pp. 1 – 21.

[28] Tomassini, M., Sipper, M., and Perrenoud, M.: On the Generation of High-Quality Random Numbers by Two-Dimensional Cellular Automata, IEEE Transactions on Computers, vol. 49, pp. 1146-1151, 2000. `https://doi.org/10.1109/12.888056`

[29] Wolfram, S.,: Cryptography with Cellular Automata. In: C. W. Hugh, ed., Proc. Conf. Advances in Cryptology—CRYPTO'85, Santa Barbara, Calif., USA, Aug. 18-22, 1985, LNCS 218, Springer-Verlag, Berlin, 1986, pp. 429-432. `https://doi.org/10.1007/3-540-39799-X_32`

[30] Wolfram, S.: Random Sequence Generation by Cellular Automata. Advances in Appl. Math., vol. 7, 1986, pp. 429-432. `https://doi.org/10.1016/0196-8858(86)90028-X`

Effects of Pooling in ParallelGlobal with Low Thread Interactions

Dániel Zombori and Balázs Bánhelyi
Department of Computational Optimization, University of Szeged

Keywords: black box, stochastic, global optimization, parallel computing

Received: March 31, 2020

The first step toward a new version of Global is discussed. It is a fully distributed algorithm. While the proposed implementation runs on a single machine, gossip based information sharing can be built into and be utilized by the algorithm. ParallelGlobal shows a feasible way to implement Global on a distributed system. Further improvements must be made to solve big real world problems with the algorithm.

Povzetek: Predstavljena je nova verzija algoritma Global z imenom ParallelGlobal.

1 Introduction

Global is an optimization algorithm built from multiple modules working in an ensemble. While older implementations viewed the algorithm as a whole, the most recent GlobalJ framework handles algorithms as a collection of interlocking modules. GlobalJ has several implementations for local search algorithms and variants of Global. Main characteristics of the single threaded version were established in [4]. In recent years Global was further developed [6] and it has several applications [5, 7] where it aids mostly other research works. To speed up optimization processes we developed an algorithm [1] that is capable of utilizing multiple computational threads of a single machine. It cannot be directly implemented for distributed systems as the millisecond order of magnitude latency in communication would significantly slow down the synchronization of threads. To mitigate this problem we propose ParallelGlobal, a parallel implementation suitable for distributed systems with high latency or even with unreliable communication channels. In this paper we introduce an experimental version whose main purpose is to test the feasibility of the proposed solution. It provides an algorithm skeleton for a real distributed implementation.

2 Global

Global is a global optimizer designed to solve black box unconstrained optimization problems with a low number of function evaluations and probabilistic guarantees [1, 2, 3, 4, 6, 8, 9, 11]. It uses local search algorithms to refine multiple sample points hence Global is a multi-start method. Global also utilizes the *Single Linkage Clustering* algorithm to make an estimation about the values of samples from the aspect of optimization.

2.1 Updated global algorithm

While the updated Global algorithm has only minor changes and in a lot of cases performs equally to the original, it is superior in execution order, therefore we consider it as the basis for improvements.

Global has an iterative framework where samples in an iteration compete with samples of previous iterations. The original version contains four phases in every iteration consisting of sampling, reduction, clustering, and local search. In the updated algorithm the clustering and local search phases are merged by an implementation which alternates them.

Algorithm 1 GLOBAL

```
 1: while $termination\text{-}criteria()$ is not $true$ do
 2:     $S \leftarrow S \cup \{uniform(lb, ub) : i \in [1, new\ samples]\}$
 3:     $S \leftarrow sort(F(s_i) < F(s_{i+1})), s_i \in S$
 4:     $R \leftarrow \{s_i : i \in [1, reduced\ set\ size]\}$
 5:     $S \leftarrow S \setminus R$
 6:     while $R$ is not $\emptyset$ do
 7:         for $C$ in $clusters$ do
 8:             $d_c \leftarrow \left(1 - \alpha^{\frac{1}{|clustered|+|R|-1}}\right)^{\frac{1}{dim(F)}}$
 9:             $N \leftarrow \left\{r_i : d_c > \|r_i - c_j\|_\infty \wedge F(r_i) > F(c_j)\right\}$
10:             if $N$ is not $\emptyset$ then
11:                 $C \leftarrow C \cup N$
12:                 $R \leftarrow R \setminus N$
13:                 repeat iteration
14:             end if
15:         end for
16:         $l \leftarrow local\text{-}search(r_1 \in R)$
17:         $C_l, d_{min} \leftarrow \underset{C \in clusters}{argmin} \left\| l - \underset{c_i \in C,}{argmin}\, F(c_i) \right\|_\infty$
18:         if $d_{min} < d_c/10$ then
19:             $C_l \leftarrow C_l \cup \{l, r_1\}$
20:         else
21:             $clusters \leftarrow clusters \cup \{\{l, r_1\}\}$
22:         end if
23:         $R \leftarrow R \setminus \{r_1\}$
24:     end while
25: end while
```

Algorithm 1 describes the updated Global in detail. In lines 2-5 the algorithm performs the sampling phase. Selection of sample points is random, using uniform distribution in the search space. The generated samples are placed in container S which has a list structure. To find the most promising samples, S is sorted and a reduced set of samples is acquired with the lowest function values. R contains the reduced set, which is removed from S.

When samples are ready to be processed, in lines 6-24 the algorithm alternates between clustering and local searches while there are unprocessed samples left. At lines 7-15 samples in R are tried against the clustered samples. To determine if $r_i \in R$ is part of cluster C we need the distance threshold d_c. This depends on the dimension of the objective function, the number of samples currently known in the clustering process, and the $\alpha \in [0, 1]$ parameter. The latter controls the decrease speed of d_c while more samples are added, in order to adapt to the expected decrease in distance between two random samples. With d_c set, sample pairs $(r_i \in R, c_j \in C)$ are evaluated to determine if r_i is part of C. The two criteria are having a clustered sample c_j with function value lower than r_i and it being closer with the infinity norm (Manhattan distance) than d_c. Samples in R satisfying both of them are moved to the current cluster C. When a point is clustered, all samples in R can potentially be clustered too, therefore $r_i \in R$ is rechecked against C. After the *for* cycle finished, samples in R cannot be the part of an existing cluster, therefore performing a local search is inevitable.

Local searches are performed in lines 16-23, where l is the local optimum reached from r_1. To determine if l is a newly found local optimum, a comparison with the cluster centers is needed. The center of a cluster is the sample in the cluster with the lowest function value. By finding the cluster with the closest center the algorithm can decide whether the optimum is already found. If the distance d_{min} to the cluster C_l with the closest center is lower than a tenth of the d_c threshold, it is considered to be the same local optimum. In this case l and r_1 are added to C_l, otherwise they form a new cluster. Since r_1 is either in an already existing cluster or in a newly created one, we can remove it from R. The *while* loop in lines 6-24 repeats until R becomes empty. With no unclustered samples left Global finished an iteration. The number of executed iterations is limited by the termination criteria.

3 ParallelGlobal

Our goal is to derive an implementation from the updated Global which is multi-threaded with few interactions between threads. The necessity for few thread interactions comes from the fact that on huge scale optimization tasks a single computer is not sufficient and in multi-computer environments the communication between machines is relatively slow compared to inter-thread communication. We address this problem by removing the synchronization of computational threads and replacing it with a message based information sharing scheme.

We can view ParallelGlobal as a naive parallelization of Global. The main idea lies in the parallel execution of Global iterations, while sharing information between computational threads. Consequently, inter-thread communication is necessary, however only a few selected data containers have to be shared. Also, the shared containers have independent data points and no deletions, therefore inconsistencies cannot arise from data insertions. These considerations make the algorithm for distributed systems viable. Luckily information gathering techniques of Global can be maintained with messaging, therefore results almost always apply to ParallelGlobal as well.

3.1 ParallelGlobal worker

Algorithm 2 describes the ParallelGlobal worker which is the implementation of a single computational thread. The worker might run on a machine by itself, or multiple workers can use the multi-threaded environment of a computer.

Algorithm 2 ParallelGlobal

```
 1: while $termination\text{-}criteria()$ is not $true$ do
 2:     $exchange\text{-}data()$
 3:     $S \leftarrow uniform(lb, ub)$
 4:     $R \leftarrow reduce\,(S)$
 5:     $d_c \leftarrow \left(1 - \alpha^{\frac{1}{|clustered|+1-1}}\right)^{\frac{1}{dim(F)}}$
 6:     for $C$ in $clusters$ do
 7:         $N \leftarrow \{r_i : d_c > \|r_i - c_j\|_\infty \wedge F(r_i) > F(c_j)\}$
 8:         $C \leftarrow C \cup N$
 9:         $R \leftarrow R \setminus N$
10:     end for
11:     $r_{min} \leftarrow \underset{r_i \in R}{argmin}\, F\,(r_i)$
12:     $l \leftarrow local\text{-}search(r_{min})$
13:     $C_l, d_{min} \leftarrow \underset{C \in clusters}{argmin} \left\| l - \underset{c_i \in C}{argmin}\, F(c_i) \right\|_\infty$
14:     if $d_{min} < d_c/10$ then
15:         $C_l \leftarrow C_l \cup \{l, r_{min}\}$
16:     else
17:         $clusters \leftarrow clusters \cup \{\{l, r_{min}\}\}$
18:     end if
19: end while
```

Similarly to Global, ParallelGlobal also runs in a loop to complete iterations until a termination criterion is met. Unlike Global, the new algorithm needs a data exchange step (line 2). At the start of every iteration, received messages can be processed and new messages can be sent according to a suitable policy. The messages contain evaluated data points arranged into clusters. These clusters can be handled as if they were evaluated locally by clustering the center point (minimizer point) of the cluster. If the center point corresponds to an existing cluster, the two clusters should be merged while duplicate points are filtered out. Otherwise, the received cluster describes a previously unknown local optimum and it can be added to the existing clusters without modifications.

Lines 3 and 4 perform sampling and reduction. In previous Global versions sampling and reduction was performed by taking a randomized sample set, then using a sorted sample pool and taking the best samples out. ParallelGlobal cannot utilize efficiently a fully synchronized common pool due to the distributed nature of the system. In this version we envisioned simple solutions providing a similar effect to a shared pool. Also with more complicated solutions closer approximates are possible. The evaluated implementations are discussed in Subsection 3.2.

In lines 5-10 the clustering occurs. It is very similar to the original clustering algorithm. The only change is that we know that no more than one sample is in R. This is also true for the local search (lines 12-18) which is identical with the local search part of the original.

3.2 Current implementation

The current implementation of ParallelGlobal only simulates the described functionality with some simplification. First, it runs on a single machine with multiple threads as a single program. Second, messaging is simulated by synchronization on the given containers while they are written, but reading operations happen simultaneously. During clustering, the cluster list is only read to a point determined before the process starts, hence new clusters will be excluded from already started searches. This also resembles the effects of messaging, like delays and losses in information spread. Because no real messaging is present, the $exchange\text{-}data()$ function is only a placeholder for now.

To evaluate the differences between Global and ParallelGlobal in more depth, we implemented the latter with two kinds of $reduce()$ functions. The naive pool-less implementation ignores the $reduce()$ function, in every iteration it creates a sample that is evaluated with clustering, and then (if not clustered) with local search. The 'pooled' implementation simulates the sample pool and reduction by generating a local pool. Along with Global, we use the notion of the sample reducing factor. Global generates a number of samples and takes a fraction of that number from the common pool. Pooled ParallelGlobal creates samples such that one sample has to be picked according to the fractional reduction.

Beyond these pooling and sample reduction strategies much more exist, for example taking a single sample every iteration and using random sample reduction, possibly aided with spatial measures on the samples information value. A more complex but possible solution would be a distributed sample pool. Samples could be transferred between local pools over reliable data connection. This would ensure that a sample is only evaluated by a single worker and would create a bigger variety of samples to choose from.

4 Results

The algorithms were examined from two aspects; comparison in the number of function evaluations and scaling of run time with additional threads. Numerical results were obtained on the following functions, definitions can be found

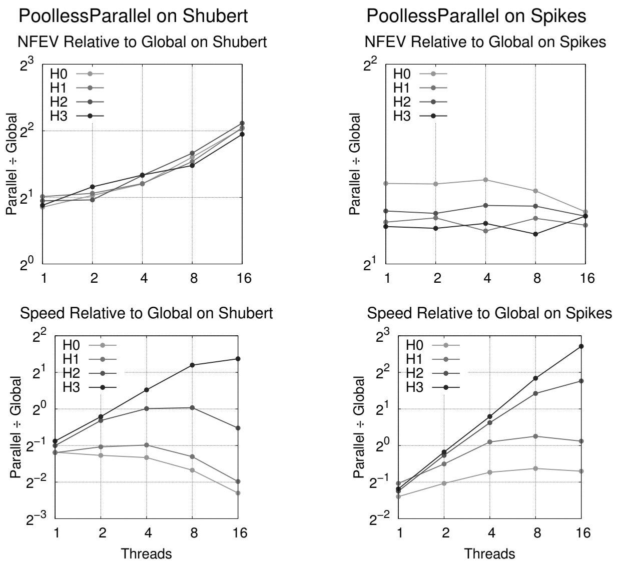


Figure 1: Numeric results of the poolless version on the Shubert (left) and Spikes (right) functions.

in [10]. *Ackley*, *Discus*, *Easom*, *Griewank*, *Levy*, *Rastrigin*, *Schaffer*, *Schwefel*, *Shekel-5*, *Shekel-7*, *Shekel-10*, *Shubert*, *Spikes*[1], and *Zakharov*. For the evaluations we used two termination criteria, the maximum number of function evaluations is 10^5 which is a soft condition therefore overshoot is possible. To check whether an optimum point is reached we use the expression

$$|F(x^*) - F(x)| < 10^{-8} + |F(x^*)| \cdot 10^{-6},$$

where x^* is a known global optimum point and x is the point in question. To emulate computationally more expensive functions we defined the hardness level. A hardness level of H means that the function will be evaluated for 10^H microseconds at the requested point. Please note that microsecond level timings can be inaccurate and only have effect in trends, while at the millisecond scale the timings show the real 10x factor.

Global is a stochastic optimizer, moreover ParallelGlobal is also affected by the operating system's thread scheduling, consequently run times and the number of function evaluations can differ largely from one optimization process to the other. To reduce the noise induced by this, we obtained data points by averaging the results of 100 runs with every configuration. The algorithm parameterizations were identical except for the number of threads.

On the left side of Figure 1 we show results for Poolless ParallelGlobal on the *Shubert* test function, namely the number of function evaluations (NFEV) and the speed of the optimization process ($1/runtime$), both relative to Global. On the horizontal axes we see the number of

[1]Spikes function definition:

$$f(x) = \begin{cases} 1000, & \text{if } \|x - (15.25, 15.75)\|_2 \leq \frac{1}{4} \\ 1002 + \Pi_{x_i} sin(2\pi x_i), & \text{otherwise} \end{cases}$$

threads. The vertical axes show the number of function evaluations and optimization processes run in unit time respectively, both divided by the result of Global on a single core. H0 to H3 denotes the hardness value for the given data series. *Shubert* is a function with many local optima and a flat global trend. In case of Global, NFEV is mostly in the $[500, 2000]$ range with an average of 900. On the top-left graph relative NFEV shows that we have an increase with a factor of two. On a single thread the multiplier of $2^1 = 2$ shows that the algorithm is by itself inferior to Global. This static multiplier is explained by the lack of a sample pool which could reduce the necessary number of local searches. They give the bulk of the NFEV and while Global uses 5 local searches on average ParallelGlobal needs much more, around 18. The dynamic growth is also explained by the local searches, combined with multi-threading. Finding the global optimum with local search takes several function evaluations in sequence. Since multiple threads start local searches independently, more evaluations can happen until one of them reaches the global optimum. Moreover when the optimum is found, the program does not terminate immediately, all local searches have to finish. This phenomenon increases the NFEV due to the intrinsic usage of multi-threading and local searches.

The bottom-left subgraph of Figure 1 shows the speedup with additional threads and different hardness values. While for H0 and H1 the additional threads caused a slowdown due to synchronization time and increased NFEV, on computationally more demanding versions we achieved a significant speedup. The results are promising because for the hardness value of 3 on a single thread a function evaluation took only $1ms$ on average. With higher evaluation times, the additional computational power would have more effect.

On the right side of Figure 1, we show the results for the *Spikes* test function which also has many local optima and a flat global trend. Poolless ParallelGlobal suffers from the lack of a sample pool on the *Spikes* function too. On the other hand, no dynamic change in NFEV is experienced. Without a sample pool ParallelGlobal had a much harder time finding the global optimum, which would often exceed the 10^5 NFEV limit. This resulted in close to constant NFEV and saturation of threads only on low hardness values. Based on the relative speed graph, we gain speed linearly with additional CPU power on higher hardness levels. Since the function is very cheap to evaluate and ParallelGlobal has to do much more evaluations, only H2 and H3 gives an advantage to the multi-threaded implementation.

On the left side of Figure 2 we show results for Pooled ParallelGlobal on the *Shubert* and *Spikes* test functions. As before the NFEV and execution speed is examined, relative to the same data on Global.

Pooled ParallelGlobal halved the NFEV on one thread, then grew to 16 times the single thread value. This phenomenon is independent of hardness, and hence curves closely match on the top-left subgraph. The single thread NFEV is halved because Global takes a set of 400 samples in every iteration and then uses another 400 samples to find the optimum summing to 800 NFEV. Pooled ParallelGlobal takes only 27 samples per iteration and also needs around 400 samples to find the optimum summing close to 400. The NFEV almost exactly grows with the number of threads, pointing to the failure of job parallelization in a way when effectively the same job is done by every thread. This is supported by the fact, that for this configuration the number of local searches matches the number of threads, and hence no parallelization is possible. This happens on functions where the optimum is found by a single local search, or less local searches than threads.

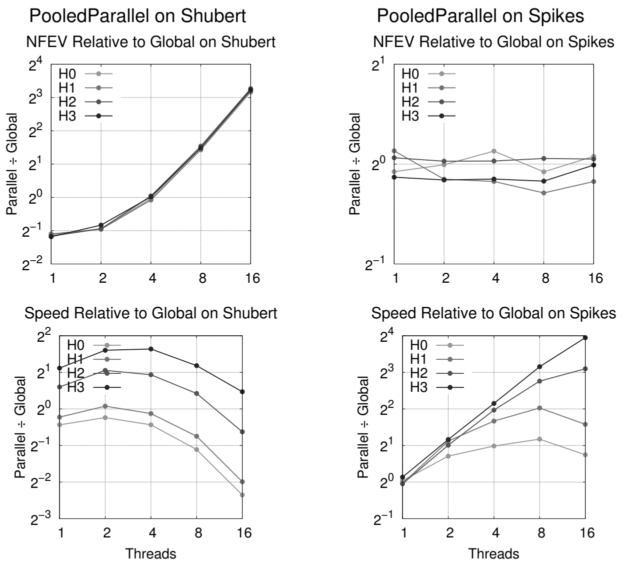


Figure 2: Numeric results of the pooled version on the Shubert (left) and Spikes (right) functions.

As the left bottom graph of Figure 2 shows, the speed gain with additional threads in case of *Shubert* is countered by the NFEV and further decreased by the thread synchronization. An interesting feature is the higher relative speed with higher hardness levels, regardless of thread count. This is caused by the limit of the evaluation timing when applying hardness. Timing of low complexity function evaluations can be very inaccurate and constant overheads in execution exist, causing a non-linear relationship between NFEV and runtime. This effect evens out lower hardnesses and smaller NFEVs more. With a T1H0 configuration (1 thread, 0 hardness) we have an execution of 110 FEVs and 27 ms runtime. The same setup also executed with 1186 FEVs and 59 ms runtime which is 10 times the evaluations but only twice the runtime. This effect is almost negligible with H3 configurations. Unlike the Poolless version, the execution speed decreases with added threads, even with higher hardnesses. This is also caused by multiple factors interacting. The Poolless version on a single thread uses on average 16 local searches which means that 16 simultaneous local searches will have almost the same effect, pushing up the number of local searches with additional threads slightly. The poolless version uses

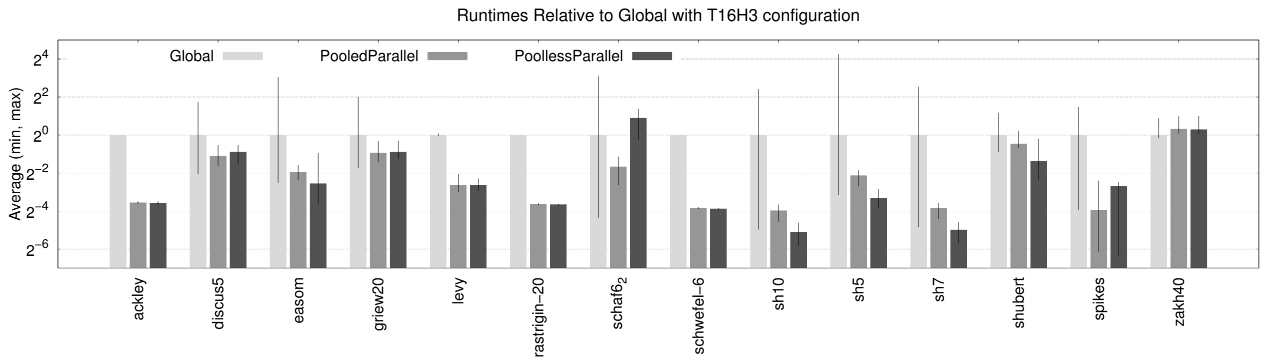


Figure 3: Relative runtimes on all test functions with 16 threads and hardness 3.

around 3000 FEVs with 16 threads that are almost entirely used in local searches. On the other hand the pooled version uses only 3 local searches on a single thread and over 3 it closely matches the number of threads. On 16 threads 1500 FEVs in 16 local searches are needed but 8000 FEVs are necessary in total. This shows that the addition of the sample pool both helped and worsened the optimizer for this particular type of function.

The top right graph of Figure 2 shows what we stated above; the pooled implementation erased the 2^1 multiplier in NFEV, it evened the Global and ParallelGlobal algorithms. Similar to the Poolless implementation we have close to linear speedup with additional threads with hardness 3 and significant speedups on lower hardness values. On this function optimizers need a lot of local searches, therefore ParallelGlobal managed to accelerate the process.

We deliberately chose *Shubert* and *Spikes* to showcase a good and bad scenario for the single thread and parallel optimizers as well. *Shubert* can be solved with few longer local searches, while *Spikes* needing only random sampling is the exact opposite. Figure 3 shows the overall picture on all of the tested functions.

On Figure 3 we show relative runtimes for the configuration of 16 threads and hardness 3 (T16H3) on every test function. Since the plot is logarithmic, 2^0 and values below mean similar and better results compared to Global. Error bars show the minimum and maximum data points from the averaged 100. On the functions which experienced slowdown either the lack of a sample pool or the intrinsic properties of ParallelGlobal prevented gains in speed. More than 50% of the functions with speedup were solved successfully, in these cases the NFEV limit had no effect.

5 Conclusion

We came to multiple important conclusions about the ParallelGlobal algorithm. The most needed change is the implementation of a distributed sample pool with sample sharing between threads. Having a set of probe points in the search space ensures that local searches only start from promising regions. This change moved the algorithm much closer to the NFEV values of Global.

Many of our results show slowdown with ParallelGlobal, but huge improvements as hardness values increase, as we have seen on the *Spikes* function. To keep our run times manageable we kept the hardness value relatively low. By going up from the current millisecond order to the second or 10 second order in function evaluations we would have a clearer image on how much speedup can we achieve. This would still undershoot the evaluation time of many practical problems, however it would be sufficient for proper testing on distributed systems.

To achieve these improvements, first the addition of a distributed framework is needed. Both the sharing of probe samples and cluster information would rely on it. It is also a key for testing on computationally expensive problems.

Acknowledgments

This work was supported by the Hungarian Government under the grant number EFOP-3.6.1-16-2016-00008. The project has been supported by the European Union, co-funded by the European Social Fund, and by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

References

[1] B. Bánhelyi, T. Csendes, B. Lévai, L. Pál, and D. Zombori. *The GLOBAL Optimization Algorithm*. Springer, 2018. `https://doi.org/10.1007/978-3-030-02375-1`.

[2] B. Betró and F. Schoen. Optimal and sub-optimal stopping rules for the multistart algorithm in global optimization. *Mathematical Programming*, 57:445–458, 1992. `https://doi.org/10.1007/BF01581094`.

[3] C. G. E. Boender and A. H. G. Rinnooy Kan. On when to stop sampling for the maximum. *Journal of Global Optimization*, 1:331–340, 1991. `https://doi.org/10.1007/BF00130829`.

[4] C. G. E. Boender, A. H. G. Rinnooy Kan, G. Timmer, and L. Stougie. A stochastic method for global optimization. *Mathematical Programming*, 22:125–140, 1982. https://doi.org/10.1007/BF01581033.

[5] T. Csendes, B. M. Garay, and B. Bánhelyi. A verified optimization technique to locate chaotic regions of Hénon systems. *Journal of Global Optimization*, 35:145–160, 2006. https://doi.org/10.1007/s10898-005-1509-9.

[6] T. Csendes, L. Pál, J. Sendin, and J. Banga. The global optimization method revisited. *Optimization Letters*, 2:445–454, 2008. https://doi.org/10.1007/s11590-007-0072-3.

[7] A. Szenes, B. Bánhelyi, L. Zs. Szabó, G. Szabó, T. Csendes, and M. Csete. Improved emission of SiV diamond color centers embedded into concave plasmonic core-shell nanoresonators. *Scientific Reports*, 7:an:13845, 2017. https://doi.org/10.1038/s41598-017-14227-w.

[8] I. Lagaris and I. Tsoulos. Stopping rules for box-constrained stochastic global optimization. *In Applied Mathematics and Computation*, 197:622–632, 2008. https://doi.org/10.1016/j.amc.2007.08.001.

[9] J. Sendín, J. Banga, and T. Csendes. Extensions of a multistart clustering algorithm for constrained global optimization problems. *Industrial & Engineering Chemistry Research*, 48:3014–3023, 2009. https://doi.org/10.1021/ie800319m.

[10] S. Surjanovic and D. Bingham. http://www.sfu.ca/~ssurjano/optimization.html.

[11] A. A. Törn. A search clustering approach to global optimization. *Towards Global Optimization 2.*, pages 49–62, 1978.

Estimating Clique Size via Discarding Subgraphs

Sándor Szabó
University of Pécs, Hungary
E-mail: sszabo7@hotmail.com

Bogdán Zaválnij
Rényi Institute of Mathematics
E-mail: bogdan@renyi.hu

Keywords: combinatorial optimization, maximum clique problem, greedy coloring, Lovász' theta function, practical solutions of NP-hard problems

Received: April 1, 2020

The paper will present a method to establish an upper bound on the clique number of a given finite simple graph. In order to evaluate the performance of the proposed algorithm in practice we carry out a large scale numerical experiment on carefully selected benchmark instances.

Povzetek: Razvita in opisana je nova metoda za določanje zgornje meje števila klik v grafu.

1 Introduction

In this paper we will work with finite simple graphs. A graph is finite if it has finitely many nodes and finitely many edges. A graph is simple if it does not have double edges and if it does not have loops. If V is the set of nodes and E is the set of edges of a finite simple graph G, then the ordered pair (V, E) completely describes the graph G.

A set of nodes I of the finite simple graph $G = (V, E)$ is called an independent set if two distinct nodes in I are never adjacent. A set of nodes U of G is called a clique if two distinct nodes in U are always adjacent. Sometimes the subgraph Δ induced by U in G is called a clique of G. If the clique Δ has k nodes, then we say that Δ is a k-clique in G. The number k is sometimes referred as the size of the clique. A k-clique Δ in G is called a maximal clique if G does not have any $(k+1)$-clique that contains Δ as a subgraph. A k-clique Δ in G is called a maximum clique if G does not contain any clique with size larger than k. This well defined integer k is called the clique number of the graph G and it is denoted by $\omega(G)$. Plainly each maximum clique in G has the same size which is equal to $\omega(G)$. Finding both maximal and maximum cliques in a given graph has important and interesting theoretical and practical applications. (For further details consult with [2, 5, 12, 23, 24].)

It is a well known result of the complexity theory of computations that the problem of determining the clique number of a given finite simple graph belongs to the NP-hard complexity class. (For a proof see [9, 17].) This can be interpreted such that computing the clique number of a given graph is computationally demanding. From this reason instead of determining the clique number exactly sometimes we settle for finding a large but not necessarily maximum clique. In this sense we distinguish exact and non-exact clique search algorithms. The non-exact algorithms are further categorized as local search or stochastic or heuristic algorithm depending on the nature of the search technique involved. An example of the exact algorithm can be found in [22] and an example of the non-exact method can be seen in [13]. A strong upper bound can be used as a quality certificate to prove that a heuristically found clique is close enough to a maximum clique.

The main result of this paper is a procedure to establish an upper bound for the clique number of a given graph. The procedure employs legal coloring of the nodes of tactically chosen subgraphs of the given graph. Although the coloring schemes we use are complex the approach is essentially combinatorial. As a consequence it does not involve floating point arithmetic and free of rounding errors. It is known that the Lovász' theta function can be used to compute upper bounds for the clique number of a given graph. This estimate in practice boils down to solve semidefinite programs which is inherently a floating point computation. We were interested in how the elementary combinatorial and the less elementary techniques compares. In the lack of adequate theoretical tools we carried out a large scale numerical experiment on carefully selected problems.

2 Bounding the clique number

Let G be a finite simple graph and let b a fixed positive integer. We color the nodes of the graph G such that the following two conditions hold.

1. Each node of G receives exactly b distinct colors.

2. Two adjacent nodes never receive the same color.

A coloring of the nodes of G satisfying these conditions is called a b-fold legal coloring of the nodes of G. For each finite simple graph G there is an integer k such that the nodes of G can be legally colored with k colors in a b-fold manner but the nodes of G do not admit a legal b-fold coloring using $(k-1)$ colors. This well defined number k is referred as the b-fold chromatic number of G and it is denoted by $\chi_b(G)$. In the $b=1$ particular case we drop the subscript b and use the notation $\chi(G)$. We refer to this number as the chromatic number of G. Coloring the nodes of a graph is an important problem with many applications and with a venerable history. (See [10].)

At a legal coloring of the nodes of G the nodes of a k-clique in G must receive k distinct colors and consequently $\omega(G) \leq \chi(G)$. In other words, legal coloring of the nodes can be used to establish upper bound for the clique number.

It is known from the complexity theory of computations that determining the chromatic number of a given graph is an NP-hard problem. A number of greedy coloring algorithms is available to construct a legal coloring of the nodes but not necessarily with an optimal number of colors. (See for instance [6, 14, 7].) Many clique search algorithms used in practice employs these greedy algorithms for upper bounding the clique size.

In this paper we will show that with some extra effort one may improve on the above upper estimates. We have carried out a number of computations in which approximate node coloring procedures were used. But a little contemplation can convince the reader that the computational scheme we use in fact can be combined with less elementary clique size estimates without major difficulty.

Let $I(G)$ denote the set of all independent sets of nodes of a graph G, and let $I(G,u)$ denote the independent sets of G that contain the node u. For each independent set I, we define a nonnegative real variable x_I. The fractional chromatic number of G, which is denoted by $\chi_f(G)$ is the minimum value of $\sum_{I\in I(G)} x_I$, subject to $\sum_{I\in I(G,u)} x_I \geq 1$ for each node u. This value provides an upper estimate of the clique number of G. (See [1].)

A connection between the fractional and the b-fold chromatic numbers is the following $\chi_f(G) = \lim_{b\to\infty} \frac{\chi_b(G)}{b}$. Therefore in practice one can use b-fold coloring [19] as an approximation.

Lemma 1. $\omega(G) \leq \frac{\chi_b(G)}{b} \leq \chi(G)$.

Proof. At any b-fold coloring of the nodes of G the nodes of a k-clique in G must receive $b\cdot k$ distinct colors. Thus $\chi_b(G)$ must be at least $b\cdot k$.

For any positive integer b and a legal node coloring of the graph with $\chi(G)$ colors one can construct a legal b-fold coloring with $b\cdot\chi(G)$ colors. For this replace each color with a list of b different colors on each node. From the conditions of legal node coloring and b-fold coloring this will yield to a proper b-fold coloring. Thus $\chi_b(G) \leq b\cdot\chi(G)$. □

Computing the b-fold chromatic number is an NP-hard problem, as for $b=1$ case it reduces to the problem of computing the chromatic number. So one looks for heuristic algorithms in case of large graphs. The b-fold coloring of a graph can be reduced to a legal coloring of a suitable auxiliary graph as described in [19, 21], so any heuristic algorithm for legal coloring of the nodes can be easily adopted for finding a b-fold coloring as well.

We may use other upper bounding techniques. For each finite simple graph G there is a well defined graph parameter $\vartheta(G)$, which is called the Lovász' theta number [15]. The Lovász' theta number of the complement of a graph G also bounds the clique number of G from above. The values of the ϑ function can be computed in polynomial time [3, 4], but the degree of the polynomial bound of the running time is high. The bounds on the clique number we have listed so far are the following [8]:

$$\omega(G) \leq \vartheta(\overline{G}) \leq \chi_f(G) \leq \frac{\chi_b(G)}{b} \leq \chi(G).$$

3 The description of the algorithm

Let $v_1,\ldots,v_n$ be all the nodes of $G=(V,E)$. For each node v_i of G we define a graph K_i of G. Namely, let K_i be the subgraph of G induced by the set of nodes $N(v_i)$. Here $N(v_i)$ is the set of neighbors of v_i in G, that is,

$$N(v_i) = \{v : v\in V, \{v_i,v\}\in E\}.$$

For each i, $1\leq i\leq n$ we compute an upper bound α_i for $\omega(K_i)$. For example using a greedy coloring algorithm we color the nodes of K_i legally and set α_i to be the number of colors the algorithm provided. Obviously one can use another upper bound as well. Using the given graph G we define a sequence of numbers $\mu_1,\ldots,\mu_n$. We pick a graph K_i with a minimum α_i. We set $\mu_n=\alpha_i$. Next we delete the node v_i from G. We repeat the whole procedure with this smaller graph and get a new quantity μ_{n-1}. When all nodes of G are deleted the computation terminates. At this stage we have a sequence $\mu_1,\ldots,\mu_n$.

Theorem 1. *Using the notation introduced above the inequality*

$$\omega(G) \leq 1+\max\{\mu_1,\ldots,\mu_n\}$$

holds.

In the remaining part of this section we justify this claim.

Let $u_1,\ldots,u_n$ be a fixed rearrangement of the nodes $v_1,\ldots,v_n$. We consider the subgraph L_i of G induced by the set of nodes $N(u_i)\cap\{u_1,\ldots,u_i\}$ for each i, $1\leq i\leq n$. Note that $u_i\notin N(u_i)$ and so

$$N(u_i)\cap\{u_1,\ldots,u_i\} = N(u_i)\cap\{u_1,\ldots,u_{i-1}\}.$$

In the $i=1$ special case we should identify $\{u_1,\ldots,u_{i-1}\}$ with the empty set. Thus L_1 is a graph without any node. In this situation we set $\omega(L_1)$ to be 0.

We will need the following result.

Lemma 2. *Using the notations introduced above the inequality*

$$\omega(G) \leq 1 + \max\{\omega(L_1), \ldots, \omega(L_n)\}$$

holds.

Proof. Set $k = \omega(G)$. Clearly, the graph G contains a k-clique Δ. On the fixed list $u_1, \ldots, u_n$ of the nodes $v_1, \ldots, v_n$ there is a unique u_i such that u_i is a node of Δ but u_j is not a node of Δ for each j, $i < j \leq n$. The set $N(u_i) \cap \{u_1, \ldots, u_{i-1}\}$ contains $k-1$ nodes of Δ and so $k - 1 \leq \omega(L_i)$ holds. From $k \leq 1 + \omega(L_i)$ it follows that

$$k \leq 1 + \max\{\omega(L_1), \ldots, \omega(L_n)\}.$$

□

The alart reader will recognize that in the statement of Lemma 2 the inequality sign can be replaced by an equation sign. But we are content with this weaker result.

Computing $\omega(L_i)$ is computationally demanding. Therefore we use an easily computable upper bound μ_i for $\omega(L_i)$ instead. Lemma 2 shows that we can freely rearrange the nodes for the estimate of $\omega(G)$. This rearrangement influences the final result. Preliminary tests showed that different rearrangements can result in quite different upper bound values. The rearrangement we propose results a rather good upper estimate.

Let us turn back to our proposed algorithm. First we locate a subgraph K_i of G for which the corresponding α_i is a minimum among $\alpha_1, \ldots, \alpha_n$. We set $u_n = v_i$. This will be the first node fixed in our rearrangement. Putting it in another way we may say that we have identified the last element u_n of the fixed list of nodes $u_1, \ldots, u_n$. We have also identified the subgraph L_n. Namely, $L_n = K_i$.

We delete the node $u_n = v_i$ from the graph G. It means we are working with a smaller new graph induced by the set of nodes $V \setminus \{u_n\}$. Next we identify the node u_{n-1} and the subgraph L_{n-1} by performing the previous step again. Continuing in this way we get the subgraphs $L_1, \ldots, L_n$. Our procedure gives that $\omega(L_i) \leq \mu_i$, for each i, $1 \leq i \leq n$ and so

$$\max\{\omega(L_1), \ldots, \omega(L_n)\} \leq \max\{\mu_1, \ldots, \mu_n\}.$$

Combining this result with Lemma 2 we get $\omega(G) \leq 1 + \max\{\mu_1, \ldots, \mu_n\}$ as stated in Theorem 1. We we call this algorithm DISCARDING.

We make some remarks about streamlining the DISCARDING procedure. Remember that first employing a greedy coloring algorithm to the graphs $K_1, \ldots, K_n$ we compute the numbers $\alpha_1, \ldots, \alpha_n$. Since we are looking for the minimum value occurring among $\alpha_1, \ldots, \alpha_n$ in certain cases we need not to complete the coloring procedure. We may abort coloring a subgraph K_i if we have already used more colors than the minimum number of colors needed so far.

Next, when we delete the node v_i from G for which α_i is minimum we may delete each v_j for which α_j is minimum. In this way we end up with a shorter list $\mu_1, \ldots, \mu_s$ instead of the longer list $\mu_1, \ldots, \mu_n$. Note that the upper estimate $\omega(G) \leq 1+\max\{\mu_1, \ldots, \mu_s\}$ is not weaker than the upper estimate $\omega(G) \leq 1+\max\{\mu_1, \ldots, \mu_n\}$. We can also delete all the v_i nodes during the procedure, where α_i less than or equal to the maximum of the previous μ values.

Note that when the node v_j is not adjacent to the deleted node v_i, then the upper bound α_j of $\omega(K_j)$ need not to be recalculated. The reason is that in this situation deleting the node v_i leaves the subgraph K_j unchanged.

We would like to point out, that calculating the α_i values can be carried out independently of each other, thus we can perform this calculation in parallel fashion. We implemented the algorithm DISCARDING in both sequential and parallel manner using OpenMP for shared memory computers. Both programs resulted the same upper bounds. It is clear, that the program could be implemented using MPI for distributed computers as well thus achieving greater scalability.

4 A simpler estimating procedure

In this section we consider a less sophisticated version of the procedure DISCARD. We will call this new procedure SEQUENTIAL. We fix an ordering $w_1, w_2, \ldots, w_n$ of the nodes $v_1, v_2, \ldots, v_n$ of the given finite simple graph $G = (V, E)$. We consider the graph L_i induced by $N(w_i) \cap \{w_1, \ldots, w_i\}$ for each i, $1 \leq i \leq n$ and compute a μ_i such that $\omega(L_i) \leq \mu_i$ holds for each i, $1 \leq i \leq n$. Now the inequality $\omega(G) \leq 1 + \max\{\mu_1, \ldots, \mu_n\}$ holds.

Let P be an auxiliary procedure for computing an upper bound $\mu(G, P)$ for the clique number $\omega(G)$ of a given finite simple graph G. We say that P has the monotonicity property if $\mu(H, P) \leq \mu(G, P)$ holds whenever H is a subgraph of G.

For example the auxiliary procedure of computing the chromatic number of a graph has the monotonicity property as the inequality $\chi(H) \leq \chi(G)$ holds for each subgraph H of G. On the other hand the auxiliary procedure of using the not necessarily optimal number of colors of a legal coloring of the nodes does not have the monotonicity property. It can happen that we use more colors to color the nodes of a subgraph H than we use for coloring the nodes of the whole graph G.

The next result reveals a certain optimality property of the DISCARDING procedure.

Theorem 2. *Let $G = (V, E)$ be a finite simple graph and suppose that the auxiliary procedure used to establish the upper bound $\omega(K_i) \leq \alpha_i$ has the monotonicity property. Then the SEQUENTIAL procedure does not provide a better estimate for the clique number of G than the DISCARDING procedure.*

Proof. Suppose that the DISCARDING algorithm gives rise to the ordering $u_1, u_2, \ldots, u_n$ of the nodes $v_1, v_2, \ldots, v_n$ of G. Assume on the contrary that the

SEQUENTIAL procedure applied to some ordering $w_1, w_2, \dots, w_n$ of the nodes $v_1, v_2, \dots, v_n$ of G leads to a strictly smaller upper estimate of $\omega(G)$.

We show that such a sequence leads to contradiction. Let us assign colors red, green, yellow to the nodes $v_1, v_2, \dots, v_n$. This coloring of the nodes is based on the sequence $u_1, u_2, \dots, u_n$, and it is not connected to the concept of legal coloring. There is a special node in this sequence, called the pivot node, for which the upper estimate attains its maximum. If the maximum is attained at several nodes we choose the last one. The pivot node of the graph will receive color green. We color the nodes in the sequence preceeding the pivot node by red, and the nodes after the pivot node by yellow.

Of course the members of the sequence $u_1, u_2, \dots, u_n$ and the sequence $w_1, w_2, \dots, w_n$ are colored with red, green and yellow, as they are just reordering of $v_1, v_2, \dots, v_n$.

The following observation will play a critical role. When in the course of the algorithm we deleted all yellow nodes from the graph G (let us denote the nodes of this new graph by V_y) we made an upper estimate for graphs using all remaining nodes, and the upper estimate for the graph using the green node was minimal. That means that the upper estimate for the graphs using the red nodes is at least as this figure after deleting the all yellow nodes of the graph.

Now we look at the other sequence $w_1, w_2, \dots, w_n$ assumed to give a better upper estimate. We locate two nodes in it, the pivot node (the green one), and the last node among the nodes that colored red. We shall denote them w_g and w_r, respectively. Note, that all nodes w_i, $i > r$ are colored yellow or green. We distinguish two cases:

Case $(g > r)$**.** In this situation all nodes $w_i, i > g$ are colored yellow, as the last red node appears *before* the green node. Let us denote the nodes of the graph we get after deleting all nodes w_i, $i > g$ from G by V_g. As the set of deleted nodes is a subset of the set of all yellow nodes, the graph induced by V_y is a subgraph of the graph induced by V_g. But the upper estimate for the graph induced by $N(w_g) \cap V_g$ is assumed to be less than the upper estimate for the graph induced by $N(w_g) \cap V_y$. This contradicts the monotonicity property of the auxiliary procedure.

Case $(r > g)$**.** In this situation all nodes $w_i, i > r$ are colored yellow, as w_r is the last red node and the only green one precedes it. Let us denote the nodes of the graph we get after deleting all nodes w_i, $i > r$ from G by V_r. As the set of deleted nodes is a subset of the set of all yellow nodes, the graph induced by V_y is a subgraph of the graph induced by V_r. But the upper estimate for the graph induced by $N(w_r) \cap V_r$ is assumed to be less than the upper estimate for the graph induced by$N(w_r) \cap V_y$. This contradicts the monotonicity property of the auxiliary procedure.

□

5 Numerical experiments

We selected altogether 43 graphs for carrying out our extended measurements. As we were aiming at problems where it is hard to calculate the $\omega(G)$, we used sources of graphs according to this, and decided to use graphs having at least 500 nodes. We used those graphs from these sources for which either the Lovász' theta program or our algorithm could finish the calculation of the upper bound in 100 000 seconds using the available 48GB of memory. The first 29 graphs[1] come from the problems of the 2nd DIMACS Challenge [11]. The second 11 graphs[2] come from various error correcting code problems [18]. (Note, that we used complement graphs of those from the webpage, as the original problem asks for the maximum independent set.) The last 3 graphs[3] are reformulated problems of monotonic matrices [20, 16]. We choose these graphs so that they would represent extremely hard clique search problems and for some we even do not know the value of the size of the maximum clique as the available clique search programs are not able to find the exact $\omega(G)$ value. There are a few ones where the exact value of the clique number is known but the existing clique solvers are not able to compute them. The so-called Johnson codes give rise to such clique problems.

As the proposed algorithm instructs us to use an arbitrary upper bound for calculating the α values we need to choose one method as a starting point. We have chosen the b-fold coloring with Culberson's iterated recoloring scheme because it gives the best result in reasonable time of couple of seconds. This way we were using several hours of computational time and hoped to achieve improvements over the original upper bound of b-fold coloring. A b-fold legal coloring of the nodes of a given graph can be reduced to legally coloring the nodes of an auxiliary graph described in [21]. In the course of our numerical experiments we performed several measurements to establish and compare different upper bounds. We collected the results in the Table 1. Namely, we did the following measurements:

1. perform a legal coloring of the graph (column: "legal coloring");
2. find the Lovász' theta value of the complement graph (column: "$\vartheta(\overline{G})$");
3. perform a b-fold coloring of the graph (column: "b-fold coloring");
4. use b-fold coloring as a base and perform the proposed algorithm (column: "DISC").

All coloring programs started with a DSatur algorithm due to D. Brélaz [6]. Then we recolored the nodes several

[1]http://iridia.ulb.ac.be/~fmascia/maximum_clique/DIMACS-benchmark

[2]https://oeis.org/A265032/a265032.html

[3]http://mathworld.wolfram.com/MonotonicMatrix.html

times using a method due to J. C. Culberson [7]. The stopping criteria was if the number of colors did not changed after a specified number of iterations. For smaller and medium graphs we used 1000 iterations; for bigger graphs (over 3000 nodes) we used 500 iterations. We used 7-fold coloring for smaller graphs (up to 2000 nodes); 5-fold coloring for medium graphs (over 2000 nodes); and 3-fold coloring for the biggest graphs (over 3000 nodes). The Lovász' theta function was computed with the program of B. Borchers [3, 4]. The measurements were performed on a computer with two Intel Xeon X5680 3.33 GHz processors – all together 12 cores – and 48GB of RAM. (This computer happens to be a node of a supercomputer, but this is of no importance for the present paper.) The legal coloring and b-fold coloring of the graphs were performed in sequential manner. The Lovász' theta calculations used all 12 cores due to the underlying BLAS implementation. Our DISCARDING algorithm used also 12 cores as it performed the calculation of α_i values independently. Note, that for BLAS the 12 core is a limit as it needs to be run in shared memory environment, while our algorithm could have used more cores in distributed parallelization on a supercomputer but for the sake of comparability we limited it to the same 12 cores.

The running times for legal coloring were smaller than a second or in the range of a few seconds. The time for performing the b-fold coloring depends on the size of the graph and the value of b. The running times for b-fold coloring algorithm was in the range of several seconds up to a few minutes. The running times of the Borchers' program on 12 cores for calculating the Lovász' theta of the complement graph is indicated in the column "time (sec) $\vartheta(\overline{G})$ (12c)". The running times of our parallel algorithm using the same 12 cores is indicated in the column "time (sec) DISC (12c)". Those values that cannot be calculated on the used test hardware we indicated in the table with "$>$ 24h", as the 100 000 seconds limit is roughly 24 hours.

6 Evaluation of the proposed algorithm

From the results in the Table 1 we may conclude that the proposed algorithm is feasible, that is, it can be computed on a nowadays computer in reasonable time. But in this section we are making also a comparison between our algorithm based on the b-fold coloring and the Lovász' theta upper bound calculation. We should point out, that this comparison is far from trivial:

1. We needed to parallelize our program and use the same number of cores by both programs. Note, that Borchers' program uses shared memory, so we could not use more cores than cores in one node of a supercomputer. On the other hand our program could be written to use a distributed supercomputer with hundreds of cores.
2. As our program, which uses b-fold coloring, using only integer and bit calculation the different architectures are not affecting the running times much. Opposed to this Borchers' program uses floating point calculations and BLAS. Thus different architectures (Intel or AMD), different compilers (icc or gcc) and different implementations of BLAS (Reference BLAS, ATLAS, OpenBLAS or Intel MKL library) all have huge effect on the running times. A modern AMD desktop PC with 8 core Ryzen processor using gcc as a compiler and OpenBLAS resulted in up to 50 times(!) slower running times compared to the times presented in Table 1 for the Lovász' theta calculations on Intel architecture, icc compiler and the MKL library. The architecture, compiler and BLAS implementation used for the results of the present paper all strongly favor the Lovász' theta calculations. A reproduction of the results on a desktop PC may result much longer running times for the Lovász' theta calculation program due to Borchers while giving similar times for our algorithm.

The reader can see that the computation of the Lovász' theta number for 22 of 43 instances could not be completed. In each of these 22 cases the program terminated with the error message: "not enough memory". As it turns out the length of the calculation of the Lovász' theta function depends on the number of edges (m) of the complement graph (or the non-edges, the missing edges of the graph). The running time and memory requirements are $O\left(m^3\right)$ and $O\left(m^2\right)$, respectively. (See [3, 4].) Using these asymptotic results we estimated the running time as $7.2 \cdot 10^{-11}m^3$ seconds and the memory requirement as $9.2 \cdot 10^{-9}m^2$ GigaBytes. This estimate was consistent with the experimental results. Clearly 48GB of memory is not enough in the missing cases. Actually, some instances, as for example the `keller6` graph, would need 1 TB(!) of memory and would run for nearly a thousand of years. If one would use a PC with nowadays usual 8-16GB of memory 6 other instances would be out of reach, thus making 28 out of 43 instances incalculable.

To sum up the results, we can say, that our algorithm gives better upper bounds than the Lovász' theta calculation for 24 out of 43 instances. In addition it runs faster for 24 out of 43 instances. It holds for the cases when the Lovász' theta computation cannot be completed. Further there are instances when our algorithm actually gave lower bound than the Lovász' theta approach. These are the graphs `MANN_a45`, `MANN_a81`, `keller4` and `p_hat300-1` from which the former two are in the table, and the later two are not as they have less than 500 nodes. Obviously, there can be other cases as well among those where the Lovász' theta could not be calculated. The graphs `johnson10-4-4` and `p_hat300-2` gave the same bound – these two also was excluded from the table because of being too small. Note, that in several cases our algorithm gave sharp upper bound which is equal to the clique number: `p_hat300-1`, latin10, c-fat500-1, queen40, queen50 and san1000 graphs.

Let us return to the question if the proposed algorithm can lower the upper bound provided by the base algorithm. One can see from the results that the proposed algorithm with few exceptions improves the upper bound of b-fold coloring, in some cases radically. We may conclude that there is computational evidence that the proposed algorithm lower the upper bound if we use more computational resources. Let us turn to the question how the proposed algorithm compares to the Lovász' theta based methodology. We are getting better result only a few times. However, we were able to compute our new bound for all but two graphs that cannot be said about the Lovász' theta number. Practically, our algorithm is so versatile that even for those cases where the upper bound could not be calculated we are able to tune it using smaller b and using smaller number of recoloring steps. Thus reducing the running time we can turn the instance calculable. Note, that one can replace the upper bounds computed by coloring the nodes of the tactically chosen subgraphs by upper bounds computed by the Lovász' theta function. The calculations involved in this situation would use much more time and need to be performed several thousand times. Thus we would expect it to be reasonable only using distributed computing on a supercomputer, as the calculations of the α values can be done independently.

Acknowledgements

The project has been supported by National Research, Development and Innovation Office – NKFIH Fund No. SNN-135643.

References

[1] E. Balas and J. Xue, *Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring,* Algorithmica 15 (1996), pp. 397–412. https://doi.org/10.21236/ada275328

[2] I. M. Bomze, M. Budinich, P. M. Pardalos and M. Pelillo, *The Maximum Clique Problem, Handbook of Combinatorial Optimization Vol. 4,* Kluwer Academic Publisher, 1999. https://doi.org/10.1007/978-1-4757-3023-4_1

[3] B. Borchers, *CSDP, A C library for semidefinite programming,* Optimization Methods and Software, 11(1) (1999), pp. 613–623. https://doi.org/10.1080/10556789908805765

[4] B. Borchers and J. G. Young, *Implementation of a primal-dual method for SDP on a shared memory parallel architecture,* Computational Optimization and Applications 37(3), (2007) pp. 355–369. https://doi.org/10.1007/s10589-007-9030-3

[5] A. Bóta and M. Krész, *A high resolution clique-based overlapping community detection algorithm for small-world networks,* Informatica, 39(2). (2015).

[6] D. Brélaz, *New methods to color the vertices of a graph,* Communications of the ACM, 22 (1979), pp. 251–256. https://doi.org/10.1145/359094.359101

[7] J. C. Culberson, *Iterated Greedy Graph Coloring and the Difficulty Landscape,* Technical Report, University of Alberta, 1992.

[8] C. Elphick and P. Wocjan, *An inertial lower bound for the chromatic number of a graph,* The Electronic Journal of Combinatorics, Volume 24, Issue 1 (2017) https://arxiv.org/abs/1605.01978 https://doi.org/10.37236/6404

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness,* Freeman, New York, 2003.

[10] F. Harary, *Graph Theory,* Addison-Wesley, 1969.

[11] J. Hasselberg, P. M. Pardalos, and G. Vairaktarakis, *Test case generators and computational results for the maximum clique problem,* Journal of Global Optimization 3 (1993), pp. 463–482. http://www.springerlink.com/content/p2m65n57u657605n https://doi.org/10.1007/bf01096415

[12] D. Kumlander, *Some Practical Algorithms to Solve the Maximal Clique Problem,* PhD. Thesis, Tallin University of Technology, 2005.

[13] S. Lamm, P. Sanders, C. Schulz, D. Strash, R. F. Werneck, Finding near-optimial independent sets at scale, *Proceedings of the 16th Meeting on Algorithm Engineering and Experimentation (ALENEX'16).* 2016. https://doi.org/10.1137/1.9781611974317.12

[14] F. T. Leighton, *A graph coloring algorithm for large scheduling problems,* Journal of Research of National Bureau of Standards 84 (1979), pp. 489–506.

[15] L. Lovász, *On the Shannon capacity of a graph,* IEEE Transactions on Information Theory, Volume 25 Issue 1, January 1979 pp. 1–7. https://doi.org/10.1109/TIT.1979.1055985

[16] P. R. J. Östergård and A. Pöllänen, *New results on tripod packings.* Discrete Comput. Geom. 61 (2019), pp. 271–284. https://doi.org/10.1007/s00454-018-0012-2

[17] C. H. Papadimitriou, *Computational Complexity,* Addison-Wesley Publishing Company, Inc., Reading, MA 1994.

[18] N. J. A. Sloane, *Challenge Problems: Independent Sets in Graphs,* `http://neilsloane.com/doc/graphs.html`

[19] S. Stahl, *n-Tuple colorings and associated graphs,* Journal of Combinatorial Theory, Series B Volume 20, Issue 2, April 1976, pp. 185–203. `https://doi.org/10.1016/0095-8956(76)90010-1`

[20] S. Szabó, *Monotonic matrices and clique search in graphs,* Annales Univ. Sci. Budapest., Sect. Comp. 41 (2013), pp. 307–322.

[21] S. Szabó and B. Zaválnij, *Reducing graph coloring to clique search,* Asia Pacific Journal of Mathematics, 1 (2016), pp. 64–85.

[22] S. Szabó and B. Zaválnij, A different approach to maximum clique search. *20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC2018)* pp. 310–316. `https://doi.org/10.1109/SYNASC.2018.00055`

[23] S. Szabó and B. Zaválnij, *Reducing hypergraph coloring to clique search.* Discrete Applied Mathematics, 264 (2019), pp. 196–207. `https://doi.org/10.1016/j.dam.2018.09.010`

[24] Q. Wu and J-K. Hao, *A review on algorithms for maximum clique problems,* European Journal of Operational Research. Volume 242, Issue 3, 1 May 2015, pp. 693–709. `https://doi.org/10.1016/j.ejor.2014.09.064`

Table 1: The test graphs and their upper bounds for $\omega(G)$ from different legal coloring methods and Lovász' theta along with the new bound from the proposed algorithm. We indicated by an * those cases where the program could not reach the result within 100,000 seconds, that is roughly in 24 hours.

	$\lvert V \rvert$	density	$\omega(G)$	legal	$\vartheta(\bar{G})$	time (sec)	b-fold	DISC	time (sec)
		%		coloring		$\vartheta(\bar{G})$ (12c)			DISC (12c)
brock800_1	800	64.93	23	117	*	> 24h	103	**68**	**12717s**
brock800_2	800	65.13	24	118	*	> 24h	103	**68**	**11670s**
brock800_3	800	64.87	25	117	*	> 24h	103	**67**	**15414s**
brock800_4	800	64.97	26	118	*	> 24h	103	**68**	**9849s**
C500.9	500	90.05	≥ 57	139	**84.20**	**109s**	130	115	10235s
C1000.9	1000	90.11	≥ 68	251	**123.49**	**6029s**	232	207	94916s
C2000.5	2000	50.02	16	194	*	> 24h	174	**90**	**68024s**
c-fat500-1	500	3.57	14	14	*	> 24h	14	**14**	**9s**
DSJC500_5	500	50.20	13	58	**22.74**	9960s	51	28	**850s**
DSJC1000_5	1000	50.02	15	107	*	> 24h	92	**49**	**10913s**
hamming10-4	1024	82.89	40	74	*	> 24h	55	**50**	**6227s**
johnson-12-5-4	792	95.58	80	140	**99.00**	**130s**	106	105	4318s
johnson-13-4-4	715	94.96	65	115	**71.50**	**107s**	76	76	1877s
johnson-13-5-4	1287	96.89	≥ 123	212	**143.00**	**792s**	155	154	19485s
keller5	776	75.15	27	31	*	> 24h	31	**31**	**598s**
keller6	3361	81.82	59	64	*	> 24h	64	**63**	**5515s**
MANN_a45	1035	99.63	345	360	356.05	**23s**	360	**353**	29869s
MANN_a81	3321	99.88	1100	1134	1126.62	**589s**	1134	**1121**	91875s
p_hat1000-1	1000	24.48	10	52	*	> 24h	46	**17**	**4041s**
p_hat700-1	700	24.93	11	40	*	> 24h	35	**13**	**977s**
p_hat700-2	700	49.76	44	86	*	> 24h	76	**52**	**31637s**
p_hat700-3	700	74.80	62	129	**72.00**	**11781s**	120	87	73518s
p_hat500-1	500	25.31	9	31	*	> 24h	27	**11**	**376s**
p_hat500-2	500	50.46	36	63	**38.97**	13677s	57	40	**8540s**
p_hat500-3	500	75.19	50	101	**58.57**	**1391s**	92	67	14654s
latin10	900	75.97	90	110	*	> 24h	93	**90**	**5529s**
queen40	1600	91.91	40	40	*	> 24h	40	**40**	**5397s**
queen50	2500	93.49	50	50	*	> 24h	50	**50**	**24986s**
san1000	1000	50.15	15	15	*	> 24h	15	**15**	**2173s**
1dc.512-c	512	92.56	52	73	**53.03**	**76s**	55	54	2007s
1dc.1024-c	1024	95.41	94	137	**95.98**	**1004s**	101	99	9171s
1dc.2048-c	2048	97.22	≥ 172	262	**174.73**	**14395s**	189	186	78627s
1et.1024-c	1024	98.17	171	215	**184.23**	**93s**	191	189	94866s
1et.2048-c	2048	98.93	316	401	**342.03**	**1026s**	356	*	> 24h
1tc.1024-c	1024	98.48	196	227	**206.30**	**64s**	216	212	23475s
1tc.2048-c	2048	99.10	352	420	**374.64**	**753s**	387	*	> 24h
1zc.512-c	512	94.72	62	93	**68.75**	**23s**	72	71	1110s
1zc.1024-c	1024	96.82	≥ 112	180	**128.67**	**295s**	136	134	13793s
2dc.1024-c	1024	67.70	16	31	*	> 24h	21	**18**	**4024s**
2dc.2048-c	2048	75.93	24	54	*	> 24h	37	**32**	**53711s**
monoton-9	729	83.52	28	45	**34.41**	**5465s**	41	37	13670s
monoton-10	1000	85.14	32	61	*	> 24h	51	**46**	**39031s**
monoton-11	1331	86.47	38	69	*	> 24h	63	**57**	**52911s**

Statistics-Based Chain Code Compression with Decreased Sensitivity to Shape Artefacts

David Podgorelec, Andrej Nerat and Borut Žalik
University of Maribor, Faculty of Electrical Engineering and Computer Science
Koroška cesta 46, SI-2000, Maribor, Slovenia
E-mail: david.podgorelec@um.si

Keywords: chain code, data compression, dynamic programming, pseudo-statistical model, shape artefact

Received: April 3, 2020

Chain codes compactly represent raster curves, but there is still a lot of room for improvement by means of data compression. Several statistics-based chain code compression techniques assign shorter extra codes to frequent pairs of consecutive symbols. Here we systematically extend this concept to patterns of up to k > 2 symbols. A curve may be represented by any of the exponentially many overlapped chains of codes, and the dynamic programming approach is proposed to determine the optimal chain. We also propose utilization of multiple averaged hard coded pseudo-statistical models, since the exact statistical models of individual curves are often huge, and they can also significantly differ from each other. A competitive compression efficiency is assured in this manner and, as a pleasant side effect, this efficiency is less affected by the curve's shape, rasterization algorithm, noise, and image resolution, than in other contemporary methods, which surprisingly do not pay any attention to this problem at all.

Povzetek: V članku predstavimo novo metodo za statistično stiskanje verižnih kod, ki dodeli posebne kode pogostim nizom do k simbolov. Optimalno izmed eksponentno mnogo rešitev izbere z dinamičnim programiranjem. Uporablja več povprečenih psevdo-statističnih modelov, ki jih ne shranjuje skupaj s krivuljo. V primerjavi z drugimi sodobnimi metodami doseže konkurenčno stopnjo stiskanja, hkrati pa je manj občutljiva na obliko krivulje, posebnosti rasterizacijskega algoritma, šum in ločljivost slike.

1 Introduction

Chain codes compactly represent curves in raster images. More than half a century ago, Freeman [3] used symbols $\sigma_i \in [0\ ..\ 7]$ to represent each curve pixel p_i with the azimuth direction ($\sigma_i \cdot 45°$) from its predecessor p_{i-1} measured anticlockwise from the positive x-axis (Fig. 1a). Each symbol is then coded with 3 bits. Alternatively, only 2 bits per pixel are required if the representation relies on 4-connectivity, i.e. the azimuth $p_i - p_{i-1}$ is ($\sigma_i \cdot 90°$), where $\sigma_i \in [0\ ..\ 3]$ (Fig. 1b). Several alternative chain code representations were later introduced, but the concept remains the same as in the pioneering *Freeman chain codes in eight (F8) or four (F4) directions*: symbols from a relatively small alphabet are assigned to subsequent primitives along a curve. In different representations, a primitive may refer to a curve pixel (as in F8 or F4), a vertex between the considered curve pixel and adjacent pixels (Vertex Chain Code – VCC [2] or Three-Orthogonal chain code – 3OT [10]), an edge separating the curve pixel from a background pixel (Differential Chain Code – DCC [9]), or a rectangular cell of pixels (in quasi-lossless representation from [9]). Meanwhile, a symbol models some local geometric relation e.g. relative position of the observed primitive with respect to the previous one. With other words, it represents a command how to navigate from one primitive to the adjacent one along the curve. All these basic chain code representations describe a raster curve efficiently, as they use only 2 or 3 bits per primitive instead of coding grid coordinates with, for example, $2 \cdot 16$ bits per pixel. Nevertheless, numerous successful methods have been proposed to additionally compress raster curves.

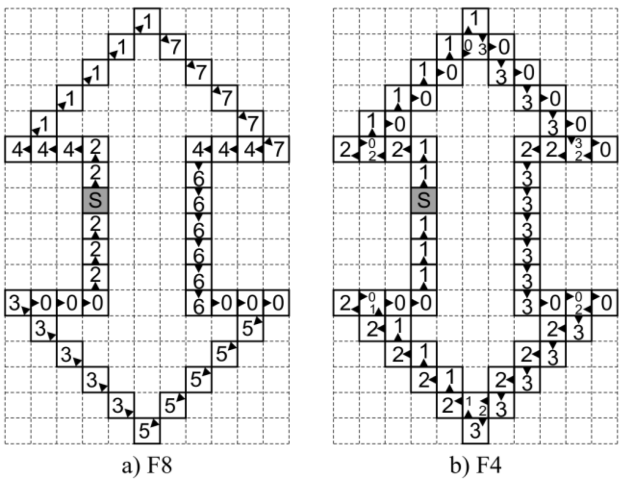


Figure 1: Freeman chain codes in 8 and 4 directions.

Statistical (Huffman or arithmetic) coding is often utilized when the symbols' probability distribution is significantly non-uniform. Further advances in statistics-based approaches were achieved by introducing extra codes for frequent pairs of primitives [7, 8], or by utilization of multiple statistical models in so-called context-based approaches [1], where the statistical model for coding a considered symbol is conditioned on the context of M (typically 1 or 2) previous symbols. On the other hand, non-statistical approaches perform various string transformations [11, 12], e.g. Burrows-Wheeler Transform (BWT) and/or Move-To-Front Transform (MTFT) to increase the number of 0's and prepare the data for efficient run-length encoding (RLE) and/or binary arithmetic coding (BAC). BWT rearranges the sequence into runs of equal symbols, while MTFT utilizes local correlations to replace the data values with their indices from typically smaller repertoire.

In this paper, we introduce a new statistics-based approach where extra codes may represent patterns of up to $k = O(1)$ symbols. Our aim was to achieve a competitive compression efficiency, but an interesting pleasant side effect was encountered and brought into focus during the method development and testing. Namely, impacts of the curve shape, image resolution, rasterization algorithm, noise, and geometric transformations on the compression ratio are significantly reduced in comparison to other contemporary methods. This topic has been so far addressed indirectly within the context-based approaches and, partially, in the non-statistical approaches, while it was completely neglected in other related works. Section 2 illustrates the overall idea of the proposed approach. Section 3 describes the preparation and utilization of multiple averaged hard coded pseudo-statistical models, crucial for the minimization of the mentioned impacts. Section 4 experimentally confirms the compression efficiency and the reduced dependence on curve's artefacts. Finally, Section 5 summarizes the presented work, and discusses future research challenges.

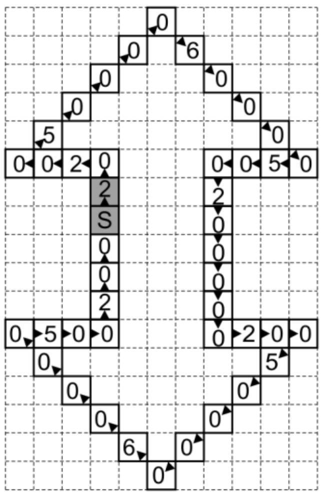


Figure 2: Freeman's chain difference coding.

2 New statistics-based method

Some years after F8 and F4, Freeman also proposed the chain-difference coding (CDC) [6]. A pixel p_i is encoded with the angle difference $\angle(p_i - p_{i-1}, p_{i-1} - p_{i-2})$. Unlike F8 where all 8 symbols have practically the same probabilities, the 0° angle difference is usually much more frequent than other 7 symbols (Fig. 2). All 8 symbols in Fig. 1a have probabilities either $6/43 \approx 13.95\%$ or $7/43 \approx 16.28\%$, and the probabilities of four symbols in Fig. 1b are either $16/63 \approx 25.40\%$ or $15/63 \approx 23.81\%$. On the other hand, the probability of symbol 0 in Fig. 2 is $32/42 \approx 76.19\%$ while the probabilities of other 7 symbols are all below 10%. Such non-uniform distribution provides a good basis for statistical coding. However, some tens of bits must be spent to store the best-fitted statistical model (BFSM) for an individual curve, which is, particularly with shorter curves, not negligible. Liu and Žalik [6] presented the directional difference chain coding (DDCC), where CDC BFSMs of over 1000 training curves are averaged into a suboptimal hard-coded statistical model (HCSM), which is then used for compression in non-training use cases. Some years later, the compressed DDCC (C_DDCC) [7] was introduced, where three extra codes for usually frequent pairs $\langle \pm 45°, \mp 45° \rangle$ and for patterns of 12 to 27 zeros were added into the HCSM. The two pairs were intuitively identified, as they periodically interrupt sequences of 0° symbols along oblique line segments (except those with slopes 45° or -45°).

Here we take a step forward by systematically extending the DDCC coding scheme with extra codes for patterns of up to $k = O(1)$ symbols, $k \geq 2$. Furthermore, we group "similar" training curves into classes and derive HCSMs separately for each class. Although this concept looks straightforward, several non-trivial issues must be considered to achieve a feasible implementation and a competitive compression efficiency. These issues can be structured into two separate phases.

1. The training phase is performed by an expert/developer in order to calibrate the algorithm for future use. A representative repertoire of training curves is first provided, and the BFSM for each curve is extracted. Features for multicriteria classification of training curves are selected (intuitively in the current implementation), and the training curves are then assigned to the classes. HCSMs are derived afterwards by separately averaging BFSMs within the classes. The detailed description follows in Section 3.
2. The exploitation phase is run by end-users in order to compress non-training curves from concrete use cases. An input curve is first analysed to determine the feature values needed to heuristically select the most appropriate of the stored HCSMs. The chosen HCSM is then utilized to compress the curve with Huffman coding. The main challenge in designing this phase is the strategy for determination of the optimal sequence (chain) of codes, which is emphasized after the definitions.

2.1 Definitions

Trail $T_{i,\,j}$ = ⟨v$_i$, ..., v$_j$⟩, $i \le j$, is a sequence of adjacent pixels (or corresponding graph vertices) along a raster curve. The length of the trail (in pixels) is $h(i, j) = j - i + 1$. The trail $T_{1,\,n}$ = ⟨v$_1$, ..., v$_n$⟩ corresponds to the entire raster curve of length n.

Trail decomposition splits the trail into one or more nested trails, whose concatenation reassembles the original trail. A trail $T_{u,\,v}$ is *nested* in $T_{i,\,j}$ if $i \le u \le v \le j$.

Symbol is a chain-code command aimed to be assigned to a single pixel along a raster curve.

Pattern (of symbols) $\Sigma_{i,\,j}$ = ⟨σ_i, ..., σ_j⟩, $i \le j$, is a sequence of symbols aimed to be assigned to pixels of a trail of the same length $h(i, j)$.

Dynamic programming graph is an edge-weighted graph (G, w), where $G = (V, E)$ is a directed graph, $V = \{v_1, ..., v_{n+1}\}$ is a vertex set, $E = \{e_{i,j}\}$ is an edge set, given by pairs of vertices $e_{i,j} = (v_i, v_j)$, $i < j$, and $w : E \to \mathbb{N}$ is a weight function. Vertices v_1, ..., v_n correspond to pixels along the raster curve, edge $e_{i,j}$ represents a trail ⟨v$_i$, ..., v$_{j-1}$⟩, and weight $w_{i,j}$ of an edge $e_{i,j}$ is the bit length of the corresponding Huffman code.

An auxiliary end vertex v$_{n+1}$ does not represent any curve pixel and, thus, there is no need to assign a symbol to it. However, this vertex enables introduction of edges $e_{i,n+1}$, $i \le n$, corresponding to trails $T_{i,\,n}$ = ⟨v$_i$, ..., v$_n$⟩.

***IN*(*i*)** is the set of start vertices of all graph edges with the end vertex v_i. $v_j \in IN(i) \Leftrightarrow e_{j,i} \in E$. Vertex v_j is a predecessor of v_j and the latter is a successor of v_j.

***OUT*(*i*)** is the set of end vertices of all graph edges with the start vertex v_i. $v_j \in OUT(i) \Leftrightarrow e_{i,j} \in E$.

Extra code is a Huffman code which replaces a pattern of two or more symbols in order to save some bits.

$p(\Sigma_{i,\,j})$ is the probability of a pattern $\Sigma_{i,\,j}$ = ⟨σ_i, ..., σ_j⟩ in a considered statistical model. If the latter corresponds to the BFSM of a curve described with $T_{1,\,n}$ = ⟨v$_1$, ..., v$_n$⟩, then we get equation (1):

$$p(\Sigma_{i,\,j}) = f(\Sigma_{i,\,j}) \,/\, (n - h(i, j) + 1), \tag{1}$$

where $f(\Sigma_{i,\,j})$ be the number of appearances of $\Sigma_{i,\,j}$ in the pattern assigned to ⟨v$_1$, ..., v$_n$⟩. However, $p(\Sigma_{i,\,j})$ in some HCSM is obtained by averaging the corresponding probabilities from all participating BFSMs.

Note that an edge $e_{i,j}$, $i < j - 1$, is added into the graph only if an extra code exists for the corresponding pattern assigned to $T_{i,\,j-1}$. On the other hand, edges $e_{i,\,i+1} = (v_i, v_{i+1})$ correspond to single-pixel trails $T_{i,\,i}$ = ⟨v$_i$⟩ and they are unconditionally added to the graph. This assures that the algorithm of parsing the curve pixels will always reach the end vertex v_{n+1}, as any other vertex has at least one successor, i.e. $i \le n \Rightarrow |OUT(i)| \ge 1$.

2.2 Exploitation phase

The existing chain code techniques construct the chain of codes by a greedy algorithm. A raster curve is parsed primitive by primitive, and each of them is immediately coded either alone or as a member of some longer pattern. If different possibilities for coding a primitive exist, the predefined priority is decisive. In C_DDCC, for example, extra codes for ⟨±45°, ∓45°⟩ pairs have higher priority than the corresponding single-pixel codes. However, such priority-based greedy algorithms cannot be simply adjusted to efficiently handle higher number of extra codes for longer patterns of symbols. In the proposed approach, each pixel can be coded with its own code or, theoretically, with one of k codes of longer trails. For $k = 6$ as used in the current implementation and tests (the decision for this value will be explained at the end of Section 3.2), these trails include two pairs, three triplets and so on till six sextets. A longer context of patterns before and behind the considered symbol determines which of the $1 + 2 + \ldots + k = k\,(k + 1) / 2$ possibilities (21 for $k = 6$) shall be used to code the pixel. We therefore have a combinatorial optimization problem where we look for an optimal chain from a large set of multiply overlapped chains. Unlike greedy algorithms, we found dynamic programming capable to provide an optimal choice. Its utilization also facilitates the so-called context dilution problem [1, 7]. Namely, introduction of extra codes for longer patterns of symbols usually extends codes of several symbols and other patterns. For example, introduction of four extra C_DDCC codes for patterns ⟨0°, ±45°⟩ and ⟨±45°, 0⟩ prolongs by 1 bit the codes for ±90°, 180°, RLE of zeros, 135° and/or –135°. Furthermore, examples of chains can be found where individual extra codes do not save any bits.

The proposed dynamic programming approach is adaptation of the so-called exon chaining algorithm from the field of bioinformatics, the simplest of the so-called similarity-based gene prediction approaches [5].

The dynamic programming optimizes the Bellman equation (2), where s_i represents the total bit length of the optimal chain from v_1 to v_{i-1}, $1 < i \le n + 1$. Additionally, s_0 is set to 0 to enable the recursive calculation of s_1.

$$s_i = min_{v_j \in IN(i)}(s_j + w_{j,i}) \tag{2}$$

The vertex pred$_i \in IN(i)$, which indeed participates to the minimum s_i, is also memorized for each v_i. The s_{n+1} represents the total bit length of the overall solution, and the optimal chain itself is then reconstructed by following the vertices pred$_i$ from v_{n+1} backwards to v_1. Bold edges in Fig. 3 represent the optimal chain for the given example. Trails $T_{1,\,2}$, $T_{3,\,5}$, and $T_{6,\,9}$ are coded with 4 + 6 + 8 = 18 bits. Note that an equivalent solution with $s_n = 18$ exists, where the first trail terminates with v_3, as demonstrated with a pair of dashed edges in Fig. 3.

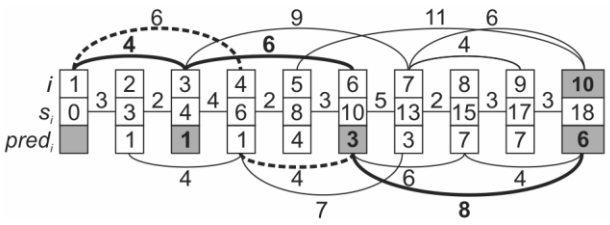


Figure 3: Dynamic programming graph.

The remarkable performance of the dynamic programming-based optimization is highlighted with Theorems 1 and 2. Although the growth of the number of solution candidates is exponential in curve length, the algorithm runs in linear time.

Theorem 1. *The number of possible decompositions of a trail grows exponentially with the trail length (in pixels) if extra codes for patterns of up to k > 1 symbols are used.*

Proof. Let $c_i(k)$ be the number of possible decompositions of $T_{1,\,i}$, where the lengths of nested trails obtained by the decomposition do not exceed k. Each of these decompositions ends with the trail $T_{i-l+1,\,i}$ of l vertices, $1 \leq l \leq min(i, k)$, preceded with one of $c_{i-l}(k)$ possible decompositions of $T_{1,\,i-l}$. Note that for $i < k$, there are less than k symbols available and, thus, the upper bound for the length of the ending trail is $min(i, k)$.

The ending trail $T_{i-l+1,\,i}$ can span through the entire $T_{1,\,i}$ (when $i = l$), resulting in an empty preceding $T_{1,\,0}$. Unlike the definition of trail in Section 2.1, we exceptionally allow $i > j$ here. This situation is indicated by $c_0(k) = 1$.

Equation (3) defines the calculation of $c_i(k)$, $i > 0$.

$$c_i(k) = \sum_{l=1}^{min(i,k)} c_{i-l}(k)\,, i > 0 \tag{3}$$

Obviously, $c_1(k) = c_0(k) = 1$ as the first pixel of $T_{1,\,i}$ can be preceded by an empty trail only in a single way. Similarly, $c_i(1) = 1$ since $T_{1,\,i}$ can be decomposed into single-pixel trails only in a single way. For $k = 2$, $i > 1$, equation (4) is obtained.

$$c_i(2) = c_{i-1}(2) + c_{i-2}(2),\ i > 1 \tag{4}$$

Let F_i represent the i-th Fibonacci number. The Fibonacci sequence is defined by $F_0 = 0$, $F_1 = 1$, and $F_i = F_{i-2} + F_{i-1}$ for $i > 1$. This recursive formula gives $F_2 = 1$ and we may thus match $c_0(k) = F_1$ and $c_1(k) = F_2$. The equation (4) then gives: $c_2(2) = F_2 + F_1 = F_3$, $c_3(2) = F_3 + F_2 = F_4$, and $c_i(2) = F_i + F_{i-1} = F_{i+1}$. As $F_{i+1} > F_i$, $i > 1$, we thus get the inequality (5).

$$c_i(2) > F_i,\ i > 1 \tag{5}$$

This result can be generalized to $k > 2$ by using the relation (6), which must be proved beforehand.

$$c_i(k + m) > c_i(k),\ m \geq 1,\ i > 1 \tag{6}$$

The proof is actually trivial. Due to the transitivity of “Is greater than”, it suffices to consider $m = 1$. The key observation is that all the decompositions counted by $c_i(k)$ are also counted by $c_i(k + 1)$ which, however, additionally counts the decompositions with at least one nested trail of length $k + 1$. The relation (5) may thus be generalized to the relation (7).

$$c_i(k) > F_i,\ k > 1 \wedge i > 1 \tag{7}$$

As the Fibonacci sequence F_i has the proven exponential growth, we may confirm that the sequence $c_i(k)$ also grows (at least) exponentially for $k > 1$. Theorem 1 is thus proved. □

Note that we assumed in the theorem, that all the trails of up to k pixels are represented by edges of the dynamic programming graph, but this is usually not a case due to the statistical model reduction (Section 3.2). The proved exponential growth therefore represents only the theoretical worst case. However, as BFSMs and particularly HCSMs typically contain the majority of the patterns of length 2 ($k = 2$ suffices for the exponential growth) and also quite a few longer patterns, the expected growth may also be considered exponential.

An interesting finding is that the recursion in equation (3) can be solved easily for $k \geq i$. Namely, substitution $c_{i-2}(k) + \ldots + c_0(k) = c_{i-1}(k)$ transforms $c_i(k) = c_{i-1}(k) + \ldots + c_0(k)$ into $c_i(k) = 2\, c_{i-1}(k)$. We may then recursively progress with such substitutions, i.e. $2\, c_{i-1}(k) = 2^2\, c_{i-2}(k) = \ldots = 2^{i-1}\, c_1(k)$, towards the equation (8).

$$c_i(k) = 2^{i-1},\ k \geq i > 0 \tag{8}$$

The result in equation (8) is expected, as any decomposition is obtained by breaking apart the trail in some interruption points between successive pixels. Since $k \geq i$, there are no limitations in the length of nested trails obtained by the decomposition. All the combinations from i single-pixel trails to a single trail spanning through entire $T_{1,\,i}$ are valid. There are $i - 1$ interruption points in a trail with i symbols and thus 2^{i-1} possible decompositions.

Theorem 2. *Optimal chain detection, based on the dynamic programming and utilization of extra codes for patterns of up to k = O(1) symbols, runs in $\Theta(n)$ time, where n is the curve length in pixels.*

Proof. The cardinality $|IN(i)|$, $1 < i \leq n$, cannot exceed k, as each v_i may only represent the end of a trail (edge) of length between 1 and k. The upper bound for time complexity of calculating s_i, $1 < i \leq n$, is thus $O(k\ n) = O(n)$ time if $k = O(1)$. The lower bound however is achieved if the statistical model contains only single-pixel symbols. But even in this case, the linear time is needed to parse the dynamic programming graph. The $\Theta(n)$ time complexity is thus proved. □

3 Training phase

Linear performance proved in Theorem 2 is not the only reason for limiting the length of patterns with attached extra codes to $k = O(1)$ symbols. This also reduces the size of the statistical model, which has a mitigating effect on the context dilution problem. In the proposed study, $k = 6$ have been chosen among different considered values. The reasons for this decision shall be explained in Section 3.2. Even in this way, the statistical model derived from the basic DDCC scheme can theoretically contain $8 + 8^2 + ... + 8^6 = 299{,}592$ entries. Although many of these patterns never appear in practice, and even if we manage to further reduce the size of the statistical model (to some tens entries in practice), there is the only practical possibility to use an averaged statistical model (or more of them). Its derivation requires a careful consideration of the following important issues.

3.1 Training set

In the reported C_DDCC tests [7], relative compression ratio to F8 only slightly varies (between 0.46 and 0.55). This may lead to a conclusion that the derived HCSM serves well for all use cases. Furthermore, similar conclusions can be adopted for practically all existing methods, no matter whether they belong to statistics-based or non-statistical approaches, and whether, in the first case, they use a HCSM or BFSMs. However, we must be aware that the training sets and testing use cases in presentations of these methods usually follow some curve creation and rasterization methodology and, thus, they share some evident common artefacts. In C_DDCC tests, for example, there were a huge probability of shorter sequences of 0° symbols, relatively high probabilities of ⟨±45°, ∓45°⟩ pairs, and rather low probabilities of ±90° symbols. In our method, we may expect even bigger impact of the curve's shape on the compression efficiency, as the distributions of longer patterns from a bigger repertoire can vary considerably from curve to curve. An averaged statistical model can thus deviate significantly from both, the BFSMs of individual training curves used to construct it in the training phase, and the distributions of patterns used to code testing curves in the exploitation phase. Since the latter directly affects the compression efficiency, we decided to use multiple averaged statistical models and, consequently, to classify the training curves and testing use cases regarding some chosen measurable artefacts. In this manner, the method gains generality, as the compression efficiency becomes less dependent on the curve creation and rasterization methodology.

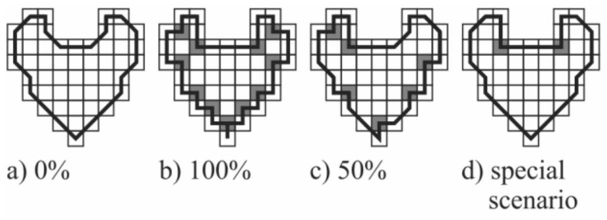


Figure 4: Different levels of forcing the 4-connectivity.

To provide an adequate training set and a relevant mixture of testing use cases, we have implemented a tool with functionalities of image rotation and scaling, manual inversions of binary values of selected pixels, and extraction of the boundary chain of a presented binary object. In this last operation, the parameter *Force-4-connectivity* controls the amount of ±90° symbols along oblique edges and, thus, simulates different rasterization methodologies. Value 0% (Fig. 4a) means that the boundary chain consists only of pixels which share edges with the object's exterior. On the other hand, value 100% (Fig. 4b) adds into the chain all the pixels which are vertex-connected with the object's exterior. Such pixel is 4-connected with both adjacent chain pixels. In Fig. 4c, half of possible pixels of this kind (coloured grey) are randomly chosen and inserted into the chain. Finally, a special scenario is supported (Fig. 4d) where a 4-connected pixel is only inserted if it represents a concave vertex between a horizontal and vertical edge (each at least two pixels long).

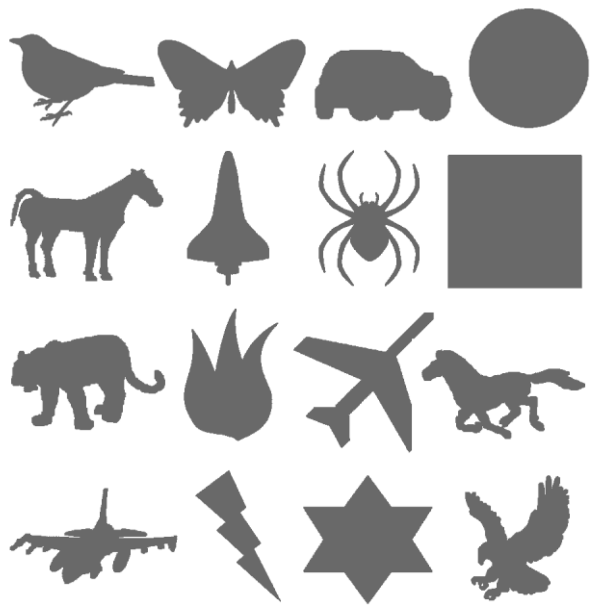


Figure 5: Examples of training and testing objects.

Basic shapes from the training set and use cases are shown in Fig. 5. They were mostly inherited from the tests made in [7, 11, 12]. Objects from the first two rows were used for testing (see Table 1), while the others belong to the training set. All together we used 50 basic shapes, i.e., 30 in the training set and 20 test cases. A variety of instances of these shapes in different orientations and scales, ranging between 150 and 20,000 boundary curve pixels, and with different levels of forcing the 4-connectivity were utilized in the experiments. There are 500 shapes in the training set.

3.2 Statistical model reduction

The first step towards reducing huge amount of data in each BFSM and mitigation of the context dilution effect was already made by limiting k to $O(1)$ symbols. We also do not have to consider patterns with probability 0. Furthermore, we may set even stronger conditions for the probability $p(\Sigma_{i,j})$ of a pattern to be accepted in a BFSM. Namely, a pattern $\Sigma_{i,j} = \langle \sigma_i, ..., \sigma_j \rangle$ is inserted in the statistical model only if $p(\Sigma_{i,j})$ is higher than the product of probabilities (weighted with w_2) of any sequence of shorter patterns whose concatenation forms $\Sigma_{i,j}$. To prevent insertion of too low probabilities, we use additional threshold w_1. The following statement considers patterns of length $l = 3$.

if ($p(\Sigma_{1,\ 3}) > max(w_1,\ w_2 * max(p(\Sigma_{1,\ 1})p(\Sigma_{2,\ 2})p(\Sigma_{3,\ 3}),\ p(\Sigma_{1,1})p(\Sigma_{2,3}),\ p(\Sigma_{1,2})p(\Sigma_{3,3})))$)
then insert (($\Sigma_{1,3}$, $w_3 * 3 * p(\Sigma_{1,3})$)) into BFSM.

As the patterns of lengths 2, 4, 5 and 6 must also be considered, as well as eventual future extensions, we generate all the concatenations algorithmically. A concatenation is obtained by breaking apart the pattern in some interruption points between successive symbols. There are $l - 1$ possible interruption points in a pattern of l symbols and thus $2^{l-1} - 1$ possible concatenations. Here the subtracted 1 represents the non-interrupted pattern. For patterns of lengths 2 to 6, we thus must test 1 + 3 + 7 + 15 + 31 = 57 products. Obviously, the method must first evaluate shorter patterns, as their probabilities are used in acceptance criteria for longer ones. As we mentioned at the end of Section 2.1, single-pixel symbols are unconditionally included in a BFSM.

Note that the weights w_1, w_2, and w_3 offer a lot of possibilities for experimentation. They were also crucial for decision to use patterns of up to $k = 6$ symbols in our tests. As the probabilities are usually decreasing with the pattern length (with possible exceptions), the value of w_1 must be decreased if $k = 7$ is used instead of $k = 6$. However, this causes that additional shorter patterns of lengths 6, 5, 4 etc. are also accepted into a BFSM, increasing the size of the BFSM and emphasizing the context dilution effect in a negative way. On the other hand, this problem appears less evident when comparing $k = 5$ and $k = 6$. Although we have not performed a complete sensitivity analysis yet, the decision for $k = 6$ seems a reasonably good choice confirmed by the results in Section 4.

3.3 Statistical vs. pseudo-statistical model

We do not wish (and neither we are able) to split probabilities of symbols and patterns among some longer patterns, as this would lead to the priority-based greedy approach, which we intentionally try to avoid. Each symbol consequently participates to probabilities of all the patterns, which include it. Strictly speaking, we use weighted probabilities (multiplied with $w_3 * l$) to reward longer patterns by assigning shorter codes to them. The sum of such weighted probabilities in a model may be as high as $(1 + 2 + 3 + 4 + 5 + 6) * w_3 = 21w_3$. It is however lower because the patterns are added selectively, but it still exceeds 1. We apparently do not deal with true statistical models but with pseudo-statistical models instead. We shall use the acronyms BFPSM and HCPSM instead of BFSM and HCSM from this point on. Nevertheless, all weighted “pseudo-probabilities” are involved in a single Huffman tree construction.

3.4 Averaging pseudo-statistical models

Averaging is a two-stage process. During the extraction and reduction of the BFPSM of a considered training curve, several simply assessed curve artefacts are computed. These are then utilized for multicriteria classification, which assigns the curve into one of the pre-defined classes. From all the assessed features that will be used in future to algorithmically select optimal classification criteria, we currently use three intuitively chosen criteria listed below, each with a single threshold.

- Average turn per pixel. Each ±45° symbol participates 1 to this value, ±90° symbols 2, ±135° symbols 3, and 180° symbols 4. The sum is then divided with the curve length in pixels. This feature separates smooth curves from more winding and noisy ones. It is negatively correlated with the probabilities of 0° symbols and their longer runs.
- Probability of ⟨±45°, ∓45°⟩ pairs is higher in curves with oblique segments than in those with mostly axis-aligned and/or ideally diagonal segments.
- Probability of ±90° symbols is usually higher in images of man-made objects than in natural objects.

Three single-threshold criteria result in 8 classes with binary indices from 000 to 111, where the first bit represents the first criterion, and the third bit refers to the last criterion. 0’s signify values below the thresholds, and 1’s those above the thresholds. In the current setting, the thresholds were computed by averaging the described quantities over the BFPSMs of all training curves.

It turns out that the classes with indices $010_{(2)}$ and $101_{(2)}$ are nearly twice more populated than others. In our training set with 500 shapes, there are 112 shapes in the class $101_{(2)}$ and 99 shapes in the class $010_{(2)}$, while the remaining six classes contain between 37 and 55 shapes. The testing use cases are also distributed in a similar way. This deviation can be explained by suboptimal training set, suboptimal thresholds selection and suboptimal classification criteria, which are all among the most important challenges for our future work. However, we may immediately establish that the currently used criteria are all correlated with the *Force-4-connectivity* value. Firstly, all additional 4-connected pixels are coded with ±90° symbols and thus increase the third criterion value. Secondly, such pixels are often inserted in the middle of ⟨±45°, ∓45°⟩ pairs, changing them into ⟨±90°, ∓90°, 0°⟩ triplets. Finally, a pair ⟨±45°, ∓45°⟩ participates 2 to the first criterion (1 per pixel), while a ⟨±90°, ∓90°, 0°⟩ triplet participates 4 (1.33 per pixel). The first and the last criterion are thus positively correlated, and there is a negative correlation between them and the second one. The indices $010_{(2)}$ and $101_{(2)}$ of above-average populated classes also confirm this finding, as the second bit is in both cases the inverse of the other two.

In the second stage, after the training curves are classified (into 8 classes in the current setting), HCSMs are derived by separately averaging BFSMs within each class. However, the BFSMs in a particular class may still significantly differ from each other, although expectedly (and confirmed by the testing results) not as much as the BFSMs from different classes. Consequently, the HCSMs must also be reduced by using the same acceptance criteria as in the BFPSM reduction (Section 3.2).

As we are aware, that the current classification is not optimal, we try to mitigate impacts of wrongly classified training curves by using soft borders between the classes. This means that averaging in an observed class also considers weighted probabilities from BFPSMs of all "adjacent" classes, distinct in one criterion from the considered one. For example, classes 001, 010, 100 are adjacent to the class 000, while, e.g., 011 is not. In the tests presented in Section 4, the probabilities are weighted in a manner that BFPSMs from an observed class contribute two thirds to the corresponding HCPSM, and those from the three adjacent classes contribute a third (a ninth each).

4 Results

In this section, we compare some typical results of the proposed method and some state-of-the-art (SOTA) chain code compression methods. 3OT, VCC, C_DDCC, and three variants of MTFT+ARLE (Move-To-Front Transform + Adaptive Run-Length Encoding) [11], i.e., MTFT+ARLE VCC, MTFT+ARLE 3OT, and MTFT+ARLE NAD (four-symbol Normalised Angle-Difference chain code) [11] were used in the tests.

The training set and use cases from Section 3.1 were used, and the weights w_1, w_2 and w_3 for the pseudo-statistical models reduction (see Section 3.2) were set to 0.02, 1.0 and 1.0, respectively. As we already stressed and explained, the length of patterns to be considered is limited to $k = 6$. The classification thresholds (Section 3.4) computed for the utilized training set were initialized to 0.92 for the average turn, 0.12 for $p(\langle\pm45°, \mp45°\rangle)$, and 0.295 for $p(\pm90°)$.

Object	Transform	Pixels	bpp (SOTA)	bpp (new method)
Basic ("user friendly") shapes				
Bird	100, 0, 0	4080	1.11$^{(1)}$	1.03
Butterfly	100, 0, 0	1122	1.45$^{(1)}$	1.33
Car	100, 0, 0	541	1.48$^{(1)}$	1.25
Circle	100, 0, 0	1831	1.13$^{(2)}$	0.99
Horse	100, 0, 0	2143	1.51$^{(3)}$	1.39
Shuttle	100, 0, 0	969	1.19$^{(1)}$	1.08
Spider	100, 0, 0	1770	1.20$^{(2)}$	1.04
Square	100, 0, 0	1088	0.30$^{(2)}$	0.35
Sophisticated instances				
Bird	10, 50, 70	671	1.60$^{(3)}$	1.31
Butterfly	140, 45, 100	2681	1.68$^{(3)}$	1.21
Car	200, 33, 50	1472	1.84$^{(3)}$	1.49
Circle	20, 0, 0	308	1.39$^{(2)}$	1.06
Horse	50, 15, 20	1284	1.93$^{(1)}$	1.51
Shuttle	100, 30, 0	980	1.31$^{(1)}$	0.94
Spider	120, 45, 25	2218	1.31$^{(1)}$	1.08
Square	100, 70, 30	1228	0.75$^{(3)}$	0.62

Table 1: Test cases and compression results [bpp]. The listed best SOTA results were obtained by C_DDCC$^{(1)}$, MTFT+ARLE NAD$^{(2)}$, or MTFT+ARLE VCC$^{(3)}$.

Table 1 shows the results for pairs of different instances of eight objects from the top two rows in Fig 5. Basic "user friendly" shapes refer to smooth, noiseless instances as being usually employed in testing the state-of-the-art (SOTA) chain code compression methods. The "sophisticated" instances were generated by transforming the basic ones with the scaling factor, rotation angle, and/or amount of additional 4-connectivity pixels different from 100%, 0°, 0%, respectively (see column *Transform*). The column *bpp (SOTA)* shows efficiency in bits per pixel (bpp) of the best of the compared SOTA methods. Comparison of the last two columns reveals that the new method is superior in most cases. The only exception is the basic axis-aligned square where all three MTFT-ARLE variants and also C_DDCC substantially benefit from long runs of 0's.

Ratios between the efficiencies of the new and best SOTA method are given in columns *A* and *B* of Table 2, separately for the basic and transformed instances. The new algorithm is mostly for 10 to 15% more efficient than SOTA in the basic configurations, and for additional 10% in the sophisticated cases. Columns *C* and *D* show ratios between the efficiencies for sophisticated and adequate basic configurations. SOTA is considered in column *C*, and the new method in column *D*. The results confirm that sophisticated curve artefacts much more affect SOTA methods (average ratio 1.22 means lower efficiency for 22%, compared to basic shapes) than the new method (average ratio 1.06). The latter even achieves better compression of some transformed shapes (butterfly and shuttle) in comparison to the basic ones. It also surpasses SOTA in the transformed square example, where all the considered methods achieve significantly worse results (omitted in the above average ratios) than in the axis-aligned instance.

Object	A	B	C	D
Bird	0.93	0.82	1.44	1.27
Butterfly	0.92	0.72	1.16	0.91
Car	0.84	0.81	1.24	1.19
Circle	0.88	0.76	1.23	1.07
Horse	0.92	0.78	1.28	1.08
Shuttle	0.91	0.72	1.10	0.87
Spider	0.87	0.82	1.09	1.04
Square	1.17	0.83	2.50	1.77

Table 2: Analysis of the compression results.

5 Conclusions

In this paper, we introduce a new statistics-based chain code compression methodology by using multiple averaged pseudo-statistical models correlated with some measurable curve artefacts, and by heuristically selecting the most appropriate of these models prior to the compression. Furthermore, the introduced models contain extra codes for systematically selected patterns of up to k symbols ($k = 6$ in the presented tests), and the dynamic programming approach replaces the common greedy method in order to determine the optimal chain of patterns. The early results are promising, but there is a plenty of work left to ultimately affirm the proposed methodology.

The methodology incorporates the training phase and the exploitation phase. The former obviously associates this research with machine learning, but classification of the training curves with respect to three intuitively pre-selected and even mutually correlated criteria is quite far from this paradigm. However, one of our future goals is to adapt the introduced methodology to other basic chain code representations (VCC, 3OT, F4, F8, and NAD), which shall certainly require more advanced and adjustable feature extraction, learning and selection, leading into optimized classification algorithms. This goal also requires an extensive sensitivity analysis by varying the number and values of classification thresholds, weights in the pattern acceptance criteria, etc. Other future goals include:

- comparison to modern non-statistical methods on both, "standard" and less "user-friendly" cases,
- improving the training set and preparation of rich repertoire of benchmarks,
- utilization of arithmetic coding instead of Huffman codes, and
- inclusion of RLE codes for longer patterns of 0's.

6 Acknowledgements

The authors acknowledge the financial support from the Slovenian Research Agency (Research Core Funding No. P2-0041). We are also grateful to our former colleague Denis Špelič for his contribution to the implementation.

7 References

[1] Akimov A.; Kolesnikov A.; Fränti P. (2007). Lossless compression of map contours by context tree modeling of chain codes, *Pattern Recognition*, Elsevier Science, vol. 40, iss. 3, pp. 944-952. https://doi.org/10.1007/11499145_33

[2] Bribiesca E. (1999). A new chain code. *Pattern Recognition*, Elsevier Sci., vol. 32, iss. 2, pp. 235-251. https://doi.org/10.1016/s0031-3203(98)00132-0

[3] Freeman H. (1961). On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers*, IEEE, vol. EC10, iss. 2, pp. 260-268. https://doi.org/10.1109/tec.1961.5219197

[4] Freeman H. (1974). Computer processing of line drawing images. *ACM Computing Surveys*, ACM, vol. 6, iss. 1, pp. 57-97. https://doi.org/10.1145/356625.356627

[5] Jones N. C.; Pevzner P. A. (2004). *An Introduction to Bioinformatics Algorithms*. The MIT Press.

[6] Liu Y. K.; Žalik B. (2005). An efficient chain code with Huffman coding. *Pattern Recognition*, Elsevier Science, vol. 38, iss. 4, pp. 553-557. https://doi.org/10.1016/j.patcog.2004.08.017

[7] Liu Y. K.; Žalik B.; Wang P.-J.; Podgorelec D. (2012). Directional difference chain codes with quasi-lossless compression and run-length encoding. *Signal Processing: Image Commun.*, Elsevier Science, vol. 27, iss. 9, pp. 973-984. https://doi.org/10.1016/j.image.2012.07.008

[8] Liu Y. K.; Wei W.; Wang P.-J.; Žalik B. (2007). Compressed vertex chain codes. *Pattern Recogn.*, Elsevier Science, vol. 40, iss. 11, pp. 2908-2913. https://doi.org/10.1016/j.patcog.2007.03.001

[9] Nunes P.; Marqués F.; Pereira F.; Gasull A. (2000). A contour based approach to binary shape coding using multiple grid chain code. *Signal Processing: Image Communication*, Elsevier Science, vol. 15, iss. 7-8, pp. 585-599. https://doi.org/10.1016/s0923-5965(99)00041-7

[10] Sánchez-Cruz H.; Bribiesca E.; Rodríguez-Dagnino R. M. (2007). Efficiency of chain codes to represent binary objects. *Pattern Recognition*, Elsevier Science, vol. 40, iss. 6, pp. 1660-1674. https://doi.org/10.1016/j.patcog.2006.10.013

[11] Žalik B.; Lukač N. (2014). Chain code lossless compression using Move-To-Front transform and adaptive Run-Length Encoding. *Signal Processing: Image Commun.*, Elsevier Science, vol. 29, iss.1, pp. 96-106. https://doi.org/10.1016/j.image.2013.09.002

[12] Žalik B.; Mongus D.; Lukač N.; Rizman Žalik K. (2018). Efficient chain code compression with interpolative coding. *Information Sciences*, Elsevier Science, vol. 439-440, pp. 39-49. https://doi.org/10.1016/j.ins.2018.01.045

Method for Estimating Tensiomyography Parameters from Motion Capture Data

Dino Vlahek, Tadej Stošić and Tamara Golob
Institute of Computer Science
Faculty of Electrical Engineering and Computer Science at the University of Maribor, Maribor, Slovenia
E-mail: dino.vlahek1@um.si, tadej.stosic1@um.si, t.golob@um.si

Miloš Kalc, Teja Ličen and Matjaž Vogrin
Institute of Sports Medicine, Faculty of Medicine at the University of Maribor, Maribor, Slovenia
E-mail: milos.kalc@ism-mb.si, teja.licen@um.si, matjaz.vogrin@um.si

Domen Mongus
Institute of Computer Science
Faculty of Electrical Engineering and Computer Science at the University of Maribor, Maribor, Slovenia
E-mail: domen.mongus@um.si

Keywords: tensiomyography, marker-based motion capture, 3D points, geometric transformation

Received: June 05, 2020

Tensiomyography is a muscle performance assessment technique that measures its mechanical responses. In this study, we explored the possibility of replacing traditional tensiomyography measurement systems with motion capture. The proposed method allows the measurement of multiple muscle points simultaneously while achieving measurements during a patient's movements. The results showed that approximately 5 *mm error was achieved when estimating maximal muscle displacement, while time delay in muscle contraction and contraction time was assessed with up to* 20 *ms error. As confirmed by physicians, the introduced errors are within the acceptable margin and, thus, the obtained results are medically valid.*

Povzetek: V članku predstavimo novo metodo, ki omogoča večtočkovno merjenje tensiomiografije. Metoda temelji na snemanju mišične kontrakcije s sistemom za zajem gibanja. Rezultati metode in pripadajoče napake so ovrednoteni s strani zdravnikov. Le-ti ocenjujejo, ali so napake še znotraj sprejemljive meje, da so rezultati medicinsko veljavni.

1 Introduction

Tensiomyography (TMG) is a non-invasive mechanomyography method that measures mechanical muscle response based on radial muscle belly displacement induced by the electrical stimulus. The measurement unit usually includes an electrical stimulator, a data acquisition subunit, a digital sensor, and muscle electrodes [28]. TMG output is a displacement-time curve evaluated with the following parameters: Delay time (T_d) is a time difference between the electrical impulse and 10% of the contraction, contraction time (T_c) is a time difference between 10% and 90% of the contraction, sustain time (T_s) is a time difference between 50% of the contraction and 50% of the relaxation, and relaxation time (T_r) is a time difference between 90% and 50% of the relaxation and maximal displacement of the muscle contraction (D_m).

TMG's resulting parameters are usually used for the evaluation of an individual's maximal speed, explosiveness, endurance, and flexibility [16]. They are also applied in the training optimization process in order to prevent negative effects of muscle asymmetry and asynchrony on an individual's performance [19]. Additionally, after an injury, muscle functional capacity can be assessed using TMG, so that the most effective rehabilitation treatment is administered [21], while its usage in medical research includes estimation of muscle composition [24], evaluation of muscle atrophy [10], measuring adaptation to different pathologies [12], and for determination of muscle fiber type composition [6].

However, TMG has significant drawbacks, as it is a fixed, static tool that can perform single-point measurements [28, 10]. Additionally, the reliability of measurement highly depends on the experiences of the measurer, since placements of sensors and electrodes could affect the reliability of the results [24], while measurements are generally performed in a static and relaxed position [28].

In order to address the above-mentioned drawbacks, we propose a method that generates output similar to TMG using marker-based motion capture. The proposed approach allows for measuring multiple points simultaneously, thus reducing the effort required in order to measure muscle contractions. In addition, the measurements can be achieved not only in the relaxed positions but also while moving, as control markers are used in order to stabilize natural limb movement in markers. Accordingly, related work in motion capture is described in Section 2. Section 3 introduces a new method that estimates TMG output from motion capture. The proposed method validation results are presented and discussed in Section 4, while section 5 concludes the paper.

2 Related work

Motion capture allows for recording the movement of objects or people. Various motion capturing systems were introduced recently, including acoustical, mechanical, magnetic, and optical ones. The most widely used systems are optical. They use a camera for recording the motions of markers attached to an object [18]. Two types of markers are used for this purpose, namely, passive and active ones. Passive markers reflect light generated by a near camera lens, while the active ones use their own light source. In any case, 3D positions of markers over time can be reconstructed using optical triangulation, and the estimated trajectories can be used for pinpointing positions of displacements for analysis, visualization, and simulation purposes [11]. Both motion capture systems have been used in the entertainment industry for years as well, where its successful implementation ranges from famous films like Avatar and Lord of the rings [1] to the gaming industry [20]. Optical motion tracking usage, with the support of virtual reality, was also demonstrated for tracking and reconstruction of hand movements for sign language interpretation and dance coaching [26]. Furthermore, optical motion capture technology is today an emerging technology in sports and medicine. For instance, its usage was examined for the purposes of facial performance acquisition [13], animation of the natural bending, bulging, and jiggling [4], reconstruction of three-dimensional rotations of human joints [7], and gait analysis [3]. Within this context, the efficiency comparison of marker-less and marker-based motion capture for gait analysis was conducted, where the authors concluded that maker-based motion capture is more suitable for clinical use. A more recent study, however, has shown that motion capture, in general, can introduce errors due to linear scaling and technology imperfection [14]. Here, the musculoskeletal models of different centers of joints, obtained from marker-based motion capture, were scaled and compared with measurements obtained from MRI images that are today believed to be the gold standard.

Nevertheless, optical motion capture is still widely used in sport gesture analysis that ranges from repetitive stresses and movements on the shoulder [23] to underwater body motions [2]. Moreover, efficient utilization of motion capture technology for medical uses was proposed in [22, 25]. In addition, motion capture technology was successfully used for the rehabilitation of patients with spastic hemiplegic cerebral palsy [15] and Duchenne muscular dystrophy [9]. Thus, as marker-based motion capture is frequently used for gait and skeleton analysis in sports medicine and animating 3D objects in the entertainment industry, it provides a solid technological foundation for our study.

3 Method

In this section, a method for estimating TMG parameters from 3D motion capture data is presented. The proposed method uses a set of markers in order to trace muscle contraction using motion capture, while TMG parameters are estimated during the following steps:

- **Point stabilization** is achieved first in order to compensate for natural limb movements and preserve only those movements that result from muscle contractions.
- **Construction of displacement-time curves** is achieved next by estimating displacement distances from stabilized 3D marker positions.
- **Extraction of TMG parameters** is finally achieved based on the estimated displacement-time curve.

Following the description of the mathematical framework, these steps are described in detail.

3.1 Theoretical background

The implementation of the proposed mathematical framework is given in the homogeneous coordinate system. This allows for implementing all the used geometric transformations, including translation, by matrix multiplication and, thus, enables efficient utilization of a graphic processing unit [8].

Let a set of markers $M = \{{}^t m_i\}$, where ${}^t m_i = [{}^t x_i, {}^t y_i, {}^t z_i, 1]$, while i is a markers index and t is the time t of its capture. A vector between points ${}^t m_i$ and ${}^t m_j$ is denoted as ${}^t\vec{v}_{i,j} = {}^t m_i - {}^t m_j$, while its projections to $XY-$ and $XZ-$planes are denoted as ${}^t u_{i,j} = ({}^t x_{i,j}, {}^t y_{i,j}, 1)$ and ${}^t w_i = ({}^t x_{i,j}, {}^t z_{i,j}, 1)$, respectively. A translation for an arbitrary vector ${}^t\vec{v}_{i,j} = ({}^t x_{i,j}, {}^t y_{i,j}, {}^t z_{i,j})$ is then given by a translation matrix M_T, defined as

$$M_T(\vec{v}_T) = \begin{bmatrix} 1 & 0 & 0 & {}^t x_{i,j} \\ 0 & 1 & 0 & {}^t y_{i,j} \\ 0 & 0 & 1 & {}^t z_{i,j} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{1}$$

In addition, rotation matrices $M_{R_y}(\Theta_y)$ and $M_{R_z}(\Theta_z)$ that define rotation around $Y-$ and $Z-$axis for given angles Θ_y and Θ_z, respectively, are denoted by

$$\begin{aligned} M_{R_y}(\Theta_y) &= \begin{bmatrix} cos\Theta_y & 0 & sin\Theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -sin\Theta_y & 0 & cos\Theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\ M_{R_z}(\Theta_z) &= \begin{bmatrix} cos\Theta_z & -sin\Theta_z & 0 & 0 \\ sin\Theta_z & cos\Theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2}$$

[27].

3.2 Point stabilization

In order to account for the natural movement of a limb, two control markers were placed on the limb joints in such a way that they were not affected by muscle contractions. They are denoted by the indices $i = 1$ and $i = 2$, while the corresponding vector ${}^t\vec{v}_{1,2}$ was used to stabilize the set of markers M (see Figure 1). In order to achieve stabilization, the origin of coordinate system was shifted to the control marker $i = 1$, while the $X-$axis was aligned with ${}^t\vec{v}_{1,2}$. Note that the latter only requires rotation around $Y-$ and $Z-$axis, while the rotation around $X-$axis can be neglected due to the nature of measurement that limits such limb movements. Thus, rotations around $Y-$ and $Z-$axis were denoted by rotation angles Θ_y and Θ_z, defined as angles between projected vectors ${}^t\vec{u}_{1,2}$ and ${}^t\vec{w}_{1,2}$ and the $X-$axis, respectively [27]. This stabilization, denoted as M_S, is formally defined as

$$M_S = M_T({}^tm_1) \cdot M_{R_y}(\Theta_y) \cdot M_{R_z}(\Theta_z) = \begin{bmatrix} cos^t\Theta_y cos^t\Theta_z & -sin^t\Theta_z cos^t\Theta_y & sin^t\Theta_y & {}^tx_1 \\ sin^t\Theta_z & 0 & 0 & {}^ty_1 \\ -sin^t\Theta_y cos^t\Theta_z & sin^t\Theta_y sin^t\Theta_z & cos^t\Theta_y & {}^tz_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3}$$

where $M_T({}^tm_1)$ denotes translation of the origin of coordinate system to control marker $i = 1$, while $M_{R_y}(\Theta_y)$ and $M_{R_z}(\Theta_z)$ rotations by Θ_y and Θ_z, respectively. Moreover, stabilized set of markers $S = \{{}^ts_i\}$, where ${}^ts_i = ({}^tx'_i, {}^ty'_i, {}^tz'_i)$ is, thus, given as:

$${}^ts_i = M_S \cdot {}^tm_i. \tag{4}$$

3.3 Construction of displacement-time curves and TMG parameters extraction

This step aims to construct a displacement time curves from S and extract the required TMG parameters. As muscle contraction is captured by the movement of stabilized markers, a displacement curve for each marker ${}^ts_i \in S$ is generated by measuring its distance ${}^td_i$ in time $t > 0$ from its starting point, given at $t = 0$. Formally, a displacement curve is given by a discrete mapping function $D : (t, i) \to \mathbb{R}$ defined by:

$$D(t, i) = \sqrt{({}^0s_i - {}^ts_i)^2}. \tag{5}$$

As Eq. 5 cannot produce negative values, it is critical that the initial measurement given at time $t = 0$ is measured in the relaxing (non-contracted) state of the muscle. $D(t, i)$, thus, provides a set of control points based on which a polynomial interpolation is achieved in order to increase the precision of the estimated TMG parameters. As polynomial interpolation is a well-know problem, it is not further discussed here. Its efficient implementation is described in [5]. Moreover, as explained in Section 1, five parameters can be extracted from a displacement curve, where most of the medically relevant information is contained in maximal contraction D_m, delay time T_d, and contraction time T_c. Given an interpolated displacement curve $d_i(t)$, definitions are as follows:

$$\begin{aligned} D_m(i) &= \max_t \ d_i(t), \\ T_d(i) &= \arg\min_t(t; d_i(t) \geq 0.1 \cdot D_m(i)), \\ T_c(i) &= T_d(i) - \arg\min_t(t; d_i(t) \geq 0.9 \cdot D_m(i)). \end{aligned} \tag{6}$$

4 Results and discussion

The proposed method's implementation was done using C++, and experiments were conducted on a workstation with Intel® Core^{TM} i5 CPU and 16 GB of main memory. Experimental data about twelve different participants were collected using a 4×5 matrix of reflective markers that were placed on the quadriceps femoris of participants' left leg, while two control markers were placed over the trochanter head and lateral condyle (see Fig. 1). The participants were instructed to lie supine on a therapeutic table where each placed its left leg on a triangular cushion that provided approximately 30° knee angle support. Then, Rectus Femoris (RF - the upper central part of the thigh) and Vastus Medialis (VM - lower internal part of the thigh) muscles were stimulated with a single electrical impulse provided by a high voltage constant current electrical stimulator, while control measurements were obtained using a traditional TMG sensor (TMG-BMC Ltd, Ljubljana, Slovenia). One series of these measurements consisted of five consecutive muscle stimulation with a 5 s interstimulus intervals in order to prevent post-activation potentiating. For each muscle, six different sets of stimulations were administrated, starting with the stimulation intensity of 30 mA, increasing the power in each measurement by 10 mA, until a maximum of 80 mA was reached. Thus, a total of 30 stimuli for each muscle were measured. At the same time, the same muscle contractions were captured from reflective markers with a Smart-D, BTS s.p.a. motion capture system. The system consisted of eight infrared cameras with 800×600 spatial and 60 Hz temporal resolution, while their position at the therapeutic table is shown in Figure 2.

At each marker, the measured motion capture data was used in order to reconstruct the displacement-time curves, while their agreement with the control TMG curve was estimated in terms of Pearson correlation coefficient [17]. The obtained results are shown in Table 1. Obviously, the displacement-time curves showed a different agreement level with the control TMG measurements, depending on the markers' proximities to the TMG sensor. On average, VM measurements displayed lower correlations with control ones than those performed on RF due to the dilated oscillations of the muscular surface, while those markers

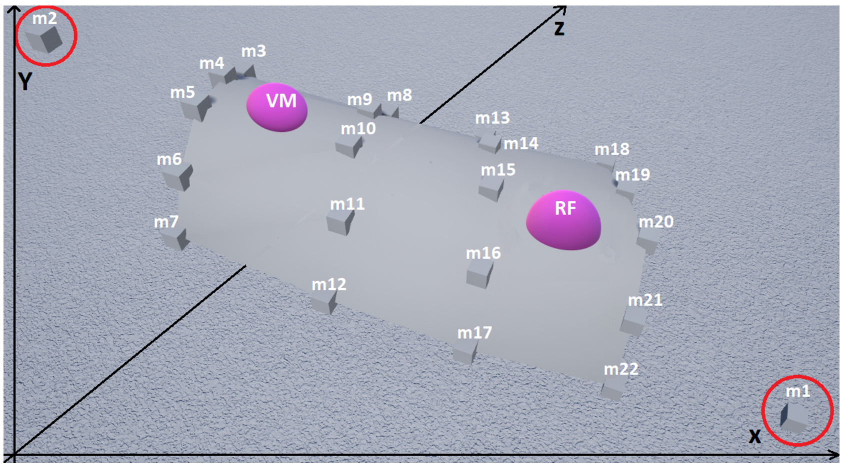


Figure 1: The placement of twenty markers and two control markers on the subject's leg. The Violet area represents the placement of the TMG sensor during measurement, while red circles indicate control markers.

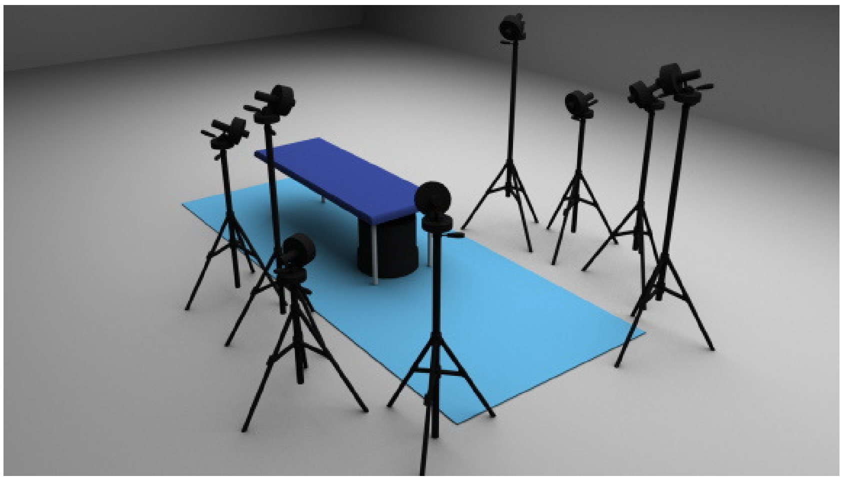


Figure 2: Position of cameras where measurements were performed.

placed near the TMG sensor displayed better correlation in both cases. As follows from Table 1, in the case of VM, the highest correlations with control TMG were measured at $m5$ and $m9$, while $m15$ displayed best results in case of RF stimulation. In addition, the results obtained at $m11$, $m16$, $m19$, and $m20$ were also statistically significant in both cases. Such results are expected as these markers were located in anatomical regions of measured muscles. Displacement-time curves from markers that show the highest agreement with corresponding control TMG curves are further presented in Figure 3, while TMG parameters were extracted from these particular markers and further examined.

In order to assess the accuracy of the extracted TMG parameter, their values were estimated from displacement-time curves generated from markers. Moreover, parameters error rates represent differences between their values and the parameters' values from corresponding control TMG. The results are shown in the appendix (Tables 2 – 7). When considering T_c and T_d for VM, the lowest error rates were observed in case of $m5$ at 50 mA and $m20$ at 50 mA with 0.2% and 0%, respectively, while error rates between $1.1 - 25.3\%$ in case of T_c and $3 - 30.4\%$ in case T_d were observed in other cases. On the other hand, no error was observed for D_m at $m20$ at 60 mA, while the error rates in other cases ranged between $1.7 - 61.7\%$. In the case of RF, T_c error rates were in the range of $0.9 - 33.7\%$, with the smallest related to $m20$ at 30 mA. However, T_d introduced inconsistent error rates, from 1.8% in case of $m16$ at 30 mA, up to 75.7% in case of $m11$ at 60 mA. D_m error rates were between $6.4 - 33.7\%$, where the lowest one is associated with $m19$ at 80 mA.

According to the evaluation provided by the medical experts, the obtained error rates were within the acceptable ranges and can be considered as medically irrelevant. The error of D_m can be explained by the fact that the TMG sensor is slightly pressed into the soft tissue, resulting in a small depression at a baseline level, causing a higher value of D_m when a traditional TMG is measured. As expected, there were high errors in the T_d parameter since the signals from motion capture and TMG were not properly synchronized. Additionally, obtained errors could be explained by

Table 1: Pearson correlation coefficients between the displacement-time curves of markers and control TMG measurements together with average result according to each marker.

	30 mA		40 mA		50 mA		60 mA		70 mA		80 mA		Average	
	VM	RF	VM	RF	VM	RF	VM	RF	VM	RF	VM	RF	VM	RF
m3	0.652	0.774	0.527	0.803	0.497	0.81	0.514	0.716	0.607	0.701	0.657	0.628	0.575	0.738
m4	0.696	0.745	0.543	0.721	0.504	0.672	0.529	0.557	0.604	0.583	0.698	0.512	0.595	0.632
m5	0.74	0.837	0.606	0.842	0.581	0.839	0.569	0.75	0.637	0.747	0.729	0.672	0.644	0.781
m6	0.604	0.77	0.508	0.76	0.584	0.785	0.538	0.699	0.647	0.732	0.681	0.638	0.594	0.731
m7	0.575	0.735	0.393	0.779	0.352	0.767	0.386	0.677	0.478	0.756	0.515	0.664	0.45	0.73
m8	0.745	0.766	0.578	0.725	0.559	0.686	0.529	0.625	0.63	0.691	0.584	0.644	0.604	0.689
m9	0.73	0.818	0.578	0.825	0.629	0.809	0.621	0.699	0.628	0.723	0.681	0.666	0.644	0.757
m10	0.734	0.732	0.596	0.789	0.578	0.761	0.555	0.767	0.646	0.801	0.718	0.697	0.638	0.757
m11	0.746	0.761	0.573	0.839	0.607	0.845	0.659	0.743	0.652	0.829	0.666	0.644	0.65	0.777
m12	0.531	0.748	0.374	0.796	0.35	0.825	0.365	0.724	0.425	0.79	0.453	0.65	0.417	0.755
m13	0.612	0.738	0.482	0.759	0.431	0.717	0.423	0.656	0.567	0.613	0.55	0.593	0.511	0.679
m14	0.647	0.732	0.525	0.727	0.516	0.668	0.494	0.626	0.605	0.635	0.607	0.654	0.566	0.674
m15	0.729	0.878	0.562	0.846	0.537	0.858	0.6	0.758	0.665	0.81	0.638	0.718	0.622	0.811
m16	0.731	0.789	0.574	0.797	0.589	0.826	0.632	0.724	0.631	0.758	0.669	0.679	0.637	0.762
m17	0.528	0.816	0.45	0.813	0.422	0.801	0.412	0.758	0.408	0.776	0.43	0.693	0.441	0.776
m18	0.567	0.558	0.427	0.603	0.381	0.639	0.406	0.558	0.536	0.535	0.523	0.496	0.473	0.565
m19	0.712	0.795	0.661	0.761	0.62	0.817	0.53	0.714	0.668	0.785	0.647	0.703	0.639	0.763
m20	0.61	0.789	0.632	0.837	0.613	0.819	0.561	0.788	0.698	0.71	0.665	0.691	0.63	0.772
m21	0.641	0.824	0.527	0.796	0.53	0.795	0.484	0.727	0.609	0.761	0.617	0.589	0.563	0.749
m22	0.483	0.76	0.439	0.801	0.542	0.774	0.504	0.688	0.617	0.801	0.576	0.67	0.527	0.749

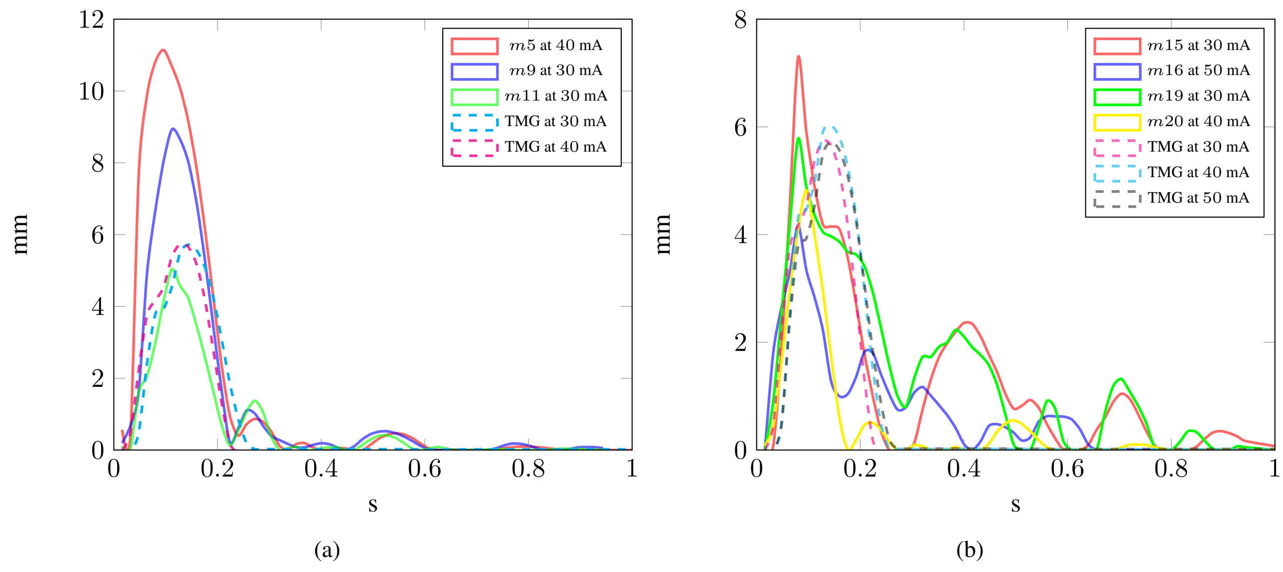


Figure 3: Displacement-time curves from traditional TMG (dashed lines) and corresponding markers (solid lines) that produced the highest level of agreement with traditional TMG for muscle a) VM and b) RF. On the $x-$axis, there is time in s, while the $y-$axis represents displacement in mm.

the fact that the TMG measurement unit provides more precise measurements because of its 1000 Hz temporal resolution when comparing it with 60 Hz of the motion capture system. On the other hand, markers $m19$ and $m11$ registered significant movements, even though they were not placed in the anatomical regions, where contraction of RF and VM was expected. Such an outcome might have different explanations:

- strong electrical stimulation can cause the propagation of the electrical stimuli in deeper tissues, causing muscle contraction of adjacent muscles,
- the passive mass, represented by inactivated muscles and adipose tissue near the stimulated region, can vibrate, causing errors in measurements.

5 Conclusion

A new method for estimating TMG parameters from 3D motion capture, proposed in this paper, allows for measurement of TMG parameters at multiple points simultaneously, while measurements can be obtained during the patient's movement. With the error rates of 5 mm when estimating maximal muscle displacement and up to 20 ms when estimating delay time and contraction time, the provided results proved to be medically relevant. Nevertheless, selection and proper placement of markers are required. One of the future tasks is a synchronization of the TMG and motion capture signals that would allow us to obtain the exact starting time of muscle contraction and, thus, further improved contraction and delay time assessment. In addition, improved point stabilization with compensating for rotations along the X-axis will be considered. Finally, as the described study provides validation of the proposed method from the engineering point of view, the extended medical one is required to prove its real value.

Acknowledgement

This work was supported by the Slovenian Research Agency under Grants J2-8176 and P2-0041, and the project "Wearable Integrated Smart Brace for Rehabilitation Monitoring and Diagnostic of Disorders in Muscular Functions - WIBRANT" that is co-financed by the Republic of Slovenia, Ministry of Education, Science and Sport, and the European Union under the European Regional Development Fund. More info: `www.eu-skladi.si/?set_language=en`.

References

[1] Rufino R. Ansara. and Chris Joslin. Adding cartoon-like motion to realistic animations. In *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP, (VISIGRAPP 2017)*, pages 137–147. INSTICC, SciTePress, 2017. https://doi.org/10.5220/0006174001370147.

[2] Gustavo R. D. Bernardina, Pietro Cerveri, Ricardo M. L. Barros, João C. B. Marins, and Amanda P. Silvatti. Action sport cameras as an instrument to perform a 3d underwater motion analysis. *PLOS ONE*, 11:1–14, 08 2016. https://doi.org/10.1371/journal.pone.0160490.

[3] Elena Ceseracciu, Zimi Sawacha, and Claudio Cobelli. Comparison of markerless and marker-based motion capture technologies through simultaneous data collection during gait: Proof of concept. *PLOS ONE*, 9(3):1–7, 03 2014. https://doi.org/10.1371/journal.pone.0087640.

[4] Caecilia Charbonnier, Frank C. Kolo, Victoria B. Duthon, Nadia Magnenat-Thalmann, Christoph D. Becker, Pierre Hoffmeyer, and Jacques Menetrey. Assessment of congruence and impingement of the hip joint in professional ballet dancers: A motion capture study. *The American Journal of Sports Medicine*, 39(3):557–566, 2011. https://doi.org/10.1177/0363546510386002.

[5] M. F. I. Chowdhury, C. Jeannerod, V. Neiger, E. Schost, and G. Villard. Faster algorithms for multivariate interpolation with multiplicities and simultaneous polynomial approximations. *IEEE Transactions on Information Theory*, 61(5):2370–2387, May 2015. https://doi.org/10.1109/TIT.2015.2416068.

[6] Raja Dahmane, Srdjan Djordjevic, Bostjan Simunic, and Vojko Valencic. Spatial fiber type distribution in normal human muscle Histochemical and tensiomyographical evaluation. *Journal of biomechanics*, 38(12):2451–2459, 2005. https://doi.org/10.1016/j.jbiomech.2004.10.020.

[7] R. Degeorges, J. Parasie, D. Mitton, N. Imbert, J.-N. Goubier, and F. Lavaste. Three-dimensional rotations of human three-joint fingers: an optoelectronic measurement. preliminary results. *Surgical and Radiologic Anatomy*, 27(1):43–50, Mar 2005. https://doi.org/10.1007/s00276-004-0277-4.

[8] Peng Du, Rick Weber, Piotr Luszczek, Stanimire Tomov, Gregory Peterson, and Jack Dongarra. From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming. *Parallel Computing*, 38(8):391 – 407, 2012. https://doi.org/10.1016/j.parco.2011.10.002.

[9] Raluca Ganea, Pierre-Yves Jeannet, A. Paraschiv-Ionescu, Nathalie Goemans, Christine Piot, Marleen Hauwe, and Kamiar Aminian. Gait assessment in children with duchenne muscular dystrophy during long-distance walking. *Journal of child neurology*, 27:30–8, 07 2011. https://doi.org/10.1177/0883073811413581.

[10] Oscar Garcia-Garcia, Alba Cuba-Dorado, Tania Alvarez-Yates, Javier Carballo-Lopez, and Mario Iglesias-Caamano. Clinical utility of tensiomyography for muscle function analysis in athletes. *Open Access Journal of Sports Medicine*, 10:49–69, 2019. https://doi.org/10.2147/OAJSM.S161485.

[11] Michael Gleicher. Animation from observation: Motion capture and motion editing. *SIGGRAPH Comput. Graph.*, 33(4):51–54, November 1999. https://doi.org/10.1145/345370.345409.

[12] K. Grabljevec, H. Burger, K. Kersevan, V. Valencic, and C. Marincek. Strength and endurance of knee extensors in subjects after paralytic poliomyelitis.

Disability and Rehabilitation, 27(14):791–799, July 2005. https://doi.org/10.1080/09638280400020623.

[13] Haoda Huang, Jinxiang Chai, Xin Tong, and Hsiang-Tao Wu. Leveraging motion capture and 3D scanning for high-fidelity facial performance acquisition. *ACM Trans. Graph.*, 30(4):74:1–74:10, July 2011. https://doi.org/10.1145/2010324.1964969.

[14] Hans Kainz, Hoa Hoang, Chris Stockton, Roslyn Boyd, David Lloyd, and Christopher Carty. Accuracy and reliability of marker based approaches to scale the pelvis, thigh and shank segments in musculoskeletal models. *Journal of Applied Biomechanics*, 33:1–21, 03 2017. https://doi.org/10.1123/jab.2016-0282.

[15] M.C.M. Klotz, L. Kost, F. Braatz, V. Ewerbeck, D. Heitzmann, S. Gantz, T. Dreher, and S.I. Wolf. Motion capture of the upper extremity during activities of daily living in patients with spastic hemiplegic cerebral palsy. *Gait & Posture*, 38(1):148–152, 2013. https://doi.org/10.1016/j.gaitpost.2012.11.005.

[16] Irineu Loturco, Saulo Gil, Cristiano Laurino, Hamilton Roschel, Ronaldo Kobal, Cesar Abad, and Fabio Nakamura. Differences in muscle mechanical properties between elite power and endurance athletes: A comparative study. *The Journal of Strength and Conditioning Research*, 29(6):1723–1728, 12 2014. https://doi.org/10.1519/JSC.0000000000000803.

[17] J. Neyman and E. S. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231:289–337, 1933. https://doi.org/10.1098/rsta.1933.0009.

[18] Haldun M. Ozaktas and Levent Onural. *Three-Dimensional Television: Capture, Transmission, Display*. Springer Science & Business Media, 2007. https://doi.org/10.1007/978-3-540-72532-9.

[19] Ezequiel Rey, Carlos Lago-Penas, and Joaquin Lago-Ballesteros. Tensiomyography of selected lower-limb muscles in professional soccer players. *Journal of Electromyography and Kinesiology*, 22(6):866 – 872, 2012. https://doi.org/10.1016/j.jelekin.2012.06.003.

[20] A. L. Rincon, H. Yamasaki, and S. Shimoda. Design of a video game for rehabilitation using motion capture, emg analysis and virtual reality. In *2016 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pages 198–204, Feb 2016. https://doi.org/10.1109/CONIELECOMP.2016.7438575.

[21] Pedro S Dias, Joan S Fort, Daniel A Marinho, Albano Santos, and MÃąrio Marques. Tensiomyography in physical rehabilitation of high level athletes. *The Open Sports Sciences Journal*, 3:47–48, 03 2014. https://doi.org/10.2174/1875399X010030100047.

[22] L. A. Schwarz, A. Mkhitaryan, D. Mateus, and N. Navab. Estimating human 3D pose from time-of-flight images based on geodesic distances and optical flow. In *Face and Gesture 2011*, pages 700–706, 2011. https://doi.org/10.1109/FG.2011.5771333.

[23] Elena Seminati, Alessandra Marzari, Oreste Vacondio, and Alberto Minetti. Shoulder 3D range of motion and humerus rotation in two volleyball spike techniques: injury prevention and performance. *Sports Biomechanics*, 14(2):216–231, 07 2015. https://doi.org/10.1080/14763141.2015.1052747.

[24] Bostjan Simunic. Between-day reliability of a method for non-invasive estimation of muscle composition. *Journal of electromyography and kinesiology*, 22:527–30, 04 2012. https://doi.org/10.1016/j.jelekin.2012.04.003.

[25] Ivo Stancic, Tamara Grujic Supuk, and Ante Panjkota. Design, development and evaluation of optical motion-tracking system based on active white light markers. *IET Science, Measurement & Technology*, 7:206–214(8), July 2013. https://doi.org/10.1049/iet-smt.2012.0157.

[26] Jeff Tang, J.C.P. Chan, and H. Leung. Interactive dancing game with real-time recognition of continuous dance moves from 3D human motion capture. *5th International Conference on Ubiquitous Information Management and Communication*, pages 50–58, 01 2011. https://doi.org/10.1145/1968613.1968674.

[27] John Vince. *Mathematics for Computer Graphics*. Springer-Verlag, Berlin, Heidelberg, 5th edition, 2014. https://doi.org/10.1007/978-1-4471-7336-6.

[28] T. Zagar and D. Krizaj. Validation of an accelerometer for determination of muscle belly radial displacement. *Medical and Biological Engineering and Computing*, 43(1):78–84, February 2005. https://doi.org/10.1007/BF02345126.

6 Appendix

Table 2: The table shows average values and associated standard deviation for parameters extracted from markers at a stimulation intensity of 30 mA. Associated errors were placed below the results. The lowest errors are highlighted.

	Vastus medialis			Rectus femoris		
	$T_c\ (ms)$	$T_d\ (ms)$	$D_m\ (mm)$	$T_c\ (ms)$	$T_d\ (ms)$	$D_m\ (mm)$
m5	39 (± 15.95)	27.2 (± 11.43)	6.5 (± 3.58)	47.2 (± 7.14)	36.7 (± 8.21)	3.4 (± 1.16)
m9	41.8 (± 17.95)	19.4 (± 3.23)	9 (± 5.08)	55.1 (± 7.56)	28.3 (± 9.84)	3 (± 1.43)
m11	43.5 (± 19.18)	21.6 (± 5.48)	5.3 (± 3.31)	59.2 (± 15.13)	27.6 (± 11.02)	2.9 (± 1.8)
m15	44.8 (± 20.08)	21.9 (± 4.64)	7.4 (± 4.74)	49.4 (± 6.01)	33.8 (± 6.75)	4.9 (± 1.8)
m16	45.3 (± 17.85)	22 (± 4.47)	5.2 (± 3.28)	50.5 (± 10.46)	26.1 (± 10.68)	3.5 (± 1.11)
m19	45.7 (± 18.22)	25.4 (± 9.35)	5 (± 3.62)	50.4 (± 6.9)	34.8 (± 11.04)	2.8 (± 1.35)
m20	41.9 (± 17.6)	29.1 (± 10.43)	4.8 (± 3.42)	51 (± 4.5)	29.8 (± 7.58)	3.3 (± 1.33)
TMG	36.5 (± 14.11)	24.2 (± 2.93)	4.9 (± 1.31)	51.5 (± 20.1)	25.6 (± 3.37)	4.3 (± 1.4)
	$Error(T_c)$	$Error(T_d)$	$Error(D_m)$	$Error(T_c)$	$Error(T_d)$	$Error(D_m)$
m5	**2.5** (6.9 %)	**3.0** (12.4 %)	**1.6** (33.7%)	4.3 (8.4%)	11.1 (43.2%)	0.9 (20.4%)
m9	5.4 (14.7 %)	4.8 (19.8 %)	4.1 (83.7%)	3.6 (6.9%)	2.7 (10.4%)	1.3 (30.9%)
m11	7.1 (19.4 %)	2.6 (10.8 %)	0.4 (8.7%)	7.7 (14.9%)	2.0 (8.0%)	1.5 (33.8%)
m15	8.3 (22.9 %)	2.3 (9.3 %)	2.5 (51.1%)	2.1 (4.1%)	8.2 (32.2%)	0.5 (12.5%)
m16	8.8 (24.2 %)	2.2 (9.0 %)	0.3 (6.8%)	**1.0** (1.9%)	**0.5** (1.8%)	**0.8** (18.8%)
m19	9.2 (25.3 %)	1.2 (5.0 %)	0.2 (3.2%)	1.1 (2.2%)	9.2 (36.0%)	1.5 (33.8%)
m20	5.4 (14.9 %)	4.9 (20.4 %)	0.1 (1.9%)	0.5 (0.9%)	4.2 (16.4%)	1.0 (22.9%)

Table 3: The table shows average values and associated standard deviation for parameters extracted from markers at a stimulation intensity of 40 mA. Associated errors were placed below the results. The lowest errors are highlighted.

	Vastus medialis			Rectus femoris		
	$T_c\ (ms)$	$T_d\ (ms)$	$D_m\ (mm)$	$T_c\ (ms)$	$T_d\ (ms)$	$D_m\ (mm)$
m5	30.9 (± 11.9)	29.1 (± 16.9)	6.9 (± 3.7)	47 (± 10.8)	35.4 (± 13.7)	4 (± 1)
m9	33.1 (± 14.5)	24 (± 14.1)	6.9 (± 5.6)	50.9 (± 7.8)	33.7 (± 10.6)	3.8 (± 1.7)
m11	35.3 (± 16)	24.8 (± 12.7)	5.5 (± 2.6)	55.3 (± 14.5)	30.3 (± 9.9)	3.8 (± 2.1)
m15	33.6 (± 15.1)	26.2 (± 13.7)	7.8 (± 4.7)	50.2 (± 7.3)	33.9 (± 7.2)	6.2 (± 1.7)
m16	36.2 (± 14)	25.9 (± 13.8)	5.3 (± 3)	54.3 (± 10)	29.1 (± 10.1)	4.3 (± 2)
m19	35.3 (± 13.4)	28.7 (± 15.5)	5.4 (± 4.1)	49.3 (± 6.1)	34.9 (± 10.8)	3.2 (± 0.8)
m20	35.6 (± 12.5)	28 (± 13.8)	5.1 (± 3.8)	49.9 (± 6.8)	32 (± 11)	4.3 (± 1.4)
TMG	31.3 (± 10.9)	23.1 (± 2.5)	5.7 (± 1.8)	45.1 (± 18.3)	24.8 (± 3.1)	4.9 (± 1.6)
	$Error(T_c)$	$Error(T_d)$	$Error(D_m)$	$Error(T_c)$	$Error(T_d)$	$Error(D_m)$
m5	0.3 (1.1 %)	6 (25.8 %)	1.2 (20.3%)	1.9 (4.2%)	10.7 (43%)	0.9 (18.2%)
m9	**1.8** (5.7 %)	**0.9** (3.8 %)	**1.2** (21.0%)	5.8 (12.9%)	8.9 (36%)	1 (21.4%)
m11	4.1 (13 %)	1.6 (7.1 %)	0.2 (3.5%)	10.2 (22.7%)	5.5 (22.3%)	1.1 (21.7%)
m15	2.3 (7.4 %)	3.1 (13.2 %)	2.1 (36.3%)	5.1 (11.3%)	9.1 (36.9%)	1.3 (26.9%)
m16	4.9 (15.8 %)	2.8 (12.1 %)	0.4 (6.2%)	**9.2** (20.5%)	**4.3** (17.6%)	**0.6** (11.8%)
m19	4 (12.8 %)	5.6 (24.1 %)	0.4 (6.2%)	4.2 (9.4%)	10.1 (40.8%)	1.6 (33.9%)
m20	4.3 (13.7 %)	4.8 (20.9 %)	0.6 (10.8%)	4.9 (10.8%)	7.2 (29.3%)	0.6 (11.9%)

Table 4: The table shows average values and associated standard deviation for parameters extracted from markers at a stimulation intensity of 50 mA. Associated errors were placed below the results. The lowest errors are highlighted.

	Vastus medialis			Rectus femoris		
	$T_c\ (ms)$	$T_d\ (ms)$	$D_m\ (mm)$	$T_c\ (ms)$	$T_d\ (ms)$	$D_m\ (mm)$
m5	28.93 (± 11.01)	23.89 (± 8.60)	7.35 (± 3.76)	48.14 (± 4.01)	38.70 (± 13.90)	12.86 (± 1.29)
m9	31.41 (± 12.99)	19.56 (± 4.33)	10.07 (± 5.65)	49.42 (± 4.20)	39.29 (± 16.25)	6.19 (± 2.01)
m11	33.45 (± 15.45)	19.91 (± 3.23)	6.62 (± 2.65)	46.98 (± 4.67)	39.21 (± 17.36)	9.83 (± 1.75)
m15	30.44 (± 14.53)	20.99 (± 5.26)	9.08 (± 4.76)	50.57 (± 6.98)	36.57 (± 9.64)	8.00 (± 1.73)
m16	33.12 (± 14.47)	21.45 (± 6.30)	6.50 (± 2.92)	48.79 (± 4.89)	39.78 (± 13.43)	4.56 (± 1.72)
m19	31.90 (± 13.95)	23.77 (± 8.49)	6.14 (± 4.32)	49.07 (± 3.34)	37.13 (± 11.97)	4.31 (± 0.78)
m20	31.93 (± 14.84)	23.11 (± 7.21)	6.09 (± 3.83)	53.01 (± 4.79)	29.9 (± 11.49)	9.15 (± 1.21)
TMG	28.86 (± 9.14)	23.08 (± 2.24)	6.31 (± 1.88)	42.04 (± 5.22)	24.44 (± 2.98)	18.97 (± 1.66)
	$Error(T_c)$	$Error(T_d)$	$Error(D_m)$	$Error(T_c)$	$Error(T_d)$	$Error(D_m)$
m5	**0.1** (0.2 %)	**0.8** (3.5 %)	**1.0** (16.5%)	6.1 (14.5%)	14.3 (58.3%)	1.2 (23.1%)
m9	2.5 (8.8 %)	3.5 (15.3 %)	3.8 (59.5%)	7.4 (17.6%)	14.8 (60.5%)	1.0 (19.5%)
m11	4.6 (15.9 %)	3.2 (13.7 %)	0.3 (4.8%)	4.9 (11.8%)	14.8 (60.5%)	0.6 (10.6%)
m15	1.6 (5.5 %)	2.1 (9.0 %)	2.8 (43.9%)	8.5 (20.3%)	12.1 (49.6%)	1.8 (33.6%)
m16	4.3 (14.7 %)	1.6 (7.0 %)	0.2 (3.0%)	6.7 (16.0%)	15.3 (62.7%)	0.3 (6.4%)
m19	3.0 (10.5 %)	0.7 (3.0 %)	0.2 (3.0%)	7.0 (16.7%)	12.7 (51.9%)	1.9 (36.0%)
m20	3.1 (10.6 %)	0.0 (0.0 %)	0.2 (3.0%)	**11.0** (26.1%)	**5.5** (22.5%)	**0.4** (8.3%)

Table 5: The table shows average values and associated standard deviation for parameters extracted from markers at a stimulation intensity of 60 mA. Associated errors were placed below the results. The lowest errors are highlighted.

	Vastus medialis			Rectus femoris		
	$T_c\ (ms)$	$T_d\ (ms)$	$D_m\ (mm)$	$T_c\ (ms)$	$T_d\ (ms)$	$D_m\ (mm)$
m5	29.5 (± 9)	25.4 (± 10.5)	7.6 (± 3.8)	47.3 (± 13.2)	35.2 (± 10.9)	4 (± 1.7)
m9	30.2 (± 12.2)	21.6 (± 8.9)	10.8 (± 5.6)	45.7 (± 4.3)	39 (± 19.2)	4.5 (± 2.4)
m11	32.8 (± 12.3)	20.7 (± 5.3)	7.6 (± 2.9)	40.9 (± 10.4)	43 (± 24.3)	5 (± 2.4)
m15	31.1 (± 14.1)	22 (± 7.1)	10.3 (± 4.8)	51.1 (± 11.7)	32 (± 6.4)	7.3 (± 3)
m16	33.6 (± 15.6)	22.8 (± 8.6)	7.5 (± 3)	45.1 (± 3)	38.8 (± 17.5)	5.1 (± 2.4)
m19	32.8 (± 13.4)	25 (± 10)	6.8 (± 4.3)	46.7 (± 4.9)	38.1 (± 18)	3.4 (± 1.3)
m20	32.8 (± 13.2)	25 (± 9.5)	7 (± 3.8)	51 (± 9)	31.9 (± 10.6)	4.7 (± 1.7)
TMG	28.2 (± 9.2)	23 (± 2.1)	6.9 (± 1.7)	41.5 (± 19.4)	24.5 (± 2.8)	5.7 (± 1.9)
	$Error(T_c)$	$Error(T_d)$	$Error(D_m)$	$Error(T_c)$	$Error(T_d)$	$Error(D_m)$
m5	**1.3** (4.7 %)	**2.4** (10.4 %)	**0.6** (9.1%)	5.8 (13.9%)	10.8 (44.1%)	1.7 (29.3%)
m9	2.1 (7.4 %)	1.4 (6.2 %)	3.9 (55.8%)	4.1 (10%)	14.6 (59.5%)	1.2 (21.4%)
m11	4.6 (16.5 %)	2.3 (9.8 %)	0.7 (10%)	0.6 (1.4%)	18.5 (75.7%)	0.7 (11.8%)
m15	3 (10.5 %)	1 (4.5 %)	3.3 (48.2%)	9.6 (23%)	7.5 (30.7%)	1.6 (27.7%)
m16	5.4 (19.2 %)	0.2 (0.9 %)	0.6 (8.2%)	3.6 (8.6%)	14.4 (58.8%)	0.6 (10.8%)
m19	4.7 (16.6 %)	2 (8.8 %)	0.1 (1.7%)	5.2 (12.4%)	13.6 (55.8%)	2.3 (41%)
m20	4.7 (16.6 %)	1.9 (8.4 %)	0 (0.0%)	**9.5** (22.8%)	**7.5** (30.6%)	**1** (17.3%)

Table 6: The table shows average values and associated standard deviation for parameters extracted from markers at a stimulation intensity of 70 mA. Associated errors were placed below the results. The lowest errors are highlighted.

	Vastus medialis			Rectus femoris		
	$T_c\ (ms)$	$T_d\ (ms)$	$D_m\ (mm)$	$T_c\ (ms)$	$T_d\ (ms)$	$D_m\ (mm)$
m5	31.3 (± 11.7)	26.5 (± 24.9)	8 (± 3.7)	50.3 (± 19.2)	23.2 (± 7.1)	4.5 (± 1.4)
m9	37.6 (± 23.8)	21.4 (± 6.6)	12 (± 5.2)	51.3 (± 10.1)	25.9 (± 6.5)	5.3 (± 1.8)
m11	38.1 (± 23.7)	22 (± 7.5)	8.4 (± 2.7)	45.2 (± 5.6)	33.8 (± 12.8)	5.6 (± 1.9)
m15	33.3 (± 16.8)	26.1 (± 17.3)	11.4 (± 4.4)	46.1 (± 5.4)	31.8 (± 9.1)	8.4 (± 2)
m16	39.8 (± 24)	22.6 (± 8.5)	8.3 (± 2.8)	45.5 (± 3.1)	32.8 (± 13.1)	5.9 (± 1.7)
m19	34.8 (± 14.1)	30 (± 23.2)	7.6 (± 4.2)	53 (± 8.3)	23.3 (± 3.3)	3.9 (± 0.9)
m20	34.5 (± 12)	30.4 (± 25.3)	7.6 (± 3.6)	46.8 (± 3.4)	27.2 (± 8.5)	5.9 (± 1)
TMG	32 (± 19.5)	23.3 (± 2.7)	7.4 (± 2)	48.2 (± 24.4)	24.7 (± 3.1)	6.4 (± 2.3)
	$Error(T_c)$	$Error(T_d)$	$Error(D_m)$	$Error(T_c)$	$Error(T_d)$	$Error(D_m)$
m5	**0.7** (2.2 %)	**3.2** (13.7 %)	**0.6** (7.7%)	2.1 (4.4%)	1.5 (6.1%)	1.9 (30.2%)
m9	5.6 (17.5 %)	2 (8.5 %)	4.6 (61.7%)	3.1 (6.4%)	1.2 (4.7%)	1.1 (16.5%)
m11	6.1 (19.1 %)	1.3 (5.8 %)	1 (13.6%)	3 (6.1%)	9 (36.5%)	0.8 (12.7%)
m15	1.3 (3.9 %)	2.8 (11.9 %)	4 (53.3%)	2.1 (4.3%)	7 (28.4%)	2 (30.9%)
m16	7.8 (24.4 %)	0.8 (3.2 %)	0.9 (11.5%)	2.8 (5.7%)	8.1 (32.6%)	0.5 (7.4%)
m19	2.8 (8.6 %)	6.7 (28.6 %)	0.1 (1.7%)	4.8 (9.9%)	1.4 (5.8%)	2.5 (39.2%)
m20	2.5 (7.7 %)	7.1 (30.4 %)	0.2 (2.6%)	**1.4** (3%)	**2.5** (10%)	**0.5** (7.5%)

Table 7: The table shows average values and associated standard deviation for parameters extracted from markers at a stimulation intensity of 80 mA. Associated errors were placed below the results. The lowest errors are highlighted.

	Vastus medialis			Rectus femoris		
	$T_c\ (ms)$	$T_d\ (ms)$	$D_m\ (mm)$	$T_c\ (ms)$	$T_d\ (ms)$	$D_m\ (mm)$
m5	26.8 (± 7.6)	21.1 (± 13.3)	8.5 (± 3.4)	39.4 (± 12.7)	20.9 (± 10.0)	4.1 (± 1.3)
m9	29.7 (± 15.3)	15.2 (± 1.7)	13.1 (± 4.6)	36.8 (± 9.4)	25.2 (± 12.7)	5.7 (± 1.4)
m11	32.8 (± 17.2)	15.8 (± 2.3)	9.3 (± 2.3)	35.2 (± 9.1)	24.4 (± 13.2)	4.8 (± 1.8)
m15	29.0 (± 14.0)	17.6 (± 6.2)	12.5 (± 3.8)	31.3 (± 9.4)	31.4 (± 11.8)	7.1 (± 2.4)
m16	32.2 (± 18.4)	16.2 (± 2.9)	9.2 (± 2.5)	31.5 (± 4.5)	31.5 (± 13.5)	4.9 (± 1.3)
m19	30.3 (± 10.7)	20.1 (± 10.9)	8.1 (± 4.0)	35.2 (± 7.7)	25.1 (± 14.6)	3.4 (± 1.3)
m20	28.5 (± 10.9)	20.5 (± 11.7)	8.3 (± 3.3)	33.9 (± 5.8)	26.1 (± 13.5)	5.2 (± 1.1)
TMG	31.4 (± 19.5)	23.8 (± 3.6)	7.6 (± 2.3)	47.6 (± 24.3)	24.8 (± 3.3)	6.7 (± 2.7)
	$Error(T_c)$	$Error(T_d)$	$Error(D_m)$	$Error(T_c)$	$Error(T_d)$	$Error(D_m)$
m5	4.6 (14.6 %)	2.7 (11.3 %)	0.9 (11.7%)	8.2 (17.1%)	3.9 (15.6%)	2.6 (39.1%)
m9	1.7 (5.4 %)	8.6 (36 %)	5.4 (71%)	**10.7** (22.6%)	**0.4** (1.6%)	**1** (15.0%)
m11	1.4 (4.5 %)	8 (33.8 %)	1.7 (22%)	12.4 (26.1%)	0.4 (1.6%)	1.9 (28.8%)
m15	2.4 (7.7 %)	6.2 (26.1 %)	4.9 (64.1%)	16.2 (34.1%)	6.6 (26.8%)	0.5 (7.1%)
m16	0.8 (2.4 %)	7.6 (32.1 %)	1.6 (20.9%)	16 (33.7%)	6.7 (27.1%)	1.8 (26.7%)
m19	**1.1** (3.5 %)	**3.7** (15.5 %)	**0.5** (6.4%)	2.4 (26%)	0.3 (1.4%)	3.3 (48.8%)
m20	2.9 (9.2 %)	3.3 (14 %)	0.6 (8.2%)	13.6 (28.6%)	1.3 (5.2%)	1.4 (21.4%)

Impact of Data Balancing During Training for Best Predictions

Suleiman Ali Alsaif and Adel Hidri
Computer Department, Deanship of Preparatory Year and Supporting Studies
Imam Abdulrahman Bin Faisal University, P.O. Box 1982, Dammam 31441, Saudi Arabia
E-mail: saalsaif@iau.edu.sa, abhidri@iau.edu.sa

Keywords: machine learning, undersampling, oversampling, data balancing, ensemble learning

Received: March 26, 2021

To protect the middle class from over-indebtedness, banking institutions need to implement a flexible analytic-based evaluation method to improve the banking process by detecting customers who are likely to have difficulty in managing their debt. In this paper, we test and evaluate a large variety of data balancing methods on selected machine learning algorithms (MLAs) to overcome the effects of imbalanced data and show their impact on the training step to predict credit risk. Our objective is to deal with data unbalance to achieve the best predictions. We investigated the performance of these methods by different learners when classification models are trained using MLAs.

Povzetek: Predstavljena je metoda strojnega učenja z neuravnoteženimi podatki za oceno tveganja prezadolžitve srednjega razreda.

1 Introduction

To become compliant with changing and stricter regulatory demands in the world, banks are focused on undertaking preventive measures for effective credit risk management. One of its drivers is to strengthen existing learning models [1, 2] to increase their predictive power and help detect future defaults in advance.

Due to the inherent complex characteristics of imbalanced data sets, learning from such data requires new understandings, principles, algorithms, and tools to transform a vast amount of raw data efficiently into information and knowledge representation. The imbalanced learning problem is concerned with the performance of learning algorithms in the presence of under-represented data and severe class distribution skews [3, 4].

Most standard algorithms assume or expect balanced class distributions. Therefore, when presented with complex imbalanced datasets, these algorithms fail to properly represent the distributive characteristics of the data and resultantly provide unfavorable accuracies across the classes of the data. The imbalanced learning problem represents a recurring problem of high importance [5, 6, 7].

In this paper, we will present the impact of data balancing strategies at the training step. We test and evaluate a large variety of data balancing methods on selected MLAs to overcome the effects of imbalanced data and show their impact at the training step to predict credit risk. Our objective is to deal with data unbalance to achieve the best predictions. We investigated the performance of these methods by different learners when classification models are trained using MLAs.

The remainder of this paper is organized as follows: in section 2, we will highlight related work related to sampling methods for imbalanced learning. Section 3 provides an overview of the evaluation metrics in machine learning. In section 4, we will describe the enhanced training approach using data balancing strategies. In section 5, a performance analysis is brought forward to set the advantages of the proposed evaluation method. Section 6 discusses the work. Finally, we will conclude in section 7 our work and bring out some insights and potential future works.

2 Related work

Typically, the use of sampling methods in imbalanced learning applications consists of the modification of an imbalanced data set by some mechanism to provide a balanced distribution. Studies have shown that for several base classifiers, a balanced data set provides improved overall classification performance compared to an imbalanced dataset. These results justify the use of sampling methods for imbalanced learning [8].

Random undersampling (RUS) removes data from the original dataset. In particular, we randomly select a set of majority class examples and remove these samples from the dataset. Consequently, undersampling readily gives us a simple method for adjusting the balance of the original dataset [6, 8]. Figure 3 shows random undersampling and oversampling.

The mechanics of RUS follow naturally from its description by adding a set sampled from the minority class: for a set of randomly selected minority examples, augment the original set by replicating the selected examples and adding them to the dataset. In this way, the number of total examples in the minority class is increased and the class distribution balance is adjusted accordingly. This provides a mech-

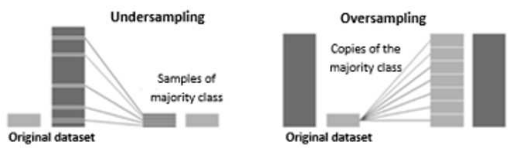


Figure 1: Random undersampling and oversampling.

anism for varying the degree of class distribution balance to any desired level.

The objective of the Easy Ensemble method [9, 10, 11, 26] is to overcome the deficiency of information loss introduced in the traditional RUS method. Easy Ensemble can be considered as an unsupervised learning algorithm that explores the majority class data by using independent random sampling with replacement [8].

The Balance Cascade algorithm takes a supervised learning approach that develops an ensemble of classifiers to systematically select which majority class examples to undersample. In Balance Cascade, the sequential dependency between classifiers is mainly exploited for reducing the redundant information in the majority class. This sampling strategy leads to a restricted sample space for the following undersampling process to explore as much useful information as possible [5].

Another example of informed undersampling uses the K-Nearest Neighbor (KNN) classifier [12] to achieve undersampling. Based on the characteristics of the given data distribution, four KNN undersampling methods were proposed, namely, NearMiss-1 and NearMiss-3 methods [5]. The NearMiss-1 method selects those majority examples whose average distance to the three closest minority class examples is the smallest [8]. The NearMiss-3 selects a given number of the closest majority examples for each minority example to guarantee that every minority example is surrounded by some majority examples [8].

SMOTE (Synthetic Minority Oversampling Technique) is an oversampling method. It works by creating synthetic samples from the minor class instead of creating copies. The algorithm selects two or more similar instances (using a distance measure) and perturbs an instance one attribute at a time by a random number within the difference to the neighboring instances [5].

Borderline-SMOTE1 and Borderline-SMOTE2 only oversample or strengthen the borderline minority examples [5]. The detailed procedure of Borderline-SMOTE1 is as follows: for every instance p in the minority class, we calculate its m nearest neighbors from the whole training set. If all nearest neighbors of p are majority examples, p is considered noise and is not operated in the following steps. If the number of p' majority nearest neighbors is larger than the number of minority ones, p is considered easily misclassified and put into a set DANGER. The examples in DANGER are the borderline data of the minority class. Then, new synthetic data are generated along the line between the minority borderline examples (data in *DANGER*) and their nearest neighbors of the same class, thus strengthening borderline examples [5].

Data cleaning techniques, such as Tomek links, have been effectively applied to remove the overlapping that is introduced by sampling methods [5]. One can use Tomek links to cleanup unwanted overlapping between classes after synthetic sampling, where all Tomek links are removed until all minimally distanced nearest-neighbor pairs are of the same class. By removing overlapping examples, one can establish well-defined class clusters in the training set, which can, in turn, lead to well- defined classification rules for improved classification performance. The Edited Nearest Neighbor (ENN) rule removes examples that differ from two of its three nearest neighbors.

3 Evaluation metrics in machine learning

In machine learning, there are a variety of metrics to evaluate the performance of classifiers. In this subsection, we will mention the most common evaluation metrics used in our work [13, 14, 15].

A confusion matrix is a specific table that is usually used to represent the performance of a classification model on test data for which the true values are known [16, 17]. As shown in Table 1, it consists of two columns and two rows that contain the number of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). Table 1 is an example of confusion matrix with two-class classifier.

The elements in the diagonal represent the number of instances for which the predicted class is equal to the true class, while off-diagonal elements are those that are misclassified by the classifier. The higher the diagonal values of the confusion matrix, the better, indicating correct predictions. In the case of over-indebtedness detection, over-indebted customers are the positive samples while no over-indebted customers are the negative samples.

Considering the confusion matrix in the table 1, we define the most basic terms as follows:

- TN: If the actual class is no over-indebted and the predicted value is also no over-indebted, so we have a TN. In practice, we want this value to be as high as possible.
- FP: If the actual class is not over-indebted and the predicted value is over-indebted, we have then a FP. In practice, we want this value to be as low as possible because it impacts customers, their banking experiences, and their lives.
- FN: If the actual class is over-indebted and the predicted value is not over-indebted, then we have a FN. Like the FP, we want this value to be as low as possible because the bank loses money when the customer who is deeply in debt, is considered as normal.

Table 1: Confusion matrix.

		Predicted class	
		No over-indebted	Over-indebted
Actual class	No over-indebted	TN	FP
	Over-indebted	FN	TP

- TP: If the actual class is over-indebted and the predicted value is also over-indebted, we have a TP. Like the TN, we want to predict all customers that will be in over-indebtedness to allow the bank to establish a support program for its consumers.

The sensitivity, also called the True Positive Rate (TPR), measures the proportion of positives that are correctly identified. The sensitivity is defined by the formula in Eq. (1) [18].

$$Sensitivity = TP/(TP + FN) \tag{1}$$

The specificity, also called the True Negatives Rate (TNR), measures the proportion of negatives that are correctly identified. The specificity is defined by the formula in Eq. (2) [18].

$$Specificity = TN/(TN + FP) \tag{2}$$

The Gini coefficient is defined as twice the area between the ROC (Receiver Operating Characteristic) curve and the chance diagonal. It takes values between 0 (no difference between the score distributions of the two classes) and 1 (complete separation between the two distributions). The ROC curve (see figure 1) is a graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold is varied.

The curve is created by plotting the TPR on the y-axis against the False Positive Rate (FPR) on the x-axis at various threshold settings. The TPR rate is also depicted as sensitivity in machine learning. The FPR rate can be calculated as follows:

$$FPR = 1 - Specificity \tag{3}$$

A classification of a new object is obtained by comparing the score s of the object with a classification threshold t. If $s > t$ the object is classified as coming from class 1 and if $s > t$ as coming from class 0.

The AUC (Area Under the Curve) is the area under the ROC curve. The performance of a classifier model is calculated by calculating the AUC [19, 20].

Related to the figure 1, we have:

$$AUC = Area\ A + Area\ B \tag{4}$$

The Gini coefficient is given by the following equation:

$$Gini = 2 * AUC - 1 \tag{5}$$

- At point $(0, 0)$, the classifier considers all instances as negatives. There are no false positives, but also no TP.

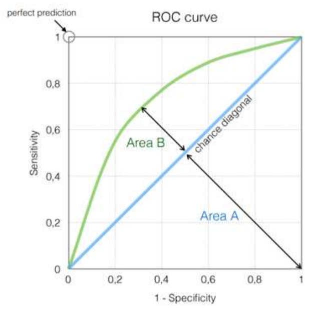


Figure 2: ROC curve.

- At point $(1, 1)$, the classifier considers all instances as positives. There are no false positives, but also no TN.

- At point $(1, 0)$, the classifier has no TP and no TN. In this case, the performance of the classifier is worse than random. Its performance can be totally improved by selectively reversing the classifiers answer.

- At point $(0, 1)$, the classifier has 100% sensitivity and 100% specificity. This point is called a perfect classification.

4 Enhancing training using data balancing strategies

The class imbalance problem corresponds to domains for which one class is represented by a large number of examples while the other is represented by only a few.

The used data consists of a collection of consumers to whom the bank gives credit. Analyzing the consumers in a binary sense, the natural classes that arise are negative or positive for a consumer who payed back or did not pay back their credit, respectively. From experience, one would expect the number of no over-indebted clients to exceed greatly the number of over-indebted clients. Indeed, this dataset contains 281,616 negative (majority class) samples and 678 positive (minority class) samples. We will have a classifier tend to provide a severely imbalanced degree of accuracy, with the majority class having close to 100% accuracy and the minority class having accuracies of 0-10% (see Figure 2).

Data are extremely imbalanced. We apply several sampling methods on the train set using the imbalanced learn package. It is a Python package offering a number of resampling techniques commonly used in datasets showing strong between-class imbalance. It is compatible with Scikit-learn and is part of the scikit-learn-contrib project [21].

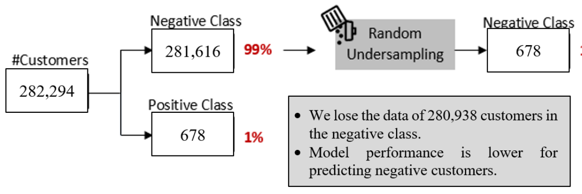

#Customers
282,294
Negative Class
281,616
99%
Positive Class
678
1%
Random Undersampling
Negative Class
678
1%
We lose the data of 280,938 customers in the negative class.
Model performance is lower for predicting negative customers.

Figure 3: Evaluation process.

Suppose a classifier achieves 10% accuracy on the minority class of the bank data set. Analytically, this would suggest that 610 minority samples are misclassified as majority samples. The consequence of this is equivalent to a 610 over-indebted clients classified as no over-indebted. In the bank, this consequence can be overwhelmingly costly.

Therefore, it is evident that for this domain, we require a classifier that will provide high accuracy for the minority class without severely jeopardizing the accuracy of the majority class.

Furthermore, this also suggests that the conventional evaluation practice of using singular assessment criteria, such as the overall accuracy or error rate, does not provide adequate information in the case of imbalanced learning. The immense hindering effects that these problems have on standard learning algorithms are the focus of most of the existing solutions. When standard learning algorithms are applied to imbalanced data, the induction rules that describe the minority concepts are often fewer and weaker than those of majority concepts, since the minority class is often both outnumbered and under-represented [8].

5 Computational results and analysis

Baseline algorithm performs the RUS for balancing data. The bank selects randomly a set from the negative (no over-indebted) class equals to the set of the positive (over-indebted) class and it removes all the other negative samples from the data. When applying random RUS in the train set, we might be losing representative samples of the negative class.

Our objective is to deal with data unbalance to achieve the best predictions for both classes and avoid favoring one of the classes over the other. We aim to get as much information as possible from the large number of majority available class examples.

The experiments were conducted using a 10-fold stratified cross-validation [22]. The experiments have been conducted to find the best sampling techniques. We investigated the performance of RUS, SMOTE, Balance Cascade, and Easy Ensemble by different learners when classification models are trained using XGB (eXtreme Gradient Boosting) [23], LR (Logistic Regression) [24], and ET (Extremely Randomized Trees) [25]. We used AUC (Area Under the Curve), Gini coefficient, and confusion matrix to evaluate the compared algorithms. We will compare the performance of the models before and after resampling to compare the increase of classifier performance due to resampling.

5.1 Undersampling methods

The bar charts of the figure 4 show the variation of Gini coefficient, sensitivity, and specificity after applying the NearMiss-1 and NearMiss-3 sampling methods to train data. The Gini coefficient achieves very high values for all algorithms except the XGB. The values for sensitivity are a little improved, but for the specificity the values are lower than the random under sampling for all algorithms.

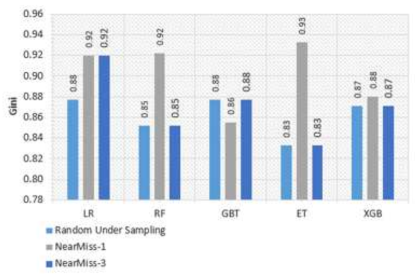
Undersampling tends to outperform the model in terms of sensitivity (see figure 4(b)), but at a very high cost of specificity (see figure 4(c)). However, while the value of sensitivity increased, the value of specificity decreased. Applying class-imbalance learning NearMiss-1 method on this data set is not necessarily beneficial.

As shown in the figure 4, the values of the three metrics are greater than the baseline. But the prediction of the positive examples is better than the negatives ones. For example, XGB algorithm gives 93.5% (see figure 4 (b)) in sensitivity whereas 88.4% in specificity (see figure 4 (c)).

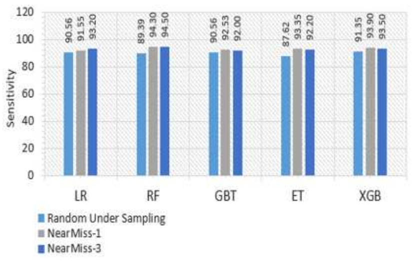
5.2 Oversampling coupled with Undersampling

We apply SMOTE methods to the data and then the random undersampling. Figure 5 shows the impact of the SMOTE and random undersampling on the performance of the models. This technique tends to produce high specificity without reducing the sensitivity when compared to other resampling techniques. As shown in Figure 5, the performance of this technique is similar to the combination of SMOTE+RUS. It tends to produce a good prediction for TN as well as the TP. The combination of SMOTEB1 with NearMiss-1 performs very poorly. The three metrics have lower values. This sampling technique is not helpful for our data and our models.

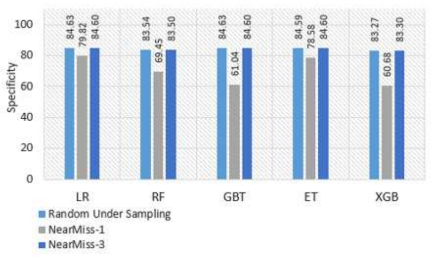
As shown in the figure 5, it can be seen that the sensitivity fell slightly. However, the Gini (see figure 5(a)) and the specificity (see figure 5 (c)) are improved. It is clear that SMOTE Tomek Links coupled with NearMiss-3 performs fairly. It tends to have a good value for the three metrics. Its results look like the results of the technique of SMOTEB2 coupled to NearMiss-3. SMOTE ENN coupled with random under sampling performs badly. It produces high specificity, but with very poor sensitivity (see figure 5(b)).



(a)



(b)



(c)

Figure 4: (a) Gini, (b) Sensitivity, and (c) Specificity of undersampling methods.

For example, we can see in the figure 5(c) that the XGB algorithm has 92.20% in specificity against 78.60% in sensitivity, which is a relatively bad value. It sacrifices high sensitivity for high specificity.

5.3 Ensemble

As shown in the figure 6, it is clear that the Easy Balance gives better results with XGB and LR algorithms in contrast to the other ones. While XGB and LR achieve very good values for both sensitivity (see figure 6(b)) and specificity (see figure 6(c)). They perform fairly for both; the other algorithms produce high specificity and very poor sensitivity.

From the figure 6, we can observe that the Balance Cascade method gives similar results to the Easy Ensemble but with a little decrease in terms of sensitivity.

6 Discussion

We compare the results of our experiments and find that the best resampling technique to use is often dataset and model dependent, certain resampling techniques tend to perform better when coupled with certain classifiers. It is obvious that there is not a best sampling technique for all models. The results show that the Easy Ensemble performs better than the others when XGB is used as a classifier.

It attains a better balance between sensitivity and specificity than almost all other methods. Instead of trading off one metric against another, it succeeds to give similar good results for all metrics. The graph in the figure 7 summarizes the effect of feature selection and data balancing with XGB algorithm on our data.

In Figure 8, we clearly see that:

– After feature selection: 4,564 clients were misclassified before as over indebted are correctly identified as normal and 3 clients were misclassified as normal are correctly identified as over-indebted.

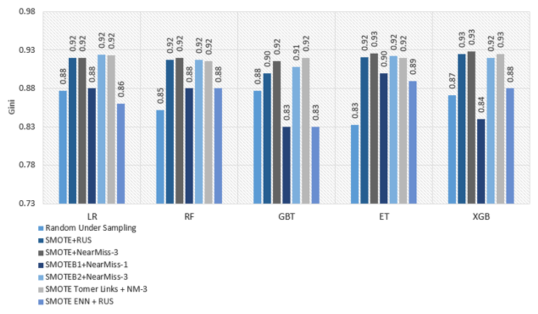
– After data balancing: we misclassified only 1 clients from the positive class to classify more 856 no over-indebted clients correctly.

7 Conclusion

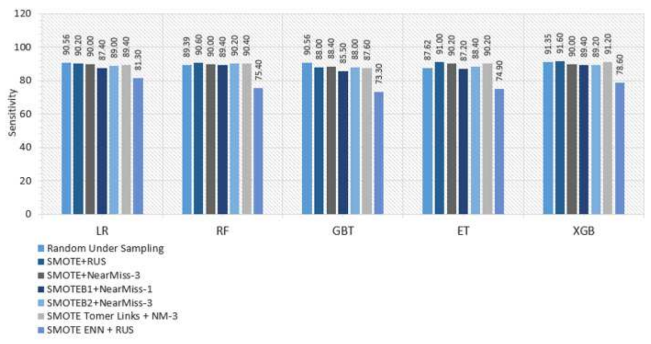
In this work, we tested and evaluated a large variety of data balancing methods on selected MLAs to overcome the effects of imbalanced data and show their impact on the training step to predict credit risk management. Our objective is to deal with data unbalance to achieve the best predictions.

After balancing the data with the easy ensemble method, we reduced the overfitting of the positive class and we reached a good balance between sensitivity and specificity. These contributions have shown that there isn't any ideal sampling method for balancing the data, and we have demonstrated through conducted experiments that it is possible to improve the overall performance of learning algorithms in the field of over-indebtedness prediction by using data balancing methods.

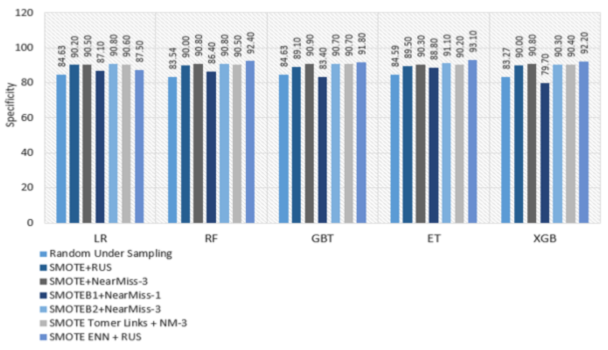
The set of conducted experiments aims at validating the importance of pre-processing the data before applying any machine learning algorithm. Balancing the data with advanced methods and changing the machine learning algorithm helped us to build a new effective and accurate pre-



(a)

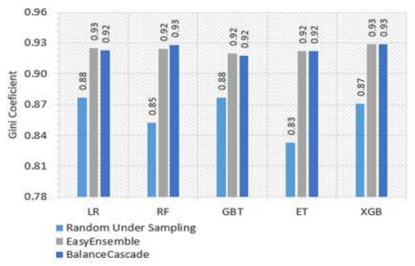


(b)

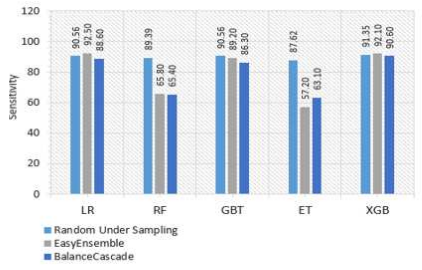


(c)

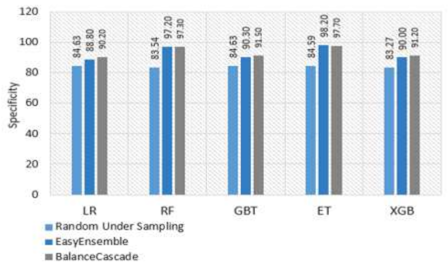
Figure 5: (a) Gini, (b) Sensitivity, and (c) Specificity of oversampling coupled with undersampling.



(a)



(b)



(c)

Figure 6: (a) Gini, (b) Sensitivity, and (c) Specificity of ensemble methods.

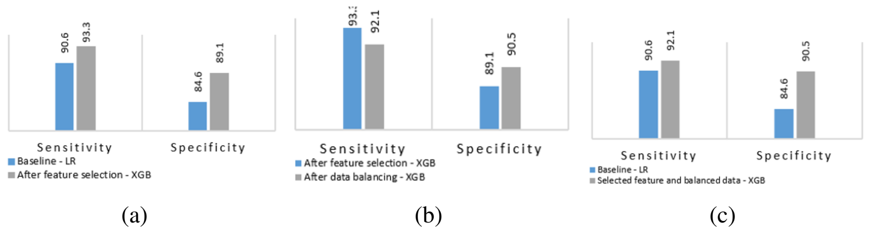
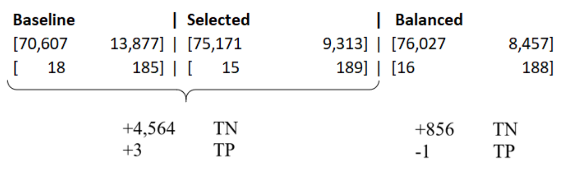


Figure 7: Improvement of sensitivity and specificity.



Baseline		Selected		Balanced	
[70,607	13,877]	[75,171	9,313]	[76,027	8,457]
[18	185]	[15	189]	[16	188]

+4,564 TN
+3 TP

+856 TN
-1 TP

Figure 8: Confusion matrix.

dictor. credit risk models need to consolidate new COVID-19 pandemic related data points to guarantee their output prevails valid and robust. Better and deeper insights can be accomplished by boring into a broader range of data sources as well as upgrading data platform technologies. This relates to our short-term work.

References

[1] A.A. Abaker and F.A. Saeed (2021) A comparative analysis of machine learning algorithms to build a predictive model for detecting diabetes complications, *Informatica (Slovenia)*, vol. 45, no. 1, pp. 117–125, `https://doi.org/10.31449/inf.v45i1.3111`.

[2] J. Sun (2021) Prediction and estimation of book borrowing in the library: Machine learning, *Informatica (Slovenia)*, vol. 45, no. 1, pp. 163–168, `https://doi.org/10.31449/inf.v45i1.3431`.

[3] L.J. Mena and J.A. Gonzalez (2009) Symbolic one-class learning from imbalanced datasets: Application in medical diagnosis, *Int. J. Artif. Intell. Tools*, vol. 18, no. 2, pp. 273–309, `https://doi.org/10.1142/S0218213009000135`.

[4] W.J. Liu, X.Y. and Z. Zhou (2009) Exploratory undersampling for class-imbalance learning, *Trans. Sys. Man Cyber. Part B*, vol. 39, `https://doi.org/10.1109/TSMCB.2008.2007853`.

[5] A. Mahani and A. Baba-Ali (2019) *Classification Problem in Imbalanced Datasets*, `https://doi.org/10.5772/intechopen.89603`.

[6] L.E.B. Ferreira, J.P. Barddal, F. Enembreck, and H.M. Gomes (2018) An experimental perspective on sampling methods for imbalanced learning from financial databases, *Proceedings of the 2018 International Joint Conference on Neural Networks*, pp. 1–6, `https://doi.org/10.1109/IJCNN.2018.8489290`.

[7] G. Goel, L. Maguire, Y. Li and S. McLoone (2013) Evaluation of sampling methods for learning from imbalanced data, *in Huang DS., Bevilacqua V., Figueroa J.C., Premaratne P. (eds) Intelligent Computing Theories*, pp. 392–401, `https://doi.org/10.1007/978-3-642-39479-9\_47`.

[8] H. He and E. Garcia (2009) Learning from imbalanced data,*IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, `https://doi.org/10.1109/TKDE.2008.239`.

[9] T.G. Dietterich (2000) Ensemble methods in machine learning, *Proceedings of the First International Workshop on Multiple Classifier Systems*, pp. 1–15, `https://doi.org/10.1007/3-540-45014-9_1`.

[10] M. Ghosh and P.G. Sanyal (2018) Performance assessment of multiple classifiers based on ensemble feature selection scheme for sentiment analysis, *Applied Computational Intelligence and Soft Computing*, pp. 1–12, `https://doi.org/10.1155/2018/8909357`.

[11] R.R. Saifan, K. Sharif, M. Abu-Ghazaleh, and M. Abdel-Majeed (2020) Investigating algorithmic stock market trading using ensemble machine learning methods, *Informatica (Slovenia)*, vol. 44, no. 3, pp. 311–325, `https://doi.org/10.31449/inf.v44i3.2904`.

[12] K. Q. Weinberger and L.K. Saul (2009) Distance metric learning for large margin nearest neighbor classification, *JMLR*, pp. 207–244, `https://doi.org/10.5555/1577069.1577078`.

[13] M. Hossin and M. Sulaiman (2019) A Review on Evaluation Metrics for Data Classification Evaluations, *International Journal of Data Mining &*

Knowledge Management Process (IJDKP), vol. 5, no. 2, pp. 1–11, https://doi.org/10.5121/ijdkp.2015.5201.

[14] M. Fatourechi, R. Ward, S. Mason, J. Huggins, A. Schlogl, and G. Birch (2009) Comparison of evaluation metrics in classification applications with imbalanced datasets, *Proceedings of the Seventh International Conference on Machine Learning and Applications*, pp. 777–782, https://doi.org/10.1109/ICMLA.2008.34.

[15] K.M. Ghori, R.A. Abbasi, M. Awais, M. Imran, A. Ullah, and L. Szathmary (2020) Performance analysis of different types of machine learning classifiers for non-technical loss detection, *IEEE Access*, vol. 8, pp. 16 033–16 048, https://doi.org/10.1109/ACCESS.2019.2962510.

[16] K. Ting (2017) *Encyclopedia of Machine Learning and Data Mining*, ser. Springer Reference, C. Sammut and G. I. Webb, Eds., https://doi.org/10.1007/978-1-4899-7687-1.

[17] O. Caelen (2017) A bayesian interpretation of the confusion matrix, *Annals of Mathematics and Artificial Intelligence*, vol. 81, https://doi.org/10.1007/s10472-017-9564-8.

[18] S. Kumar (2020) Sensitivity, specificity, generalizability, and reusability aspirations for machine learning (ML) models in MHealth, *Proceedings of Deep Learning for Wellbeing Applications Leveraging Mobile Devices and Edge Computing*, pp. 1, https://doi.org/10.1145/3396868.3402495.

[19] P. Flach, J. Hernández-Orallo, and C. Ferri (2011) A coherent interpretation of auc as a measure of aggregated classification performance, *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pp. 657–664, https://doi.org/10.5555/3104482.3104565.

[20] L.E. Raileanu and K. Stoffel (2004) Theoretical comparison between the gini index and information gain criteria, *Annals of Mathematics and Artificial Intelligence*, vol. 41, no. 1, pp. 77–93, https://doi.org/10.1023/B:AMAI.0000018580.96245.c6.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.* (2011) "Scikit-learn: Machine learning in python, *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, https://doi.org/10.5555/1953048.2078195.

[22] S. Raschka, J. Patterson, and C. Nolet (2020) Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence, *Information*, vol. 11, no. 4, pp. 193, https://doi.org/10.3390/info11040193.

[23] H. Drucker and C. Cortes (1995) *Boosting decision trees*. Cambridge, MA, USA: MIT Press, pp. 479–485, https://doi.org/10.5555/2998828.2998896.

[24] D.W. Hosmer and S. Lemeshow (2013) *Applied logistic regression*. John Wiley and Sons, https://doi.org/10.1002/9781118548387.

[25] C. Latha and S. Jeeva (2019) Improving the accuracy of prediction of heart disease risk based on ensemble classification techniques, *Informatics in Medicine Unlocked*, vol. 16, pp. 100203, https://doi.org/10.1016/j.imu.2019.100203.

[26] E.K. Ampomah, Z. Qin and G. Nyame and F.E. Botchey (2020) Stock market decision support modeling with tree-based AdaBoost ensemble machine learning models, *Informatica (Slovenia)*, vol. 44, no. 4, pp. 477–489, https://doi.org/10.31449/inf.v44i4.3159.

Performance Analysis of Test Path Generation Techniques Based on Complex Activity Diagrams

Walaiporn Sornkliang
Management of Information Technology, School of Informatics
Walailak University, Nakhon Si Thammarat, Thailand
E-mail: vlaiporn@gmail.com

Thimaporn Phetkaew
School of Engineering and Technology, Walailak University, Nakhon Si Thammarat, Thailand
Informatics Innovation Center of Excellence, Walailak University, Nakhon Si Thammarat, Thailand
E-mail: pthimapo@wu.ac.th, thimaporn.p@gmail.com

Keywords: coverage criteria, software testing, test path generation, concurrency test scenario, UML activity diagram

Received: February 5, 2020

Effort reduction in software testing is important to reduce the total cost of the software development project. UML activity diagram is used by the tester for test path generation. It is hard to select the appropriate test path generation technique to diminish the effort of software testing. In the experiment, we compared the efficiency of 12 commonly-used test path generation techniques with both simple activity diagrams and the constructed complex activity diagrams. The experimental results summarized in four aspects. (1) The most appropriate test path generation technique for path testing generates the number of paths equivalent to the target number of all possible paths. (2) The suitable test path generation technique for the concurrency test scenario. (3) The techniques that can generate test paths covering basis path coverage in the case that testing all possible paths for the large or complex object-oriented method is laborious. (4) To compare the efficiency of test path generation algorithms, the percentage test path deviation to the target number of all possible paths is calculated for the constructed complex activity diagrams. We also recommended suitable test path generation methods for each manner of the UML activity diagram.

Povzetek: *Avtorji so analizirali uspešnost dvanajst metod za iskanje poti preverjanja programske opreme z enostavnimi in kompleksnimi diagrami aktivnosti.*

1 Introduction

Software testing is part of the most important phases of the software development life cycle. Test planning or test design specifications usually occur at the beginning of the system development process. The program can be tested according to the software requirements specification (SRS), or detailed design documents, which reduces the time and cost of software development. Today, most software is developed using Object-Oriented technology and the Unified Modeling Language (UML). UML activity diagram describes the workflow of a sequence of software activities, or concurrent software activities, from the initial activity to the end. An activity diagram is flowchart-like that can be used to generate test paths.

According to Linus's law, given large enough beta-tester and co-developer bases, almost every problem will be characterized quickly and the fix will be obvious to someone [1]. Green software testing takes into consideration the number of people and the amount of equipment allocated to test based on predefined test cases related to energy consumption [2]. Software testing needs to generate test cases to determine the expected output for any program path. There are many methods to generate test paths from a UML activity diagram. Each method is different and yields distinct results because the paths of the program can be traversed in a variety of techniques; thus, selecting the appropriate test path generation method is challenging. If there are too many test paths. It is the cause that uses a lot of effort to design test cases and to execute the program path.

Although general method in the object-oriented programming is not much complex, in case of we want to compare the efficiency of test path generation algorithms, we have to apply those with the complex models of UML activity diagram. There are many types of the control structure of the program such as selection control structure consisting of single-way selection, two-way selection, and nested selection; iteration control structure consisting of pre-test loop and post-test loop; and fork-join structure consisting of simple fork-join, fork-merge concurrent, part-join concurrent, and no-join concurrent. The complex UML activity diagrams are constructed to evaluate test path results of the algorithms by coverage criteria such as statement coverage, branch coverage, activity path coverage, basis path coverage, and path coverage.

The test path generation techniques from UML activity diagram usually depend on graph or tree theory: the test path generation techniques based on tree structure, such as the Dependent Flow Tree [3], the Fault Success Tree Analysis [4], and the Activity Tree [5]; the test path generation techniques based on graph structure, such as the Intermediate Black Box Model [6], the Activity Graph [7], the Activity Flow Table [8], the Activity Convert Grammar [9], the Activity Flow Graph [10], the Test Case Generation Based on Activity Diagram [11], the Activity Dependency Graph [12], and the Intermediate Testable Model [13]. In addition, the test path generation from UML activity diagram can be used the heuristic algorithm e.g., ant colony [14].

Currently, the advantages of the test path generation are applied to several real-world problems, for example, (1) to generate test data using the neighborhood search strategy [15] and using the genetic algorithm [16], (2) to generate test paths for effective chatbot software testing using customized response [17], (3) to optimize test cases by generating test path and selecting test data using Cuckoo search and Bee colony algorithm [18], and (4) to generate optimized test data for saving both testing cost and time [19].

To reduce the effort of software testing, the test paths must be non-redundant, adequate and complete. The purpose of this study was to compare the current test path generation techniques using complex UML activity diagrams constructed by the researcher.

The remainder section of this paper is organized as follows. Section 2 introduces the software testing, test path generation, UML activity diagram, coverage criteria, and literature review. Section 3 explains the experimental setup for this study. The experimental results and discussion of this paper are included in Section 4. Finally, the paper is concluded in Section 5.

2 Background

This section provides a brief overview of software testing, test path generation, UML activity diagram, coverage criteria, and research articles related to this study.

2.1 Software testing

Testing is concerned with bugs or errors, defects or faults, failures, and incidents [20]. Software testing is a crucial part of software quality assurance; it concerns the examination of specifications, designs, and codes [21]. Testing aims to find the detects early in system development. If the fault is found early, then the computational cost will be lower than if the fault is found in the implementation phase. The testing is carried out to assure the quality and reliability of the software. To find the defect as soon as possible, testing activities should start as early as the requirements are derived and should continue until the software is completed [22]. Good test cases have attributes such as a high probability to find bug; the test should not be redundant or too simple or complex [21].

2.2 Test path generation

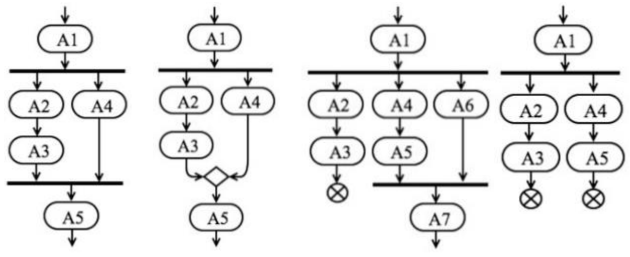
A test case is a path that covers specific system requirements and data [23]. A test case is made up of a set of test inputs, execution conditions, and expected results developed for the set of objectives [24]. Test cases can be generated automatically from requirements and specifications, design, or the source code [25, 26]. A good test case is a part that has a high probability of finding an as-yet-undiscovered error [27]. To check if the application produces correct outputs, a series of test variables are used as inputs by a tester. Test case generation and also test path generation are an important issue in the software testing field.

2.3 UML activity diagram

An activity diagram is importantly a flowchart that chronologically organizes a set of activities that show the workflow from a start point to the finish that takes place over time [28]. An activity diagram shows the flow from one activity to another. The diagram symbols consist of activities, initial activity, final activity, transition, decision, merge, fork, join, and swimlane, as shown in Table 1.

Symbols	Name	Description
⬭	Activity	The process being modelled
●	Initial activity	The flow starts in the UML activity diagram
◉	Final activity	The final step in the UML activity diagram
→	Transition	Control flow
◇	Decision	Alternative activities
◇	Merge	Brings together one or more incoming flows to accept the single outgoing flow
┬	Fork	Split transition into multiple fork activities
┴	Join	The combination of multiple fork activities
╤	Swimlane	Classification of activities' duty

Table 1: The symbols of the UML activity diagram.



(a) Simple fork-join. (b) Fork-merge concurrent. (c) Part-join concurrent. (d) No-join concurrent.

Figure 1: Types of the concurrent structure in UML activity diagram.

For a concurrent structure in the UML activity diagram [6, 13], the most common form is classified into four forms as shown in Figure 1. First, the simple fork-join in Figure 1(a) is a pair of fork node and join node and all activity between a fork and join symbol. Second, the fork-merge concurrent in Figure 1(b) has a merge node that is placed instead of a join which allows multiple flows. The paths of the fork node can also be traversed in the same way as a selection structure. Third, the part-join concurrent in Figure 1(c) has two parts: the first part, it has a join node that is used to converge outgoing flows of a fork node; the second part, it is no converging node at the end of these flows. And last, the no-join concurrent in Figure 1(d) is no converging node at the end of these flows.

2.4 Coverage criteria

The coverage criterion is the degree, expressed as a percentage or a specified coverage item that needs to be exercised by a test suite [29]. For concurrency test scenario, the distinction between paths and their respective coverage criteria will help to effective effort estimation and test management [30].

2.4.1 Normal test scenario

The coverage criteria items can be classified into five items. First, statement coverage requires that all statements must be executed at least once. This condition suggests that an error in a statement cannot be revealed without executing the faulty statement. Second, branch coverage requires every decision of the program to be covered by at least one test path. Thus, each decision leads to two test requirements, the decision is either true or false. If every method must be called at least once, each method leads to one test requirement. Third, activity path coverage is the test path that ensures all activities are tested at least once, and all possible paths are tested for all activities [5]. An activity path is flow of activities from the start activity into the final activity in the activity diagram. Fourth, path coverage is the test path that ensures all paths of the program work. To determine path coverage, all paths need to be covered from start to end. And last, basis path coverage is the test path that ensures the optimal test path is covered. The number of paths to ensure the basis path coverage criterion is satisfied can be determined from Cyclomatic complexity [26]. Cyclomatic complexity is the quantitative software metric of the complexity in a program. The cyclomatic complexity value determines the number of independent paths in a basic program set and the maximum amount of testing needed to ensure that all basis paths are covered at least once. Cyclomatic complexity has a foundation in graph theory and is computed in one of three ways. By definition, $V(G)$ is the complexity of the control flow graph. It can be calculated according to each of the following, as in Equation (1), (2) [21, 26], and Equation (3) [31].

$$V(G) = (E - N) + 2, \quad (1)$$

where E is the number of edges between nodes and N is the number of nodes.

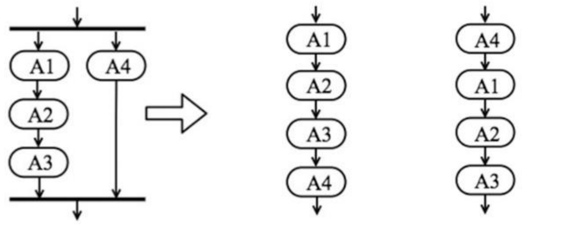
$$V(G) = (P + 1), \quad (2)$$

where P is the number of predicate nodes in control flow graph.

$$V(G) = (R + 1), \quad (3)$$

where R is the number of closed regions of control flow graph.

For the concurrent regions in the UML activity diagram, all possible paths must be traversed from two directions that are left-to-right and right-to-left. For the UML activity diagram in Figure 2(a), there are two possible paths. In the first path in Figure 2(b), it traverses from the left thread to the right thread. In the second path in Figure 2(c), it traverses from the right thread to the left thread. Each thread is listed from the activity of top-level to activity of low-level [32].



(a) Concurrent region. (b) First path. (c) Second path.

Figure 2: The all possible paths in the concurrent regions.

2.4.2 Concurrency test scenario

Execution of activities in the concurrent region leads to concurrency errors if the implementation does not include required restrictions on some of the interleaving paths by setting appropriate synchronization primitives [30].

For the Interleaving Activity Path Coverage (IAPC) criterion [30], let *IP* be the set of interleaving activity paths inside a fork-join structure, and *TS* be the set of test scenarios generated from the activity diagram. The test set *TS* satisfies interleaving activity path coverage, if and only if, $\forall\ p \in IP,\ \exists\ t \in TS$ such that when the program is executed using test scenario 't', the interleaving activity path 'p' of the activity diagram is executed, fully or partially. The interleaving paths are calculated using Equation (4):

$$\left(\sum_{i=1}^{m} n_i\right)! / \prod_{i=1}^{m} (n_i!), \quad (4)$$

Where m is the number of threads, n_i is the number of activities in thread i.

2.5 Literature review

In software testing, test paths must be generated to test the software for the expected results. The techniques for test path generation from UML activity diagrams usually depend on tree or graph theory, which are listed in the subsection below.

2.5.1 The test path generation techniques based on tree structures

A tree is a simple hierarchical graph that links together one edge between two nodes. It starts from the root node and ends at the leaf node. Many tree-based test path generation methods have been developed recently. We describe three methods that are relevant to this research study.

The Dependent Flow Tree [3] is a method that generates test paths from the activity diagram by a constructing dependent flow tree with stores all the information extracted from the XML file of the diagram through the help of a parser. The dependency flow tree consists of nodes and edges. After that, the test paths were generated by using a Depth First Search algorithm that visiting all the nodes and edges exactly once. In this study, the generated test paths include branch coverage and path coverage.

The Fault Success Tree Analysis [4] is a method that generates test paths from an activity diagram by considering the decision of the activity diagram to build a tree. The classification of each tree is built by considering all possible decisions and the paths they took before approaching the next decision. Each decision can pass conditions on the way to the next decision. If conditions were found along the way, the decision conditions are identified in the classification tree and ordered accordingly before reaching the next decision. Then, the test paths are generated from the root node to the leaf node. Next, the fault tree diagram can be generated from the invalid paths of classification tree and the success tree diagram can be generated from the valid path of the classification tree. In this study, the generated test paths include branch coverage.

The Activity Tree [5] is a method that builds test paths from activity diagrams. This suggests the test paths generated from the activity diagrams are transformed into an ordered test flow tree, from the initial activity to the final activity. If activity loops are found in the activity tree, then the activity before the next loop was used as the last activity at the end of the loop. After that, the possible test paths were identified by using a Depth First Search algorithm. If the test paths had loops, this algorithm could search for test paths from the initial node to the last node of the test flow tree only once. In this study, the generated test paths include activity path coverage.

2.5.2 The test path generation techniques based on graph structures

Graph (G) is made up of a set of ordered pairs, $G = (V, E)$, where V is set of nodes, and E is the set of edges, which are the links between nodes. We describe a relevant set of eight methods that use graphs to generate test paths.

The Intermediate Black Box Model [6] is a method that generates test paths from unstructured activity diagrams. The unstructured module is including the loop structure and fork-join structure. The method begins by constructing an activity graph from an activity diagram. Then, the activity graph is classified into a set of groups, including the loop and fork-join structure. The nodes in each group are then combined into a single node. The test paths then searched the graph using the Depth First Search algorithm. If a node has an iteration structure, then the path goes through the loop least once. If all nodes in a fork-join structure, then the path goes through all possible activity. In this study, the generated test paths include path coverage.

The Activity Graph [7] is a method that generates test paths from the activity diagrams. The activity diagram is transformed into an activity graph. The activity graph is a direct graph and is replaced by each node of the activity diagram. The activity graph is ordered using the control flow from the chronological list of activities, including the branch, decision, iteration, and fork-join activities. The test paths then searched the graph using the Depth First Search algorithm that visiting all the nodes and edges exactly once. In this study, the generated test paths include activity path coverage.

The Activity Flow Table [8] is a method that generates test paths from the activity diagram. The activity diagram is used to construct the activity flow table that describes the symbols by numbers given to each activity. Then an activity flow graph is constructed and used the symbols ordered on an activity flow table. The activity flow graph searches all possible paths by using a Depth First Search algorithm that compares each path to a set of criteria using basis path coverage. If a node has an iteration structure, then the path goes through the loop only once. In this study, the generated test paths include basis path coverage and activity path coverage.

The Activity Convert Grammar [9] is a method that generates test paths from activity diagrams and activity convert grammar by constructing an activity dependency table and a decision dependency table from the sets of testing data. Then, the activity dependency table and the decision dependency table construct test paths using the grammar method. The grammar is divided into activities on the Left-Hand Side (LHS) that is used as the dependent activity and activities on the Right-Hand Side (RHS) that tests the branch and fork activity. To generate test paths, the activities are tested chronologically. In the case of a fork activity, the activities are prioritized from left to right and right to left. In this study, the generated test paths include path coverage.

The Activity Flow Graph [10] is a method that generates test paths from activity diagrams. The activity diagrams are used to construct the control flow activity table and values using the conditions of each activity. Then an activity flow graph is constructed and ordered using the control flow from the chronological list of activities including the branch, decision, iteration, and fork activities. The test paths then searched the graph using the Depth First Search. If a node has an iteration structure, then the path goes through the loop only once. Each test path generates a test path. In this study, the generated test paths include activity path coverage.

The Test Case Generation Based on Activity Diagram [11] is a method that generates test paths from activity diagrams according to the following steps. First, the activity dependency table is constructed and used to create an activity dependency graph covering all activities. The activity dependency graph searches all possible paths by using a Depth First Search that compares each path to a set of criteria using basis path coverage. If activity in a loop structure is encountered, the loop is only traversed once.

In this study, the generated test paths include basis path coverage and path coverage.

The Activity Dependency Graph [12] improves the test path generation technique from activity diagrams by constructing an activity dependency table and an activity dependency graph, respectively. Then, the dependency graph is improved by removing the activities which have the same names, decision symbols, fork symbols, join symbols and merge symbols. The improved activity dependency graph is used to construct test paths, which generate the test paths. In this study, the generated test paths include basis path coverage and branch coverage.

The Intermediate Testable Model [13] studies the synthesis of testing situations from activity diagrams. The method begins by constructing a control flow graph from an activity diagram. Then, the control flow graph is classified into a set of groups including the selection, loop, and fork-join structures. The nodes in each group are then combined into a single node. When generating a test path, the tester chooses the structure of interest to build sub-paths. When all the constructs are used to generate the test paths, then that test paths have covered all possible paths. In this study, the generated test paths include path coverage.

The test paths then searched the graph using the Depth First Search algorithm that visiting all the nodes and edges exactly once. In this study, the generated test paths include activity path coverage.

2.5.3 The test path generation techniques based on heuristic algorithm

Orientation-based Ant Colony algorithm (OBACO) [14] is proposed to generate test paths for a concurrent segment of UML activity diagram. Parsing of XMI code takes UML activity diagram as input and results into individual sub-queues of activity nodes present under the fork-join structure. The input of the orientation based ant colony optimization is sub-queues under the fork-join structure of an activity diagram is used for generating combinations between the activity nodes of the sub-queues. The next activity node in the path is decided by pheromone, heuristic values, and orientation factor.

3 Experimental setup

This research looks at 12 commonly-used test path generation techniques. There are three tree-based test path generation algorithms, which are the Dependent Flow Tree (DFT) [3], the Fault Success Tree Analysis (FSTA) [4], and the Activity Tree (ActTree) [5]. There are eight graph-based test path generation algorithms, which are the Intermediate Black Box Model (IBM) [6], the Activity Graph (AG) [7], the Activity Flow Table (AFT) [8], the Activity Convert Grammar (ACG) [9], the Activity Flow Graph (AFG) [10], the Test Case Generation Based on Activity Diagram (TCBAD) [11], the Activity Dependency Graph (ADG) [12], and the Intermediate Testable Model (ITM) [13]. And there is one heuristic-based test path generation algorithm, which is the Orientation-based Ant Colony algorithm (OBACO) [14]. It is difficult to select the appropriate test path generation techniques to reduce the effort of software testing. The researcher has conducted the research by comparing the test path generation techniques using complex UML activity diagrams created by the researcher.

3.1 The UML activity diagrams used in the experiment

To evaluate the test path results of the 12 algorithms, we used both real-world activity diagram and constructed activity diagram. The two real-world activity diagrams, which are a Shopping Mall System activity diagram in Figure 3 and a Library Management System activity diagram in Figure 4. The selection criteria are the activity diagram which describes the daily life work and easy to understand. Two activity diagrams created by the researcher are called a Complex Concurrent Structure activity diagram in Figure 5 and a Complex Control Structure activity diagram in Figure 6. No previous work in the literature review generated test paths from the complex activity diagram. So, we create a complex activity diagram to compare the efficiency of each algorithm in terms of the covered coverage criteria. The control variables of four activity diagrams are the number of paths covered by path testing and basis path coverage. The four activity diagrams employed in the experiment are as follows.

3.1.1 The shopping mall system activity diagram

Figure 3 shows the Shopping Mall System activity diagram, applied from [33, 34], which consists of sequence structure, selection structure, and iteration structure. In this activity diagram does not has the fork-join structure. In this system, a user can select the item, the system can make the billing, and check the member card. The user can select to pay by cash or card. Besides, the user can select the gift service or collect stamp.

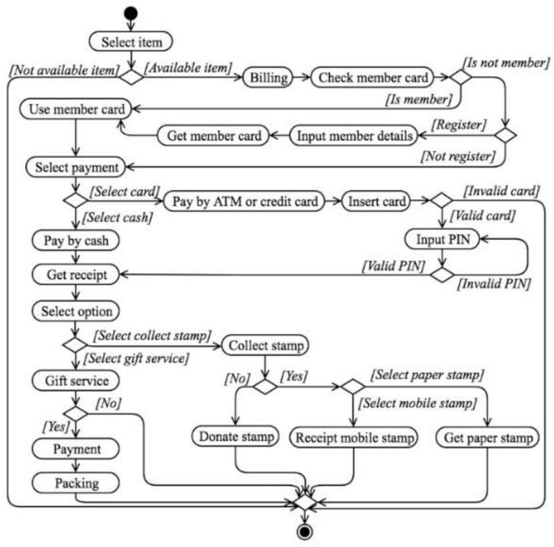


Figure 3: The Shopping Mall System activity diagram.

3.1.2 The Library Management System activity diagram

Figure 4 shows the Library Management System activity diagram, applied from [35, 36], which consists of sequence structure, selection structure, iteration structure, and fork-join structure. In this system, the user inserts the card, inputs the password, and then the system checks the account. If it is the account created, the system checks the status of the account, and the user can decide to return or borrow the book. If it is not the account created, the user registers the new account. The system checks the availability of the book. If the book is available, the system will decrease book availability and increase the number of the book borrowed.

3.1.3 The Complex Concurrent Structure activity diagram

Figure 5 shows the Complex Concurrent Structure activity diagram that is constructed to generated test path focus on fork-join structure. This activity diagram has control structure such as sequence structure, selection structure, and fork-join structure. In this activity diagram does not has the iteration structure. For the selection structure, there is a nested selection. For the fork-join structure, we use fork-join structure classification of the concurrent module i.e., simple fork-join, fork-merge concurrent, part-join concurrent, and no-join concurrent.

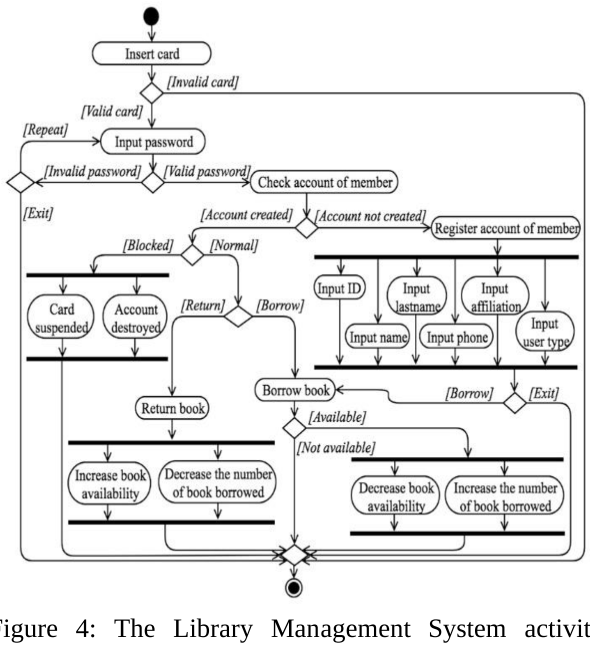


Figure 4: The Library Management System activity diagram.

3.1.2 The Complex Control Structure activity diagram

Figure 6 shows the Complex Control Structure activity diagram consists of all type of control structure. This activity diagram is comprised of the sequence structure, selection structure, iteration structure, and fork-join structure. For selection structure, there is one-way, two-way, and nested selection. Iteration structure consisting of pre-test loop and post-test loop. Within the fork-join structure, there are one-way selection, two-way selection, and nested selection, pre-test loop, post-test loop, simple fork-join, fork-merge concurrent, part-join concurrent, and no-join concurrent.

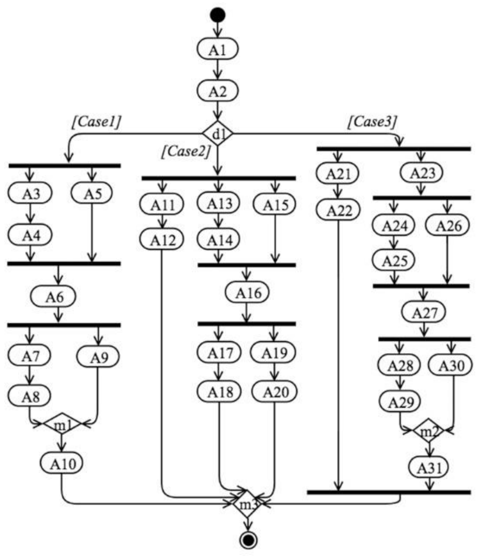


Figure 5: The Complex Concurrent Structure activity diagram.

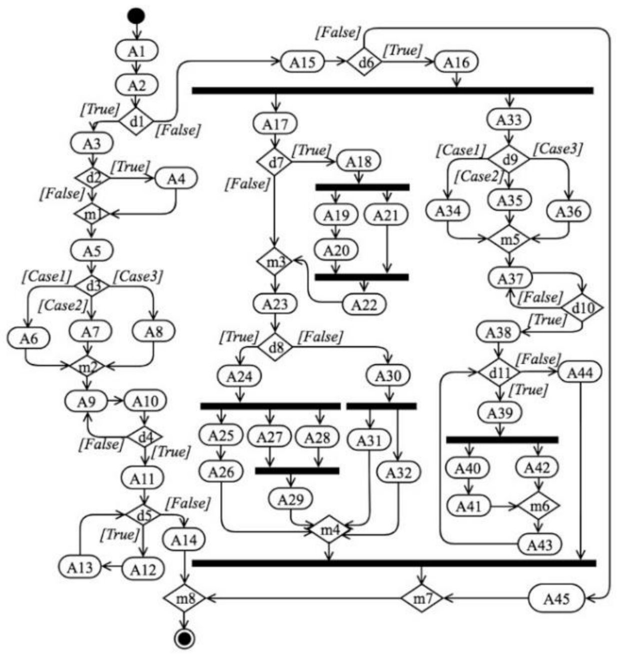


Figure 6: The Complex Control Structure activity diagram.

3.2 The target number of test path

To compare the efficiency of test path generation algorithms, four UML activity diagrams are used to construct in the form of the control flow graphs (CFG), and then each control flow graph is searched to find the target number of all possible path using a Depth First Search algorithm and to find the number of basis path using Equation (1). For path coverage in the case of the concurrent regions, all possible paths must be traversed from two directions that are left-to-right and right-to-left [32]. The first direction was started from the activity on the left transition and goes as far as it can down a given path to reach the join symbol, then backtracks until it finds an unexplored path, and then explores it. These procedures are repeated until the entire concurrent region has been explored. For the second direction, it was started from the activity on the right transition. For the concurrency test scenario, the target number of all possible path in the fork-join structure is calculated by using Equation (4).

4 Experimental results and discussion

In this section, we show the experimental results for 12 test path generation techniques with four activity diagrams. For simplicity to show the result, we use the word "Shopping" short for Shopping Mall System activity diagram, the word "Library" short for Library Management System activity diagram, "Concurrent" short for Complex Concurrent Structure activity diagram, and "Complex" short for Complex Control Structure activity diagram. The results of test path generation techniques based on tree structures, graph structures, and heuristic algorithm are shown in Table 2. The target number of all possible paths (TAP) and the target number of basis paths (TBP) are shown in the table to compare with the number of test paths of each technique. The coverage criterion used in the experiment are statement coverage (SC), branch coverage (BC), activity path coverage (AP), basis path coverage (BP), and path coverage (PC). The symbol ✓ means the technique found out coverage of that criteria, whereas symbol × means the technique is not satisfied with that criteria.

For the concurrency test scenario, we show the experimental results for the Intermediate Black Box Model (IBM) and the Intermediate Testable Model (ITM) which are the test path generation methods focused on the concurrency region. Table 3 shows the coverage percentage of test path generation techniques for the interleaving activity path coverage (IAPC) criteria. Both of the Intermediate Black Box Model (IBM) and the Intermediate Testable Model (ITM) generate the same number of paths as the target paths.

Test path generation techniques based on tree structures																				
Activity diagram	**Target**		**DFT**						**FSTA**						**ActTree**					
	TAP	TBP	Test paths	Coverage criteria					Test paths	Coverage criteria					Test paths	Coverage criteria				
				SC	BC	AP	BP	PC		SC	BC	AP	BP	PC		SC	BC	AP	BP	PC
Shopping	49	11	49	✓	✓	✓	✓	✓	49	✓	✓	✓	✓	✓	37	✓	✓	✓	✓	×
Library	33	17	65	✓	✓	✓	✓	✓	17	✓	✓	✓	✓	×	10	✓	✓	✓	×	×
Concurrent	17	10	14	✓	✓	✓	✓	×	3	✓	✓	✓	×	×	7	✓	✓	✓	×	×
Control	565	20	58	✓	✓	✓	✓	×	73	✓	✓	✓	✓	×	115	✓	✓	✓	✓	×

Test path generation techniques based on graph structures																				
Activity diagram	**Target**		**IBM**						**AG**						**AFT**					
	TAP	TBP	Test paths	Coverage criteria					Test paths	Coverage criteria					Test paths	Coverage criteria				
				SC	BC	AP	BP	PC		SC	BC	AP	BP	PC		SC	BC	AP	BP	PC
Shopping	49	11	49	✓	✓	✓	✓	✓	49	✓	✓	✓	✓	✓	34	✓	✓	✓	✓	×
Library	33	17	5,777	✓	✓	✓	✓	✓	65	✓	✓	✓	✓	✓	33	✓	✓	✓	✓	✓
Concurrent	17	10	139	✓	✓	✓	✓	✓	14	✓	✓	✓	✓	×	14	✓	✓	✓	✓	×
Control	565	20	60,505	✓	✓	✓	✓	✓	58	✓	✓	✓	✓	×	30	✓	✓	✓	✓	×
Activity diagram	**Target**		**ACG**						**AFG**						**TCBAD**					
	TAP	TBP	Test paths	Coverage criteria					Test paths	Coverage criteria					Test paths	Coverage criteria				
				SC	BC	AP	BP	PC		SC	BC	AP	BP	PC		SC	BC	AP	BP	PC
Shopping	49	11	49	✓	✓	✓	✓	✓	50	✓	✓	✓	✓	✓	49	✓	✓	✓	✓	✓
Library	33	17	33	✓	✓	✓	✓	✓	18	✓	✓	✓	✓	×	65	✓	✓	✓	✓	✓
Concurrent	17	10	17	✓	✓	✓	✓	✓	7	✓	✓	✓	×	×	14	✓	✓	✓	✓	×
Control	565	20	565	✓	✓	✓	✓	✓	173	✓	✓	✓	✓	×	58	✓	✓	✓	✓	×
Activity diagram	**Target**		**ADG**						**ITM**											
	TAP	TBP	Test paths	Coverage criteria					Test paths	Coverage criteria										
				SC	BC	AP	BP	PC		SC	BC	AP	BP	PC						
Shopping	49	11	34	✓	✓	✓	✓	×	49	✓	✓	✓	✓	✓						
Library	33	17	26	✓	✓	✓	✓	×	5,777	✓	✓	✓	✓	✓						
Concurrent	17	10	14	✓	✓	✓	✓	×	139	✓	✓	✓	✓	✓						
Control	565	20	24	✓	✓	✓	✓	×	60,505	✓	✓	✓	✓	✓						

Test path generation technique based on heuristic algorithm								
Activity diagram	**Target**		**OBACO**					
	TAP	TBP	Test paths	Coverage criteria				
				SC	BC	AP	BP	PC
Shopping	49	11	49	✓	✓	✓	✓	✓
Library	33	17	65	✓	✓	✓	✓	✓
Concurrent	17	10	17	✓	✓	✓	✓	✓
Control	565	20	565	✓	✓	✓	✓	✓

Table 2: A comparison of the result of test path generation techniques.

Activity diagram	Target	IBM		ITM	
	IAPC	Test paths in the fork-join structures	Interleaving activity path coverage criteria (%)	Test paths in the fork-join structures	Interleaving activity path coverage criteria (%)
Library	5,772	5,772	100	5,772	100
Concurrent	139	139	100	139	100
Control	60,480	60,480	100	60,480	100

Table 3: A comparison of the result of test path generation techniques for the concurrency test scenario.

The percentage deviation (PD) of each test path results can be calculated to determine how much each method deviates from all possible paths, generated from the Complex Concurrent Structure activity diagram and the Complex Control Structure activity diagram. The equation to calculate the percentage deviation is listed below, as in Equation (5).

$$PD = \left(\frac{TP - N}{N}\right) x100, \quad (5)$$

where *PD* is the percentage deviation, *TP* is the number of test paths by each method, and *N* is the target number of the all possible paths of the activity diagram.

For example, from Table 2 the Dependent Flow Tree (DFT) could generate 14 paths of the Complex Concurrent Structure activity diagram, but the target number of all possible paths of the Complex Concurrent Structure activity diagram was 17 paths. So, the percentage deviation of DFT in Equation (5) is ((14-17)/17) x 100 = -17.65%, as shown in Table 4. A positive value indicates that the number of generated test paths is larger than the target number of all possible paths. A negative value indicates that the number of generated test paths is smaller than the target number of all possible paths.

Table 4 shows the percentage deviation of test path generation techniques for the Complex Concurrent Structure activity diagram and the Complex Control Structure activity diagram.

Test path generation techniques	Complex Concurrent Structure activity diagram (TAP=17 paths)			Complex Control Structure activity diagram (TAP=565 paths)		
	Number of test paths	Number of the different test path	Percentage deviation (%)	Number of test paths	Number of the different test path	Percentage deviation (%)
DFT	14	-3	-17.65	58	-507	-89.73
FSTA	3	-14	-82.35	73	-492	-87.08
ActTree	7	-10	-58.82	115	-450	-79.65
IBM	139	122	717.65	60,505	59,940	10,608.85
AG	14	-3	-17.65	58	-507	-89.73
AFT	14	-3	-17.65	30	-535	-94.69
ACG	17	0	0.00	565	0	0.00
AFG	7	-10	-58.82	173	-392	-69.38
TCBAD	14	-3	-17.65	58	-507	-89.73
ADG	14	-3	-17.65	24	-541	-95.75
ITM	139	122	717.65	60,505	59,940	10,608.85
OBACO	17	0	0.00	565	0	0.00

Table 4: The percentage deviation of test path generation techniques for the Complex Concurrent Structure activity diagram and the Complex Control Structure activity diagram.

Figure 7 shows the percentage deviation of test paths generated of the Complex Concurrent Structure activity diagram. Figure 8 shows the percentage deviation of test paths generated of the Complex Control Structure activity diagram. The positive value indicates that the number of generated test paths is larger than the target number, whereas a negative value indicates that the number of generated test paths is smaller than the target number.

In case of the Complex Concurrent Structure activity diagram, the percentage deviation of the Intermediate Black Box Model (IBM), and the Intermediate Testable Model (ITM) is 717.65%, this means that they generated excessive test paths. The percentage deviation of the Fault Success Tree Analysis (FSTA) is -82.35%, this means that it generated inadequate test paths. The percentage deviation of the Activity Convert Grammar (ACG) and the Orientation-based Ant Colony algorithm (OBACO) are 0%, this means that they generate the equivalent test paths.

In case of the Complex Control Structure activity diagram, the percentage deviation of the Intermediate Black Box Model (IBM), and the Intermediate Testable Model (ITM) is 10,608.85%, this means that they generated excessive test paths. The percentage deviation of the Activity Dependent Graph (ADG) is -95.75%, this means that it generates inadequate test paths. The percentage deviation of the Activity Convert Grammar (ACG) and the Orientation-based Ant Colony algorithm (OBACO) are 0%, this means that they generate the same number of paths as the target paths.

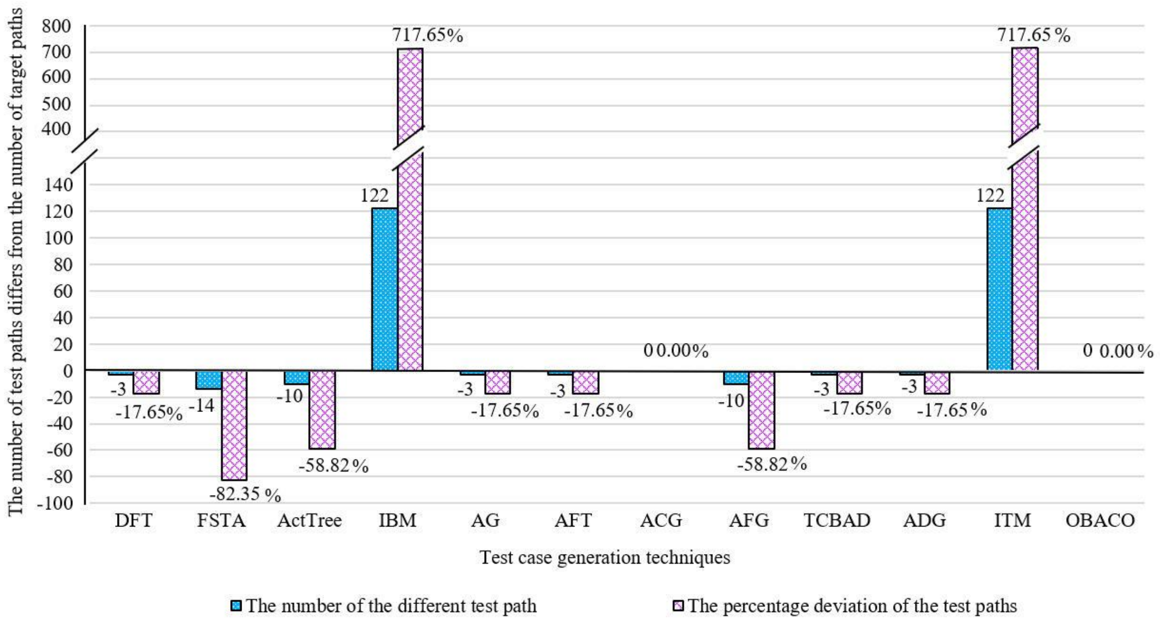


Figure 7: A comparison of the percentage deviation between the number of test paths generate from the Complex Concurrent Structure activity diagram and the target number of all possible paths.

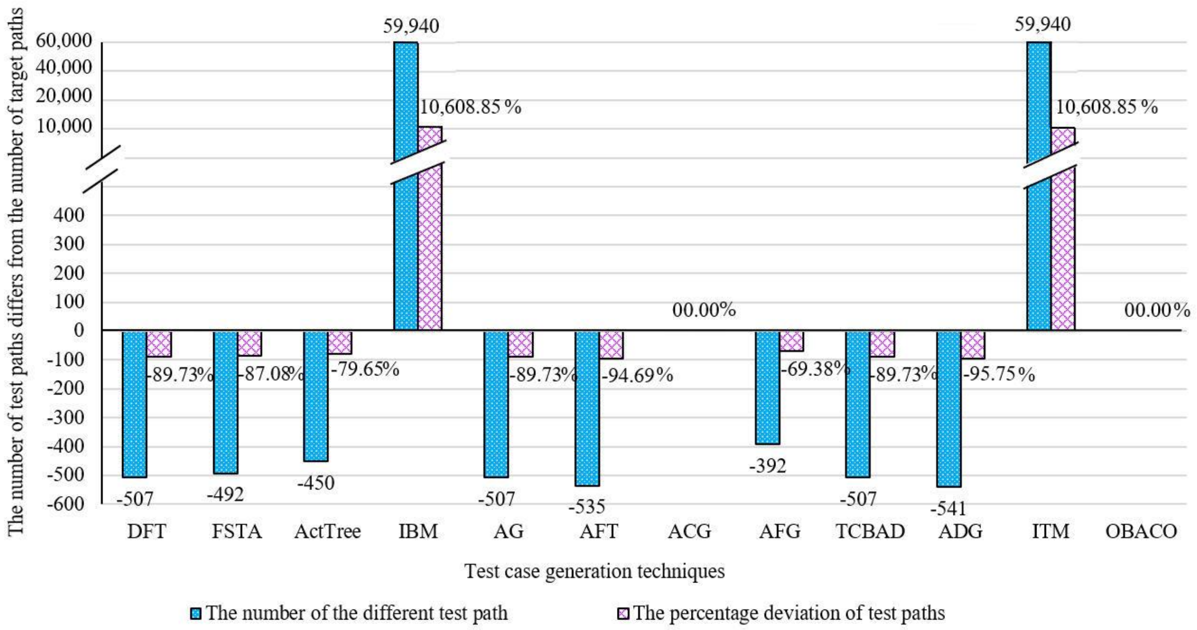


Figure 8: A comparison of the percentage deviation between the number of test paths generate from the Complex Control Structure activity diagram and the target number of all possible paths.

The difference between the test path results and the target number of all possible paths is mainly occurred at the fork-join structure. A fork-join symbol in a UML activity diagram is a control node that splits a transition into multiple concurrent transitions. Test path generation of the activities within the fork-join of the 12 test path generation techniques is different. The number of test paths depends on the number of transitions and the number of activities in each transition. The path traversal to find test paths for the fork-join structure is shown in Table 5.

The experimental results from Table 2 - 4 can be summarized into four aspects: (1) the aspect of covering the path coverage for both simple (real-world) activity diagrams and our proposed complex activity diagram, (2) the aspect of covering the interleaving activity path coverage for the concurrency test scenario, (3) the aspect of covering the basis path coverage, and (4) the aspect of the efficiency of test path generation algorithms applying with the complex activity diagram. And, we also suggested the proper activity diagram for each test path generation method, as shown in Table 6.

Test path generation techniques	The path traversal in fork-join structure
DFT, AG, AFT, ADG, TCBAD	It starts from the top activity on the left thread, traverses through the activities in the low level in the same thread till it reaches the join symbol, and then it goes out of the join symbol.
ACG	There are two directions. In the first direction, it traverses from the left thread to the right thread. And in the second direction, it traverses from the right thread to the left thread. Each thread is listed from the activity of top-level to activities of low-level.
FSTA, ActTree, AFG	It starts from the top activity on the left thread and goes as far as it can down a given path to reach the join symbol, then backtracks until it finds an unexplored path, and then explores it. These procedures are repeated until it traverses through all thread and finally it goes out of the join symbol.
IBM, ITM	The calculation of all possible paths in the fork-join structure is $N!/(n!*n!)$, where N is the sum of all activities in the fork-join, and n is the sum of all activities in each transition.
OBACO	The paths in the fork-join structure are generated through the use of separate ant agents for performing traversal with the sub-transition.

Table 5: The path traversal in the fork-join structure in test path generation techniques.

Test path generation techniques	Suitable UML activity diagram
DFT, AG, TCBAD	Simple activity diagram with sequence, selection, iteration, or fork-join structure.
ActTree, AFT, ADG	Simple activity diagram with sequence and selection structure.
FSTA	Simple activity diagram with sequence, selection, and iteration structure.
AFG	Simple or complex activity diagram focusing on loop testing.
IBM, ITM	Simple or complex activity diagram focusing on concurrency test scenario in the fork-join structure.
ACG, OBACO	Simple or complex activity diagram with sequence, selection, iteration, or fork-join structure.

Table 6: UML activity diagrams which are suitable for the test path generation techniques.

(1) For path coverage in the case of the simple (real-world) activity diagram, the Dependent Flow Tree (DFT), the Intermediate Black Box Model (IBM), the Activity Graph (AG), the Activity Convert Grammar (ACG), the Test Case Generation Based on Activity Diagram (TCBAD), the Intermediate Testable Model (ITM), and the Orientation-based Ant Colony algorithm (OBACO) could generate the number of test paths that satisfied the path coverage, while the Fault Success Tree Analysis (FSTA), the Activity Flow Table (AFT), and the Activity Flow Graph (AFG) could generate the number of test paths that occasional satisfied the path coverage. For path coverage in the case of the constructed complex activity diagrams, only the Intermediate Black Box Model (IBM), the Activity Convert Grammar (ACG), the Intermediate Testable Model (ITM), and the Orientation-based Ant Colony algorithm (OBACO) could generate the number of test paths that satisfied the path coverage. There is an exceptional case for the Activity Tree (ActTree), which satisfied to activity path coverage and the Activity Dependency Graph (ADG), which satisfied for basis path coverage.

(2) For the concurrency test scenario, in both cases of the simple (real-world) activity diagram and the constructed complex activity diagrams, the Intermediate Black Box Model (IBM) and the Intermediate Testable Model (ITM) could generate the number of test paths that satisfied 100% interleaving activity path coverage.

(3) In case that it is difficult and take a lot of effort to test all possible paths for complex activity diagram, the tester should select the basis paths coverage instead. The experimental results depict that the Dependent Flow Tree (DFT), the Intermediate Black Box Model (IBM), the Activity Graph (AG), the Activity Flow Table (AFT), the Activity Convert Grammar (ACG), the Test Case Generation Based on Activity Diagram (TCBAD), the Activity Dependency Graph (ADG), the Intermediate Testable Model (ITM), and the Orientation-based Ant Colony algorithm (OBACO) could generate test paths that covered the basis paths.

(4) For efficiency comparison of test path generation algorithms with the constructed complex activity diagrams., Figure 7 and 8 depict that the Activity Convert Grammar (ACG) and the Orientation-based Ant Colony algorithm (OBACO) could generate the equivalent test paths to the target number of all possible paths. Whereas, the Intermediate Black Box Model (IBM), and the Intermediate Testable Model (ITM) could multiply the number of test paths if there were a lot of activities in the fork-join structure.

5 Conclusion

This paper presents the performance analysis of test path generation algorithms. To compare the efficiency of test path generation algorithms, we applied 12 commonly-used techniques with both simple (real-world) activity diagrams and the constructed complex activity diagrams. In this research, we constructed two complex activity diagrams, i.e., the Complex Concurrent Structure activity diagram and the Complex Control Structure activity diagram.

The experimental results show that to test all possible paths, the Activity Convert Grammar (ACG) and the Orientation-based Ant Colony algorithm (OBACO) are the most appropriate test path generation technique which can generate the number of paths equivalent to all possible paths. Besides, other test path generation techniques such as the Intermediate Black Box Model (IBM) and the Intermediate Testable Model (ITM) can cover path coverage, but there are too many test paths. However, these two methods can cover 100% interleaving activity path coverage for the concurrency test scenario.

Testing all possible paths for the large or complex object-oriented method is laborious, the tester should select the basis paths coverage instead. The Dependent Flow Tree (DFT), the Intermediate Black Box Model (IBM), the Activity Graph (AG), the Activity Flow Table (AFT), the Activity Convert Grammar (ACG), the Test Case Generation Based on Activity Diagram (TCBAD), the Activity Dependency Graph (ADG), the Intermediate Testable Model (ITM), and the Orientation-based Ant Colony algorithm (OBACO) are the appropriate test path generation techniques that can cover the basis path coverage of both the simple and complex activity diagram.

References

[1] Matjaž Gams and Tine Kolenik. Relations between electronics, artificial intelligence and information society through information society rules. Electronics, 10(4): 514, 2021. https://doi.org/10.3390/electronics10040514

[2] Mahdi Dhaini, Mohammad Jaber, Amin Fakhereldine, Sleiman Hamdan, and Ramzi A. Haraty. Green computing approaches - A survey. Informatica, 45(1): 1-12, 2021. https://doi.org/10.31449/inf.v45i1.2998

[3] Oluwatolani Oluwagbemi and Hishammuddin Asmuni. Automatic generation of test cases from activity diagrams for UML based testing (UBT). Jurnal Teknologi (Science & Engineering), 77(13): 37-48, 2015. https://doi.org/10.11113/jt.v77.6358

[4] Pimthip Paiboonkasemsut and Yachai Limpiyakorn. Reliability tests for process flow with fault tree analysis. In 2015 2nd International Conference on Information Science and Security (ICISS), IEEE, 14-16 December, Seoul, South Korea, pp. 1-4, 2015. https://doi.org/10.1109/ICISSEC.2015.7371028

[5] Ranjita Kumari Swain, Vikas Panthi, Durga Prasad Mohapatra, and Prafulla Kumar Behera. Prioritizing test scenarios from UML communication and activity diagrams. Innovations in Systems and Software Engineering, 10(3): 165-180, 2014. https://doi.org/10.1007/s11334-013-0228-5

[6] Yufei Yin, Yiqun Xu, Weikai Miao, and Yixiang Chen. An automated test case generation approach based on activity diagrams of SysML. International Journal of Performability Engineering, 13(6): 922-936, 2017. https://doi.org/10.23940/ijpe.17.06.p13.922936

[7] Namita Khurana, Rajender Singh Chhillar, and Usha Chhillar. A novel technique for generation and optimization of test cases using use case, sequence, activity diagram and genetic algorithm. Journal of Software, 11(3): 242-250, 2016. https://doi.org/10.17706/jsw.11.3.242-250

[8] Ajay Kumar Jena, Santosh Kumar Swain, and Durga Prasad Mohapatra. A novel approach for test case generation from UML activity diagram. In 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), IEEE, 7-8 February, Ghaziabad, India, pp. 621-629, 2014. https://doi.org/10.1109/ICICICT.2014.6781352

[9] Kanjanee Pechtanun and Supaporn Kansomkeat. Generation test cases from UML activity diagram based on AC grammar. In 2012 International Conference on Computer and Information Science (ICCIS), IEEE, 12-14 June, Kuala Lumper, Malaysia, pp. 895-899, 2012. https://doi.org/10.1109/ICCISci.2012.6297153

[10] Ranjita Kumari Swain, Vikas Panthi, and Prafulla Kumar Behera. Generation of test cases using activity diagram. International Journal of Computer Science and Informatics, 4(1): 35-44, 2014. https://www.interscience.in/ijcsi/vol4/iss1/8

[11] Chanda Chouhan, Vivek Shrivastava, and Parminder S Sodhi. Test case generation based on activity diagram for mobile application. International Journal of Computer Applications, 57(23): 4-9, 2012. https://doi.org/10.5120/9436-3563

[12] Pakinam N. Boghdady, Nagwa L. Badr, Mohamed A. Hashim, and Mohamed F. Tolba. An enhanced test case generation technique based on activity diagrams. In The 2011 International Conference on Computer Engineering & Systems, IEEE, 29 November - 1 December, Cairo, Egypt, pp. 289-294, 2011. https://doi.org/10.1109/ICCES.2011.6141058

[13] Ashalatha Nayak and Debasis Samanta. Synthesis of test scenarios using UML activity diagram. Software and Systems Modeling, 10: 63-89, 2011. https://doi.org/10.1007/s10270-009-0133-4

[14] VinayArora, Maninder Singh, and Rajesh Bhatia. Orientation-based ant colony algorithm for synthesizing the test scenarios in UML activity diagram. Information and Software Technology, 123: 106292, 2020. https://doi.org/10.1016/j.infsof.2020.106292

[15] Sapna Varshney and Monica Mehrotra. A hybrid particle swarm optimization and differential evolution based test data generation algorithm for data-flow coverage using neighborhood search strategy. Informatica, 42(3): 417-438, 2018. https://doi.org/10.31449/inf.v42i3.1497

[16] Aman Jaffari, Aman Jaffari, and Jihyun Lee. Automatic test data generation using the activity diagram and search-based technique. Applied Sciences, 10(10): 3397, 2020. https://doi.org/10.3390/app10103397

[17] Mani Padmanabhan. Sustainable test path generation for chatbots using customized Response. International Journal of Engineering and Advanced Technology, 8(6): 149-155, 2019. https://doi.org/10.35940/ijeat.D6515.088619

[18] Lakshminarayana P and T V SureshKumar. Automatic generation and optimization of test case using hybrid cuckoo search and bee colony algorithm. Journal of Intelligent Systems, 30(1): 59-72, 2021. https://doi.org/10.1515/jisys-2019-0051

[19] Manju Khari and Prabhat Kumar Khari. An effective meta-heuristic cuckoo search algorithm for test suite optimization. Informatica, 41(3): 363-377, 2017.

http://www.informatica.si/index.php/informatica/article/view/1174/1069

[20] Paul C. Jorgensen. Software testing a craftsman's approach. 4th ed., CRC Press, London, 2014.

[21] Roger S. Pressman. Software engineering: a practitioner's approach. 7th ed., McGraw-Hill, New York, NY, 2010.

[22] Subashni, S. and Satheesh Kumar N. Software testing using visual studio 2010. Packt Publishing, Birmingham, UK, 2010.

[23] Mahesh Shirole and Rajeev Kumar. UML behavioral model based test case generation: a survey. ACM SIGSOFT Software Engineering Notes, 38(4): 1-13, 2013. https://doi.org/10.1145/2492248.2492274

[24] IEEE Computer Society. 829-2008-IEEE standard for software and system test documentation. The Institute of Electrical and Electronics Engineers, New York, NY, pp.1-84, 2008. https://doi.org/10.1109/IEEESTD.2008.4578383

[25] Itti Hooda and Rajender Chhillar. A review: study of test case generation techniques. International Journal of Computer Applications, 107(16): 33-37, 2014. https://doi.org/10.5120/18839-0375

[26] Md. Abdur Rahman, Md. Abu Hasan, Khaled Shah, and Md. Saeed Siddik. Multipartite based test case prioritization using failure history. International Journal of Advanced Science and Technology, 129: 25-42, 2019. http://sersc.org/journals/index.php/IJAST/article/view/1353/1084

[27] B.B. Agarwal, S.P. Tayal, and M. Gupta. Software engineering & testing. Jones and Bartlett, Sudbury, Massachusetts, pp.157-164, 2010.

[28] Grady Booch, James Rumbaugh, and Ivar Jacobson. The unified modeling language user guide. Addison-Wesley Longman, Reading, Massachusetts, pp. 311-331, 1998.

[29] Mauro Pezz and Michal Young. Software testing and analysis: process, principles, and techniques. John Wiley & Sons, Hoboken, NJ, pp. 211-230, 2008. https://ix.cs.uoregon.edu/~michal/book/Samples/book.pdf

[30] Mahesh Shirole and Mahesh Shirole. Concurrency coverage criteria for activity diagram. IET Software, 15(1): 43-53, 2021. https://doi.org/10.1049/sfw2.12009

[31] Frank Tsui, Orlando Karnal, and Barbara Bernal. Essentials of software engineering. 4th ed., Jones & Bartlett Learning, Burlington, Massachusetts, pp. 170-171, 2017.

[32] Farid Meziane and Sunil Vadera. Artificial intelligence applications for improved software engineering development: new prospects. Information Science Reference, Hershey, New York, NY, pp. 248, 2010. https://doi.org/10.4018/978-1-60566-758-4

[33] Prateeva Mahali and Prateeva Mahali. Model based test case prioritization using UML activity Diagram and evolutionary algorithm. International Journal of Computer Science and Informatics, 4(2): 76-81, 2014. https://doi.org/10.47893/ijcsi.2014.1177

[34] Sonali Khandai, Sonali Khandai, and Sonali Khandai. Prioritizing test cases using business criticality test value. International Journal of Advanced Computer Science and Applications, Science and Information Organization, 3(5): 103-110, 2011. https://doi.org/10.14569/IJACSA.2012.030516

[35] Oluwatolani Oluwagbemi, Hishammuddin Asmuni. An approach for automatic generation of test cases from UML diagram. International Journal of Software Engineering and Its Applications, 9(8): 87-106, 2015. https://www.earticle.net/Article/A252751

[36] Monalisha Khandai, Arup Abhinna Acharya, and Durga Prasad Mohapatra. Test case generation for concurrent system using UML combinational diagram. International Journal of Computer Science and Information Technology, 2(3): 1172-1182, 2011. http://ijcsit.com/docs/Volume%202/vol2issue3/ijcsit2011020344.pdf

Stock Market Prediction with Gaussian Naïve Bayes Machine Learning Algorithm

Ernest Kwame Ampomah
School of Information & Software Engineering,
University of Electronic Science and Technology of China, China
E-mail: ampomahke@gmail.com

Gabriel Nyame
Department of Information Technology Education
Akenten Appiah-Menka University of Skills Training and Entrepreneurial Development, Kumasi-Ghana
E-mail: kwakuasane1972@gmail.com

Zhiguang Qin
School of Information & Software Engineering,
University of Electronic Science and Technology of China, China
E-mail: qinzg@uestc.edu.cn

Prince Clement Addo
School of Management and Economics, University of Electronic Science and Technology of China, China
E-mail: prince@std.uestc.edu.cn

Enoch Opanin Gyamfi
School of Information & Software Engineering,
University of Electronic Science and Technology of China, China
E-mail: enochopaningyamfi@outlook.com

Michael Gyan
Department of Physics Education, University of Education, Winneba-Ghana
E-mail: mgyan173@gmail.com

Keywords: machine learning, gaussian naïve Bayes, stock price, feature extraction, scaling

Received: January 8, 2021

The stock market is one of the key sectors of a country's economy. It provides investors with an opportunity to invest and gain returns on their investment. Predicting the stock market is a very challenging task and has attracted serious interest from researchers from many fields such as statistics, artificial intelligence, economics, and finance. An accurate prediction of the stock market reduces investment risk in the market. Different approaches have been used to predict the stock market. The performances of Machine learning (ML) models are typically superior to those of statistical and econometric models. The ability of Gaussian Naïve Bayes ML algorithm to predict stock price movement has not been addressed properly in the existing literature, hence this attempt to fill that gap in the literature by evaluating the performance of GNB algorithm when combined with different feature scaling and feature extraction techniques in stock price movement prediction. The performance of the GNB models set up were ranked using the Kendall's test of concordance for the various evaluation metrics used. The results indicated that, the predictive model based on integration of GNB algorithm and Linear Discriminant Analysis (GNB_LDA) outperformed all the other models of GNB considered in three of the four evaluation metrics (i.e., accuracy, F1-score, and AUC). Similarly, the predictive model based on GNB algorithm, Min-Max scaling, and PCA produced the best rank using the specificity results. In addition, GNB produced better performance with Min-Max scaling technique than it does with standardization scaling techniques

Povzetek: Predstavljena je metoda Gausovega naivnega Bayesa za borzne napovedi.

1 Introduction

The stock market is one of the key sectors of a country's economy. It provides investors with an opportunity to invest and gain returns on their investment. Predicting the stock market has attracted serious interest from researchers from many fields such as statistics, artificial intelligence, economics, and finance. An accurate prediction of the stock market reduces investment risk in the market. Different opinions exist as regards to the predictability of the stock market. The efficient market hypothesis (EMH) states that all available information is fully incorporated by current market price immediately, therefore, changes in price of the stocks are as a result of new information [1]. The EMH implies that stock prices would trail a random walk pattern, hence, the stock market cannot be forecasted from past data to make any meaningful returns [2]. However, numerous researches have been conducted since the beginning of the 21st century which contradicts the EMH and show that the stock market can be predicted to some extent [3-5]. Exploration of many prediction algorithms in stock market forecasting has taken place and showed that the behavior of stock prices can be forecast [6]. The prediction of the stock market behavior is a very difficult task since this market is very complex, non-linear, and evolutionary. The market is influenced by situations such as investors' sentiments, political events, and overall economic conditions [7]. Three main approaches: fundamental analysis, technical indicators, and machine learning (ML) are used to forecast the stock market. In fundamental analysis, the value of a stock is derived from the general economic and financial factors such inflation, return on equity (ROE), price to earnings (PE) ratios, and debt levels. In technical analysis approach, technicians use charts and market statistics from historical price data to identify market trends and patterns so that they can make fairly accurate forecast of the trajectories of the stock market behavior [8]. The machine learning approach offers system the ability to learn and improve automatically from massive amount of historical data without them being explicitly programmed. Machine learning models have been shown to perform better than both fundamental, and technical analyses in the literature [9-11]. Distributional assumptions are not required by ML models. Also, ML models are able find hidden patterns in time series data [12-13]. Several machine learning algorithms exist, but the focus of this study is on Gaussian Naïve Bayes (GNB) algorithm. GNB is a probabilistic classifier based on Bayes' theorem with assumption of strong (naïve) independence between the features [14]. GNB algorithm is very simple and easy to implement and does not require too many training data. It is highly scalable (it scales linearly with the number of features and data points), not sensitive to irrelevant features and able to deal with missing data very effectively. A major weakness with GNB algorithm is the assumption of independence between predictors. GNB assumes that all the predictors are mutually independent. This assumption is hardly true in real life especially with financial data. However, this assumption can be met by applying feature extraction techniques to extract independent predictors from the given data. Many feature extraction techniques are available in the literature which can be used to achieve this goal. Hence, this work assesses the performance of GNB with different feature scaling and feature extraction techniques in predicting the direction of movement of stock prices.

2 Related studies

Many ML algorithms have been used in the literature of forecasting the direction of stock price. A review of some of those works is provided. Ampomah et al, (2020) [14] studied the effectiveness of tree-based AdaBoost ensemble ML models (namely, AdaBoost-DecisionTree (Ada-DT), AdaBoost-RandomForest (Ada-RF), AdaBoost-Bagging (Ada-BAG), and Bagging-ExtraTrees (Bag-ET)) in predicting stock prices. The experimental results showed that AdaBoost- ExtraTree (Ada-ET) model generated the highest performance among the tree-based AdaBoost ensemble models studied. Kumar and Thenmozhi (2006) [15] carried out a study to forecast the direction of S&P CNX NIFTY Market Index of the National Stock Exchange (NSE). Random forest, linear discriminant analysis, artificial neural network, logit, and SVM machine learning algorithms were used by the researchers. The experimental results indicated that SVM is the best performer among the classification algorithms used. Ou and Wang (2009) [16], studied and applied ten different data mining techniques to forecast stock price movement of Hang Seng index of Hong Kong stock market. The techniques included neural network, Linear discriminant analysis (LDA), Logit model, Quadratic discriminant analysis (QDA), K-nearest neighbor classification, Naïve Bayes based on kernel estimation, Bayesian classification with Gaussian process, Tree based classification, SVM and Least squares support vector machine (LS-SVM). The empirical results presented indicate that the performance of SVM and LS-SVM models are superior to those of the other models. Subha and Nambi (2012) [17] examined the predictability of the movement of BSE-SENSEX and NSE-NIFTY stock indices of the Indian Stock Market by using k-Nearest Neighbours algorithm (k-NN) and Logistic Regression model to predict the daily movement of the indices. Data for the period between January 2006 to May 2011were used. The research outcome shows that the k-NN classifier performed better than the logistic regression model in all the model evaluation metrics used. Saifan et al, (2020) [18] applied the Quantopian algorithmic stock market trading simulator to evaluate ensemble models performance in daily prediction and trading. The ensemble models used are Extremely Randomized Trees, Random Forest, and Gradient Boosting. The models were trained using multiple technical indicators and automatic stock selection. The results showed a significant returns relative to the benchmark and large values of alpha were generated from all models. A study to verify whether modified SVM classifier can be applied successfully in prediction of short-term trends in the stock market was undertaken by Zikowski, (2015) [19]. The author computed and used several technical indicators and statistical measures as input features. Fisher's method was applied to perform feature selection. The study outcome shows that using the modified SVM in conjunction with feature selection enhance significantly the trading strategy results in terms of the total rate of return, as well as the maximum drawdown during a trading period. Patel, et al (2015) [20], compared the performance of Artificial Neural Network

(ANN), support vector machine (SVM), random forest and Naive-Bayes with two different approaches for input data to the models in forecasting the direction of movement of stock and stock price index. The first approach to input data computed ten technical indicators from the stock trading data (open, high, low & close prices) and the second approach represent the technical indicators as trend deterministic data. They evaluated the models with 10 years of historical stock data from 2003 to 2012 of Reliance Industries, Infosys Ltd, CNX Nifty and S&P Bombay Stock Exchange (BSE) Sensex. The outcome of the study shows that for the first approach random forest outperforms other three prediction models on overall performance. Also, that the performance of all the prediction models improved when these technical indicators are represented as trend deterministic data. Sun et al, (2018) [21], proposed a hybrid ensemble learning model combining AdaBoost and LSTM network to predict financial time series. Daily datasets of two major exchange rate and two stock market indices were used for evaluating the model. The experimental outcome shows that the AdaBoost-LSTM ensemble model outperformed the other single forecasting models and ensemble models that were compared with it. Khan et al, (2020) [22] assessed the impact of social media and financial news data on stock market prediction accuracy. The authors performed feature selection and spam tweets reduction on the data sets. Experiments were performed to find stock markets that are difficult to predict and those that were more influenced by social media and financial news. They compared the results of different algorithms to find a consistent classifier. Deep learning and some ensemble classifiers were used. The experimental results indicated that highest prediction accuracies of 80.53% and 75.16% were achieved using social media and financial news, respectively. Also, New York and Red Hat stock markets were difficult to predict, New York and IBM stocks are more influenced by social media, while London and Microsoft stocks by financial news. Random forest classifier was found to be consistent and highest accuracy of 83.22% was achieved by its ensemble. Bhandare et al, (2020) [23] used the Naive Bayes classifier to provide analyse and quantify the performance of stock market analysts by providing ratings. The recommendations given by the analysts was analysed and factors relevant to the success or failure of the recommendation extracted. The Naive Bayes classifier was used provide a rating on the factors thus extracted. The results indicated that the system efficiently analyse the performance of an analyst given their passed records by matching it with the actual stock prices and provide a rating for the analyst using the Naive Bayes classifier. The performance of the system is optimal when Gaussian Naive Bayes Classifier was used. From the above discussion, and to the best of our knowledge, the ability of Gaussian Naïve Bayes to predict stock price movement has not been addressed properly in the existing literature. Hence, a gap study aims to fill in that gap by evaluating the impact of feature scaling and feature extraction techniques on GNB algorithm in prediction of stock price movement.

3 Method

3.1 Experimental design

Stock Data set used for the study were gathered randomly from three different stock market (NYSE, NASDAQ and NSE) through yahoo financial application programming interface (API). Daily data of seven stocks were gathered. Details of the stock data used are given in Table A1 in the Appendix. Forty (40) technical indicators were computed from the raw stock data which comprise of open price, low price, high price, close price and volume. The computed technical indicators were used as input features for the GNB models. Details of these technical indicators are presented in Table A2-A4 in the Appendix. Each data set was split into training and test set. Initial seventy percent (70%) of the data was used as the training set, and the final thirty percent (30%) of the data was used as the test set. In this work, the ability of GNB algorithm in combination with different feature scaling techniques (i.e., Standardization scale and Min-Max scale) and different feature extraction techniques (i.e., PCA, LDA, and FA) to forecast stock price movement were evaluated.

The following GNB models were evaluated and compared: (i) GNB model, (ii) Integrated model based on GNB algorithm and standardization scaling (GNB_Z-Score) (iii) Integrated model based on GNB algorithm and Min-Max normalization (GNB_Min-Max) (iv) Integrated model based on GNB algorithm and principal component analysis (GNB_PCA) (v) Integrated model based on GNB algorithm and factor analysis (GNB_FA) (vi) Integrated model based on GNB algorithm and linear discriminant analysis (GNB_LDA) (vii) Integrated model based on GNB algorithm, standardization scaling, and principal component analysis (GNB_Z-Score_PCA) (viii) Integrated model based on GNB algorithm, standardization scaling, and factor analysis (GNB_Z-Score_FA) (ix) Integrated model based on GNB algorithm, Min-Max normalization, and principal component analysis (GNB_Min-Max_PCA) (x) Integrated model based on GNB algorithm, Min-Max normalization, and factor analysis (GNB_Min-Max_FA).

GNB model applies the GNB algorithm to the raw stock data without any feature scaling or feature extraction to make prediction. GNB_Z-Score model first used the standardization scaling technique to scale the data, and then the GNB algorithm was applied to forecast the movement of stock price. GNB_Min-Max model applied Min-Max scaling technique to scale the data before the GNB algorithm was applied to the scaled data to make predictions. With GNB_PCA model, the PCA was first applied to the unscaled stock data to extract important features from the data, and then the GNB algorithm was applied to the extracted data to make prediction. GNB_FA model initially applied FA technique to the unscaled stock data to extract relevant features from the original data, and then applied the GNB algorithm to the extracted data to make predictions. GNB_LDA model first used LDA technique to extract relevant features from the initial input data, and then applied the GNB algorithm to the extracted data. GNB_Z-Score_PCA model first applied standardization scaling technique to the initial data to scale it, after that it then applied PCA to extract important features from the scaled data and finally applied the GNB to the extracted scaled stock data to make predictions. GNB_Z-Score_FA model initially used standardization scaling technique to scale the data, then applies the FA technique to extract relevant features from the scaled data and the GNB algorithm was applied to the extracted scaled data to make predictions. GNB_Min-Max_PCA model initially scaled the input data with Min-Max scaling technique, then applied PCA to extract important feature from the original stock data, and then applied the GNB algorithm to the extracted scaled data to make predictions. GNB_Min-Max_FA uses Min-Max to first scaled the data, then applied the FA technique to extract relevant features from the scaled data, and finally applied the GNB algorithm to make predictions.

3.2 Feature scaling techniques

Feature Scaling is a way of standardizing the independent features that are present in the data within a fixed range. The two most widely used feature scaling techniques are standardization scaling and Min-Max Normalization.

3.2.1 Standardization scaling

Standardization scaling (Z-score) is a scaling method that centers the values around the mean with a unit standard deviation. The data is scaled to a specific area to enable a thorough analysis. The variables are rescaled to have a mean of zero and the resulting distributions have a unit standard deviation. The standardized scaling is expressed by the formula below.

$$X' = \frac{X-\mu}{\sigma} \tag{1}$$

μ = mean of the feature values

σ = standard deviation of the feature values

3.2.2 Min-max normalization

Min-Max normalization (Min-Max) is a scaling approach in which features are re-scaled so that the data will fall in the range of zero and one. It undertakes a linear alteration on the initial data [24]. In the Min-Max scaling, the minimum value of every feature is converted to zero, and the maximum value of each feature is converted to one. The formula below expresses how the normalized form of each feature is computed.

$$X' = \frac{X-X_{min}}{Xmin_{max}} \tag{2}$$

X_{min}= minimum value of the feature, $X_{\max}$ = maximum value of the feature

3.3 Feature extraction techniques

Feature extraction is a dimensionality reduction process that extracts important features or attributes of the data in order to reduce the initial set of data to generate a more concise description of the data for processing. There are many feature extraction techniques in existing literature, however, in this study the principal component analysis (PCA), Linear discriminant analysis (LDA), and factor analysis (FA) were applied.

3.3.1 Principal component analysis

Principal Component Analysis (PCA) is a dimensionality-reduction technique that transform higher data sets to a lower dimensional set. It transforms a data set of interrelated features, into a new set of uncorrelated features called principal components (PCs) and the initial few of these PCs hold most of the variation present in the entire data set [25]. The PCs are linear combination of the actual features in such a way that the first PC has the largest amount of variation and the second PC is orthogonal to the first PC and has the most variance among the remaining PCs. The subsequent PCs follow in that order. The underlying assumption in PCA is that the coordinates with the large variants demonstrate the divergence between sample points, while the coordinates with lesser variants may be a source of noise, which must be ignored or suppressed. The correlation between two dimensions denotes irrelevant information, which will not be presented. This is why PCA requires the subsequent coordinates to be orthogonal to previous coordinates [26]. PCA is sensitive to scaling.

3.3.2 Linear discriminant analysis

Linear Discriminant Analysis (LDA) is a supervised linear transformation technique that computes the linear discriminants (directions) that will represent the axes which maximize the differences between multiple classes. The objective of the technique is to maximize the ratio of the between-group variance and the within-group variance. When the ratio is maximum, then the instances within each group have the least possible scatter and the groups are separated from each other the most. LDA is used to map features in higher dimension space into a lower dimension space while keeping the class-discriminatory information [27]. LDA is not sensitive to scaling, hence, the performance of LDA remains the same with or without scaling. The LDA uses two criteria to generate a new axis: (i) maximize the distance between means of the two classes, (ii) minimize the variation within each class.

3.3.3 Factor analysis

Factor analysis (FA) is a data reduction technique that describes variability among observed, correlated features in terms of a potentially smaller number of unobserved (latent) features called factors. The observed features are modeled as linear combinations of the factors plus error. FA extracts maximum common variance from all features and place them under a common score. This score as an index of all features can be used to do further analysis. FA evaluates how much of the variability in the data is as a result of common factors. The main goals of FA are to display multidimensional data in a lower dimensional space with minimum loss of information and to extract the independent latent of the data [28]. The FA technique makes the following assumptions: linear relationship

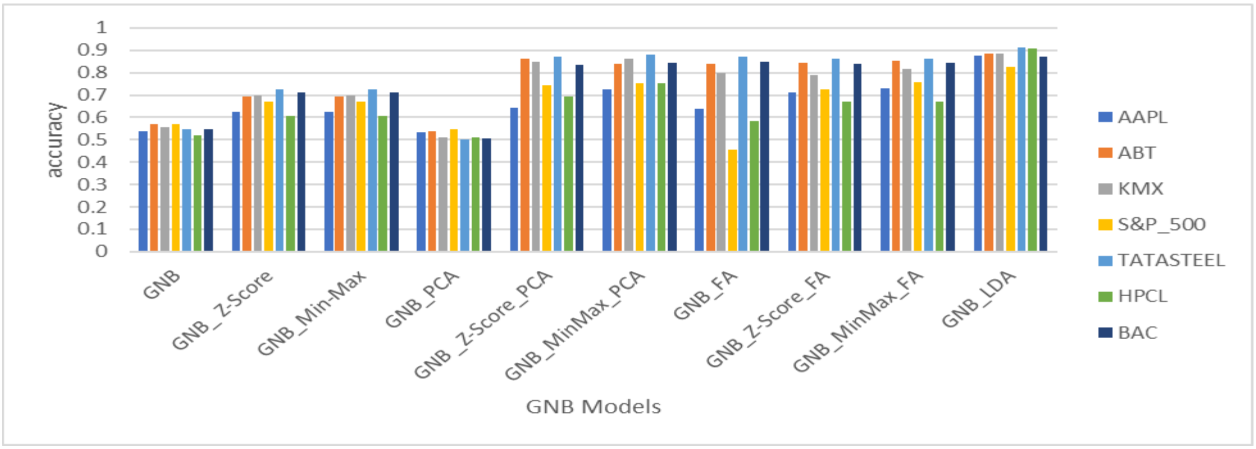


Figure 1: Accuracy results of the GNB models on the different stock data sets.

DataSets	GNB	GNB_ Z-Score	GNB_ MinMax	GNB_ PCA	GNB_ Z-Score_PCA
AAPL	0.5361	0.6241	0.6241	0.5342	0.6444
ABT	0.5713	0.6954	0.6954	0.5361	0.8639
KMX	0.5583	0.6982	0.6982	0.5111	0.8509
S&P_500	0.5722	0.6704	0.6704	0.5472	0.7444
TATASTEEL	0.5461	0.7232	0.7232	0.5012	0.8713
HPCL	0.5197	0.6085	0.6085	0.5126	0.6953
BAC	0.5472	0.7111	0.7111	0.5056	0.8333
Mean	**0.5501**	**0.6758**	**0.6758**	**0.5211**	**0.7862**

DataSets	GNB_ MinMax_ PCA	GNB_ FA	GNB_ Z-Score_FA	GNB_ MinMax_FA	GNB_ LDA
AAPL	0.7241	0.6370	0.7111	0.7314	0.8769
ABT	0.8398	0.8407	0.8462	0.8528	0.8861
KMX	0.8648	0.7963	0.7907	0.8176	0.8870
S&P_500	0.7546	0.4537	0.7269	0.7583	0.8259
TATASTEEL	0.8809	0.8701	0.8616	0.8637	0.9142
HPCL	0.7548	0.5832	0.6700	0.6700	0.9092
BAC	0.8435	0.8509	0.8407	0.8454	0.8713
Mean	**0.8089**	**0.7188**	**0.7782**	**0.7913**	**0.8815**

Table 1: Accuracy results recorded by the GNB models.

exists between the observed features and the common factors, no multi-collinearity is present, it includes relevant features into analysis, and there is true correlation between features and factors.

3.4 Evaluation metrics

The performances of the models were evaluated using the following evaluation metrics:

Accuracy: The percentage of entire instances rightly predicted by the model.

$$accuracy = \frac{tp+tn}{tp+tn+fp+fn} \quad (3)$$

F1-score: This is a harmonic mean of precision and recall

$$F1 - score = \frac{2 \times precision \times recall}{precision+recall} \quad (4)$$

Specificity: The proportion of negative instances rightly predicted by the classifier out of the total instances that are actually negative. This shows a model's ability to classify true negative instances as negative.

$$specificity = \frac{tn}{tn+fn} \quad (5)$$

Area Under Receiver Operating Characteristics Curve (AUC): Measures the ability of the classifier to distinguish between the positive and negative classes. A perfect classifier will have AUC of one. AUC measures tradeoff between specificity and recall.

Kendall's coefficient of concordance (W): is a metric that uses ranks to establish an agreement among raters. It measures the agreement among different raters who are evaluating a given set of objects [29]. Depending on the area where it is being used, the raters can be variables, characters, and so on. The raters are the different data sets in this work.

4 Experimental results

Table 1 provides the accuracy results generated by the various GNB models on the different stock data sets used. GNB_LDA model produced accuracy results which were better than all the other GNB models on each of the stock data used. The highest accuracy value recorded by any of

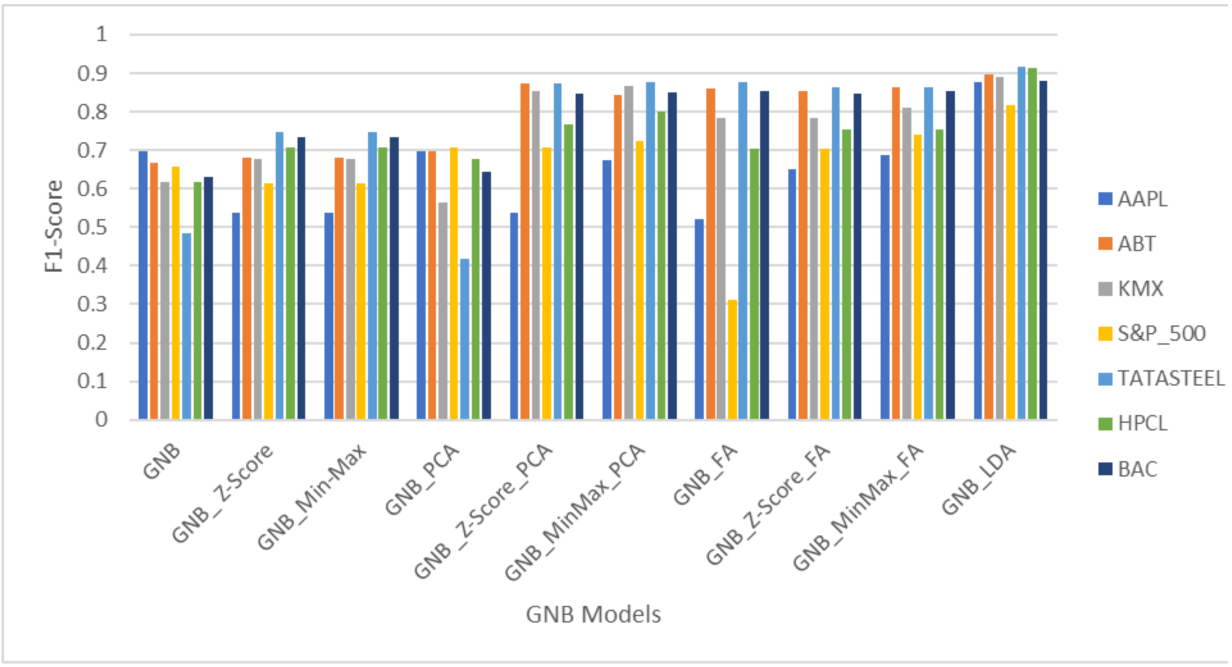


Figure 2: F1-score of the GNB models on the different stock data sets.

DataSets	GNB	GNB _ Z-Score	GNB _ MinMax	GNB _ PCA	GNB_ Z-Score_PCA
AAPL	0.6962	0.5365	0.5365	0.6964	0.5362
ABT	0.6662	0.6803	0.6803	0.6980	0.8740
KMX	0.6175	0.6766	0.6766	0.5629	0.8540
S&P_500	0.6583	0.6139	0.6139	0.7074	0.7058
TATASTEEL	0.4860	0.7461	0.7461	0.4166	0.8747
HPCL	0.6161	0.7074	0.7074	0.6777	0.7659
BAC	0.6304	0.7342	0.7342	0.6454	0.8454
Mean	**0.6244**	**0.6707**	**0.6707**	**0.6292**	**0.7794**

DataSets	GNB_ MinMax_PCA	GNB_ FA	GNB_ Z-Score_FA	GNB_ MinMax_FA	GNB_ LDA
AAPL	0.6740	0.5220	0.6494	0.6875	0.8783
ABT	0.8443	0.8590	0.8534	0.8635	0.8976
KMX	0.8653	0.7835	0.7839	0.8108	0.8891
S&P_500	0.7237	0.3114	0.7053	0.7398	0.8175
TATASTEEL	0.8760	0.8762	0.8626	0.8648	0.9167
HPCL	0.7990	0.7031	0.7532	0.7532	0.9119
BAC	0.8516	0.8546	0.8473	0.8542	0.8790
Mean	**0.8048**	**0.7014**	**0.7793**	**0.7963**	**0.8843**

Table 2: F1-scores recorded by the GNB model.

the models is 0.9142 generated by the GNB_LDA model on the TATASTEEL data. The least accuracy value recorded by any of the models is 0.5012 by GNB_ PCA model on the TATASTEEL stock data. The mean accuracy value of GNB_LDA model (0.8815) was the highest mean accuracy value, and GNB_ PCA produced the least mean accuracy value (0.5211). Figure 1 provides the column chart of the accuracy values produced by the GNB models on the different stock data.

Table 2 shows the outcome of F1-scores evaluation metric of the GNB models on the different stock data sets used. The F1-score of GNB_LDA model was better than all the other GNB models on each of the stock data. The highest F1-score recorded by any of the models was 0.9167 generated by the GNB_LDA model on the TATASTEEL data. The least F1-score value recorded by any of the models was 0.4166 produced by GNB_ PCA on the TATASTEEL stock data. The mean F1-score of GNB_LDA model (0.8815) was the highest mean F1-score among the GNB models, and the mean F1-score of the GNB model (0.6244) was the least mean F1-score among the various models. Figure 2 represents the column chart of the F1-score evaluation metric outcome for the GNB models on the different stock data.

Table 3 presents the specificity results of the models on the different stock data sets used. GNB_Z-Score_FA

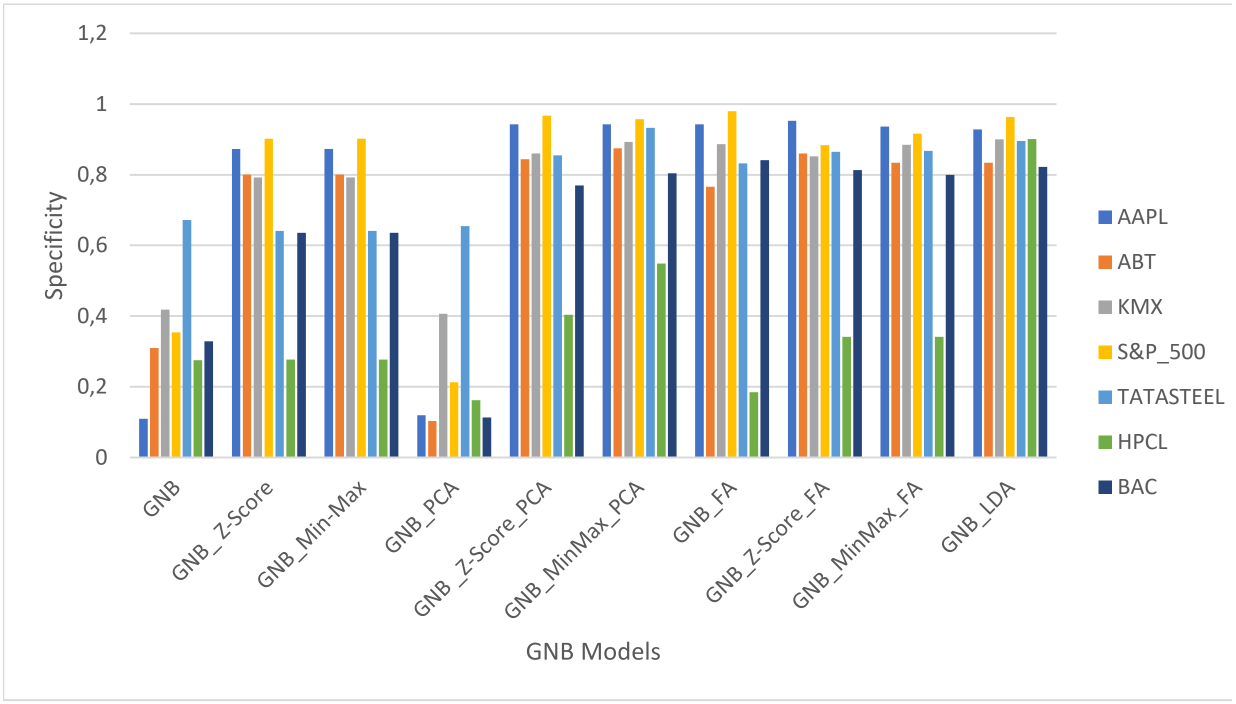


Figure 3: Specificity of the GNB models on the different stock data sets.

DataSets	**GNB**	**GNB_ Z-Score**	**GNB_ Min-Max**	**GNB_ PCA**	**GNB_ Z-Score_PCA**
AAPL	0.1099	0.8728	0.8728	0.1200	0.9423
ABT	0.3094	0.8004	0.8004	0.1030	0.8443
KMX	0.4184	0.7927	0.7927	0.4069	0.8599
S&P_500	0.3538	0.9018	0.9018	0.2131	0.9672
TATASTEEL	0.6717	0.6413	0.6413	0.6544	0.8544
HPCL	0.2754	0.2774	0.2774	0.1621	0.4037
BAC	0.3283	0.6359	0.6359	0.1132	0.7698
Mean	**0.3524**	**0.7032**	**0.7032**	**0.2532**	**0.8059**

DataSets	**GNB_ Min-Max_PCA**	**GNB_ FA**	**GNB_ Z-Score_FA**	**GNB_ Min-Max_FA**	**GNB_ LDA**
AAPL	0.9423	0.9424	0.9523	0.9363	0.9284
ABT	0.8743	0.7665	0.8603	0.8343	0.8343
KMX	0.8925	0.8868	0.8522	0.8848	0.9002
S&P_500	0.9571	0.9800	0.8834	0.9161	0.9632
TATASTEEL	0.9326	0.8326	0.8652	0.8674	0.8957
HPCL	0.5487	0.1843	0.3416	0.3416	0.9006
BAC	0.8038	0.8415	0.8132	0.8000	0.8226
Mean	**0.8502**	**0.7763**	**0.7955**	**0.7972**	**0.8921**

Table 3: Specificity values recorded by the GNB models.

model outperformed the other models on AAPL. GNB_Min-Max_ PCA model performed better than the rest of the models on ABT, and TATASTEEL stock data. GNB_LDA model recorded better specificity results than the rest of the models on KMX, and HPCL stock data. GNB_FA model produced better specificity results than the other models on S&P_500 and BAC stock data. The highest specificity value recorded by any of the GNB models was 0.9800 generated by the GNB_FA model on the S&P_500 data. The least specificity value recorded by any of the models was 0.1030 produced by GNB_ PCA on the ABT stock data. The mean specificity of GNB_LDA model (0.8921) was the highest mean specificity among the GNB models, and the mean specificity of GNB_PCA model (0.2532) was the least mean specificity among the GNB models. Figure 3 presents the column chart of the specificity results of the GNB models on the different stock data.

Table 4 provides the AUC results of the GNB models on the different stock data sets used. The performance of GNB_LDA model on each of the stock data was better than the rest of the GNB models. In general, the highest AUC value recorded by any of the models was 0.9743 generated by the GNB_LDA model on the TATASTEEL

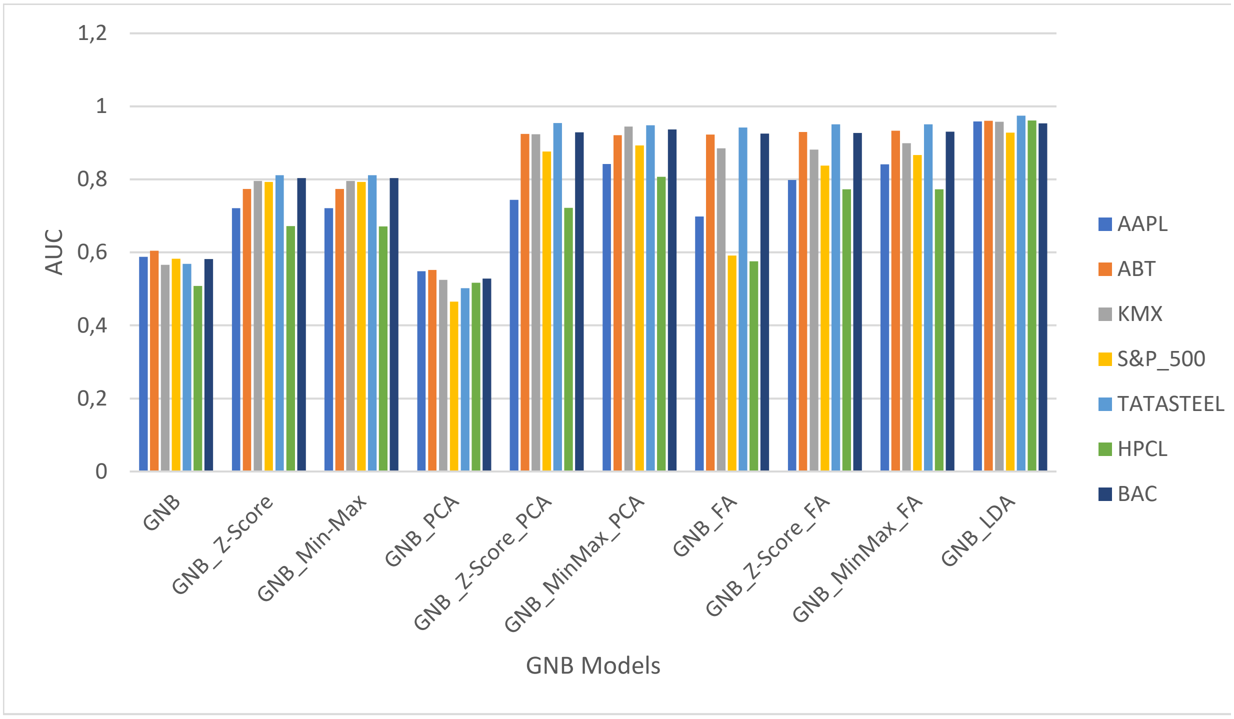


Figure 4: AUC values of the GNB models on the different stock data sets.

DataSets	**GNB**	**GNB_ Z-Score**	**GNB_ Min-Max**	**GNB_ PCA**	**GNB_ Z-Score_PCA**
AAPL	0.5883	0.7216	0.7216	0.5487	0.7438
ABT	0.6046	0.7736	0.7736	0.5517	0.9246
KMX	0.5661	0.7958	0.7958	0.5251	0.9238
S&P_500	0.5830	0.7927	0.7927	0.4649	0.8764
TATASTEEL	0.5688	0.8115	0.8115	0.5022	0.9540
HPCL	0.5085	0.6725	0.6708	0.5172	0.7225
BAC	0.5819	0.8037	0.8037	0.5281	0.9291
Mean	**0.5716**	**0.7673**	**0.7671**	**0.5197**	**0.8677**

DataSets	**GNB_ Min-Max_PCA**	**GNB_ FA**	**GNB_ Z-Score_FA**	**GNB_ Min-Max_FA**	**GNB_ LDA**
AAPL	0.8421	0.6986	0.7982	0.8416	0.9586
ABT	0.9211	0.9232	0.9296	0.9329	0.9603
KMX	0.9447	0.8850	0.8817	0.8994	0.9581
S&P_500	0.8927	0.5914	0.8382	0.8663	0.9282
TATASTEEL	0.9483	0.9421	0.9507	0.9511	0.9743
HPCL	0.8067	0.5757	0.7726	0.7726	0.9617
BAC	0.9364	0.9251	0.9268	0.9305	0.9532
Mean	**0.8989**	**0.7916**	**0.8711**	**0.8849**	**0.9563**

Table 4: AUC values recorded by the GNB models.

data. The smallest AUC value recorded by any of the models was 0.4649 by GNB_ PCA on the S&P_500 stock data. The mean AUC value of GNB_LDA model (0.9563) was the highest mean AUC recorded among the GNB models. The mean AUC of the GNB_PCA model (0.5197) was the least mean AUC among the various models. Figure 4 represents the column chart of the AUC evaluation metric result for the GNB models on the different stock data.

The ROC curves of the various GNB models on AAPL, ABT, KMX, S&P_500, TATASTEEL, HPCL, and BAC stock data sets are presented by Figure 5 to Figure 11 respectively.

Table 5 to Table 8 present the Kendall's coefficient of concordance rankings of the GNB models using accuracy, F1 score, specificity, and AUC evaluation results respectively. The study used a cutoff value of 0.05, and the Kendall's coefficient is considered significant and able to assign ranks to the models when $p < 0.05$ and $\chi^2 >$ 16.919.

From Table 5, the Kendall's coefficient was significant to rank the GNB models using the accuracy

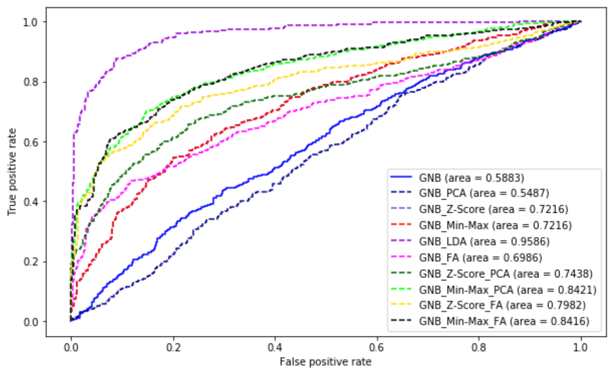


Figure 5: ROC Curves of the GNB models on the AAPL stock data set.

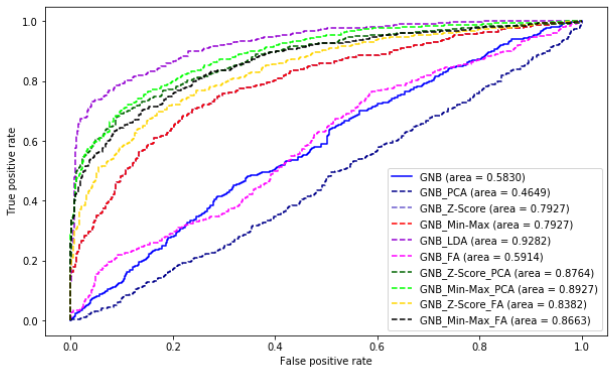


Figure 8: ROC Curves of the GNB models on the S&P_500 index stock data set.

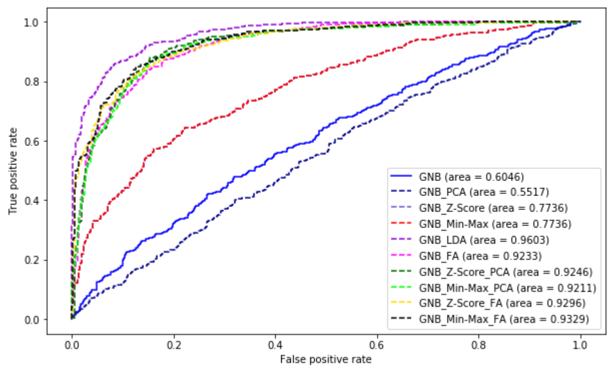


Figure 6: ROC Curves of the GNB models on the ABT stock data set.

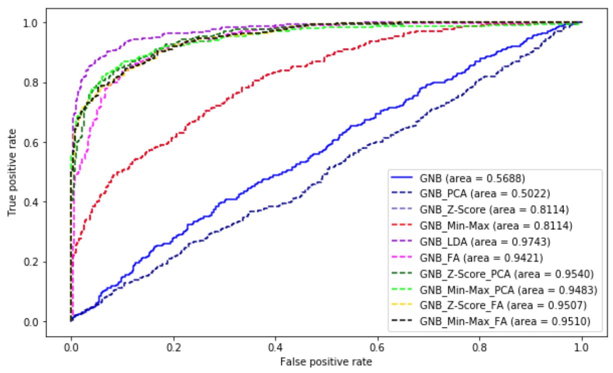


Figure 9: ROC Curves of the GNB models on the TATASTEEL stock data set.

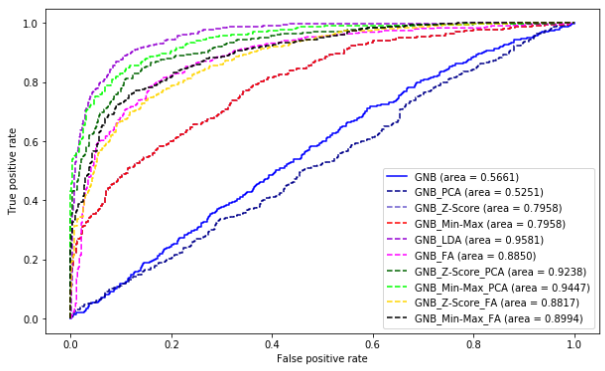


Figure 7: ROC Curves of the GNB models on the KMX stock data set.

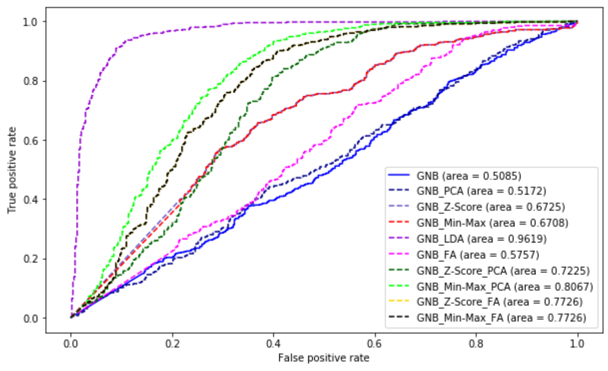


Figure 10: ROC Curves of the GNB models on the TATASTEEL stock data set.

results. GNB_LDA model attained the highest rank. The overall ranking of the models was:

GNB_LDA > GNB_Min-Max _PCA > GNB_Min-Max_FA > GNB_Z-Score_PCA > GNB_Z-Score_FA > GNB_FA > GNB_Z-Score = GNB_Min-Max > GNB > GNB_PCA

Table 6 shows that Kendall's coefficient was significant to rank the GNB models using the F1-Score metric. GNB_LDA model generated the highest rank. The overall ranking was given as:

GNB_LDA > GNB_Min-Max _PCA > GNB_Min-Max_FA > GNB_Z-Score_PCA > GNB_Z-Score_FA > GNB_FA > GNB_Z-Score = GNB_Min-Max > GNB_PCA > GNB

Table 7 indicates that Kendall's coefficient is significant to rank the GNB models using specificity

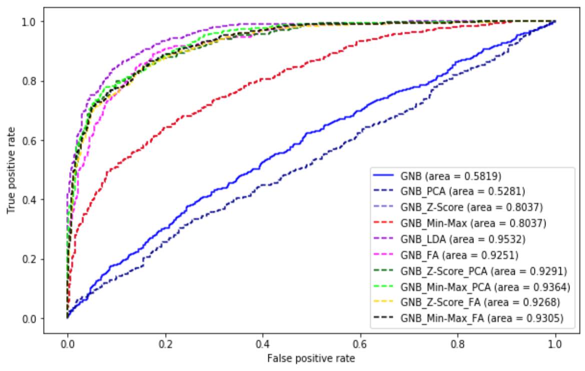


Figure 11: ROC Curves of the GNB models on the HPCL stock data set

Metric	W	$\chi 2$	p	Ranks					
Accuracy	0.84	53.07	0.00	**Technique**	GNB	GNB _ Z-Score	GNB_ Min-Max	GNB_ PCA	GNB _ Z-Score_ PCA
				Mean Rank	2.14	3.79	3.79	1.14	7.29
				Technique	GNB_ Min-Max _PCA	GNB_ FA	GNB_ Z-Score _FA	GNB_ Min-Max _FA	GNB_ LDA
				Mean Rank	7.86	5.29	6.07	7.50	10.00

Table 5: Kendall's W Test of Concordance Rankings of the GNB models using the accuracy metric.

Metric	W	$\chi 2$	p	Ranks					
F1-Score	0.62	39.22	0.00	**Technique**	GNB	GNB_ Z-Score	GNB_ Min-Max	GNB_ PCA	GNB_ Z-Score _PCA
				Mean Rank	2.71	3.36	3.36	3.14	6.43
				Technique	GNB_ Min-Max _PCA	GNB_ FA	GNB_ Z-Score _FA	GNB_ Min-Max _FA	GNB_ LDA
				Mean Rank	7.43	5.00	6.07	7.36	10.00

Table 6: Kendall's W Test of Concordance Rankings of the GNB models using the Specificity metric.

Metric	W	$\chi 2$	p	Ranks					
Specificity	0.70	43.76	0.00	**Technique**	GNB	GNB _ Z-Score	GNB _ Min-Max	GNB_ PCA	GNB _ Z-Score _PCA
				Mean Rank	2.29	3.64	3.64	1.43	7.07
				Technique	GNB_ Min-Max _PCA	GNB _ FA	GNB _ Z-Score _FA	GNB_ Min-Max _ FA	GNB _LDA
				Mean Rank	8.50	6.71	6.93	6.57	8.21

Table 7: Kendall's W Test of Concordance Rankings of the GNB models using the Specificity metric.

Metric	W	$\chi 2$	p	Ranks					
AUC	0.90	56.95	0.00	**Technique**	GNB	GNB _ Z-Score	GNB_ Min-Max	GNB _PCA	GNB _ Z-Score _PCA
				Mean Rank	1.86	4.00	3.86	1.14	7.29
				Technique	GNB_ Min-Max _ PCA	GNB _ FA	GNB_ Z-Score _FA	GNB_ Min-Max _FA	GNB _ LDA
				Mean Rank	8.00	4.43	6.64	7.79	10.00

Table 8: Kendall's W Test of Concordance Rankings of the GNB models using the AUC metric.

metric. GNB_Min-Max _PCA model produced the highest rank. The overall ranking of the models was: GNB_Min-Max _PCA > GNB_LDA > GNB_Z-Score_PCA > GNB_Z-Score_FA > GNB_Min-Max_FA > GNB_FA > GNB_Z-Score = GNB_Min-Max > GNB > GNB_PCA

Table 8 shows that Kendall's coefficient was significant to rank the GNB models using AUC metric. GNB_LDA generated the highest rank. The overall ranking of the GNB models was: GNB_LDA > GNB_Min-Max _PCA > GNB_Min-Max_FA > GNB_Z-Score_PCA > GNB_Z-Score_FA > GNB_FA > GNB_Z-Score > GNB_Min-Max > GNB > GNB_PCA.

The ROC curves of the various GNB models on AAPL, ABT, KMX, S&P_500, TATASTEEL, HPCL, and BAC stock data sets are presented by Figure 5 to Figure 11 respectively.

Table 5 to Table 8 present the Kendall's coefficient of concordance rankings of the GNB models using accuracy, F1 score, specificity, and AUC evaluation results respectively. The study used a cutoff value of 0.05, and the Kendall's coefficient is considered significant and able to assign ranks to the models when $p < 0.05$ and $\chi^2 > 16.919$.

From Table 5, the Kendall's coefficient was significant to rank the GNB models using the accuracy results. GNB_LDA model attained the highest rank. The overall ranking of the models was:
GNB_LDA > GNB_Min-Max _PCA > GNB_Min-Max_FA > GNB_Z-Score_PCA > GNB_Z-Score_FA > GNB_FA > GNB_Z-Score = GNB_Min-Max > GNB > GNB_PCA

Table 6 shows that Kendall's coefficient was significant to rank the GNB models using the F1-Score metric. GNB_LDA model generated the highest rank. The overall ranking was given as:
GNB_LDA > GNB_Min-Max _PCA > GNB_Min-Max_FA > GNB_Z-Score_PCA > GNB_Z-Score_FA > GNB_FA > GNB_Z-Score = GNB_Min-Max > GNB_PCA > GNB

Table 7 indicates that Kendall's coefficient is significant to rank the GNB models using specificity metric. GNB_Min-Max _PCA model produced the highest rank. The overall ranking of the models was: GNB_Min-Max _PCA > GNB_LDA > GNB_Z-Score_PCA > GNB_Z-Score_FA > GNB_Min-Max_FA > GNB_FA > GNB_Z-Score = GNB_Min-Max > GNB > GNB_PCA

Table 8 shows that Kendall's coefficient was significant to rank the GNB models using AUC metric. GNB_LDA generated the highest rank. The overall ranking of the GNB models was:
GNB_LDA > GNB_Min-Max _PCA > GNB_Min-Max_FA > GNB_Z-Score_PCA > GNB_Z-Score_FA > GNB_FA > GNB_Z-Score > GNB_Min-Max > GNB > GNB_PCA.

5 Conclusion

This study assessed how the GNB algorithm performed with different feature scaling (i.e., standardization scaling, and Min-Max scaling techniques) and feature extraction techniques (i.e., PCA, LDA, and FA) in predicting the direction of movement of stock price using stock data randomly collected from different stock markets. The performance of the various GNB models were evaluated using accuracy, F1-Score, specificity and AUC evaluation metrics. Kendall's W test of concordance was used to generate ranks for the GNB models using the evaluation metrics.

The experimental results indicated that application of scaling techniques improved the performance of the GNB model. Models based on integration of GNB algorithm, feature scaling technique and feature extraction technique generated results which were superior to results produced by models based on either integration of GNB algorithm and feature scaling technique or GNB algorithm and feature extraction technique with the exception of GNB_LDA. In general, the model based on integration of GNB algorithm and Linear Discriminant Analysis (GNB_LDA) outperformed all the other models of GNB considered in three of the four evaluation metrics (i.e., accuracy, F1-score, and AUC). Similarly, the predictive model based on GNB algorithm, Min-Max scaling, and PCA produced the best rank using the specificity results. In addition, GNB produced better performance with Min-Max scaling technique than it does with standardization scaling techniques.

6 Acknowledgement

This work was supported by the NSFC-Guangdong Joint Fund (Grant No. U1401257), National Natural Science Foundation of China (Grant Nos. 61300090, 61133016, and 61272527), science and technology plan projects in Sichuan Province (Grant No. 2014JY0172) and the opening project of Guangdong Provincial Key Laboratory of Electronic Information Products Reliability Technology (Grant No. 2013A061401003)

7 Reference

[1] Fama E. F, Fisher L, Jensen M, Roll R (1969) The adjustment of stock price to new information. Int Eco Rev 10(1):1–21

[2] Yeh, I.-C., & Hsu, T.-K. (2014). Exploring the dynamic model of the returns from value stocks and growth stocks using time series mining. Expert Systems with Applications, 41, 7730–7743. https://doi.org/10.1016/j.eswa.2014.06.036

[3] Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. Journal of Computational Science, 2, 1–8. https://doi.org/10.1016/j.jocs.2010.12.007

[4] Smith, V. L. (2003). Constructivist and ecological rationality in economics. American Economic Review, 93, 465–508. https://doi.org/10.1257/000282803322156954

[5] Gandhmal, D. P., & Kumar, K. (2019). Systematic analysis and review of stock market prediction techniques. Computer Science Review, 34, 100190. https://doi.org/10.1016/j.cosrev.2019.08.001

[6] Huang, C.-J., Yang, D.-X., & Chuang, Y.-T. (2008). Application of wrapper approach and composite classifier to the stock trend prediction. Expert Systems with Applications, 34(4), 2870–2878. https://doi.org/10.1016/j.eswa.2007.05.035

[7] Huang, W., Nakamori, Y., &Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. Computers & Operations Research, 32(10), 2513–2522. https://doi.org/10.1016/j.cor.2004.03.016

[8] Maragoudakis M., Serpanos D. (2015). Exploiting Financial News and Social Media Opinions for Stock Market Analysis using MCMC Bayesian Inference. Computational Economics. DOI 10.1007/s10614-015-9492-9. https://doi.org/10.1007/s10614-015-9492-9

[9] Iqbal, N., & Islam, M. (2019). Machine learning for dengue outbreak prediction: A performance

evaluation of different prominent classifiers. Informatica, 43(3), 361 -371. https://doi.org/10.31449/inf.v43i3.1548

[10] Abaker, A. A., & Saeed, F. A. (2021). A Comparative Analysis of Machine Learning Algorithms to Build a Predictive Model for Detecting Diabetes Complications. Informatica, 45(1), 117 -125. https://doi.org/10.31449/inf.v45i1.3111

[11] Zhang, Y., & Wu, L. (2009). Stock market prediction of s&p 500 via combination of improved bco approach and bp neural network. Expert Systems with Applications, 36 (5), 8849–8854. https://doi.org/10.1016/j.eswa.2008.11.028
Meesad, P., & Rasel, R. I. (2013). Predicting stock market price using support vector regression. In Informatics, electronics & vision (iciev), 2013 international conference on informatics. IEEE, 2013, 1–6. https://doi.org/10.1109/iciev.2013.6572570

[12] Zhou, Z., Gao, M., Liu, Q., & Xiao, H. (2020). Forecasting stock price movements with multiple data sources: Evidence from stock market in China. Physica A: Statistical Mechanics and its Applications, 542, 123389. https://doi.org/10.1016/j.physa.2019.123389

[13] Ampomah, E. K., Qin, Z., Nyame, G., & Botchey, F. E. (2021). Stock market decision support modeling with tree-based AdaBoost ensemble machine learning models. Informatica, 44(4), 363 – 375 https://doi.org/10.31449/inf.v44i4.3159

[14] Kumar, M., & Thenmozhi,M. (2006). Forecasting Stock index movement: A comparison of support vector machines and random forest. SSRN Scholarly Paper. Rochester, NY: Social Science Research Network, January 24, 2006. https://doi.org/10.2139/ssrn.876544

[15] Ou, P., & Wang, H. (2009). Prediction of stock market index movement by ten data mining techniques. Modern Applied Science, 3(12), 28. https://doi.org/10.5539/mas.v3n12p28

[16] Subha, M. V., & Nambi, S. T. (2012). Classification of Stock Index movement using K-nearest neighbours (k-NN) algorithm. WSEAS Transactions on Information Science & Applications, 9(9), 261–270. P259.

[17] Saifan R, Sharif K, Abu-Ghazaleh M, Abdel-Majeed M. Investigating Algorithmic Stock Market Trading Using Ensemble Machine Learning Methods. Informatica. 2020 Sep 15;44(3), 311-325 https://doi.org/10.31449/inf.v44i3.2904

[18] Zikowski, K. (2015). Using volume weighted support vector machines with walk forward testing and feature selection for the purpose of creating stock trading strategy. Expert Systems with Applications, 42, 1797–1805. https://doi.org/10.1016/j.eswa.2014.10.001

[19] Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques. Expert Systems with Applications, 42, 259–268. https://doi.org/10.1016/j.eswa.2014.07.040

[20] Sun, S., Wei, Y., & Wang, S. (2018). AdaBoost-LSTM Ensemble Learning for Financial Time Series Forecasting. Computational Science – ICCS 2018, 590–597. https://doi:10.1007/978-3-319-93713-7_55

[21] Khan, W., Ghazanfar, M.A., Azam, M.A. et al. (2020). Stock market prediction using machine learning classifiers and social media, news. J Ambient Intell Human Comput. https://doi.org/10.1007/s12652-020-01839-w

[22] Bhandare, Y., Bharsawade, S., Nayyar, D., Phadtare, O., & Gore, D. (2020). SMART: Stock Market Analyst Rating Technique Using Naive Bayes Classifier. 2020 International Conference for Emerging Technology (INCET). https://doi:10.1109/incet49848.2020.9154002

[23] Saranya C. Manikandan G. (2013). A study on normalization techniques for privacy preserving data mining. International Journal of Engineering and Technology (IJET). 5(3):2701-2714

[24] Abdi H. Williams L. J. (2010). Principal component analysis. Wiley interdisciplinary reviews: computational statistics. 2(4):433-459 https://doi.org/10.1002/wics.101

[25] Bro R. Smilde A. K. (2014) Principal component analysis. Analytical Methods. 6(9):2812-2831

[26] Tharwat A. Gaber T. Ibrahim A. Hassanien A. E. Linear discriminant analysis: A detailed tutorial. AI communications. 2017, 30(2):169-190 https://doi.org/10.3233/aic-170729

[27] Maskey R. Fei J. Nguyen H. O. (2018), Use of exploratory factor analysis in maritime research. The Asian journal of shipping and logistics. 34(2):91-111 https://doi.org/10.1016/j.ajsl.2018.06.006

[28] Kendall, M. G. Babington, S. B. (1939). The Problem of m Rankings, The Annals of Mathematical Statistics, 10, 275-287.

8 Appendix

Data Set	Stock Market	Time Frame	Number of Sample
AAPL	NASDAQ	2005-01-01 to 2019-12-30	3773
ABT	NYSE	2005-01-01 to 2019-12-30	3773
BAC	NYSE	2005-01-01 to 2019-12-30	3773
S&P_500	INDEXSP	2005-01-01 to 2019-12-30	3773
HPCL	NSE	2005-01-01 to 2019-12-30	3278
KMX	NYSE	2005-01-01 to 2019-12-30	3773
TATASTEEL	NSE	2005-01-01 to 2019-12-30	3476

Table A1: Detail of stock data sets used.

Volume Indicator	Description
Chaikin A/D Line (ADL)	Estimates the Advance/Decline of the market.
Chaikin A/D Oscillator (ADOSC)	Indicator of another indicator. It is created through application of MACD to the Chaikin A/D Line
On Balance Volume (OBV)	Uses volume flow to forecast changes in price of stock

Table A2: Description of Volume Indicators used in the study.

Price Transform Indicator	Description
Median Price (MEDPRICE)	Measures the mid-point of each day’s high and low prices.
Typical Price (TYPPRICE)	Measures the average of each day’s price.
Weighted Close Price (WCLPRICE)	Average of each day's price with extra weight given to the closing price.

Table 3: Description of Price Transform Function.

Overlap Studies Indicators	Description
Bollinger Bands (BBANDS)	Describes the different highs and lows of a financial instrument in a particular duration.
Weighted Moving Average (WMA)	Moving average that assign a greater weight to more recent data points than past data points
Exponential Moving Average (EMA)	Weighted moving average that puts greater weight and importance on current data points, however, the rate of decrease between a price and its preceding price is not consistent.
Double Exponential Moving Average (DEMA)	It is based on EMA and attempts to provide a smoothed average with less lag than EMA.
Kaufman Adaptive Moving Average (KAMA)	Moving average designed to be responsive to market trends and volatility.
MESA Adaptive Moving Average (MAMA)	Adjusts to movement in price based on the rate of change of phase as determined by the Hilbert transform discriminator.
Midpoint Price over period (MIDPRICE)	Average of the highest close minus lowest close within the look back period
Parabolic SAR (SAR)	Heights potential reversals in the direction of market price of securities.
Simple Moving Average (SMA)	Arithmetic moving average computed by averaging prices over a given time period.
Triple Exponential Moving Average (T3)	It is a triple smoothed combination of the DEMA and EMA
Triple Exponential Moving Average (TEMA)	An indicator used for smoothing price fluctuations and filtering out volatility. Provides a moving average having less lag than the classical exponential moving average.
Triangular Moving Average (TRIMA)	Moving average that is double smoothed (averaged twice)

Table A3: Description of Overlap Studies Indicators used in the study.

Momentum Indicators	Description
Average Directional Movement Index (ADX)	Measures how strong or weak (strength of) a trend is over time
Average Directional Movement Index Rating (ADXR)	Estimates momentum change in ADX.
Absolute Price Oscillator (APO)	Computes the differences between two moving averages
Aroon	Used to find changes in trends in the price of an asset
Aroon Oscillator (AROONOSC)	Used to estimate the strength of a trend
Balance of Power (BOP)	Measures the strength of buyers and sellers in moving stock prices to the extremes
Commodity Channel Index (CCI)	Determine the price level now relative to an average price level over a period of time
Chande Momentum Oscillator (CMO)	Estimated by computing the difference between the sum of recent gains and the sum of recent losses
Directional Movement Index (DMI)	Indicate the direction of movement of the price of an asset
Moving Average Convergence /Divergence (MACD)	Uses moving averages to estimate the momentum of a security asset
Money Flow Index (MFI)	Utilize price and volume to identify buying and selling pressures
Minus Directional Indicator (MINUS_DI)	Component of ADX and it is used to identify presence of downtrend.
Momentum (MOM)	Measurement of price changes of a financial instrument over a period of time
Plus Directional Indicator (PLUS_DI)	Component of ADX and it is used to identify presence of uptrend.
Log Return	The log return for a period of time is the addition of the log returns of partitions of that period of time. It makes the assumption that return**s** are compounded continuously rather than across sub-periods
Percentage Price Oscillator (PPO)	Computes the difference between two moving averages as a percentage of the bigger moving average
Rate of change (ROC)	Measure of percentage change between the current price with respect to a at closing price n periods ago.
Relative Strength Index (RSI)	Determines the strength of current price in relation to preceding price
Stochastic (STOCH)	Measures momentum by comparing closing of a security with earlier trading range over a specific period of time
Stochastic Relative Strength Index (STOCHRSI)	Used to estimate whether a security is overbought or oversold. It measures RSI over its own high/low range over a specified period.
Ultimate Oscillator (ULTOSC)	Estimates the price momentum of a security asset across different time frames.
Williams' %R (WILLR)	Indicates the position of the last closing price relative to the highest and lowest price over a time

Table A4: Description of Momentum Indicators used in the study.

Load Balancing Mechanism Using Mobile Agents

Sarra Cherbal
University of Ferhat Abbas Setif 1, Department of Computer Science, 19000 Setif, Algeria
E-mail : sarra_cherbal@univ-setif.dz

Keywords: distributed systems, load balancing, mobile agents, agents' technology

Received: July 8, 2020

Distributed systems need to perform load balancing on their hosts, so that the computation runs as quickly as possible. Research in this field has seen the emergence of mobile agents' paradigm as a promising solution. In this present work, we use this paradigm to propose a load balancing approach that benefits from the advantages of agent mobility, more particularly, in the information-gathering phase. In which, we aim to have an overall system vision while reducing network communication overhead, as well as other benefits such as fault tolerance and extensibility for large scale networks. Thus, the purpose of our contribution is to improve the distribution of loads in a balanced way to get loads as close as possible to the system average load. The experimental results show the efficiency of our approach on balancing the loads and in decreasing the response time.

Povzetek: V prispevku je opisana nova agentna metoda za uravnoteženje obremenitev v porazdeljenih sistemih.

1 Introduction

The trend of computer world towards "distributed systems" is no longer envisaging the operation of one single computer, without interacting or cooperating with other computers. These systems must be designed to meet the new requirements [1] [2] [3]. In this work, we present two related research areas: "Mobile agents" and "load balancing".

The technology of Mobile software agent is one of the known technologies in the field of distributed computing. It has emerged as an alternative to the classic "client / server" paradigm that presents the most widely used approach in building distributed applications [4] [5]. Mobile agent technology has interesting prospects for various application areas, among which, we find e-commerce, web-based information retrieval, and the domain of load balancing (LB).

In order to achieve better performances in distributed systems, the load balancing problem has been intensively studied by researchers [6] [7] [8]. Balancing allows maximum use of available resources, and this can be achieved by distributing tasks in a smart manner.

In this work, we study the use of mobile agents in the load-balancing domain. Based on this study, we propose an approach that aims to balance the system load using agent mobility by adopting three agents, and defining tasks for each of these agents. Where, the basic idea is to have local loads close to the system average load.

The rest paper is organized as follows. In section 2, we present an overview of related works and we give a corresponding discussion and the motivation of the proposed approach. Section 3 summaries the benefits of using mobile agents in distributed systems. The details of the proposal are explained in Section 4. In section 5, we present our simulation results to prove the efficiency of the proposed techniques. Finally, Section 6 concludes the paper.

2 Related work

2.1 Load balancing mechanisms

In [9], the authors employ a new concept of transferring virtual loads, which allows nodes to predict the future loads they will receive in the subsequent iterations. Accordingly, the notified node will take into account this predicted load when transferring a part of its actual load to some of its neighbors.

The work of [10] introduces a resource scheduling and a load balancing approach for efficient cloud service provisioning. They increase the use of virtual machines, and achieve load balancing by dynamically selecting a request from a class using Multidimensional Queuing Load Optimization algorithm.

The paper of [11] presents a load balancing methodology for container loading problem in road transportation. They propose a random-key genetic algorithm (BRKGA), with a new fitness function that takes static stability and load balance into account.

Authors of [12] propose a Dynamic Data Replication Algorithms (DDRA) of three phases to improve the information duplication under the cloud storage system. The first two phases are to determine the adequate service nodes to achieve the workload balance based on the nodes' workloads. The third phase presents a scheme of a dynamic duplication deployment proposed to realize a higher access performance and a better load balancing. In the first phase, for realizing the initial LB, the service nodes with probability lower than a defined threshold are filtered.

The authors of [13] propose a new variant of directed diffusion routing protocol in wireless sensor networks. This variant tries to improve the existing protocol by using a load balancing mechanism in order to balance the energy of sensors.

2.2 Mobile agents in load balancing

Authors of [14] propose a dynamic load balancing in a small world P2P network using mobile agents. Firstly, they cluster the peers that have the same set of sharing resources. Then, they balance the query loads in intra-cluster nodes to avoid network congestion. For this purpose, three agents are defined, Host Agent (HA), Detection Agent (DA) and the Reconnection Agent (RA). The DA migrates between the intra-nodes to detect node congestions. This agent is generated and controlled by the HA that walks through all the groups. The RA is generated to reconnect the links between congested nodes and under-loaded nodes of the same cluster. An attractiveness parameter is measured for each node according to some criteria as node degree, processing capacity and the resources contained in the node. The DA migrates to the node with high attractiveness parameter, when the DA arrives, it checks if the node is overloaded or underloaded. However, the selection algorithm chooses a partner node that may refuse to accept the DA agent requests if it is overloaded. In this case, the DA agent migrates to other nodes until finding the right node. Thus, the right partner may not be correctly chosen before migration process.

In [15] they propose two different models of load management in large-scale distributed systems. The first model is based on mobile agents in collecting the status of node actual resources (RAM and CPU). To balance the loads, they integrate a second model based on fuzzy function. The agent system is consisting of three modules as Agent monitoring module (AMM) to collect the available status of current resources. The Agent Decision module (ADM) used to make a decision of migration when the node is overloaded if the measured probability crosses the threshold. The Agent migration module (AMMO) selects the destination node where to migrate, according to some criteria like the high level of processing and computing resources. However, in the agent's model, when defining the node status (overloaded/under-loaded), the authors did not mention how the used threshold is defined or measured. Thus, if it is a constant or a variable value and if it is measured locally or globally according to different node status.

In [16], load balancing is realized using mobile agents that migrate through all nodes based on a credit value factor. This factor is defined at initial stage to decide the node selection for migration. The selection of the destination node to which the agent will be migrated, is based on comparing the local load to a threshold value. If the destination node refuses the load reception, the agent will be directed to the next selected node. However, how to measure the threshold value is not defined. Besides, the agents keep migrating until finding the right partner for each overloaded node.

The paper of [17] presents a load balancing algorithm for parallel virtual architectures. They use a host agent to perform a diagnostic test to evaluate the computation performance and latency of each node. Each parallel task is assigned to a VPU (virtual processing unit) presented by a mobile agent. These VPUs communicate by exchanging messages containing data and tasks to be performed. However, the communication between VPUs of different nodes leads to more traffic.

The paper of [18] distinguishes five populations of agents. The authors use the ant-colony optimization approach to distribute the tasks between the worker agents in a parallel way. Such as, a dispatcher agent (load balancer agent) collects the necessary information for the scheduling decisions, and then the ant colony approach is used to take this decision. In the context of their research, the scheduler is used as an ant that chooses for the current job, the machine having the higher rate of pheromone. To measure this pheromone probability, each worker machine sends via its controller agent, to the dispatcher agent, the number of tasks that it has in its queue.

In [19], the authors combine the least time of first byte algorithm (LFB) and the mobile agent concept. Where, the mobile agent role is monitoring the state of each server resources. However, the authors focus on achieving a reliable approach without taking into consideration the system throughput and latency.

The paper of [20] presents a load balancing scheme in heterogeneous P2P systems. They propose three agents. The first mobile agent is for information gathering and the second one is to decide the receiver partner for the overloaded node. The third is a stationary agent, which resides on each node to update its routing table. Its role is to inform all the network nodes of the other nodes' addresses and spread the node's failure information in the network. This mechanism of using messages to spread information each time to all the nodes lead to more overhead.

A load balancing algorithm for heterogeneous P2P systems is proposed in [21]. In which, one type of mobile agent is used with its essential components such as collection, analysis and location. A utilization rate is measured according to the load and capacity of each node. However, to migrate the overload, the authors propose to choose the neighbor node with the smallest utilization rate. Thus, this one under-loaded node is going to be chosen by the most of overloaded nodes or all of them. Consequently, the chosen node will be overloaded or will refuse the receiving tasks, which leads to rerun location process and try again with other nodes, thus, more time is wasting in location, migration and execution.

2.3 Discussion

In this section of related work, we have reviewed different papers of load balancing approaches existing in the literature. We can notice that this field in general still relevant with the mentioned recent approaches. Sub-section 2.2 concerns propositions of load balancing based on mobile agents. According to this literature study, in our proposed approach, we aim to avoid and improve some existing limits or weak points, mentioned as follows.

In the selection process, it concerns selecting the partner node to which the overload will be migrated, i.e. selecting an under-loaded node for an overloaded node. In this process, in [14], the authors propose to migrate a mobile agent (MA) between a set of nodes until finding an under-loaded node. When the MA arrives to a node, it checks if this node can receive the holding overload, if it rejects the reception, the MA will pass to another from a sorted list [14] [15] [21] and repeat the same verification steps. Thus, this method leads to a wasting time in migration process especially that the MA is holding the overload when migrating, which makes its migration slower. Thus, the system latency is increased and it could find a good partner but not the suitable one. Therefore, in our work, we propose a selection process that aims to find the suitable partner node for each overloaded node and that before launching the load migration to avoid the mentioned disadvantages. In other words, our migration agent goes directly to a defined and a right node destination. In addition, in the selection process, some works like [21] propose to choose the node with the smallest load to be the partner node. consequently, most of overloaded nodes going to choose the same partner, this partner is going to be overloaded and the next receiving agents will be rejected and they should migrate to find another partner. To avoid this, in our approach we propose a method to choose one under-loaded node for each overloaded node without causing its overload.

In general, the node state is defined as overloaded or under loaded according to a threshold value. In some works like [15] [16], there is no mention on how the threshold is defined or measured. Such, if this threshold is measured locally according to the node local information or globally according to the system global information. Therefore, in our approach, we measure the threshold value according to the global overview, based on all the states of nodes. This, in order to have almost equally states for all the nodes, which is the main principle of load balancing mechanism.

Unlike [19], in our approach, we are interested not only by achieving a balanced system but also by avoiding the increase of throughput and by reducing the latency. Thus, we use mobile agents and their migration in a way to avoid exchanging messages between the nodes, unlike [17] [20].

In the next section, we summary the benefits of using mobile agents in distributed systems.

3 Mobile agents and load balancing

Mobile agents' technology provides a load balancing support with three main characteristics:

- They can move from one platform to another
- They can move across platforms of heterogeneous nature (e.g. OS, capacity of CPU, Storage, …)
- They carry the application specific code, instead of requiring a pre-installation of this code on the destination machine.

The two main advantages that mobile agent approach brings to load balancing are:

- Reducing network traffic
- Migrating processes from one site to another.

4 The proposed architecture

In this section, we present our proposed contribution of load balancing based on mobile agent paradigm.

In the following, we assume that the nodes are in a cooperative network, which allows us to study only the characteristics of the considered algorithm, and we assume that the tasks are independent i.e. there is no communication between them.

In this work, we are interested by the "dynamic" load balancing, in which, the movement of tasks depends on the current load of the processors. For the balancing decision, we chose the "source-initiative" approach in which a site called source tries to transfer its surplus (excess) towards a weakly charged site called receiver. For the choice of this latter, we propose a method that tries to obtain local loads close to the average load (AL) of the system, which is the goal of load balancing.

To determine the state of each machine, thresholds must be defined: there are static thresholds that are unchangeable fixed values and other dynamic thresholds that are varied according to the evolution of the system. In our approach, we use the second solution where the threshold is defined according to the average load of the system, which is more suitable to a dynamic environment.

Our contribution consists of three types of agents, a fixed agent and two mobile agents. Each one implements one or more load balancing policies (Figure 1):

- ***Observer Agent (OA):*** a fixed agent located on each site of the system. It evaluates and monitors the local load of the site and so launches the migration process to the partner site (which is indicated by the SMA agent).
- ***Supervisor Mobile Agent (SMA):*** a mobile agent that moves from one site to another in order to build a global vision of the system. It puts the collected information (the local load value + the site identifier) in a table. After finishing the course between the sites, SMA calculates the average load, according to which it builds two tables: a table of overloaded sites (in a descending order) and other table of under loaded sites (in an ascending order). With these two tables, SMA determines the receiving site for each overloaded site (section 4.2.2.).
- ***Transporter Mobile Agent (TMA):*** a mobile agent that migrates between two unbalanced machines in order to balance the system load. It is launched by the OA of an overloaded site, in order to transport the excess load (tasks) of this site to the defined partner by SMA.

The scenario of each of these agents is explained in the following of this section (section 4.2).

4.1 Suggestions and critics

In this sub-section, we aim to explain the benefit of each proposed process in this approach, by showing the different cases that can arise.

If we eliminate the concept of "tables and their ordering" (i.e. SMA agent only calculates the average), three cases arise:

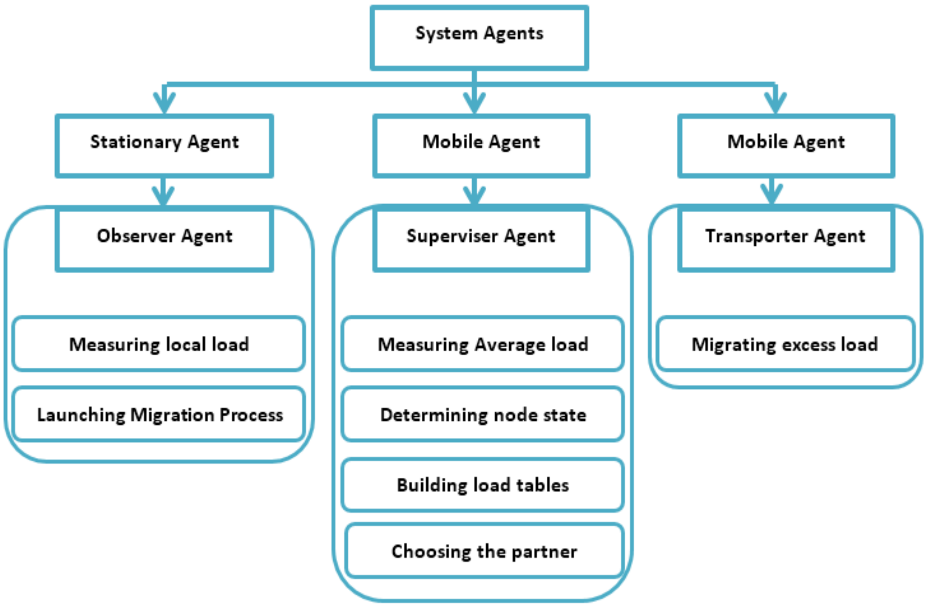


Figure 1: The role of each proposed agent.

4.2 Suggestions and critics

In this sub-section, we aim to explain the benefit of each proposed process in this approach, by showing the different cases that can arise.
If we eliminate the concept of “tables and their ordering” (i.e. SMA agent only calculates the average), three cases arise:

4.2.1 First case

When an observer agent OA_i detects that its site is under-loaded, it sends a broadcast message to all other sites in the system (including underloaded sites) to inform them of its state. In case of an overloaded site Sj, its corresponding OAj sends a message to OA_i to see if it can transmit its excess load. Three cases can be distinguished:

a. OA_i has already accepted the offer of a site S_k, so it will reject the offer of OAj.
b. The excess load of site Sj can cause the overloading of Si, thereby, a refusal message will be sent to the agent OAj,
c. The site S_i accepts the offer of Sj, so an acceptance message will be sent to Sj.

Disadvantages
A very high cost of communication, especially in case of a large network (N sites for example). There will be:

- N-1 broadcast messages for each imbalance ,
- M messages from overloaded sites (M < N),
- M messages of refusal / acceptance.

4.2.2 Second case

TMA agent is responsible for determining the partner node of the overloaded node Sj on which it was created. TMA moves from one node to another while handling the surplus of the node Sj, then, it chooses the first node that has a less load than Sj.
Disadvantages

a. The size of TMA becomes larger when it is handling the excess load, and it has no vision about its destination, thus, it can go through several overloaded sites before arriving to the suitable one, which is not favored in a dynamic load-balancing system. The size of TMA should be as small as possible, in order to speed up the load transfer before that the local loads of the sites change again, otherwise, this transfer becomes unnecessary.
b. This surplus can cause the overloading of the recipient site, while it is possible to find a more suitable recipient. This latter can receive the surplus while still having a load close to the system AL (which is the purpose of load balancing).

4.2.3 Third case

If we eliminate the order of tables (ascending/descending), the agent OAj of an overloaded site Sj will choose the node with the lowest load to ensure effective balancing.
Disadvantage
All OAs that are on overloaded nodes will choose the same node that is weakly loaded, thus, this latter becomes overloaded.

4.3 Description of the proposed agents

4.3.1 Observer Agent (OA)

The aim of our algorithm is to make the system in a balanced state, and thus having balanced machines. It is the role of the OA to observe the machine load, on which it is located:

a. Measuring the node local load

There are two methods to measure the load, one is on demand, and the other is periodic:

- On demand: in this case, the OA measures the node load when the SMA asks him to do it (i.e. when the

SMA arrives on this node). Thus, OA must know the arrival time of SMA, which means that SMA must inform OA with a signaling message. Otherwise, SMA must wait until OA calculates the local load, which increases the load collection time of SMA.

- Periodic: in this case, OA does not need to know the arrival time of SMA since it measures its site local load each period Tinfo. When SMA arrives on this site, the OA provides him the last measured load value. The Tinfo must be an appropriate period to avoid system overload. If the loads change quickly, then the Tinfo must be small. Otherwise, it is necessary to increase the duration of Tinfo.

In our contribution, we chose to use the second method because it is simple to implement, it does not require remote message exchanges between SMA and OA and it reduces the time allocated to the information collection policy.

b. Start the migration process

When the OA of site Sj receives a message from SMA, he gets that his site is overloaded. Therefore, this OA launches the TMA agent, by indicating the surplus to transport, using the average load sent by SMA. In addition, the destination site is declared in the message of SMA.

This load transport between two unbalanced machines makes their loads closer to the system average load, in order to make them better balanced.

Algorithm.1: Observer Agent script

```
For each site $S_k$ {
For each period $T_{info}$ {
Measure local load $L_k$  //$L_k$ is the Local load of site $S_k$
}
Receive_info(SMA, Sr, AL);   //SMA indicates the recipient site Sr.
                              //AL is the average load
If (Received_info != NULL) {
        Launch the TMA agent and send it to the site $S_r$
Else                    //$S_k$ is not overloaded
        OA declares: "My site is lightly loaded"
OA declares: "I am waiting for the reception of transporter agents from other sites"
}
}
```

4.3.2 Supervisor Mobile Agent (SMA)

The SMA moves cyclically between the various sites of the network. For each site, SMA retrieves the site name and its load value and put them in a table (Tab_System). After finishing this movement, SMA makes the two next steps.

a. Calculation of the system average load (AL)

At the arrival of the SMA agent on a site, it contacts the OA, to retrieve the local load. Then, SMA adds this load value to the values already collected. When the SMA finishes its trajectory, it calculates the average load "AL". This AL allows to :

- Achieve a global load balancing, since each machine load is compared to the system overall load.
- Manage load variations that may occur in the system, which is not the case when determining a fixed threshold.

b. Determining the node state and building the two tables

We use the average load (AL) as "threshold": according to which we can distinguish 3 states:

- The site is Overloaded If its load > AL;
- The site is Underloaded If its load < AL;
- Neutral If its load = AL.

From the Tab_System table and the calculated average load, the SMA agent builds two sub-tables:

- Tab_Less : contains underloaded sites.
- Tab_plus : contains overloaded sites.

After that, SMA will order the load values for each of these tables:

- Tab_Less: in ascending order (from the smallest load value to the greatest).
- Tab_Plus: in descending order.

c. Choosing the partner

After these two steps, a global vision is built, and now SMA plays the role of a distributor. It determines for each overloaded site, a partner (a receiver under-loaded site), in a way that the overloaded site and its partner have the same table index, respectively in tab_plus and tab_less.

In a detailed way, the method that we proposed to make the choice of partner is to assign the first entry of Tab_Less to the first entry of Tab_Plus. This, in order to assign the site of the biggest load to the site of the smallest load), and assign the second entry of Tab_Less to the second entry of Tab_Plus, and so on. I.e. each underloaded site Tab_Less [i] is the partner of the overloaded site Tab_Plus [i].

Noting that the mobile agent have to be fast during its move so that the system load information doesn't change during this movement i.e. the agent must be up to date, thus, the size of the SMA code must be minimized.

In our architecture, we propose to use a single supervisor agent but we can always consider multiplying the number of these agents and this depending on the network size (number of participating machines in the application). In this case, it is necessary to provide a cooperation mechanism between the different supervisory agents in order to build a global vision on the system load.

Algorithm.2: script of Supervisor Mobile Agent

```
Som=0 ; AL=0 ;
For m = (from 1 to nbSites) {  // nbSites is the number of machines or nodes
Request the local load from the OA of site Sj
Add(site name Sj + Lj load) as an entry in tab_system
Som=Som+Lj    // Lj: local load provided by agent OA of site Sj
doMove ($S_j$, $S_k$)    // SMA moves to the next node
}
AL = Som / nbSites ;
While Tab_Sys[i] not_empty {
        If (Tab_System [i] > AL)
```

```
                Add this site as an entry to tab_plus
        Else {
                If (Tab_Systeme [i] < AL)
                Add this site as an entry to tab_less
          }
i++;   //next hut in tab_sys
}
Ordering Tab_Less in ascending order;
Ordering Tab_Plus in descending order;
While tab_plus [i] not_empty {  //indicating partners
$S_o$ = Tab_Plus[i]  // source overloaded site
$S_u$ = Tab_Moins[i]  //receiver underloaded site
Send_info ($S_k$, $S_j$ ,AL)  // SMA informs the overload site by the receiver of its surplus load
}
```

4.3.3 Transporter Mobile Agent (TMA)

It is a mobile agent launched by OA agent of an overloaded site and it represents the surplus of this latter. The TMA purpose is to transport this surplus to the recipient site, on which the task(s) will be executed.
It is the OA agent that indicates to the TMA the work to handle. This work presents a set of tasks. The number of TMA agents depends on how many tasks they can handle.

Algorithm.3: script of Transporter Mobile Agent

```
doMove($S_k$,$S_j$) ;
// TMA moves from the overloaded site to the receiver site
Send (L_Plus$_k$, $S_k$, $S_j$) ;            // $S_k$ : the source site name
                                    // L_Plus$_k$ : excess load of site $S_k$
      // Sj: the recipient site name (indicated in SMA script)
```

An overview of our system architecture with a scenario of the three agents on 4 sites, is presented in Figure 2.

5 Experimental results

5.1 State of loads

To test the behavior and to evaluate the performances of the proposed approach, our agents' algorithms are developed using Java eclipse IDE on top of the JADE agent platform. We use the "FSMBehaviour" for OA and SMA agents. This behavior serves to present complex tasks, it responds to the needs of the compound behaviors of these two agents. For the third TMA agent that has only one task to perform, a simple behavior meets the need, so, in this case it is the "OneShotBehaviour". The source code is composed of three classes and of their sub-classes according to the roles in Figure 1.

In order to launch our algorithm execution under Jade, we first launch the observer agents (we implement eight agents OA0,OA1,…), each on a container. Then, we launch SMA agent on the main container. Next, the AMT agent is launched by the corresponding OA agent (that of an overloaded site).
A load ratio is assigned randomly to each container. Figure 3 presents the loads' state in the three phases: before applying LB, and when applying the proposed LB with and without table approach. Where, in the third phase (without table approach), the TMA transports the overload to the first found under-loaded site. Figure 3 shows that in the first phase, the loads are unbalanced (from very high to very low loads). In the third phase, the loads start to be balanced which shows the impact of the LB approach. Thereafter, in the second phase, the loads are more balanced compared to the system overall state, which demonstrates the efficiency of the proposed table approach in choosing the right partner to receive the overload.

5.2 The impact of LB on response time

In distributed computing, the response time is a significant issue and the fact of reducing it is a very important requirement in improvement approaches. However, when the loads are bigger, the system response time is higher. In this type of environment, a set of tasks is distributed between some nodes of the system. The system response time is the time taken by all the participating nodes to realize this set of tasks.
We measure the response time using the following formula:

$$T = (NbTasks \times HighL \times ReqAT)/100 \qquad (1)$$

Such that:

- NbTask is the total number of tasks,
- HighL is the higher load percentage in this system,
- ReqAT is the required average time to execute one task.

Figure 4 shows the change of response time in the three phases, while increasing the number of tasks to be executed by this system (between 200, 300, 400 & 500). The results prove the efficiency of our proposed LB algorithm in reducing the response time comparing to the two other phases.

5.3 Impact of our proposed selection technique

The main purpose of applying a LB process in a distributed system is that all the nodes work with almost equal workloads. In other words, minimize the difference in workload percentage between nodes and so avoid having a node with a high workload while another node is under-loaded. In a distributed system, where the nodes work in collaboration to complete a specific set of tasks, balancing the workload and thus the effort between the nodes is essential to reduce the execution time (or response time) of a task and thus of all the launched tasks. Therefore, in a system, we measure the difference of workload rates between the most heavily loaded nodes and the least loaded ones. Thus, the difference between the highest and the lowest workload rate.

In the selection technique, an overloaded node (site) selects one of the under-loaded nodes where to transmit the overload, what we call a partner. In our approach, we

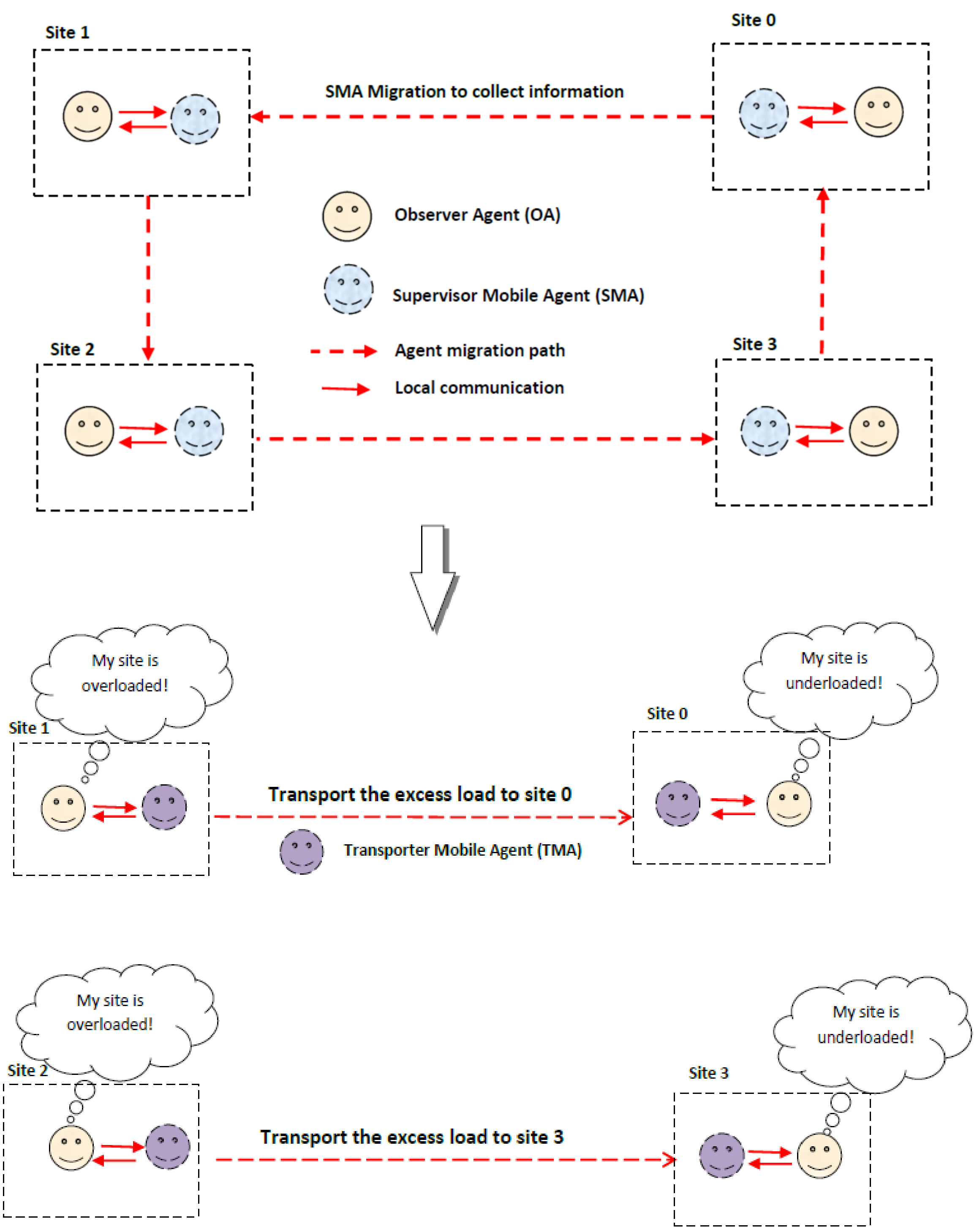


Figure 2: Architecture of the proposed system.

propose a selection technique based on a defined table as explained earlier in this paper.

To prove the efficiency of our proposal, in this sub section, we compare it to the selection technique proposed in [14] and in [21] in terms of workload differences and response time.

The selection technique of [14] is based on sorting the under-loaded nodes in a list, according to an attractiveness parameter and in [21] the list is sorted according to the utilization rate. Then, each overloaded node uses this list to find the least charged node to transmit him the overload. When the migration agent arrives to the selected node, it checks if this node is overloaded so it accepts the received overload or reject it if it is under-loaded. In case of rejection, the migration agent moves to the next node from the sorted list.

Figure 5 presents the difference of workload rates between the most heavily loaded site (node) and least loaded site in each simulation (case). This parameter is compared in three phases, the first phase is before applying LB, the second is when applying our proposed approach and the third is when applying LB with the

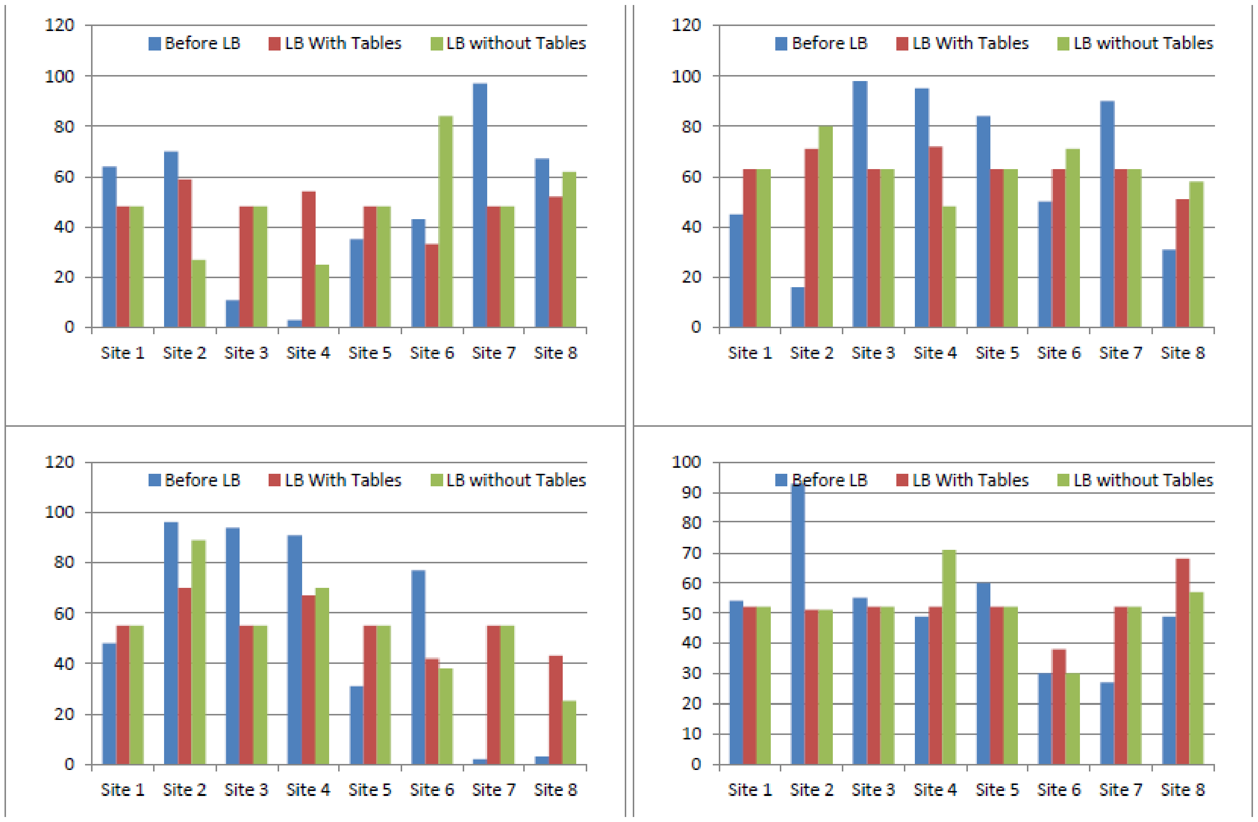


Figure 3: State of loads in the three phases.

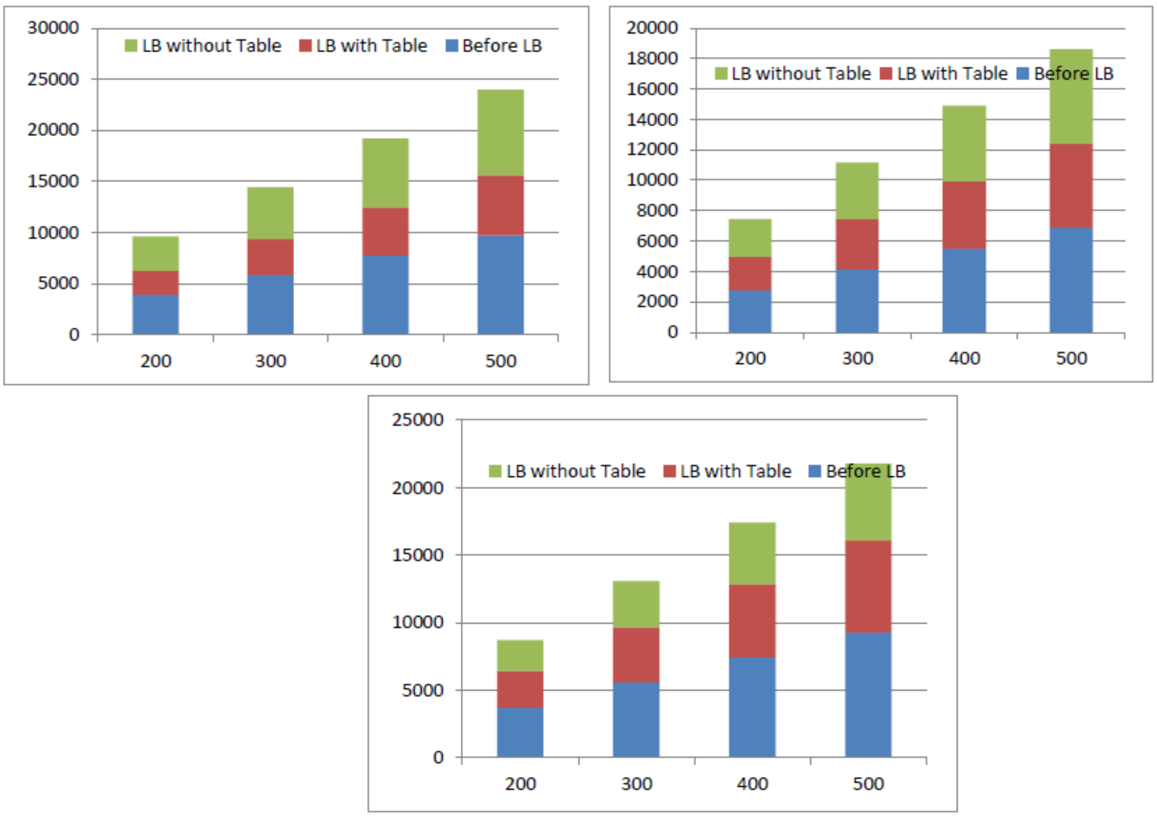


Figure 4: Response time as function of number of tasks.

selection techniques of [14][21]. A workload rate is assigned to each node in a random manner, and we have launched eight simulations (8 cases). The results in Figure 5 show that the differences in workload rates are more reduced after applying a LB approach and then are much reduced when applying our proposed approach. These results prove the efficiency of our proposed selection technique compared to existing techniques in selecting the most suitable under-loaded node for each overloaded node and thus all the nodes work with almost equal workload rates.

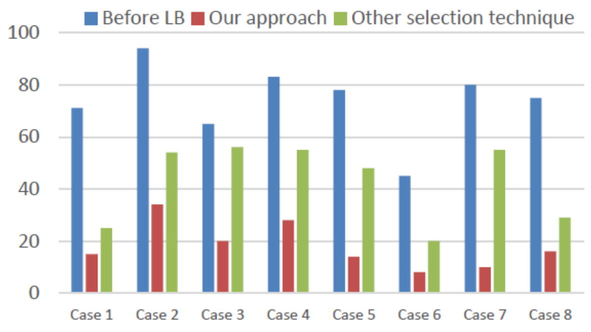


Figure 5: Differences in workload rates between most heavily loaded sites and least loaded sites

6 Conclusion

Load balancing is one of the keystones for the improvement of system performances. Its main objectives are to improve the execution time of tasks and to take advantage of the maximum system resources.

In this work, we are interested in the use of mobile agent technology in the field of load balancing. One of the important motivations of this paradigm is the minimization of remote communications thus saving the bandwidth consumption, which is favorable for an efficient load balancing system. For this reason, we have integrated in our proposed solution a mobile agent whose role is to collect the loads information to build a global vision on the system load, which reduces the communication cost compared to the classic collection method. In addition, the stationary agents must be aware

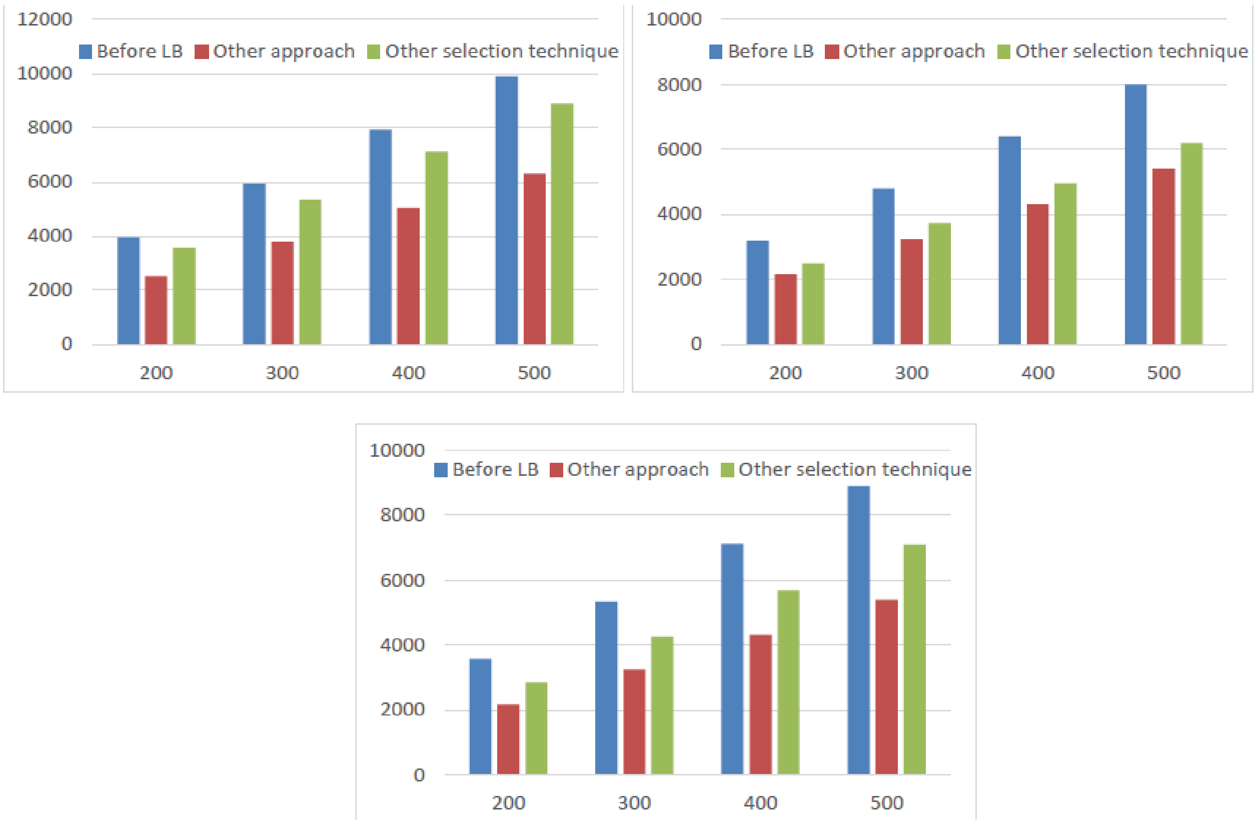


Figure 6: Impact of selection technique in response time as function of number of tasks.

Figure 6 shows the impact of the used selection technique on the Response time while changing the number of tasks from 200, 300, 400 to 500. In sub-section 5.2, we explain the signification of response time and we define formula (1) to measure it. The results presented in Figure 6 correspond to three launched simulations with different random workloads. Figure 6 shows that the response time increases with the increase of number of tasks with more workloads to perform. Besides, our proposed approach presents the most reduced response time in three launched simulations. This proves the efficiency of our selection technique in choosing the partner node and thus all the nodes participate with almost same effort to finish a task, and then the set of tasks achievement is done faster.

of the global load so they can detect the machine balance state and so select the tasks to be migrated. In our solution, the mobile supervisory agent only informs overloaded hosts which conducts in a traffic reduction compared to those approaches that inform all the hosts of the system.

Other benefits of mobile agents are robustness and fault tolerance, which are necessary in a load balancing system so that it can continue to operate when one of the members is disconnected. Mobile agents also offer the advantage of scalability, they adapt well to small networks as to large-scale networks. These benefits conduct us to use mobile agents in our solution, to have an extensible and a robust load balancing system.

In a large-scale network, increasing the number of supervisory agents is possible to reduce the agent migration time and to avoid increasing the agent code size. This allows an improvement in load balancing. However, we find that determining the number of agents needs another study. Furthermore, we have implemented our proposal on Jade platform. By a comparative evaluation, the results showed the efficiency of the load balancing approach. Besides, we have shown its impact on reducing the execution time latency, which is an important factor in distributed systems. In addition, we have shown the impact of our proposed selection techniques compared to existing selection techniques in balancing the workloads and in reducing the system response time.

As a perspective, we aim to implement the proposal architecture on mobile nodes to show the effect on energy consumption.

References

[1] Van Steen M, Tanenbaum AS (2016) A brief introduction to distributed systems. Computing. vol. 98, no. 10, pp. 967-1009. https://doi.org/10.1007/s00607-016-0508-7

[2] Anjomshoa MF, Salleh M, Kermani MP (2015) A Taxonomy and Survey of Distributed Computing Systems. Journal of Applied Sciences, Vol. 15, pp. 46-57. DOI: 10.3923/jas.2015.46.57

[3] Samolej S. and Rak T. (2009) Simulation and Performance Analysis of Distributed Internet Systems Using TCPNs, Informatica, vol. 33, pp. 405-415.

[4] Hakansson A, Hartung R (2013) Book: Agent and Multi-Agent Systems in Distributed Systems - Digital Economy and E-Commerce. Publisher: Springer-Verlag Berlin Heidelberg. DOI: 10.1007/978-3-642-35208-9

[5] Hajduk M, Sukop M, Haun M (2019) Cognitive Multi-agent Systems : Structures, Strategies and Applications to Mobile Robotics and Robosoccer. In Series of Studies in Systems, Decision and Control. DOI: 10.1007/978-3-319-93687-1

[6] Wided A. and Okba K. (2019) A novel Agent Based Load Balancing Model for maximizing resource utilization in Grid Computing, Informatica, vol. 43, no. 3. https://doi.org/10.31449/inf.v43i3.2944

[7] M. A. Salehi, H. Deldari, and B. M. Dorri (2009) Balancing load in a computational grid applying adaptive, intelligent colonies of ants, Informatica, vol. 33, no. 2, pp. 159–168.

[8] Aghdashi A, Mirtaheri SL (2019) A Survey on Load Balancing in Cloud Systems for Big Data Applications. In: Grandinetti L., Mirtaheri S., Shahbazian R. (eds) High-Performance Computing and Big Data Analysis. TopHPC 2019. Communications in Computer and Information Science, vol 891. Springer, Cham. DOI: https://doi.org/10.1007/978-3-030-33495-6_13

[9] Couturier R, Giersch A, Hakem M (2018) Best effort strategy and virtual load for asynchronous iterative load balancing. Journal of Computational Science. Vol. 26, Pages 118-127. DOI: https://doi.org/10.1016/j.jocs.2018.04.002

[10] Priya V, Kumar CS, Kannan R (2019) Resource scheduling algorithm with load balancing for cloud service provisioning. Appl Soft Comput Vol. 76, pp. 416–424. https://doi.org/10.1016/j.asoc.2018.12.021

[11] Ramos AG, Silva E, Oliveira JF (2018) A new load balance methodology for container loading problem in road transportation. European Journal of Operational Research, Vol. 266(3), pp. 1140–1152. https://doi.org/10.1016/j.ejor.2017.10.050

[12] Hsieh H, Chiang M (2019) The Incremental Load Balance Cloud Algorithm by Using Dynamic Data Deployment. J Grid Computing Vol. 17, pp. 553–575. https://doi.org/10.1007/s10723-019-09474-2

[13] Semchedine F, Bouallouche-Medjkoune L, Tamert M, Mahfoud F, Aïssani D (2015) Load balancing mechanism for data-centric routing in wireless sensor networks. J. Comput. Electrical Eng. Vol. 41, pp. 395–406. https://doi.org/10.1016/j.compeleceng.2014.03.005

[14] Shen X-J et al (2014) Achieving dynamic load balancing through mobile agents in small world P2P networks, Comput. Netw., vol. 75, pp. 134-148, Dec. https://doi.org/10.1016/j.comnet.2014.05.003

[15] Ali M, Bagchi S (2019) Design and analysis of distributed load management: Mobile agent. based probabilistic model and fuzzy integrated model. Appl Intell, Vol. 49, pp. 3464-3489. https://doi.org/10.1007/s10489-019-01454-z

[16] Metawei MA, Ghoneim SA, Haggag SM, Nassar SM (2012) Load balancing in distributed multi-agent computing systems, Ain Shams Engineering Journal, Vol. 3, pp. 237-249, May 2012. https://doi.org/10.1016/j.asej.2012.03.001

[17] Youssfi M, Bouattane O, Bensalah, M (2015) Efficient Load Balancing Algorithm for Distributed Systems Using Mobile Agents , Advanced Studies in Theoretical Physics Vol. 9, no. 5, pp.245 – 253. DOI:10.12988/ASTP.2015.5110

[18] Younes H, Bouattane O, Youssfi M, Illoussamen E (2017) New load balancing framework based on mobile AGENT and ant-colony optimization technique. Intelligent Systems and Computer Vision (ISCV), Fez, 2017, pp. 1-6. DOI: 10.1109/ISACV.2017.8054961

[19] Afriansyah MF, Somantri M, Riyadi MA (2017) Model of load balancing using reliable algorithm with multi-agent system. IOP Conference Series: Materials Science and Engineering, Volume 190, conference 1. doi: 10.1088/1757-899X/190/1/012033

[20] Nehra N, Patel RB, Bhat VK (2008) Load Balancing in Heterogeneous P2P Systems using Mobile Agents. International Journal of Electrical and Computer Engineering Vol:2, No:8, pp. 2740-2745.

[21] Li H (2011) Load Balancing Algorithm for Heterogeneous P2P Systems Based on Mobile Agent. in IJCA Proceedings on International Conference on Electronics, Information and Communication Engineering (ICEICE) (Foundation of Computer Science, New York, 2011), pp. 1446–1449. DOI: 10.1109/ICEICE.2011.5777758

https://doi.org/10.31449/inf.v45i2.3465

Value-Based Retweet Prediction on Twitter

Surbhi Kakar, Deepali Dhaka and Monica Mehrotra
Computer Science, Jamia Millia Islamia University
E-mails: kakar.surbhi3@gmail.com, deepali.dhaka@gmail.com, mmehrotra@jmi.ac.in

Keywords: retweet prediction, twitter, value system, emotions, sentiments, topic- specific emotion, retweeting, social networks, retweet behavior, topic

Received: March 7, 2021

Retweeting is an online activity done on the twitter social network. This activity leads to sharing of opinions and ideas from one person to another. Predicting retweet decision has been an interesting and challenging task since the past decade. Past studies have shown that emotions, sentiments and topic specific emotions can influence the retweet decision of the user. However, value systems of an individual can also be an important and crucial aspect in predicting the decision of user. Hence, through our work, we propose to study retweet prediction as a function of value systems. Our work also presents an experimental comparative study with the features used in previous studies. The experimental results using the different machine learning algorithms shows that value-systems have a higher performance in predicting retweet decision of the user as compared to emotions, sentiments and topic-specific emotions.

Povzetek: Z metodami strojnega učenja je analiziran problem uporabnikovega odgovora na Twitterju.

1 Introduction

Social networks are a platform where people meet each other virtually. Such platforms allow people to share their ideas, thoughts, opinions with each other freely and leads to diffusion of information within the network [47]. As the content shared by the users is an expression of their feelings, sentiments and values, this content can be used to predict the user behavior [4].

Twitter is a social network famous for micro blogging where users express their interests by using the Tweet button. These tweets can further be shared by anyone who feels or experiences a connection with the author of the tweet[32], thus initiating the retweeting mechanism.

Users on Twitter can have a follower-followee relationship between them. If a user A is inspired by another user B or finds their interests similar to them, A can then opt to follow that user. In such a case, A is said to be a follower of B. The user B may or may not follow the former user, in the scenario where B does not follow A, he/she is said to be a followee of A.

A retweet is a tweet that is re-shared by a user. A tweet prefixed with a symbol RT represents a retweet.
Research around retweeting mainly addresses three research problems:

1. *Whether a tweet will be retweeted by a user*

   This problem can be redefined as: Given a tweet, whether a user will retweet the tweet or not.
   Studies done in this area focuses on exploring features that impact user's retweet behavior followed by building retweet prediction models for the same. [33][51][1][47].

2. *Finding users who will retweet a tweet*

   This research problem focuses on finding which users will retweet a given target tweet. [26][23].

3. *Factors that affect the retweet frequency of the tweet*

   A lot of research has been done around finding why a specific tweet is retweeted more in comparison to other tweets. [35][42][5][32][17].

This work will be focusing on addressing the first research problem calling it as the retweet prediction problem.

Retweet prediction can be defined as a problem of predicting the retweet decisions of a user. This has been a very challenging problem as tweets innately are noisy and complex. It is through the retweet mechanism that the diffusion of information takes place. Understanding retweet behaviors and the virality of tweets on social media can help us identify influential people who can spread the information at a faster pace. This insight is useful in applications such as viral marketing and emergency response. Predicting which tweets will be retweeted by a user can also help in providing recommendations to a user to create a personalized experience for them.

The most popular approach for retweet prediction starts with building a user profile[8][10][24][25][28]. The profile of the user can be extracted from their tweet/retweet data. Several factors like URL's, hashtags can be used directly from the timeline of the user to build their profile. However, certain information is latent/hidden in the content shared by the user. At such places, topic extraction can be very useful.

The topic of the tweet has been found to be a promising factor in capturing interests of the user [8][10][28][15][46]. In addition to these factors, emotions and sentiments can also be employed for this task.

Emotion represents the mental state of a human being whereas sentiments can be viewed as an opinion towards a person or an object. The content written by a user is an expression of how he/she feels, making it a good representative of their emotions and sentiments. Several theories have been proposed to classify human emotions but for our work, we use the well-accepted theory by [37]. According to this theory, emotions can be classified into eight basic types: Anger, Joy, Surprise, Anticipation, Sadness, Disgust, Fear and Trust. Sentiments, on the other hand can be categorized into either positive or negative. For our work, we use the NRC word-emotion lexicon[31] to label emotions and sentiments in the content of the tweet as it is a well-accepted lexicon for labeling emotions and sentiments. The impact of emotions and sentiments on retweet prediction have been studied in various research[17][21][32][36][11].

[11] in their work compared the conjunctive effect of using emotion and topic with topic-specific emotion model in predicting the retweet decision of the user. They proposed that not just the topic, but the emotions and sentiments expressed by a user on a topic also correlates with user's decision. They, in their future work, proposed to study value systems for the purpose of retweet prediction task. This formed the inspiration of our work around value systems.

A value system of an individual denotes the beliefs a user carries in their life. This can be learnt from their environment including places like family and school. As per [41][7], value systems can fall into the following classes:

- *Self-Transcendence*: This type of value system represents values of benevolence and universalism. The core beliefs in this category are those of wisdom, peace, spirituality, and welfare of general public.
- *Self-Enhancement*: This is the category where people are more interested in their own enhancement and growth. They are also inclined towards power and authority.
- *Conservation*: People who carry this type of value system are more traditional and believe in cultural values and religion. They tend to conform to the rules of the society and are concerned about their family security as well as national security.
- *Openness to Change*: This value system represents people who are adventurous and daring, someone who is independent and self-directed.
- *Hedonism*: People carrying this type of value system tend to be involved in pleasure seeking activities.

Value systems have been shown as an important factor in influencing user decisions as per past studies[2][48][39][22][29] ranging from shaping leadership styles to influencing voting preferences of a user.

Hence, our work proposes to explore the impact of using value systems on retweet decisions of a user.

Overall, the contributions of this paper can be stated as below:

- Proposing a novel value-based model which uses value system related features to predict retweet behavior.
- Proposing feature extraction methodology for value related features.
- Comparing emotion, sentiment, topic-specific emotion and value-based models.
- Experimental results demonstrate the higher performance of value-based models as compared to emotion, sentiment and topic-specific emotion models used.

The remaining paper is structured as follows. Section 2 outlines the previous studies performed in this field. Section 3 summarizes the statistics and the approach of the data collection. Section 4 presents the methodology used and the process of feature generation. Section 5 discusses the experiments performed followed by Section 6 and 7, discussing the results and summarizing the conclusion of our work respectively.

2 Related works

This section reviews the past work done in the context of retweet prediction discussing various features like emotions, sentiments and value systems which are potential predictors for modeling retweet decisions of a user. Retweet Prediction can be approached as either a classification problem or a recommendation problem. Our work would be using classification to approach the problem of retweet prediction.

The earlier research in retweet prediction were mainly studying the factors affecting the retweet mechanism. [4] studied the reasons and the conventional styles of retweeting. [42] proved the impact of using URL's, hashtags, number of followers and followees on the retweet frequency of the tweet. It was shown by [17][36] that emotions and sentiments also affect the virality of the tweet. The topic of interest was seen as a potential factor by [28].

Recent studies are focused around predicting retweet behavior. [47] used a factor graph model and concluded that time of the tweet, user information and the content of the tweet can be effective predictors in predicting retweet behavior of the user. [33] used the temporal information to study the retweeting activity. They used conditional random fields for their work. Other research also exploited the temporal information of the tweet [14][51].

The topical information of the tweet was also studied to gauge the influence of topic on the retweet decision of the user. [50] used a factor graph model considering user attributes, topic information and instantaneity to study the retweet behavior. [10] captured short term interests of the user by ranking top three topics as the hot topics that the user is interested in, at that point in time. [8] used a collaborative-based recommendation algorithm considering topic as a feature to capture user interests. [11] studied the impact of emotions specific to a topic, emphasizing that the same person can have different emotions for different topics and showed that topic-specific emotion feature correlates with the retweeting behavior of the user.

Other researchers also showed the importance of topic as a factor in retweet mechanism[15][46][28][27]. Author information has also been shown to have influence on retweet behavior of the user.

Recent authors view the retweet prediction as a recommendation problem and use matrix factorization techniques for the same [45][44][18][19][51].

Sentiments and Emotions also have a big impact on user posting/re-posting behavior. Emotions represent the state of mind of an individual at a specific point in time. As per [37] it can be categorized into eight basic types of anger, joy, sadness, disgust, trust, anticipation, surprise and fear. Sentiments can be viewed as positive, negative opinions of people over an event/person/object. It has been proved by several studies that sentiments and emotions have an impact on predicting the decisions of the user [17][32][36]. [36] in their work, demonstrated that the intensity of emotions expressed in a tweet is directly proportional to its retweet frequency. [21] used emotions and user related features to predict the retweet behavior. They concluded that tweets reflecting sadness and anger are the most dominating emotions to be retweeted.

Several tools and techniques have been proposed to detect emotions from the content of the tweet.

These include tools based on parsers, tree taggers and lexicon-based techniques to label emotions in text [40][21][34][38]. For our work, we employ the use of lexicon-based methods to label emotions, sentiments and value systems in the content of the tweet.

An individual learns their personal values from their environment since childhood. The value system of an individual can be viewed as the core beliefs held by them towards someone or something.

[41] classified value systems into five types of self-transcendence, conservation, openness to change, hedonism, self-enhancement.

Several works show that the content shared reflects the value system of the user [7][12][43][16][9].

[43] used the text of the speech to infer values of the user. Another work used human annotations and machine-learning to identify values in text [16]. Some authors built a word map for the people reflecting traits of being conservative and liberal [9]. [7] also confirmed that the content of the tweet can have potential influence for labeling the value system of the user by analyzing the words related to a specific value system category. Several research show that value systems can help shape personal decisions of people. [2] showed that personal values of an individual can shape their style of leading teams. Another work studied value systems and concluded that they can impact the travel decisions of young adults [48]. Several other works confirm that value systems have a potential influence on voting decisions, foreign policy orientations and health decisions of an individual [22][39][29].

Hence, our work attempts to use value systems to explore the impact on retweet decisions of the user comparing them with previous state of art models used viz, emotions, sentiments, topic-specific emotions. For our work, we will be using the valueDict lexicon [20] for labeling value systems in the content written by users as it is one of the first lexicons to be proposed for the purpose of labeling value systems. This lexicon contains words associated with each of the value system categories. The lexicon was created by taking a set of seed users. For these users, their value system and the associated words for each value system category were inferred by investigating the content written by them in their descriptions and the tweets. The strength of the lexicon was increased by generating synonyms using word2vec embeddings. A validation of the lexicon was applied on an additional set of users taking their descriptions as the ground truth label for their value system. Table 1 summarizes the findings of past studies around sentiments, emotions and topic-specific emotions in addition to other user and tweet features used.

S. No	Feature used	Author	Finding
1.	Sentiments	[32][17]	Showed the importance of sentiments in predicting retweets. Tweets with negative sentiment are more likely to be retweeted.
2.	Emotions	[21]	Demonstrated the impact of emotions on retweet behaviors. Concluded that tweets with anger and sadness correlate highly with retweet probability.
3.	Topic specific emotions	[11]	Concluded that topic specific emotions also have an impact on retweet behavior in addition to sentiments and emotions.

Table 1: Summary of Related Works.

3 Data collection

The data for this research was collected through the Twint API [49][3]. For this study, we selected a set of 126 seed users manually based upon the average activity of the user per day. These users had an activity of posting a tweet/retweet on an average 4-5 times per day. For these users, their latest 700 tweet/retweet data were collected which was used in constructing the user-interest profile.

As a next step, a list of 60 followees each was fetched for these users from the Twint API.

To create the target dataset, we needed to create positive and negative samples for each user. We considered all the retweets of the user within a given time as the positive samples for the target dataset.

To create the negative samples, we fetched the list of retweet authors for each of the seed users.

A retweet author is the author whose tweets, a seed user has retweeted. If these retweet authors were also a followee of the seed user, we collected latest 700 tweets of the author within the same timeline of user's retweets.

Both datasets were merged together to form a list of interesting and non-interesting tweets for each user. This process was repeated for all the seed users. The final dataset amounted to 17,180 users with 2,15,312 tweet/retweet data. Table 2 presents a summary of the data statistics used.

Number of tweets	2,15,312
Number of users	17180
Number of positive samples	179525
Number of negative samples	35787

Table 2: Data Statistics.

4 Methodology

To prepare the data for retweet prediction problem, for each of the seed user, we prepared a list of interesting and non-interesting tweets. The interesting tweets were the tweets that the user found interesting and retweeted. Whereas, to collect the non-interesting tweets, we collected tweets of their followees where the seed user did not retweet. This enabled us to create positive and negative samples for each of the user. The methodology resulted in a total of 2,15,312 tweet/retweet data with 17,180 users.

4.1 Feature generation

4.1.1 Value systems

Value system of an individual is a potential predictor of the decisions, they are likely to take in their life. Past studies have shown the significance of content-based analysis of value systems. Hence, our work uses content of the tweet to determine the value system of the user. We use the valueDict lexicon for labeling the users with their value system [20]. The authors in this study proposed this lexicon which contains words relative to each of the value system categories. They then used it to study the prominent value systems in developing and developed regions of the world.

The following strategy is used to calculate the value system for a user in our study: The value system of a user can be represented by w dimensions, (where, w=5) namely: self-transcendence, self-enhancement, conservation, hedonism, openness to change

$$V = \{V_1, V_2, \ldots V_w\}$$

Suppose, $U = \{U_1, U_2, ..U_n\}$ be the set of users

And let $Tw_i = \{Tw_{i1}, Tw_{i2} \ldots Tw_{iz}\}$ be the set of tweets for i^{th} user, where

$$1 \le i \le n$$
$$1 \le j \le z_i$$

Then let n_{ijk} be the number of hits found in the lexicon, for each v_k where $v_k \in$ V, for a tweet Tw_{ij},

The score of v_k for a user i, can be then calculated as:

$$S_{ik} = \sum_{j=1}^{z_i} n_{ijk}$$

Hence, the total score of the value system for a user i, can be represented as:

$$S = \max \{S_{ik}\}$$

The user is labeled with the value system which has the score S. The value system of a tweet is calculated similarly.

4.1.2 Value similarity score

This feature intends to capture how similar a target tweet is, to user's past interests.

Suppose a target tweet for a user u, has a value system V_j, then the similarity score for this tweet can be calculated as:

$$Similarity_{score} = Total_j / X$$

Where $Total_j$ is the total number of tweets/retweets in the u's profile reflecting value system V_j, and X is the total number of tweets/retweets posted by u.

4.1.3 Emotions and sentiments

Human emotions can be represented by eight basic emotions namely, anger, disgust, sadness, trust, joy, surprise, fear and anticipation [37]. Our work uses this theory of emotions proposed by the author.

Sentiments can be viewed as the opinions of people on certain objects/events. It can be classified into positive and negative sentiments.

For our work, we use the NRC word-emotion lexicon [31], to determine the emotion and sentiment score of the tweet. Our methodology of extracting emotions and sentiments is inspired by [11].

For simplicity, we treat emotions and sentiments together and calculate a single score for them, therefore, we may sometimes refer to this combined score as the emotional score of the tweet. Let emotions and sentiments be represented by a 10-dimensional vector for a tweet Tw_i for a user i:

$$E = \{ES_{i1,} ES_{i2,\ldots\ldots} ES_{i10}\}$$

Let n_{ki} be the number of hits found in the NRC lexicon for emotion dimension, ES_k,

where $ES_k \in E$, then the emotion/sentiment score for ES_k can be determined by:

$$S_{ik} = n_{ki}$$

To give more weight to the emotions that are dominating, we calculate the fraction of the matching words found in the lexicon for a tweet Tw_i, multiplying with the number of matching words found in the lexicon.

The resultant emotional vector for tweet, Tw_i, is of the form:

$$\{S_{ik1}, S_{ik2} \ldots S_{ik10}\}$$

To further simplify, we convert the scores in this vector to binary scores based on a threshold as past studies have shown that a tweet may reflect more than one emotion. We consider the threshold as the mean of the emotional scores in the vector. If a score is greater than the threshold, we mark it as a 1 else a 0. However, for the sentiments, we consider the bigger of the two sentiment value based on if the tweet reflects a higher Positive value or a higher negative value.

4.1.4 Topic-specific emotion

This feature, given the topic of a tweet reflecting certain emotional states, captures its similarity with the user's emotional states on this topic.

To create this feature, we first extracted topic out of a tweet using LDA GIBBS sampling method[13]. We then used conditional probabilities to extract topic specific emotions for the target tweet using the method suggested by [11].

A target tweet, for a user, in such a case can be represented by a vector containing conditional probabilities, $\{P(ES_1|T_i), P(ES_2|T_i) \ldots. P(ES_{10}|T_i)\}$, for all emotion dimensions given a specific topic the user is interested in.

These conditional probabilities can be defined as:

$$P(ES_j|T_i) = P(ES_j, T_i)/P(T_i)$$

Where $P(ES_j, T_i)$ is the probability of emotion dimension ES_j and topic T_i occurring together in user's profile and,

$P(T_i)$ is the probability of user's tweets/retweets reflecting topic T_i.

Mathematically, it can be written as:

$$P(ES_j, T_i) = Total_{ij}/X$$

$$P(T_i) = Total_i/X$$

where, $Total_{ij}$ are the total number of tweets/retweets where emotion ES_j and Topic T_i co-occur in user's profile, X is the total number of tweets/retweets posted by the user and $Total_i$ is the total number of tweets/retweets of the user on topic T_i.

4.1.5 Conventional features

URL's and Hashtags

URL's and Hashtags have been an important factor in determining the retweet decision of the user [46][42][1]. For our work, we checked if the URLs and hashtags in the target tweet is similar to their user profile. If so, we create a score of 1 else a 0. The URLs and hashtags interest were taken from the user interest profile.

User Interest Vector

Text Similarity is a well-known algorithm for the task of retweet prediction [46][42][26][1]. To compute text similarity between the target tweet and the past tweets/retweets of the user, we create user interest vector and interest vector for the target tweet by using word2vec algorithm [30]. Cosine Similarity is used to further calculate the text similarity between the two vectors.

5 Experiment

We performed separate experiments to evaluate value-based, emotion/sentiment based and topic-specific emotion-based models for the task of retweet prediction. Conventional features were used in conjunction with these models.

To perform the experiment, the target tweet/retweet dataset was divided into training and test set with a test ratio of 0.3. Each model was trained on the train set and evaluated on the test set. All the models were run using four different classifiers: Random Forest, Logistic Regression, XGB and GBT. A 10-fold cross-validation was performed to get optimal parameter values for the models in order to avoid overfitting.

These experiments were implemented using python 3 with a PyCharm editor on a machine with a processor of 2.2 GHz 6-core Intel Core i7 and memory of 16 GB.

5.1 Data checks and preprocessing

To prepare the modeling data, several data checks and preprocessing techniques were applied including skewness checks, handling null values and encoding the categorical features. As the target label was highly imbalanced, we used SMOTE sampling to balance out the imbalance between the class labels [6].

5.2 Modeling

For our work, we built the following models to compare the value-based model with previously used models for retweet behavior prediction, namely, emotion/sentiment-based model, topic-specific emotion-based model. We also compared our work with one of the baseline models proposed in previous studies [15][11]. This baseline model is called as the user-interest model.

5.2.1 Value-based Model (VM)

This model explores the impact of using user's value systems on their retweet decisions. The model uses features based on the value systems viz target value system and the similarity score between the target value system and value system in the user interest profile.

5.2.2 Emotion-based Model (EM)

This model intends to capture the effect of user's emotions and sentiments on their retweet behavior. The model uses the 10-dimensional emotion and sentiment score extracted by the process described in the Feature generation section.

5.2.3 Topic-Specific Emotion Model (TSM)

The topic-specific model was built to investigate the effect of topic specific emotions on user's retweeting decision, as different users can express different emotions for a specific topic. It uses the 10-dimensional conditional probabilities score to predict the retweet decision of the user. The probabilities are calculated using conditional probability of an emotion dimension, given a specific topic. This tells us how likely the user is to express an emotion given a specific topic.

5.2.4 User Interest Model (UIM)

This model is used as a baseline model and intends to explore the text similarity between the user interest vector and the target tweet. The vectors are created using the word2vec algorithm. Cosine Similarity is used to infer the similarity between user interest vector and target tweet vector.

To calculate the accuracy of our retweet predictions models, we used the accuracy metric which can be defined as the ratio of number of correctly classified instances to the total number of instances.

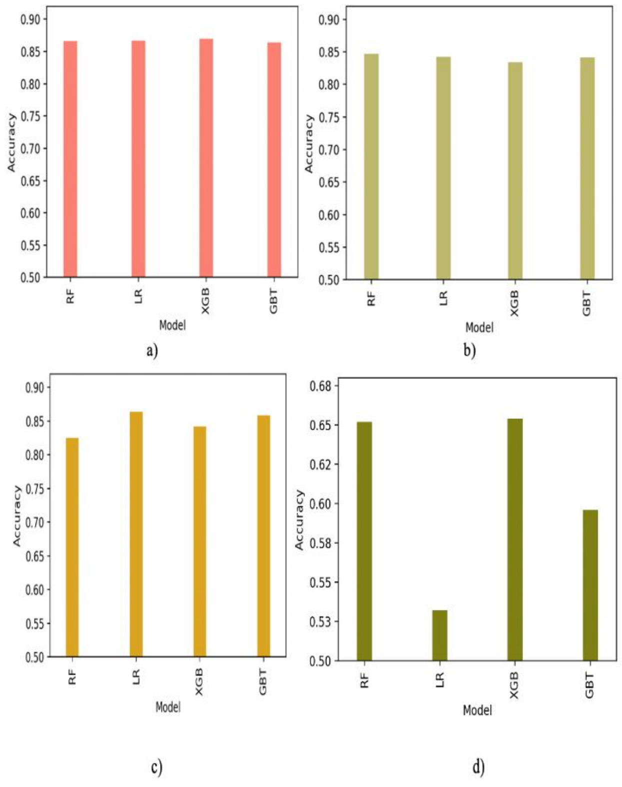


Figure 1: Model Accuracies for a) Value based Model b) Emotion based model c) Topic-specific emotion-based model d) User Interest Model.

6 Results and discussion

All the above models were initially evaluated on the accuracy metric.

Figure 1 shows the accuracies of value-based models along with previously used models for the task of predicting retweet decision of users. The accuracies are calculated using classifiers namely, Random Forest (RF), Logistic Regression (LR), XGB (Extreme gradient boosting trees), GBT (Gradient boosting trees). The figure shows 4 sub-parts demonstrating the accuracies of value-based model, emotion-based model, topic-specific emotion-based model and user interest model respectively.

The value-based model (VM model) uses value system and the value similarity score between target tweet and user profile. This model has a comparable performance across all the classifiers used, with XGB performing slightly better than others.

The emotion-based model (EM model) simply uses the 10-dimensional emotion vector as a feature for the prediction. As seen in Figure b), this model as well has a comparable performance across all classifiers used with a slight improvement with the random forest classifier.

Figure c) shows the topic-specific emotion-based model (TSM model) accuracy for the task of retweet prediction. It uses the topic-specific emotion feature for predicting retweet behavior by using the conditional probability of an emotion given a topic in the target tweet.

For this model, it can be seen that logistic regression performs the best when drawn a comparison with other classifiers.

The user-interest model (UIM model) uses the cosine similarity between the user interest vector and the target vector as a feature. The accuracy of this model varies with the type of classifier used. We can see that when using logistic regression classifier, this model performs the worst but shows a great improvement when tested with other classifiers.

	Models			
Classifier	TSM	EM	VM	UIM
Random Forest	.825	.847	**.866**	.652
XGB	.842	.834	**.870**	.654
GBT	.859	.841	**.864**	.596
Logistic Regression	.864	.842	**.867**	.532

Table 3: A comparison of various models on the basis of accuracy.

Classifier	**Random Forest**			**XGB**			**GBT**			**Logistic regression**		
Model	P	R	F1	P	R	F1	P	R	F1	P	R	F1
TSM	.709	.712	.710	.713	.716	.715	.744	.714	.727	.757	.713	.732
EM	.737	.716	.725	.702	.717	.709	.713	.717	.715	.713	.715	.714
VM	**.760**	**.718**	**.736**	**.770**	**.724**	**.743**	**.755**	**.720**	**.735**	**.763**	**.714**	**.734**
UIM	.558	.597	.547	.559	.598	.548	.554	.597	.520	.550	.591	.483

Table 4: A comparison of various models based on precision, recall and F1 score.

Table 3 and 4 shows the comparative performance between value-based model with previously used retweet prediction models. Table 3 draws a comparison between different models based on accuracy. We used four classifiers viz, Random Forest, XGB, GBT and Logistic Regression, for each of the models to be compared.

The user-interest model (UIM model) uses the cosine similarity between the user interest vector and the target vector as a feature. The topic-specific emotion-based model (TSM model) uses the topic-specific emotion feature for predicting retweet behavior. It uses the conditional probability of an emotion given a topic in the target tweet. Emotion-based model (EM model) simply uses the 10-dimensional emotion vector as a feature for the prediction. Value based model (VM model) on the other hand, uses value system and the value similarity score between target tweet and user profile.

As it can be seen, the UIM model achieves the worst performance as compared to other models. This is in conformance to our expectation as previous studies use this model as a baseline [15][11]. TSM model shows a better performance than the EM model across all classifiers, however, EM model has an improved accuracy with Random Forest classifier. This indicates that mutual effect of topic and emotions can be treated as a comparable feature to the use of emotions for predicting retweet behaviors.

Comparing VM model to TSM and EM model, VM model has an improved accuracy across all classifiers.

This indicates that using value systems of an individual can prove to be potential predictor of their retweet decision. Also, we believe that the use of word2vec model to generate similar words for the valueDict lexicon captures the underlying contextual information in the content which when used to label the value systems of users helps in having a higher performance for the retweet prediction task.

Accuracy is a good metric when the distribution of our target is balanced. However, in case of imbalanced classes, it is good to evaluate our test set based on other metrics like precision, recall and F1 score.

Precision is the ratio of correctly classified true instances to total classified instances as positive. Recall is the ratio of correctly identified true positives to the total instances that were originally positive.

Precision can be also written as:

$$Precision = True\ Postives/(True\ Positives + False\ Positives)$$

Recall can also be expressed as in:

$$Recall = True\ Positives/(True\ Positives + False\ Negatives)$$

F1 score is a metric that represents the harmonic mean of precision and recall. It is important to look at this metric as a model with a very high precision and a very low recall is also not considered to be a useful model. Hence F1 score provides a mean to judge the performance of both metrics.

Hence, we present a comparison based on evaluation metrics that we used for our test set, namely, precision, recall and F1 score in Table 4.

The Table presents a comparison between our proposed value-based model with the previous state of art models used for the given classification task.

A similar pattern as that in accuracy can be seen in these metrics while comparing across the different models. The precision of the VM model proves to be higher as compared to TSM, EM and UIM models across all classifiers. This again proves the ability of using value systems as a feature for the retweet prediction.

Comparing the TSM and EM model in terms of precision, we can see that TSM model shows a higher precision when used with all classifiers except Random Forest. This proves again that both features can be said to be potential predictors rather than one being superior to the other.

Looking at the recall, we see that almost all the models have a comparable performance, with VM model having a slightly better performance than others.

However, to have a look at both the measures jointly, we consider the harmonic mean of precision and recall, used as F1 score for our evaluation. Through the results, we can see that VM model has a higher F1 score as compared to all the other models.

As expected, the baseline using UIM Model has a lower performance for all the metrics.

These results confirm the importance of value systems as a potential predictor of retweet behaviors in addition to state of art features previously used: emotions, sentiments, and topic-specific emotions. This work can be used in all the applications of retweet prediction including viral marketing, emergency response and tweet recommendation. Value Systems of an individual can also be used in practice to identify spammers.

7 Conclusion

Predicting retweet decisions of a user is a challenging problem. The retweet behavior of a user correlates with factors like emotions, sentiments, topic-specific emotions as studied and showed by the past studies. Value systems have also been shown in the past studies as an important predictor of user decisions, however, its impact was not yet explored in the domain of retweet prediction. Hence, in this work, our objective was to explore the impact of value systems on the retweet decisions of the user. Value Systems, being a latent attribute of a user have a potential to have a good predictive power in deciphering retweet behavior of the user. We presented a value-based model explaining the methodologies to extract value related features. We also compared our model with previous state of art models used. Through different experiments, our work shows that value systems, are indeed an important factor in predicting retweeting decisions of the user. The future work of our paper includes studying and comparing other state of art models with value-based models.

References

[1] Abel, F., Gao, Q., Houben, G. J., Tao, K. 2011. Analyzing user modeling on twitter for personalized news recommendations. In international conference on user modeling, adaptation, and personalization, pages 1–12. Springer. https://doi.org/10.1007/978-3-642-22362-4_1

[2] Ali, S., Katoma, V., Tyobeka, E. 2015. Identification of key values and behaviours influencing leadership orientation in Southern Africa. Journal of Emerging Trends in Educational Research and Policy Studies, 6(1):6–12.

[3] Bonsón, E., Perea, D., Bednárová, M. 2019. Twitter as a tool for citizen engagement: An empirical study of the Andalusian municipalities. Government Information Quarterly, 36(3):480–489. https://doi.org/10.1016/j.giq.2019.03.001

[4] Boyd, D., Goder, S., Lotan, G. 2010. Tweet, tweet, retweet: Conversational aspects of retweeting on

twitter. In 2010 43rd Hawaii International Conference on System Sciences, pages 1–10. IEEE. https://doi.org/10.1109/hicss.2010.412

[5] Can, E. F., Oktay, H., & Manmatha, R. (2013, October). Predicting retweet count using visual cues. In Proceedings of the 22nd ACM international conference on information & knowledge management (pp. 1481-1484). https://doi.org/10.1145/2505515.2507824

[6] Chawla, N. V., Bowyer, K. W., Hall, L. O., Kegelmeyer, W. P. 2002. SMOTE: Synthetic Minority Over- sampling Technique. Journal of Artificial Intelligence Research, 16:321–357. https://doi.org/10.1613/jair.953

[7] Chen, J., Hsieh, G., Mahmud, J. U., Nichols, J. 2014. Understanding individuals' personal values from social media word use. In Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing, pages 405–414. ACM. https://doi.org/10.1145/2531602.2531608

[8] Chen, K., Chen, T., Zheng, G., Yao, J. O., Yu, E., Y 2012. Collaborative personalized tweet recommendation. Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval 2012, ACM, pages 661–670. https://doi.org/10.1145/2348283.2348372

[9] Dehghani, M., Gratch, J., Sachdeva, S., Sagae, K. 2011. Analyzing conservative and liberal blogs related to the construction of the 'Ground Zero Mosque'. Proceedings of the Annual Meeting of the Cognitive Science Society, 33.

[10] Deng, Z., Yan, M., Sang, J., Xu, C. 2015. Twitter is faster: personalized time-aware video recommendation from Twitter to YouTube. ACM Trans Multimed Comput Commun Appl (TOMM), 11(2):31–31. https://doi.org/10.1145/2637285

[11] Firdaus, S. N., Ding, C., Sadeghian, A. 2019. Topic specific emotion detection for retweet prediction. International Journal of Machine Learning and Cybernetics, 10(8):2071–2083. https://doi.org/10.1007/s13042-018-0798-5

[12] Fleischmann, K. R., Oard, D. W., Cheng, A.-S., Wang, P., Ishita, E. 2009. Automatic classification of human values: Applying computational thinking to information ethics. Proceedings of the American Society for Information Science and Technology, 46(1):1–4. https://doi.org/10.1002/meet.2009.1450460345

[13] Griffiths, T. L. Steyvers, M. 2004. Finding scientific topics. Proceedings of the National Academy of Sciences, 101(Supplement 1):5228–5235. https://doi.org/10.1073/pnas.0307752101

[14] Hong, O., Dan, B. D., Davison 2011. Predicting popular messages in twitter. Proceedings of the 20th international conference companion on World wide web, pages 57–58. https://doi.org/10.1145/1963192.1963222

[15] Huang, D., Zhou, J., Mu, D., Yang, F. 2014. Retweet behavior prediction in twitter. 2014 IEEE Seventh international symposium computational intelligence and design (ISCID), 2:30–33. https://doi.org/10.1109/iscid.2014.187

[16] Ishita, E., Oard, D. W., Fleischmann, K. R., Cheng, A.-S., Templeton, T. C. 2010. Investigating multi-label classification for human values. Proceedings of the American Society for Information Science and Technology, 47(1):1–4. https://doi.org/10.1002/meet.14504701116

[17] Jenders, M., Kasneci, G., Naumann, F. 2013. Analyzing and predicting viral tweets. Proceedings of the 22nd international conference on world wide web 2013, ACM, pages 657–664. https://doi.org/10.1145/2487788.2488017

[18] Jiang, B., Lu, Z., Li, N., Wu, J., Jiang, Z. 2018. Retweet prediction using social-aware probabilistic matrix factorization. International Conference on Computational Science, pages 316–327. https://doi.org/10.1007/978-3-319-93698-7_24

[19] Jiang, B., Yi, F., Wu, J., Lu, Z. 2019. Retweet prediction using context- aware coupled matrix-tensor factorization. International Conference on Knowledge Science, Engineering and Management, pages 185– 196. https://doi.org/10.1007/978-3-030-29551-6_17

[20] Kakar, S., Dhaka, D., Mehrotra, M. 2020. Value-Based Behavioral Analysis of Users Using Twitter. In Inventive Communication and Computational Technologies, Lecture Notes in Networks and Systems, Springer, volume 145. https://doi.org/10.1007/978-981-15-7345-3_23

[21] Kanavos, A., Perikos, I., Vikatos, P., Hatzilygeroudis, I., Makris, C., Tsakalidis, A. 2014. Modeling retweet diffusion using emotional content. In IFIP International conference on artificial intelligence applications and innovations, pages 101–110. Springer. https://doi.org/10.1007/978-3-662-44654-6_10

[22] Kaufmann, E. 2016. It's NOT the economy, stupid: Brexit as a story of personal values. British Politics and Policy at LSE.

[23] Lee, K., Mahmud, J., Chen, J., Zhou, M., & Nichols, J. 2015. Who will retweet this? detecting strangers from twitter to retweet information. ACM Transactions on Intelligent Systems and Technology (TIST), 6(3), 1-25. https://doi.org/10.1145/2700466

[24] Lee, W. J., Oh, K. J., Lim, C. G., Choi, H. J. 2014. User profile extraction from twitter for personalized news recommendation. 16th International conference on advanced communication technology, pages 779–783. https://doi.org/10.1109/icact.2014.6779068

[25] Lu, C., Lam, W., Zhang, Y. 2012. Twitter user modeling and tweets recommendation based on Wikipedia concept graph. Workshops at the Twenty-Sixth AAAI conference on artificial intelligence.

[26] Luo, Z., Osborne, M., Tang, J., Wang, T. 2013. Who will retweet me? Finding retweeters in Twitter. Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, pages 869–872.

https://doi.org/10.1145/2484028.2484158
[27] Ma, R., Hu, X., Zhang, Q., Huang, X., Jiang, Y. G. 2019. Hot topic-aware retweet prediction with masked self-attentive model. Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 525–534. https://doi.org/10.1145/3331184.3331236
[28] Macskassy, S. A. Michelson, M. 2011. Why do people retweet? Anti-homophily wins the day! In 5th international AAAI conference on weblogs and social media, pages 209–216.
[29] Mazzi, M. A., Rimondini, M., van der Zee, E., Boerma, W., Zimmermann, C., Bensing, J. 2018. Which patient and doctor behaviours make a medical consultation more effective from a patient point of view. Results from a European multicentre study in 31 countries. Patient Education and Counseling, 101(10):1795-1803. https://doi.org/10.1016/j.pec.2018.05.019
[30] Mikolov, T., Chen, K., Corrado, G., Dean, J. 2013. Efficient estimation of word representations in vector space.
[31] Mohammad, S. M. Turney, P. D. 2013. CROWDSOURCING A WORD-EMOTION ASSOCIATION LEXICON. Computational Intelligence, 29(3):436–465. https://doi.org/10.1111/j.1467-8640.2012.00460.x
[32] Naveed, N., Gottron, T., Kunegis, J., Alhadi, A. C. 2011. Bad news travel fast: A content-based analysis of interestingness on twitter. In Proceedings of the 3rd international web science conference, pages 1–7. ACM. https://doi.org/10.1145/2527031.2527052
[33] Peng, H. K., Zhu, J., Piao, D., Yan, R., Zhang, Y. 2011. Retweet modeling using conditional random fields. 2011 IEEE 11th International conference on data mining workshops (ICDMW), pages 336–343. https://doi.org/10.1109/icdmw.2011.146
[34] Perikos, I. Hatzilygeroudis, I. 2013. Recognizing emotion presence in natural language sentences. In International conference on engineering applications of neural networks 2013, pages 30–39. Springer. https://doi.org/10.1007/978-3-642-41016-1_4
[35] Petrovic, S., Osborne, M., & Lavrenko, V. (2011, July). Rt to win! predicting message propagation in twitter. In Proceedings of the International AAAI Conference on Web and Social Media (Vol. 5, No. 1).
[36] Pfitzner, R., Garas, A., Schweitzer, F. 2012. Emotional divergence influences information spreading in Twitter. Sixth international AAAI conference on weblogs and social media, 12.
[37] Plutchik, R. 2001. The Nature of Emotions. American Scientist, 89(4):344–344.
[38] Rao, Y., Li, Q., Wenyin, L., Wu, Q., Quan, X. 2014. Affective topic model for social emotion detection. Neural Networks, 58:29–37. https://doi.org/10.1016/j.neunet.2014.05.007
[39] Rathbun, B. C., Kertzer, J. D., Reifler, J., Goren, P., Scotto, T. J. 2016. Taking Foreign Policy Personally: Personal Values and Foreign Policy Attitudes. International Studies Quarterly, 60(1):124–137. https://doi.org/10.1093/isq/sqv012
[40] Roberts, K., Roach, M. A., Johnson, J., Guthrie, J., Harabagiu, A. M. 2012. Empatweet: annotating and detecting emotions on Twitter. LREC 12, 12:3806–3813.
[41] Schwartz, S. H. 1994. Are there universal aspects in the structure and contents of human values? Journal of social issues, 50(4):19–45. https://doi.org/10.1111/j.1540-4560.1994.tb01196.x
[42] Suh, B., Hong, L., Pirolli, P., Chi, E. H. 2010. Want to be retweeted? large scale analytics on factors impacting retweet in twitter network. In and others, editor, IEEE Second International Conference on Social Computing. https://doi.org/10.1109/socialcom.2010.33
[43] Templeton, T. C., Fleischmann, K. R., Boyd-Graber, J. 2011. Simulating audiences: Automating analysis of values, attitudes, and sentiment. 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing, pages 734–737. https://doi.org/10.1109/passat/socialcom.2011.238
[44] Wang, Q., Li, L., Wang, D. D., Zeng 2017. Incorporating message embedding into co-factor matrix factorization for retweeting prediction. International Joint Conference on Neural Networks (IJCNN), pages 1265–1272. https://doi.org/10.1109/ijcnn.2017.7965998
[45] Wang, W., Zuo, Y., Wang 2015. A multidimensional nonnegative matrix factorization model for retweeting behavior prediction. Mathematical Problems in Engineering. https://doi.org/10.1155/2015/936397
[46] Xu, Z. Yang, Q. 2012. Analyzing user retweet behavior on twitter. Proceedings of the 2012 international conference on advances in social networks analysis and mining, pages 46–50. https://doi.org/10.1109/asonam.2012.18
[47] Yang, Z. 2010. Understanding retweeting behaviors in social networks. CIKM, pages 1633–1636. https://doi.org/10.1145/1871437.1871691
[48] Ye, S., Soutar, G. N., Sneddon, J. N., Lee, J. A. 2017. Personal values and the theory of planned behaviour: A study of values and holiday trade-offs in young adults. Tourism Management, 62:107–109. https://doi.org/10.1016/j.tourman.2016.12.023
[49] Zacharias, C. 2017. Twint-twitter intelligence tool.
[50] Zhang, J., Tang, J., Li, J., Liu, Y., Xing, C. 2015a. Who influenced you? predicting retweet via social influence locality. ACM Trans. Knowl. Disc. Data (TKDD), 9(3):25–25. https://doi.org/10.1145/2700398
[51] Zhang, K., Yun, X., Liang, J., Zhang, X. Y., Li, C., Tian, B. 2016. Retweeting behavior prediction using probabilistic matrix factorization. IEEE Symposium on Computers and Communication (ISCC). https://doi.org/10.1109/iscc.2016.7543897
[52] Zhang, Q., Gong, Y., Guo, Y., Huang, X. 2015b. Retweet behavior prediction using hierarchical

dirichlet process. Twenty-Ninth AAAI Conference on Artificial Intelligence.

Towards a Formal Multi-Agent Organizational Modeling Framework Based on Category Theory

Abdelghani Boudjidj
Ecole nationale Supérieure d'Informatique (ESI), BP 68M, 16270, Oued-Smar Algiers, Algeria
ICOSI Lab University, Abbes Laghrour khenchela BP 1252 El Houria 40004 Khenchela, Algeria
E-mail: a_boudjidj@esi.dz

Elkamel Merah and Mohammed El Habib Souidi
ICOSI Lab University, Abbes Laghrour khenchela BP 1252 El Houria 40004 Khenchela, Algeria
E-mail: ekmerah@gmail.com, souidi.mohammed@univ-khenchela.dz

Keywords: multi-agent systems, organizational models, category theory, formal semantics

Received: September 23, 2019

Multi-agent organizational modeling frameworks can be considered as an efficient solution regarding the distributed applications' problems such as task bundle mechanisms, supply chain management, as well as air traffic control. The main objective of this paper is to provide a solution based on a solid mathematical theory for the modeling, the analysis and the verification of artificial organizations properties, and particularly those of Multi-Agent Systems (MASs). Category theory is a mathematical formalism used to categorically study the logics of organizations in different societies. Therefore, it can be projected on artificial organizations in a categorical way. Our work is revolved around the idea of modeling Multi-Agents organization using category theory. In other words, it consists on the transformation of Agent-Group-Role (AGR) organizational model in a categorical way in order to obtain a formal semantics model describing the MAS organization. This formal model allows the analysis, the verification and also the validation of the main concepts of an organization at a high level of abstraction.

Povzetek: Predstavljena je matematična formulacija agentnih sistemov na osnovi kategorij.

1 Introduction

Multi-agent systems (MASs), in particular, organizational modeling frameworks represent an advanced technology for modern applications that are dynamic, open and distributed. The development of these applications requires the exploration of the related design and analysis methods.

MASs are useful for the modeling and development of distributed computer systems; they have emerged to solve the problems of organizations in large-scale software systems.

Their characteristics allow a better structuration of complex systems, a wide use of MASs in several areas of computer and engineering, such as e-commerce, e-learning, communication, data mining, simulation, robotics, system transport, grid computing …

An organization is a social entity that has a specific number of entities (people, a computer system or an institution). The main objective of the organization is to permit the coordination of these entities. Each member or entity recognizes its role and those of the others, in order to achieve a collective goal [1].

The organization is supposed to have structural and strategic components. These two components are linked, an organizational strategy related to characteristics such as size, innovation, versatility and geographical distribution of the organization. These characteristics imply different coordination mechanisms. Different organizational structures are trained to ensure the coordination mechanisms, and the dependence of different types of tasks [2].

Several organizational models of multi-agent systems have been proposed. They are based on the social structure and organizational concepts for the complex systems construction, in order to propose a solution to the problem of the heterogeneity of languages.

The Organizational MASs Model transformation is an adaptation (making connections between categorical concepts and units of an organization MAS such as AGR), and a way of categorically describing an artificial model, which allows us to have a mathematical model. This model allows us to perform the analysis and the formal verifications of certain properties of organizational MASs.

Many studies have addressed organizational models in MASs, such as the AGR model (Agent Group Role), representing the former Aalaadin model evolution proposed in [3].

In this model, the organization is defined as a framework of activities. The interactions are based on group agents' notions, roles and their relations. The AGR model focuses on the organization structure and both groups and roles definitions, this model does not deal with the internal architecture of the agent but it focuses on the function of this agent within an organization (its role) [3].

We can cite three extensions of AGR organizational model, the model Agents Group Role Environment [4], known as AGRE, which takes into consideration the physical and social environments placed in domains, known as spaces.

The second extension is the Agent Group Service Role (AGSR) organizational model [5], represents another conceptual and architectural framework for organizing open and dynamic multi-agent systems.

And finally, the Agent-Group-Role-Membership Function (AGRMF) [6]. In this model, the agent is an entity capable of acting and communicating. The group is equipped with a membership function managing its flexibility (access of the agent to the group) and also represents the primitive notion of the combination of the agents.

There exist other models such as, Model of Organization for multI-agent SystEms (MOISE). Furthermore, The Roadmap organizational model is more similar to agent-oriented methods, which is based on a role, a protocol and an interaction model [7].

Organizational Model for Normative Institutions (OMNI) [8], the organizational model of YAMAM [8], based on four different concepts: Agent, Competence, Role and Task.

As a first approximation, we can say that the category theory is the mathematical study of function algebras, arises from the idea of functions system between certain objects [9]. The category theory was introduced and used as a framework in many cases. In many fields of computer science and software engineering [10], it has a rich body of theory for reasoning about structures (objects and relations between objects), and it is sufficiently abstract to represent a wide range of different specification languages.

In this paper, we are interested in modeling organizational multi-agent systems (MASs), which represent the backbone of our research activity, by using category theory. Primarily focusing on organization-based methods, these methodologies are still fairly new, and mainly focused on the analysis phase. Otherwise, the design and implementation phases are either missing or redirected to agent-based methodologies, which do not offer enough tools to model organizational concepts. Therefore, there is still a gap between the analysis and the design, which must be clearly, correctly, and completely specified.

In our work we will use category theory to interpret MAS organizational models. In other words, our objective is to transform and to validate organizational concepts of AGR (semi-formal model) to the categorial formal model with category theory. The following diagram details the different stages of our approach.

After the exploration and the Analysis of the MAS organizational concepts, we have seen the usefulness in the transformation of AGR organizational model. Once our objective is fixed, we proceed to its categorical modeling. A categorical modeling aims to construct categories for AGR model with its organizational concepts (Agents Groups and Roles) and the relation between them. We study and validate the different proprieties emerging for the agent 'interactions in the categorical model. If the properties of the chosen model are not satisfactory, it is necessary to return to the formal model.

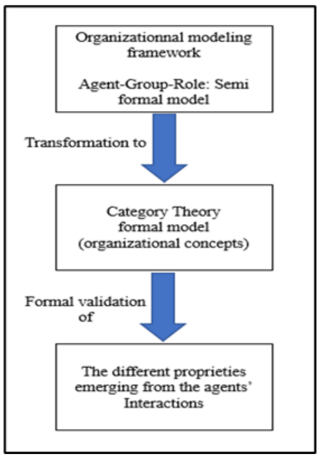

Organizationnal modeling framework
Agent-Group-Role: Semi formal model
Transformation to
Category Theory formal model (organizational concepts)
Formal validation of
The different proprieties emerging from the agents' Interactions

Figure 1: Diagram of different stages of our approach.

Firstly, we present category theory. The category is a bunch of objects. These objects are linked together with arrows known as morphisms. It can be compared to a set of objects but a category is larger than sets.

A morphism or arrow can be defined as a link between two objects. So an object A, object B and an arrow between them called path.

An object can be considered as primitive in this theory. It has no properties and does not have an internal structure. An object can be compared with an atom (it is like a point without properties). A morphism R is also a primitive, it has no properties except that each arrow has a beginning and an end. In fact, the reason of having objects is to mark the ends of the arrows (morphisms). In other words, if we have objects *a* and *b*, we can have zero or more arrows linking between them [11].

For each pair of objects, we need to specify the arrows that link between them with different names, some objects are not connected with arrows, other are connected with an arrow, and other objects are connected with an infinite number of arrows. The number of arrows can be an incalculable linking from two objects to the others. There exists a morphism of identity for each object in a category.

Our contribution will be reflected by the introduction of the composition with the aim of obtaining a composition table that composes the morphisms, and provides different categories [11].

Category theory is general and aims to unify mathematical modeling languages that provides many universal building principles [12]. This framework offers a structure for formalizing large specifications and provides primitive compositions in algebraic languages and specification languages of temporal logic [13].

There exist different types of categories such as:

- Set Category, in which the objects are considered as sets. The arrows or morphisms linking between sets A and B represent all functions from A to B [14].

- Discrete Category, is a category where morphisms are only morphisms of identity. For example, we suppose that X and Y are different objects in category C, the morphism of X to X can only be the morphism of identity of X, and the morphism of X to Y does not exist, which means :$(X, X) = \{id_X\}$ for all X objects, and mor$(X, Y) = \emptyset$ for all $X \neq Y$ objects [15].

- Path Category, before presenting this Path we need to have basic knowledge about directed graphs. A directed graph G is a set O of objects called vertices or nodes, and a set A of ordered pairs of vertices are called arrows or directed edges [16]. Each arrow diagram or directed graph can be interpreted as a category named Path, whose morphisms are sequences (Paths) of arrows.

Other types of examples that we often see in mathematics are the categories of structured sets. In other words, sets with another "structure" and functions which "preserve" them, where these notions are determined independently such as: groups and group homomorphisms, vector spaces and linear maps, graphs and homomorphisms of graphs, the real numbers $\mathbb{R}$ and the continuous functions $\mathbb{R} \rightarrow \mathbb{R}$, the open subsets $U \subseteq \mathbb{R}$ and the continuous functions $f: U \rightarrow V \subseteq \mathbb{R}$ defined on them, topological spaces and continuous maps, differential manifolds and smooth maps, the natural numbers $\mathbb{N}$ and all recursive functions $\mathbb{N} \rightarrow \mathbb{N}$, or as in the example of continuous functions, we can take partial recursive functions defined on the subsets $U \subseteq \mathbb{N}$.

The paper is organized as follows: in the first section, we will discuss the use of Category Theory (CT) in relation to the multi-agent systems. In the second section, we will describe the problem, in which we detail the different concept forming AGR organizational model, and also the concept of category theory. The third section is devoted to the AGR model interpretation via the utilization of category theory. In the fourth section, we will present a case study for the validation of the categorical model of the AGR organization. In the final section of this paper, we will conclude by brief notes reflecting the importance of the proposed work.

2 Related work

Several research activities are based on categories theory and their mathematical concepts at a very abstract level, in agent-oriented or organization-oriented multi-agent systems.

In a recent work [17], they studied human organization using the category theory in a philosophical way. They also represented the human society concepts via the category theory concepts, and put links between the two concepts. According to this work, the categories theory with a high level of abstraction allows the formalization of social (collective) models within the framework of social theory in order to explore their interaction, to express the organizational concepts of MASs and software components with the same categorical terms and also their interactions. Category theory helps in organizational models' validation. Once the MAS organizational categorical model is generated, it is possible to analyze and study some properties of organizations.

In [18], they proposed a high-level agent-oriented modeling. The authors have categorically detailed the structure of a Belief Desire Intention (BDI) agent, through the modeling of these concepts. After a design and modeling of an Autonomous Reactive System (ARS) by a multi-agent system, the Category Theory will be applied to formalize the autonomous behavior of MAS and also that of the ARS. The mapping of ARS and MAS via category theory allows the formalization of the autonomous behavior of these lasts with the aim of proving that the two categorical representations mapped from ARS and MAS are isomorphic. This step will guarantee that the autonomous behaviors of ARS and MAS are the same. This work presented the possibility of coupling MASs and CT.

In [19], the authors introduced a MAS categorical generic model, leading to the MAS category. In this category, objects are agents of different types and the morphisms represent all kinds of relations between agents, called communication and general cooperation arrows. This general structure of communication and cooperation is represented by a corresponding arrow diagram, called Basic Diagram of MAS. The first formulation of the idea is based on the categorical modeling of relations such as the category Path(X, R), a natural description of the basic diagram of a MAS in categorical notions arises.

In [20], they introduced a formal verification of a concurrent system based on category theory. In other words, they managed the consistency between design and implementation in the phase of a concurrent system's development.

As a first step, they designed the concurrent system using communicating sequential processes (CSP). The next step focused on the implementation of the concurrent system with a process-oriented programming language known as Erasmus.

They categorically verified the implementation against the design of the concurrent system as a last step.

In [21], the authors proposed a formal approach, named Reactive Autonomic Systems Framework (RASF), based on category theory, to face the challenges such as group behavior that emerges complex and unexpected, which be in need of a formal specification and verification, they focus on the formal specification of substitutability property for the fault-tolerance, their approach was illustrated via Mars-world case study.

They used RASF to model the Reactive Autonomic Systems (RAS). They built a RAS meta-model with categorical specification as a first step. Secondly, they transformed it to be applicable in the case of exploring the Mars-world. Finally, they implemented these RAS models through the Multi-Agent Systems (MAS) using JADEX, where they have five types of agents (robots) as Manager, Supervisor, Sentry, Producer and Carry agents. They have proved in this work that category theory is largely used to capture constructs and their relations in a system, in a formal way and in a single categorical presentation.

In [22], the contribution is reflected through the construction of a transformation system for Multi-Agent System (hereafter, MAS) modeled and based on category theory as a common linguistic formal support. They introduced the Category MAS of all Multi-Agent Systems, the objects represent agents and morphisms represent the relations between the agents. The relations between agents can change according to the MAS dynamism. Therefore, it is justifiable to define the Category MAS of All MASs, where the objects are Multi Agent Systems and the morphisms are MAS's morphisms. They associate a base diagram to each MAS representing the complete relational structure. The Double Pushout Method (hereafter, DPO), a concept widely developed in the field of algebraic graph transformations, they proposed a MAS semantic to the DPO approach. They proved in this work that there is a strong need for formalization of MAS, and their goal was to develop a toolbox for MAS modeling using category theory notions.

3 Problem description

The use of organizations in multi-agent systems is an attractive research activity requiring a formal framework to mathematically manage it, and proceed to the validation of the different properties.

Several models have been realized in order to reflect the importance of organization in multi-agent systems, and to lead to efficient solutions regarding the complex problems. New concepts have been used in MASs according to the proposed model such as AGR, which we are going to detail.

The category theory provides many mathematical aspects and concepts at a very abstract level. This fact will allow us to represent or transform the concepts related to organizational MAS. This formalization will lead us to a categorical model for the AGR model (formal model).

Also, category theory will allow us to examine the AGR in an abstract way by formalizing the system as collections of objects (categories) and morphisms with the aim of reasoning about these objects and their relations or interactions (morphisms). This point is very suitable for agent-based systems like organizational MAS. The formal obtained model will lead itself to formal verifications of interesting properties of organizational SMA.

Several studies have presented the possibility of using CT in relation to MAS, and consequently, the modeling of the AGR model (Organization Oriented).

The problem studied is to reformulate AGR organization with CT and to validate this approach. An instance of market organization is used in order to clarify the use of AGR model, to find an appropriate supplier among the set of participating suppliers to the market. Specifically, this case study will be used to test our AGR formalization where groups and roles will interact and communicate with each other. The agents engaged and involved in the market will proceed to negotiation in order to appoint the supplier. Properties will be presented in this example such as the flexibility of agents, negotiation between agents playing the roles, as well as an agent playing one or more roles.

3.1 An agent in agent group role (AGR) model

This model represents the agent as an autonomous entity that can communicate with other agents, and can play one or more roles in one or more groups. No constraint is placed on the internal architecture of the agent to allow each designer to choose the adequate agent according to the processed problem. A simple semantic structure formula of Agent $(A)$is a tuple:

A:$< Id, N, PLs >$

Id: The Agent Identity.

N: Agent Name.

PLs: roles played by the agent.

3.2 Group in agent group role (AGR) model

The group represents a set of agents that shares characteristics, such as usual MAS, where each agent can be a member of one or more groups. The group is used to divide the organization where each group has an activity; and any agent can start a group, a semantic structure formula of a group (G) represented as:

$G < Gn, r >$

Gn:group name,

r: resident roles in the group.

Figure 2 represents a class diagram of AGR.

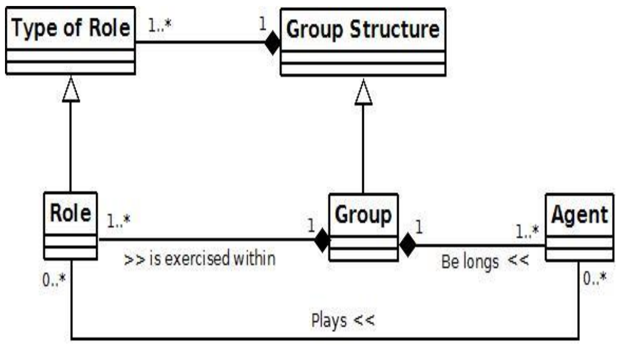


Figure 2: AGR meta-model.

3.3 Role in agent group role (AGR) model

The Role can be considered as the abstract representation of the function of an agent within a particular group. A Role is local in a group where an agent can have one or more roles. A Role is a set of tasks, and in turn a task is a set of actions. The Role (R) is given by a semantic structure as follows:

R:$< Rn, Gn >$

Rn: role name

Gn: group name where role reside.

3.4 Category theory concepts

3.4.1 A category

Includes the following data [14],

• *Objects*: A, B, C, etc.

• *Morphisms* (Arrows): $f, g, h,$ etc.

• Domain (dom) and Codomaine (cod): For each arrow f, we give objects: $dom\ (f)$, $cod(f)$ called domain and codomaine of f, we write:

$f: A \rightarrow B$ to indicate that$A = dom(f)$ and $B = cod(f)$.

• Composition($\circ$): From the arrows $f: A \rightarrow B$ and $g: A \rightarrow C$, that is to say with: $cod(f) = dom(g)$, we have a given arrow: $g \circ f$: A → C.

• Identity: For each object A there is a given arrow $Id_A: A \rightarrow A$, called identity arrow of A.

These components are required to comply with the following laws:

Associativity: $h \circ (g \circ f) = (h \circ g) \circ f$, for all $f: A \rightarrow B$, $g: B \rightarrow C$, $h: C \rightarrow D$.

Unit: $f \circ Id_A = f = Id_B \circ f$, for all $f: A \rightarrow B$.

3.4.2 Isomorphism

In any category C, an arrow $f: A \rightarrow B$ is called an isomorphism if there is an arrow $g: B \rightarrow A$ in C such that $g \circ f = Id_A$ and $f \circ g = Id_B$.

Since inverses are unique (proof), we write $g = f^{-1}$. We say that A is isomorphic to B, written $A \approx B$, if there exists an isomorphism between them [9].

3.4.3 Functor

Definition of a functor [9]:

$F: C \rightarrow D$ between categories C and D is a mapping of objects to objects and arrows to arrows, in such a way that:

(a) $F(f: A \rightarrow B) = F(f): F(A) \rightarrow F(B)$,

(b) $F(Id_A) = Id_{F(A)}$,

(c) $F(g \circ f) = \mathrm{F}(g) \circ F(f)$.

That is, F preserves domains and codomains, identity arrows, and composition.

A functor $F: C \rightarrow D$ thus gives a sort of picture —perhaps distorted—of C in D.

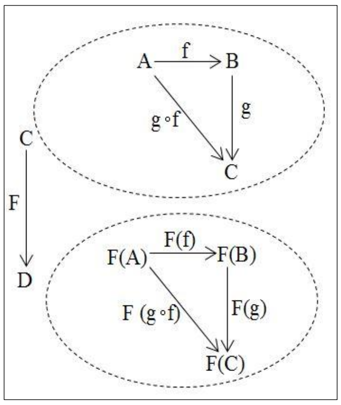


Figure 3: F functor from category C to category D.

3.5 Tool supporting the approach

Category theory is known for its ability to organize the key abstractions that make up most areas of mathematics, and it becomes useful for writing elegant and maintainable code through categorical ideas. Haskell is a programming language used to stimulate the construction of category theory (www.haskell.org).

4 Interpretation of agent group role in category theory

In this AGR model, the organization represents the AGR itself including all the existing groups. A group represents a set of agents and roles. The AGR model gives us the possibility to choose the type of appropriate agent (reactive, interactive, cooperative ...) according to the system or the addressed problem.

Our work focuses on a formal transformation of a specific organizational model. Therefore, we have based our work on reactive agents. An agent is represented by a set of goals, skills, and characteristics. The goals that the agent works for their accomplishment are the performance of a set of actions and also the play of one or more roles. Agents may have new goals after a system update causing new actions to be performed and roles to be played.

On the one hand, the role is a sequence of tasks to be performed by one or more agents (playing the same role). On the other hand, a task is a sequence of actions to be executed in a discrete-time.

The units of the AGR organization (agent groups) can communicate with each other via messages (send and receive). The exchange of messages can provoke new goals and events.

The use of category theory allows us to focus on the morphisms or relations between objects instead of focusing on the internal representations of objects. In the AGR, as it is mentioned in section problem description, each group contains a set of roles played by the belonging agents. The agents belonging to the same group cooperate with each other in order to perform different tasks.

In a group of agents, there exist two different types of communication: local communication and global communication. Local communication occurs only within a group, and worker agents communicate with each other to cooperate. Global communication occurs between agents of different groups.

In this section, we will present the categorical modeling of AGR. We will examine the agent structure and role. Moreover, we will represent the main concepts such as Actions, tasks, role, groups, agent, and their relations via category theory. Then a return on the global system, which is the set of groups, the wholes represents the AGR with the help of constructions of category theory.

The following section contains definitions of the different concepts allowing the formalization of the groups in an AGR organization.

4.1 The role

A role is represented by a sequence of tasks to be sequentially executed. In turn, a task is represented by a set of actions sequentially executed in a discrete-time, presented in a Task category. So, to represent the tasks and their big category TASK, we will first define the category ACTIONS.

4.1.1 Category ACTIONS

ACTIONS is a category that contains all the required actions to perform in order to build a task, objects, and morphisms in that category. These proprieties are defined as follows:

- Objects: are a set of executable actions designated by $A1$, $A2$, ...

- Morphisms: morphisms of identity

ACTIONS is a discrete category that contains identity morphisms. Actions are defined as an abstraction of agents' reaction to environmental events. The following figure shows the ACTIONS category.

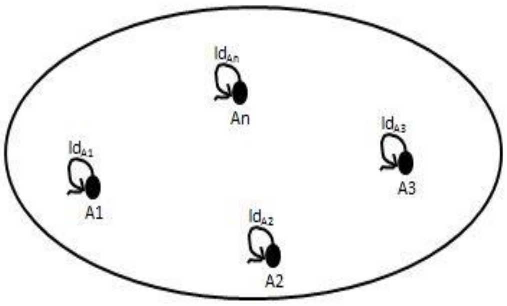


Figure 4: ACTIONS Category.

4.1.2 Category Task

Task is a category representing the sequence of actions where objects are of type $A1$, $A2$ ... and morphisms are the morphisms of object identities, and morphisms of type Before, this type of morphism ensures the order of execution of the actions.

An object named A_{null} means a null action. This type is used to capture exceptions, and it only has the identity morphism. The order of executions of the actions is interesting in this category and by obligation the morphisms of length one, will be accepted ensuring the partial order of execution of the actions. The Figure 5 shows an example of a Task category

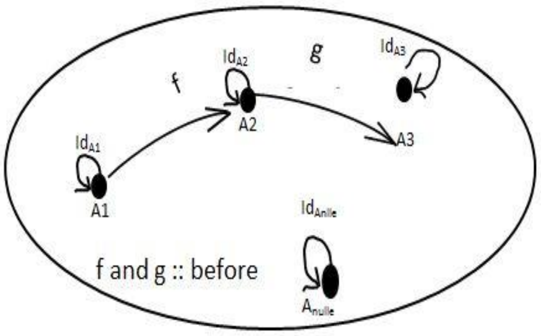


Figure 5: ACTIONS Category.

Where:

Objects: $A1$, $A2$ and $A3$, represent the execution of tasks $A1$, $A2$ and $A3$ respectively.

Morphism: morphisms of identities of objects (Id).

Morphisms between objects of type $(::)$ *before*, the morphism before assures us that the action $A1$ executed during t_1 before $A2$ which is executed in t_2, and vice versa.

The task category is represented by a special category called *Path* which accepts morphisms (arrows), of length one.

In the following figure, we will showcase the non-displayed path of length equal to 2, which is the composition of the two morphisms f and g. As we have noted before, the category theory is based on the composition of the morphisms between the objects and the composition of the functors between the categories.

Morphism before $:: g \circ f = h: A1 \rightarrow A3$. The length of this morphism is equal to 2, it will not be displayed.

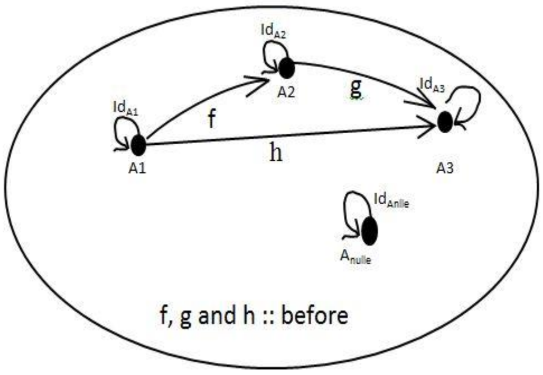


Figure 6: Task Category without the rule application of the PATH category

The morphism h represents the composition of the two morphisms f and g satisfying the laws of associativity and unity. Therefore, the validity of the Task Category is proven.

4.1.3 Functor Sequence_A

This functor maps objects and morphisms from the category ACTIONS to a category as follows:

- Objects: actions in the ACTIONS category maps to the Action objects in the Task Category.
- Morphisms: all the morphisms of identities of the objects $A1$, $A2$, ..., of the category ACTIONS towards the objects $A1$, $A2$, ..., of the category Task.

The figure 7 shows the categories ACTIONS, Tasks of type Task, and functors of type *Sequence_A.*

The preceding diagram presents two categories of the type Task, which were injected from the *Category ACTIONS* by the functors F and G of type *Sequence_A*,

F Functor:

Objects

$F(A1) = task1.A1$

$F(A2) = task1.A2$

$F(A3) = task1.A3$

Morphisms

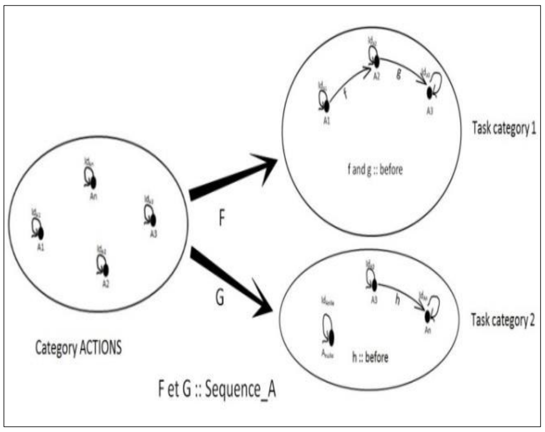


Figure 7: Diagram shows an example of two categories of tasks extracted from the ACTIONS category by two functors.

$F(Id_{A1}) = Id_{F(A1)}$
$F(Id_{A2}) = Id_{F(A2)}$
$F(Id_{A3}) = Id_{F(A3)}$
G Functor:
Objects
$G(A3) = task2.A3$
$G(AN) = task2.AN$
Morphisms
$G(Id_{A3}) = Id_{G(A1)}$
$G(Id_{AN}) = Id_{G(AN)}$

4.1.4 Category TASK

The *Category TASK* represents a large category where, the objects are: Tasks of the categories of the type Task x, $(x = 1 \; or \; 2 \dots)$.
Morphisms are: the morphisms of identities.

4.1.5 Functor collapses_T

Functor that collapses a category of type Task to a T object of the big *Category TASK*:

All the objects of a category of the type Task x, will be collapsed towards an object T in the Category TASK, and all the morphisms will be also collapsed towards a single morphism of identity. The object T of Category TASK will represent a black hole collapsing a whole category of Task x type into a single point.

The figure 8 shows an example of *Task* type categories that would melt into T objects of the large *Category TASK*.

Example functor collapses_T (H):
Objects
$H(task1.A1) = T1$
$H(task1.A2) = T1$
$H(task1.A3) = T1$
Morphisms
$H(Id_{A1}) = Id_{H(T1)}$
$H(Id_{A2}) = Id_{H(T1)}$
$H(Id_{A3}) = Id_{H(T1)}$
$H(f) = Id_{H(T1)}$
$H(g) = Id_{H(T1)}$

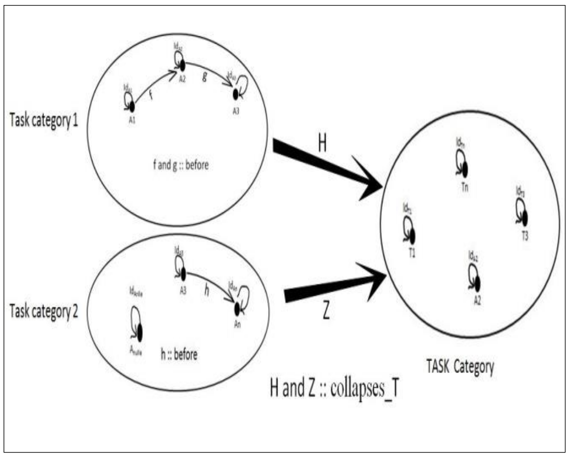


Figure 8: Collapse of *Task x* Categories in the major *Category TASK*.

4.1.6 The Category Role

The *Role* represents a sequence of *Tasks* to be executed sequentially from the large *Category TASK*. We can have several Categories *Role* from the *Category TASK*.

4.1.7 Functor Sequence_T

This Functor maps
Objects: T from *Category TASK* to *Role T* objects.
Morphisms: identity morphisms *Id* of *Category TASK* to identity morphisms *Id Role*.

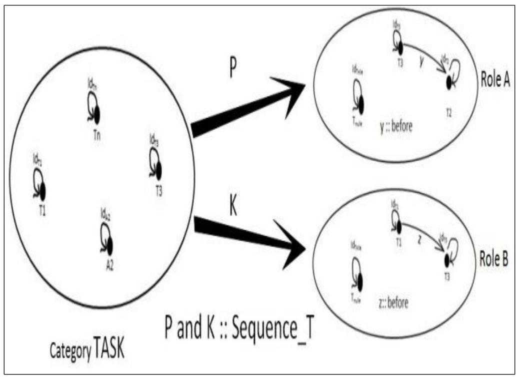


Figure 9: Extraction of *Roles* from the broad *Category TASK*.

4.2 Category Group

Category Group is a category that contains Roles residing in a Group x, with their relations,

4.2.1 Functor Collapse_G

It represents a functor collapsing a category of type *Role* x, in an object *R* of the *Category Group* x,
Objects: The *Role categories* to *R objects* of the *Category Group*.
Morphisms: the morphisms of identities of the *Role categories* towards the morphisms of identity of the objects of the *Category Group*.

All objects of a category of type *Role* x, would collapse to an *object* R in the *Category Group* x, and all morphisms would also collapse to a single identity morphism of the same object, the *object* R of *Groupe* x will also represent a black hole that will collapse an entire category of type *Role* x ($x = A$ *or* $B, \ldots$), into a single point that represents the object.
Figure 10 shows an example of *Role* type categories that would melt into *R objects* in the *Category Group* x.

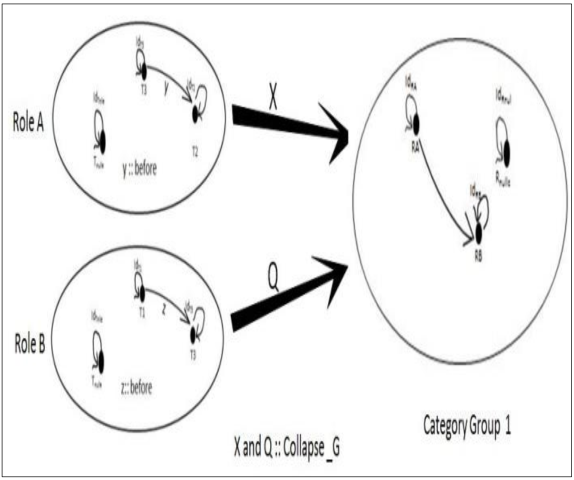


Figure 10: Collapse of Roles x categories in a Category Group x.

From the existing categories *Role*, we can have several categories *Group*. Every action depends on the *Roles* belonging to the *Groups*. This ensures that a *Role x* is resident in one or more *Groups*.

The diagram (figure 11) shows the categories: *ACTIONS*, *Task*, *TASK*, *Role* and *Group*

And the Functors: *Sequence_A*, *collapses_T*, *Sequence_T* and *collapse_G*.

4.3 Category Agents

In this work, the agent is a communicating entity capable of playing one or more Roles in one or more Groups, Category Agents is a category where:

The Agents are represented by the objects. Furthermore, the morphisms only reflect the identities of each object (Agent).

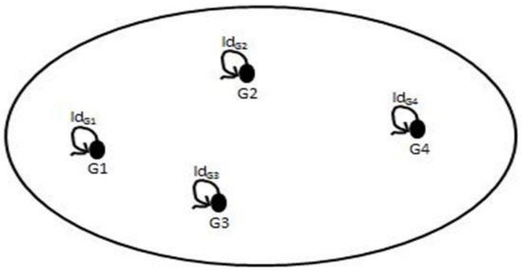


Figure 12: Category Agents.

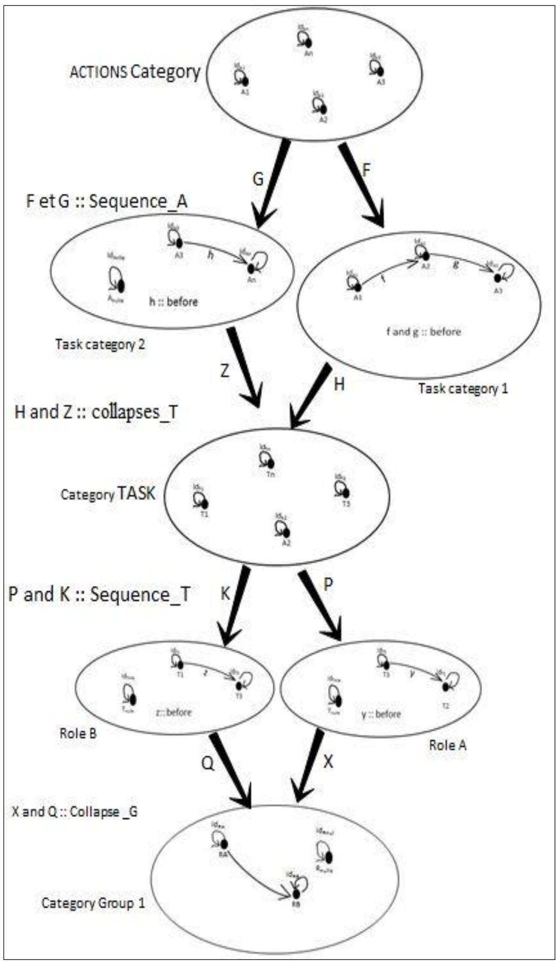


Figure 11: Representation of the categories studied and the relations between them (Functors).

4.3.1 Functor plays_role

This functor allows the Agent to play one or more Roles in a Group or Groups. It maps the objects from the Category Agents to the objects of the Category Group.
Morphisms: It maps the morphisms' identity "Id " of agents to the morphisms of identities of the roles in the group category.

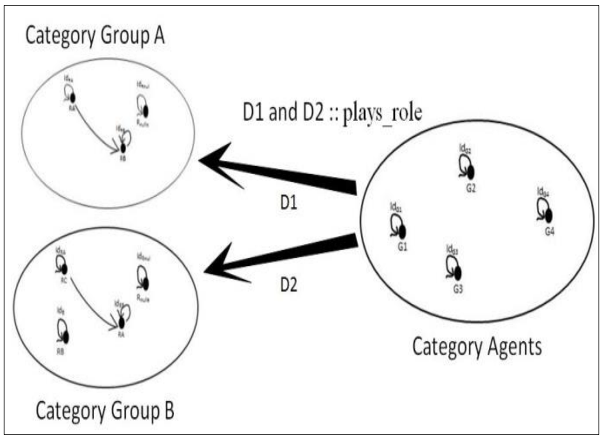


Figure 13: Relationship between Agents category and Groups categories.

The figure 13 shows the Agents category, the plays_role morphism, and the Group category.

We have presented all the categories that represent the AGR model: ACTIONS, Task, TASK, Role, Group and Agents, as well as the relations between them by the functors, Seqence_A, Collapses_T, Sequence_T, Collapse_G and plays_rôle. In Figure 14, an example of 3 Roles, Role A, Role B and Role C residing in two different Groups, Group A and Group B, and 4 Agents A1, A2, A3 and A4 playing these roles within the meaning of these groups. The example of Figure 14 shows us the important flexibility given to Roles and Agents by this categorical representation. A Role belongs to one or more groups. An Agent plays one or more Roles in one or more Groups representing the initial definition of AGR model.

5 Validation

We will validate now our interpretation of AGR organizational modeling framework in terms of category theory, through a case study of a market organization instance [3]. A set of agents represent the different elements forming the *market* structure. The Figure 15 shows the three groups in the *Market* organization, a *Group of Customers*, a *Service Provider Group*, and a *Contract Group*.

Group of Service Providers: in this group it resides the role *Broker* and *service*, the *broker* role is the representative of the *suppliers*.

Customer Group: Gathers the customers whose *customer* role we find, and to find a suitable *supplier*, *customers* interact with the *Service Provider Group* via *Broker*.

Group contract: in this group it resides the roles *seller* and *buyer*, its structure is made to bring together the agents who engaged and involved to move to the negotiation.

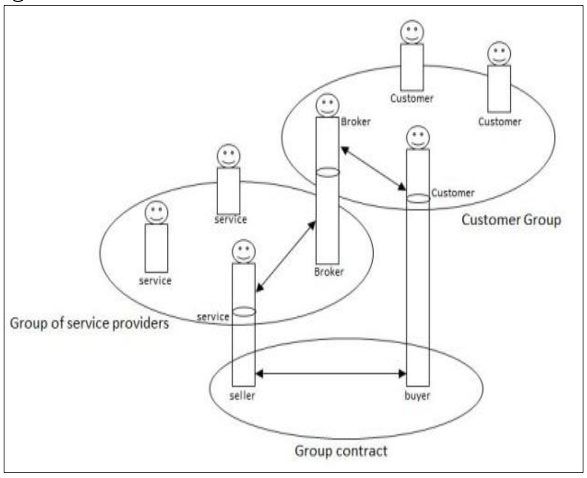


Figure 15: A market organization instance, [3].

In our categorical AGR model an agent can belong to both the client group and the service provider group (message passing). So, it is possible to play multiple roles in multiple groups by an agent and perform tasks. As shown in the last example, an agent will play the Broker role in two different groups.

The figure 16 shows a projection of the Market organization instantiated from the AGR categorical model, including the three groups with their roles.

The category Agents includes all the agents of our organization with the supposition that there exist 7 Agents. Three Agents will play the Role *Customer* in the *Category Customer Group* via the functor:

D1 ∷ *plays_role*, which maps the Agents with the Roles. Also, it linked one of the Agents with the *Broker* Role in the same category.

Another functor *D1* ∷ *plays_role*, links other Agents with *service* Roles in the *Provider Services Group Category*, and also maps the same Agent that plays the *Broker* role in the Category *Customer Group* to the *broker* Role of the Group category served customer he plays the same role in two different groups.

In the *Category Customer Group*, the *customers* are represented by the objects. This category contains the role, *broker* and also the roles *customer*. The morphisms between the role *broker* and the other roles *customer* represent the link between them, and among these links only one *customer* has a response from the *Broker* via a morphism back (an Isomorphic relation), the same thing in the group category served by providers links are generated via morphisms. The *contract group category* will bring together the Agents playing the roles of *seller* and *buyer* via the functor *D3* ∷ *plays_role.*

In this example we have not detailed the categories *ACTIONS* and *TASK*, we have assumed that the actions and tasks presented via each *Role* are predefined.

Each category has an identity morphism that ensures the update (id_update) of the category in case of change, the appearance of a *New Role* for example or one of the Agents does not want to play a *Role* and vice versa. This categorical transformation ensures the flexibility of Agents between Groups, which is a very important property in multi-agent organizational systems.

5.1 Comparison with the existing approaches

The table 1 contains the related works using category theory and multi-agent systems. Through the latter, a comparison study between these approaches was made according to a set of criteria such as the orientation of the multi-agent system (agent-oriented or organization-oriented), the type of agent used, the dynamism of the system, and the stability when changing the role (if one exists) of Agent.

5.2 Evaluation

The table illustrates the differences between the proposed approaches, for example, the approach of Olga Ormandjieva line 4, which is an agent-oriented based work, does not deal with the organization concept, which is the base of our approach. Otherwise, some approaches are based on the use of the organization especially in the MASs.

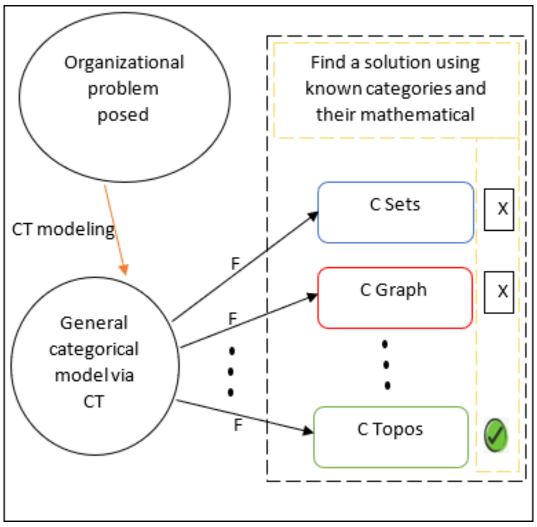

Organizational problem posed
CT modeling
General categorical model via CT
F
F
F
Find a solution using known categories and their mathematical
C Sets
X
C Graph
X
C Topos
✓

Figure 17: Using of Categorical model.

The work of Abderrahim. S, is oriented organization which has studied the link between category theory and human society, the type of agent used is generalized, while our work is based on the model of artificial organization AGR with a specific type of agents.

After the generation of the categorical model, the category theory allows us to switch to several known modeling modes such as graphs or sets. This very important link represented by a functor (F) allows us to switch between the mathematical representation, which give us the possibility to reformulate the studied problem via graphs, sets, topos, and this allows us to use the characteristics and properties of each domain to solve the starting problem (Figure 17).

So, if we categorically model a problem linked to an organization, we can solve it with the sets. If we will not arrive at a solution with these, we can change our mathematical tools and use the topos for example. For that, we just need to go back to our categorial model and change the functor to the topos. General categorical modeling gives us more mathematical choices for problem solving.

6 Conclusion

In this paper, we have proposed a formal semantics of AGR organizational modeling framework in terms of category theory, with its concepts and structures based on Agents, Groups and Roles. This formal semantics allows the analysis, the verification and the validation of the emergent properties.

The main objective is to use category theory to formalize the structures of the AGR as a toolbox in the

Table 1: Comparison the existing approaches in related work.

Authors	Titled work	Agent-centered MAS	Organization-centered MAS	Reactive Agent	Cognitive Agent	Dynamism	Stability
Pfalzgraf, J. 2005	On categorical and logical modeling in multiagent systems. Anticipative Predictive Models in Systems Science. [8]	✓	X	Not mentioned		X	X
Pfalzgraf, J. and T. Soboll,. 2009	On a general notion of transformation for multiagent systems and its implementation. [9]	✓	X	Not mentioned		✓	X
Kuang, H., et al. 2010.	Formal Specification of Substitutability Property for Fault-Tolerance in Reactive Autonomic Systems. in SoMeT. [10]	✓	X	X	✓	✓	✓
Olga Ormandjieva et al.,2015	Modeling multi-agent systems with category theory. [11]	✓	X	X	✓	X	X
Abderrahim, S. and R. Maamri, Dec,2018	A Category-theoretic Approach to Organization-based Modeling of Multi Agent Systems on the Basis of Collective Phenomena and Organizations in Human Societies. [12]	X	✓	Not mentioned		X	✓
Our work	Towards a formal multi-agent organizational modeling framework based on category theory.	X	✓	✓	X	✓	X

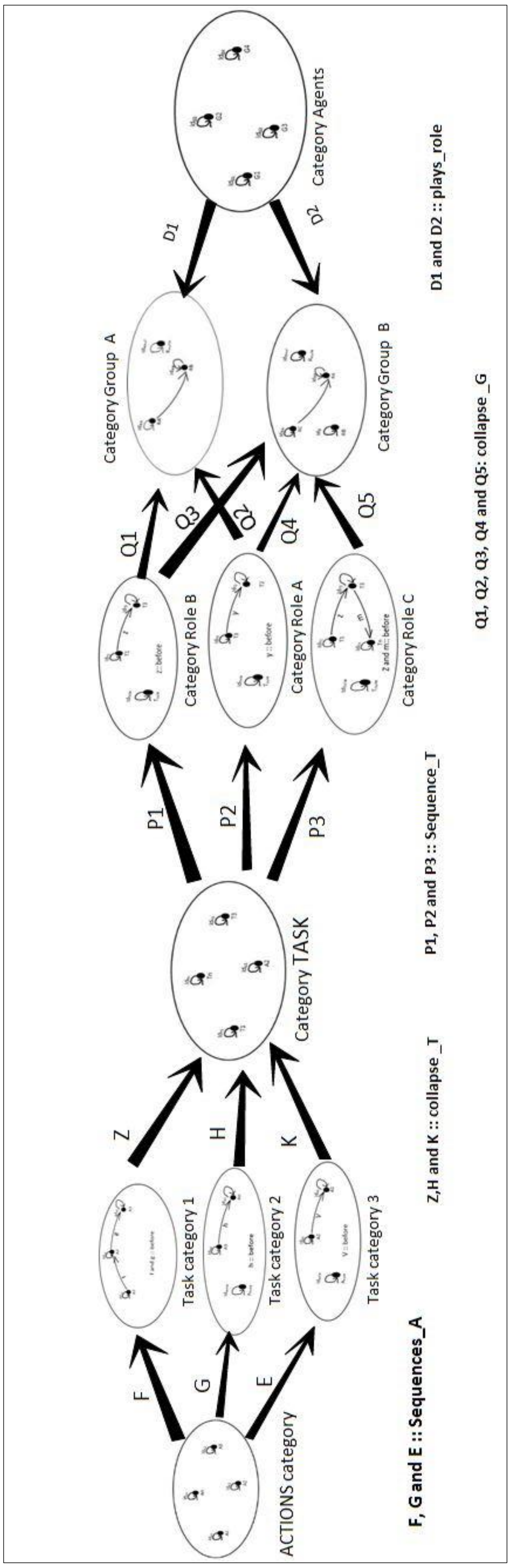


Figure 14: AGR model with Category Theory (categorical representation).

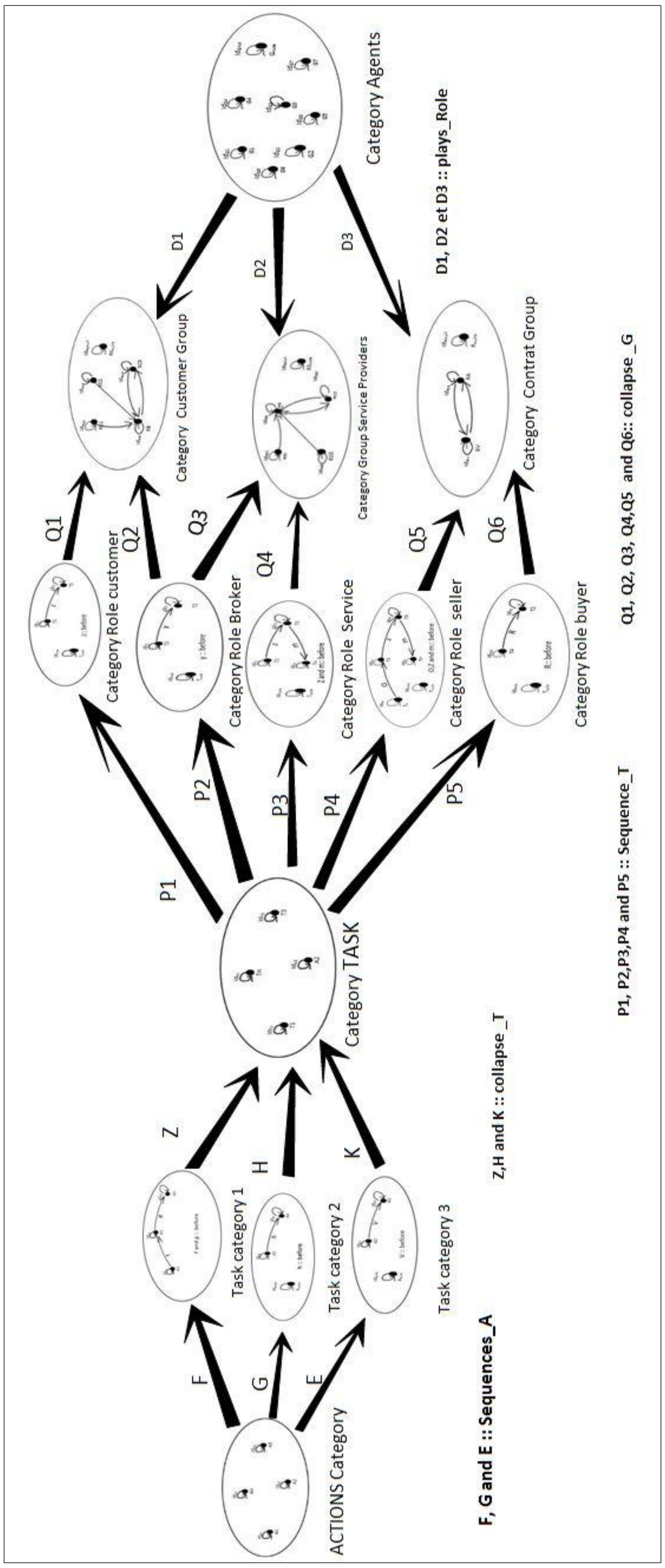


Figure16: Transforming the Market Organization Using the AGR Categorical Model.

field of MAS organization. The category theory allows us to formally achieve this goal in order to represent the different concepts in the same categorical framework. This representation is effectuated via the use of objects, morphisms and their compositions, and also functors between categories of the framework.

As shown in section 3, the different concepts of AGR can be expressed and combined. Our categorical interpretation allows us to check some properties of

MAS, such as flexibility of agents, stability of the groups, as well as the emergence of new roles. The validation of these properties will depend on the final result of the modeling and the processed problem.

In future work, we will focus on the modeling of AGR extensions, such as AGRE, AGRS and AGRMF, as well as other organizational MAS models. Moreover, we will try to establish a relation and a combination between agent-oriented modeling and organization-oriented modeling. Also, we plan to automate the transformation of AGR concepts in concepts of category theory by using Model-Driven Engineering (MDE) and Model Transformation.

References

[1] March, J.G. and R.I. Sutton, Crossroads—organizational performance as a dependent variable. Organization science, 1997. 8(6): p. 698-706. https://doi.org/10.1287/orsc.8.6.698

[2] Donaldson, L., The contingency theory of organizations. 2001: Sage. DOI: 10.1016/B978-0-08-097086-8.73110-2

[3] Ferber, J. and A. Gutknecht. *Un méta-modèle organisationnel pour l'analyse, la conception et l'execution de systèmes multiagents*. In Proceedings of Third International Conference on Multi-Agent Systems ICMAS. 1998. Un meta-modèle organisationnel pour l'analyse ... - LIRMM

[4] Ferber, J., F. Michel, and J. Báez. AGRE: Integrating environments with organizations. in International Workshop on Environments for Multi-Agent Systems. 2004. Springer. DOI: 10.1007/978-3-540-32259-7_2

[5] Mansour, S. and J. Ferber. Agent Groupe Rôle et Service : Un modèle organisationnel pour les systèmes multi-agents ouverts. In Actes du colloque JFSMA. 2007. Agent Groupe Rôle et Service - MaDKit

[6] Souidi, M.E.H., et al., Multi-agent cooperation pursuit based on an extension of AALAADIN organisational model. 2016. 28(6): p. 1075-1088. https://doi.org/10.1080/0952813x.2015.1056241

[7] Vázquez-Salceda, J., V. Dignum, and F. Dignum, Organizing multiagent systems. Autonomous Agents Multi-Agent Systems, 2005. 11(3): p. 307-360. https://doi.org/10.1007/s10458-005-1673-9

[8] Marion, N., Étude de modèles d'organisation sociale pour les environnements virtuels de formation. ftp://ftp.idsa.prd.fr/local/caps/DEPOTS/BIBLIO2006/rapbiblio_marion_nicolas.pdf

[9] Awodey, S., Category theory. 2010: Oxford University Press. Category Theory (Oxford Logic Guides, 52): Awodey, Steve

[10] Fiadeiro, J.L., Categories for software engineering. 2005: Springer Science & Business Media. Categories for Software Engineering - Research - Royal ...

[11] Milewski, B., Category theory for programmers. 2018: Bartosz Milewski. Category Theory for Programmers: The Preface | Bartosz

[12] Lane, S.M., Categories for the Working Mathematician. 1998. Graduate Texts in Mathematics, 1998. Categories for the Working Mathematician - Springer

[13] Wirsing, M., Algebraic specification. Handbook of Theoretical Computer Science (J. van Leeuwen, ed.). 1990. DOI: 10.1016/B978-0-444-88074-1.50018-4

[14] Awodey, S., Category theory. 2006: Oxford University Press. USA. DOI: 10.1093/acprof:oso/9780195307221.001.0001 Published ...

[15] Easterbrook, S., An introduction to Category Theory for Software Engineers. 1999, Department of Computer Science, University of Toronto. An introduction to Category Theory for Software Engineers*

[16] Pfalzgraf, J. and T. Soboll, On a General Notion of Transformationfor Multiagent Systems. Integrated Design Process Technology, IDPT- printed in the United States of America, 2007. K42: Building a Complete Operating System - cs.uni-salzburg.at

[17] Abderrahim, S. and R. Maamri, A Category-theoretic Approach to Organization-based Modeling of Multi Agent Systems on the Basis of Collective Phenomena and Organizations in Human Societies. Informatica, 2018. 42(4). https://doi.org/10.31449/inf.v42i4.1282

[18] Ormandjieva, O., et al., Modelling multi-agent systems with category theory. Procedia Computer Science, 2015. 52: p. 538-545. https://doi.org/10.1016/j.procs.2015.05.031

[19] Pfalzgraf, J., On categorical and logical modeling in multiagent systems. Anticipative Predictive Models in Systems Science, 2005. 1: p. 93-98. On Categorical and Logical Modeling in Multiagent Systems ...

[20] Zhu, M., P. Grogono, and O. Ormandjieva., Using category theory to verify implementation against design in concurrent systems. Procedia Computer Science, 2015. 52: p. 530-537. https://doi.org/10.1016/j.procs.2015.05.030

[21] Kuang, H., et al. Formal Specification of Substitutability Property for Fault-Tolerance in Reactive Autonomic Systems. in SoMeT. 2010. DOI:10.3233/978-1-60750-628-7-357

[22] Pfalzgraf, J. and T. Soboll, On a general notion of transformation for multiagent systems and its implementation. Electronic Communications of the EASST, 2009. 12. DOI:10.14279/tuj.eceasst.12.259.267

Analysis of Deep Transfer Learning Using DeepConvLSTM for Human Activity Recognition from Wearable Sensors

Stefan Kalabakov
Department of Intelligent Systems, Jožef Stefan Institute, Jamova 39, Ljubljana, Slovenia
International Postgraduate School, Jamova 39, Ljubljana, Slovenia
E-mail: stefan.kalabakov@ijs.si

Martin Gjoreski
Faculty of Informatics, Università della Svizzera Italiana (USI), Lugano, Switzerland
E-mail: martin.gjoreski@usi.ch, martingjoreski.github.io

Hristijan Gjoreski
Faculty of Electrical Engineering, University Ss. Cyril and Methodius, Skopje, Macedonia
E-mail: hristijang@feit.ukim.edu.mk, dis.ijs.si/hristijan

Matjaž Gams
Department of Intelligent Systems, Jožef Stefan Institute, Jamova 39, Ljubljana, Slovenia
E-mail: matjaz.gams@ijs.si, dis.ijs.si/mezi

Keywords: transfer learning, deep learning, human activity recognition, accelerometer data

Received: July 15, 2021

Human Activity Recognition (HAR) from wearable sensors has gained significant attention in the last few decades, largely because of the potential healthcare benefits. For many years, HAR was done using classical machine learning approaches that require the extraction of features. With the resurgence of deep learning, a major shift happened and at the moment, HAR researchers are mainly investigating different kinds of deep neural networks. However, deep learning comes with the challenge of having access to large amounts of labeled examples, which in the field of HAR is considered an expensive task, both in terms of time and effort. Another challenge is the fact that the training and testing data in HAR can be different due to the personal preferences of different people when performing the same activity. In order to try and mitigate these problems, in this paper we explore transfer learning, a paradigm for transferring knowledge from a source domain, to another related target domain. More specifically, we explore the effects of transferring knowledge between two open-source datasets, the Opportunity and JSI-FOS datasets, using weight-transfer for the DeepConvLSTM architecture. We also explore the performance of this transfer at different amounts of labeled data from the target domain. The experiments showed that it is beneficial to transfer the weights of fewer layers, and that deep transfer learning can perform better than a domain-specific deep end-to-end model in specific circumstances. Finally, we show that deep transfer learning is a viable alternative to classical machine learning approaches as it produces comparable results and does not require feature extraction.

Povzetek: V prispevku je raziskan vpliv števila učnih primerov pri prenesenem učenju, ki kaže izboljšano delovanje pri malem številu primerov.

1 Introduction

With numerous healthcare, smart-home and security applications on the horizon, human activity recognition (HAR) is a field which has gained significant traction in the past two decades. Most of the research done on this topic has been aimed at understanding human activity using data from wearable sensors, primarily inertial measurement units (IMUs). This is in large part due to the rapid development of wearable devices, allowing researchers to perform complex tasks on them, as well as their ubiquitous and unobtrusive nature.

Until recently, researchers in the field of HAR were mainly focused on using traditional pattern-matching methods as well as machine learning to detect activities [1]. In order to work properly, these methods require the extraction of features which in turn requires considerable domain knowledge in order to extract a diverse and information-rich feature set. However, in recent years, as the benefit of using deep neural networks is apparent in domains such as computer vision and NLP [2], research has shifted towards deep learning [3]. The focus has mostly been centered around Convolutional Neural Networks (CNNs). These networks are capable of automatically capturing hierarchical feature representations of the data [4], i.e. they produce features which range from general to application specific as one goes deeper in the network. Another emerging trend in HAR is the use of LSTM cells which are able to better capture the temporal dependencies between sensor readings [5][6]. Finally, an interesting approach which yields arguably the best result is the creation of hybrid models such as the one proposed by Ordóñez et al., where they combine convolutional layers with LSTM cells in order to exploit both the spatial and the temporal analysis which those types of layers provide [7].

Nevertheless, as is the case in many other fields where it is applied, aside from the benefits, deep learning also brings certain challenges. These challenges are usually even more emphasized when working on HAR as opposed to fields such as image classification. For example, large amounts of (diverse) data are required to train a deep end-to-end classifier that can accurately predict human activities. These large amounts of data are usually difficult to collect as it takes a lot of time and sometimes money to do so. In addition, deep neural networks require a lot of time to train in order to reach their full potential, which hampers the ability to quickly create prototypes that can be further built upon. Finally, the source and target data in HAR can be very different, as different users perform the same activities differently depending on their personal preferences. This makes building end-to-end deep learning models a difficult task.

Given these challenges, it is clear that providing a solution to them could accelerate the development of HAR models for specific activity domains and make these models more adaptable to users and data that they have not previously seen. To this end, we explore transfer learning, a learning paradigm that deals with transferring knowledge acquired in one (source) domain to another related (target) domain, as a method that could help mitigate these issues.

There have been several works in the past which showed transfer learning to be beneficial in the HAR domain, but most of them used classical ML methods [8][9][10]. An extensive analysis of conventional transfer learning methods can be found in [11]. However, in contrast to this, there aren't many works addressing deep transfer learning. Morales et al. presented the pioneering deep transfer learning approach for HAR in [12]. In this paper the authors worked with the PAMAP2 and Skoda Mini Checkpoint datasets and investigated the transfer of weights of the DeepConvLSTM model between the two domains. Unfortunately, they get negative transfer results and conclude that the two domains are just too different from each other. In addition, Hoelzemann et al. have a similar transfer learning setup to the one presented in [12] and in their experiments they conclude that transfer between sensor locations in the same domain is feasible, but transfer between datasets is accompanied with significant performance losses. The authors in [13] propose a method which is able to achieve good results when transferring between HAR datasets which have the same set of tasks. Finally, in our previous work [14] we showed promising results for a transfer learning system using the MultiResNet architecture [15] at different adaptation set sizes.

In order to provide more detailed insights into some of the open questions, this paper is going to explore the performance of a transfer learning system, using two intuitively similar datasets which consist of activities of daily living (ADL). The deep learning architecture we are going to be using in this work is the DeepConvLSTM architecture. In more detail, we are going to explore: (i) the performance of transfer learning when transferring the weights of different numbers of convolutional layers; (ii) the performance of transfer learning when using different sizes of labeled adaptation sets; (iii) how transfer learning performs in comparison to domain-specific classical machine learning approaches and domain-specific end-to-end learning.

The rest of this paper is organized as follows. In Section 2 we introduce the datasets which are used in our experiments. Section 3 describes the preprocessing and feature extraction steps which were performed on the raw signal data before it was presented to the algorithms. The following section (Section 4) briefly describes the deep learning architecture that we chose to use. Following that, in Section 5 we give an introduction into our experimental setup, and in Section 6 we present and discuss the results. Finally, our work is concluded in Section 7.

2 Datasets

For the experiments we chose two datasets that are intuitively similar to each other. Both the Opportunity [16] and the JSI-FOS dataset [17][18], consist of activities which users commonly perform in their daily routines. Both of these datasets contain at least one 3D accelerometer worn on the right wrist, which allows us to discard the sensor modality and location as variables in our analysis. Another important characteristic of these datasets is the fact that they consist of data from several different users, which allows us to test the generalization capabilities of our models by using a Leave-One-Subject-Out (LOSO) evaluation.

Although they have a lot of similarities, the JSI-FOS and Opportunity datasets differ in several areas: (i) the number of activities, with JSI-FOS having 18 and Opportunity having 21 distinct activities (reduced to 10 and 14 after the preprocessing steps described in Section 3); (ii) sampling rate, which is equal to 50Hz and 30Hz, respectively; (iii) the overall duration of the data, which is around 3 times larger in the JSI-FOS dataset and amounts to around 20 hours (after the preprocessing steps described in Section 3). Furthermore, the two datasets also differ slightly in the types of activities, with Opportunity focusing on gestures and not just locomotion activities. Figure 1 and Figure 2, show the distribution of the activities we selected from both datasets.

Finally, a summary of the information about both datasets is given in Table 1.

Dataset	Type	#Subjects	Sampling rate	#Activities	#Selected activities	# of examples
JSI-FOS	ADL	10	50Hz	18	10	36060 [20h]
Opportunity	ADL	4	30Hz	21	14	10822 [6h]

Table 1: Overview of the two datasets used in our experiments

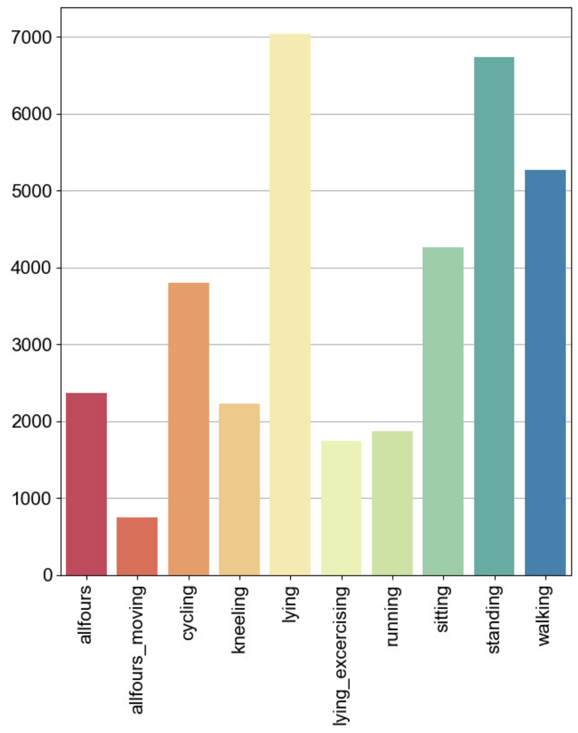


Figure 1: The number of examples per selected activity in the JSI-FOS dataset.

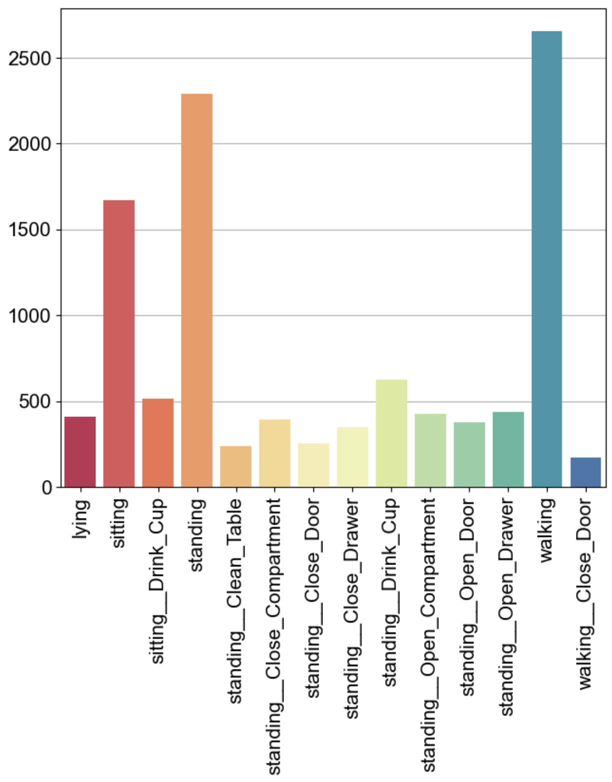


Figure 2: The number of examples per selected activity in the Opportunity dataset.

3 Preprocessing and feature extraction

In order to unify the way data is represented in both datasets, we constructed a fairly simple preprocessing pipeline. The first step in this pipeline is the selection of data which comes from a 3D accelerometer, worn on the right wrist. In addition to the three channels of data that these accelerometers produce, we also calculate the magnitude as a virtual fourth channel. Although it is available, we disregarded data from other sensors in order to simplify the model and make the analysis easier.

The second step in the preprocessing pipeline is the resampling of the data from the accelerometers to a sampling rate of 25Hz. We chose this relatively low sampling rate to make our transfer learning setup more suitable for potential use with wearable devices since a lower sampling rate results in better energy efficiency. Furthermore, the authors in [15] show that there are no significant performance differences when using sampling rates between 25Hz and 100Hz. The data from both datasets is downsampled to a common sampling rate to ensure that the filters we transfer from the source model to the target model work as intended. Otherwise, if the source and target data have different sampling frequencies, each of the transferred convolutional filters would work on a piece of data that has a different temporal length in comparison to what it was trained on. Next, we also convert the units of the measurements to a common unit, "g" (9.81 m/s^2). After we take care of the raw data format, we turn our attention to the activities in each dataset. In the JSI-FOS dataset, we only perform some simple aggregations of the original activities. For example, the activities *lying_back*, *lying_left_side*, *lying_right_side*, and *lying_stomach* are all aggregated to one activity, *lying*. This is why, from the original 18 activities, we end up selecting only 10 distinct ones. Table 2 shows all the aggregations used for this dataset.

On the other hand, when working with the Opportunity dataset, whenever a locomotion and gesture label are simultaneously available we create a new activity label. This new activity is simply the concatenation of the locomotion

Original	Aggregation
lying_back	lying_back
lying_left_side	
lying_right_side	
lying_right_stomach	
allfours	allfours
allfours_still	
standing	standing
standing_leaning_still	
transition_up	Null
transition_down	

Table 2: Aggregations of activities used for the JSI-FOS dataset

Original	Aggregation
*_open_door1	*_open_door
*_open_door2	
*_close_door1	*_close_door
*_close_door2	
*_open_fridge	*_open_compartment
*_open_dishwasher	
*_close_fridge	*_close_compartment
*_close_dishwasher	
*_open_drawer[1/2/3]	*_open_drawer
*_close_drawer[1/2/3]	*_close_drawer

Table 3: Aggregations of activities used for the Opportunity dataset

and gesture label, for example, walking (locomotion) while drinking from a cup (gesture). After this, we perform the same aggregation process as with JSI-FOS. Table 3 shows the aggregation rules for the Opportunity dataset. An asterisk is used in this table as a placeholder for a potential locomotion label.

The penultimate step of the pipeline is segmenting the raw data into windows of fixed size. Each window in this work contains 100 sensor readings, which represents 4 seconds of data at a sampling frequency of 25Hz. There is a 50% overlap between two windows. Windowing is performed for each channel separately, which means that after this step, both datasets are represented as sets of quadruples (4 channels).

Finally, as the last step, from both datasets we remove the windows with a Null activity label and disregard all activities with fewer than 100 windows (3.3 seconds). At this point, the preprocessing steps for the DeepConvLSTM model end, and the quadruples, stacked vertically, can be fed into the model in order to be processed.

3.1 Feature extraction

In order to be able to use classical machine learning algorithms we need to further change the form of the aforementioned windows by extracting features. In order to provide the algorithms with an information-rich representation, from each quadruple of windows (4 channels), we extract around 2400 features based on the related work on HAR. The majority of the features come from the TSFRESH package, which allows for the extraction of general-purpose time-series features. On top of the features extracted with TSFRESH, we also extracted a set of frequency-domain features which was previously shown to work well in other HAR applications [19][20]. This set of features is based on the Power Spectral Density (PSD) of the signal and include its binned distribution, entropy, energy, magnitude, and first four statistical moments of the PSD, among others.

4 Model architecture

In this work we chose to use the DeepConvLSTM framework, proposed by Ordóñez et al. in [7]. This architecture consists of stacked convolutional layers which are followed by LSTM cells. This allows the network to extract hierarchical feature representations and model the temporal dependencies between them. The network architecture was chosen primarily for its simplicity, which allows for an easier evaluation of how some changes affect the transfer learning performance, as well as its frequent use in other deep transfer learning studies in the field of HAR [12] [21].

The input, in our implementation of the network, is expected to consist of 4 stacked windows of signal data (one per sensor channel). This input is then processed through 4 convolutional layers, each with 64 feature maps. The convolutional layers use the ReLU activation function to compute their output.

Following them are two pairs of dropout and LSTM layers. Each dropout layer has a rate of 0.5 and each LSTM layer has 128 cells. Finally, a softmax layer is attached to the last LSTM layers in order to produce the final predictions. A diagram of this architecture can be seen on Figure 3.

5 Experimental setup

In this paper we adopt the following transfer learning approach: (i) train a (source) model on the Opportunity dataset; (ii) transfer the weights of all layers (except the softmax layer) of that model to a new (target) model in which the softmax layer fits the number of classes of the JSI-FOS dataset; (iii) freeze a certain number of convolutional layers and allow all the rest to be fine-tuned; (iv) fine-tune the rest of the layers using some number of instances (adaptation set) from the JSI-FOS dataset. Both the source and target models in this approach are trained using a batch size of 64 and a learning rate of 0.001. However, there is a difference in the number of training epochs between the source and target models and those numbers of epochs are

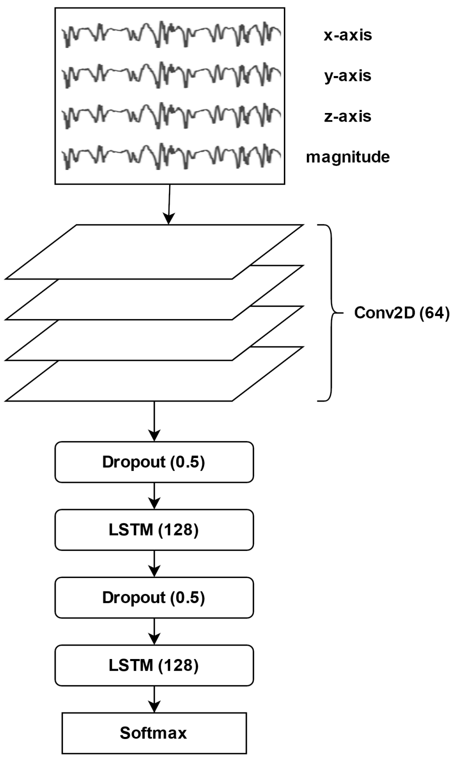


Figure 3: A diagram of our implementation of the DeepConvLSTM architecture.

100 and 70, respectively. These numbers were determined experimentally.

Another important detail to explain, before we move on to the experiments, is the adaptation set. This is the set of instances from the JSI-FOS dataset we use to either train or fine-tune a model. Since we wanted to explore the efficacy of transfer learning at different amounts of labeled data from the target domain, we repeat each of our experiments several times, using different sizes of the adaptation set. The adaptation set sizes range from 100 instances to 12000 instances, that is, between 3.33 minutes and 6.66 hours of labeled data.

Furthermore, it is important to note that, since we use Leave-One-Subject-Out (LOSO) evaluation, the adaptation set is produced in a stratified manner, using all subjects except for the one which is selected for testing and the two subjects (randomly chosen from the remaining set of subjects) which are selected for validation. This means that an adaptation set of 100 instances, will be produced at least 10 times, once in each iteration of the LOSO evaluation. In order to make the results more relevant and minimize randomness, we repeat the LOSO evaluation several times at each adaptation set size and report the average of these results. For example, given an adaptation set of 100 instances, we repeat the LOSO evaluation 4 times, which means that a random adaptation set will be produced a total of 40 times (JSI-FOS has 10 subjects).

Finally, we should note that when we train models on the target dataset (JSI-FOS) we use early stopping based on the loss value on the validation set (two randomly selected train-users in each iteration).

Experiment 1
The first experiment is aimed at exploring the optimal number of convolutional layers to freeze in step (iii) of the transfer learning approach. To this end, we repeat the transfer learning approach 4 times, and in each of them we freeze a different number of convolutional layers, ranging from 1 to 4.

Experiment 2
The second experiment is aimed at comparing the performance of deep transfer learning, deep end-to-end models and classical ML. As an example of a classical ML algorithm we chose Random Forest, as it often shows state-of-the-art results and does not require extensive hyper-parameter tuning [15]. This model is trained only on the instances from the adaptation set, using the features extracted in Section 3.1. The end-to-end model is also trained using only the examples in the adaptation set and uses the same architecture as the transfer learning model, but its weights were initialized randomly and no transfer of knowledge has taken place. Finally, the transfer learning model is trained using steps (i) through (iv) and the number of convolutional layers transferred between models is based on the results from the first experiment.

6 Results and discussion

The results from ***experiment 1*** can be seen on Figure 4. The x-axis of that graph, shows the number of instances in the adaptation set, while the y-axis of the graph, shows the macro F1-score. Based on the results from this experiment, it seems that there isn't a huge difference in performance when freezing different numbers of convolutional layers from the DeepConvLSTM architecture. However, the setup in which we only freeze the first convolutional layer and allow all others to be fine-tuned, performs marginally, but consistently, better than the rest. This finding seems to be in line with what was concluded by [12] and supports the claim that convolutional layers deeper in the model, extract features which are just too dataset (domain) specific.

The results from ***experiment 2*** can be seen on Figure 5. Here we compare the performance of a RF classifier, an end-to-end (E2E) DeepConvLSTM model and a DeepConvLSTM model trained using transfer learning. As is expected, the E2E model shows very poor performance when the adaptation set size is very low, and gradually, improves as the adaptation set grows. It is interesting to note that although it comes close, the E2E model never really matches the performance of the Random Forest (RF) classifier. On the other hand, the RF classifier produces strong results on all adaptation set sizes, except the smallest one. This is probably due to the fact that relevant features were extracted by hand. Lastly, it is interesting to see that the DeepConvLSTM model trained using transfer learning produces

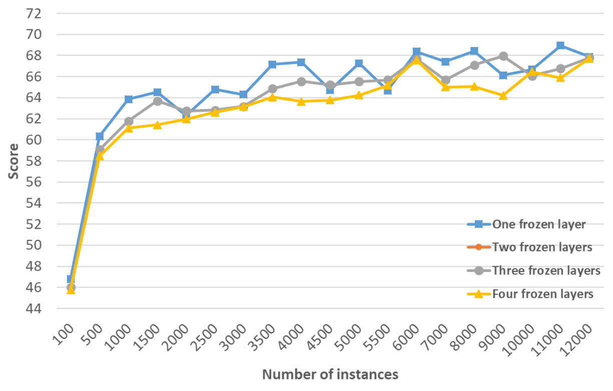


Figure 4: Performance of the DeepConvLSTM architecture when freezing different numbers of transferred layers.

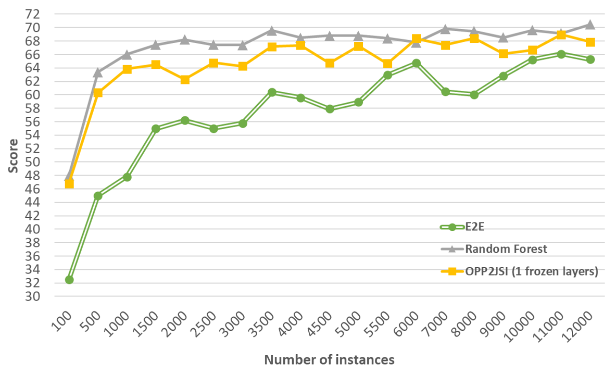


Figure 5: A comparison between the performances of a classical ML algorithm, an end-to-end deep learning model and a transfer learning model, at different adaptation sizes.

results which are quite similar to the ones produced by the RF classifier and that it manages to beat the results of the E2E model across all adaptation set sizes. This seems to support the idea that relevant features were already extracted in the first convolutional layer (trained on the source domain) and that even small adaptation sets contain enough data for the model to make sense of those features.

Finally, Figure 6 shows a per activity comparison between the performances of the deep E2E model and the deep transfer learning model. Each row represents a different adaptation set size, while the columns represent the different activities in the target dataset (JSI-FOS). The value in each of the cells, is the difference in activity F1-score between the deep transfer learning model and the domain-specific deep E2E model. This produces positive values, whenever the deep transfer learning model is better and negative values whenever the opposite is true. As we can see, there are very few situations in which the deep E2E model manages to perform better than the deep transfer learning model, which is to be expected based on the results shown on Figure 5. This figure also, quite clearly shows the gradual decline in performance gains (as we increase the size of the adaptation set) for the *allfours*, *allfours_moving*, *cycling*, *standing*, *walking* activities.

7 Conclusion

In this paper we use the DeepConvLSTM architecture to explore the benefits of transferring knowledge (represented by model weights) from the Opportunity dataset, to the JSI-FOS dataset. Unlike in several previous works, we explore transfer learning between datasets which come from intuitively similar domains and both contain activities from the daily lives of users. In this work we aim to create a head-to-head comparison of classical ML, end-to-end deep learning and deep transfer learning. From the results, we can conclude that it is better to transfer the weights of fewer convolutional layers, as there was already extracted a set of diverse features which only get more domain specific as we transfer more layers. Furthermore, we also show that deep transfer learning is able to produce better results in comparison to a deep end-to-end model trained on the same amount of labeled data. Finally, we show that with the use of deep transfer learning one can produce results comparable to those of a RF classifier without the need for feature extraction done by hand.

Acknowledgement

Part of the study was supported by the project WideHealth - Horizon 2020, under grant agreement No 95227. Additionally, Stefan Kalabakov would like to thank the Slovene Human Resources Development and Scholarship Fund (Ad futura) for the support.

8 References

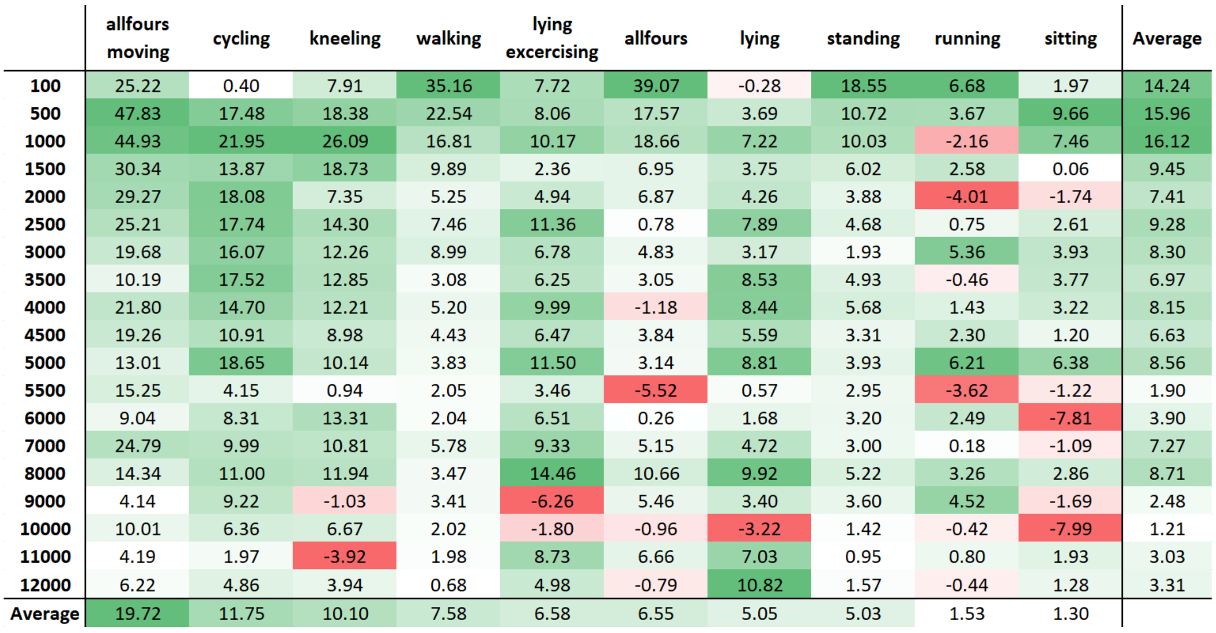
References

[1] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE communications surveys & tutorials*, vol. 15, no. 3, pp. 1192–1209, 2012.
https://doi.org/10.1109/SURV.2012.110112. 00192.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
https://doi.org/10.1038/nature14539.

[3] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A survey," *Pattern Recognition Letters*, vol. 119, pp. 3–11, 2019.
https://doi.org/10.1016/j.patrec.2018.02.010.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
https://doi.org/10.1109/5.726791.



	allfours moving	cycling	kneeling	walking	lying excercising	allfours	lying	standing	running	sitting	Average
100	25.22	0.40	7.91	35.16	7.72	39.07	-0.28	18.55	6.68	1.97	14.24
500	47.83	17.48	18.38	22.54	8.06	17.57	3.69	10.72	3.67	9.66	15.96
1000	44.93	21.95	26.09	16.81	10.17	18.66	7.22	10.03	-2.16	7.46	16.12
1500	30.34	13.87	18.73	9.89	2.36	6.95	3.75	6.02	2.58	0.06	9.45
2000	29.27	18.08	7.35	5.25	4.94	6.87	4.26	3.88	-4.01	-1.74	7.41
2500	25.21	17.74	14.30	7.46	11.36	0.78	7.89	4.68	0.75	2.61	9.28
3000	19.68	16.07	12.26	8.99	6.78	4.83	3.17	1.93	5.36	3.93	8.30
3500	10.19	17.52	12.85	3.08	6.25	3.05	8.53	4.93	-0.46	3.77	6.97
4000	21.80	14.70	12.21	5.20	9.99	-1.18	8.44	5.68	1.43	3.22	8.15
4500	19.26	10.91	8.98	4.43	6.47	3.84	5.59	3.31	2.30	1.20	6.63
5000	13.01	18.65	10.14	3.83	11.50	3.14	8.81	3.93	6.21	6.38	8.56
5500	15.25	4.15	0.94	2.05	3.46	-5.52	0.57	2.95	-3.62	-1.22	1.90
6000	9.04	8.31	13.31	2.04	6.51	0.26	1.68	3.20	2.49	-7.81	3.90
7000	24.79	9.99	10.81	5.78	9.33	5.15	4.72	3.00	0.18	-1.09	7.27
8000	14.34	11.00	11.94	3.47	14.46	10.66	9.92	5.22	3.26	2.86	8.71
9000	4.14	9.22	-1.03	3.41	-6.26	5.46	3.40	3.60	4.52	-1.69	2.48
10000	10.01	6.36	6.67	2.02	-1.80	-0.96	-3.22	1.42	-0.42	-7.99	1.21
11000	4.19	1.97	-3.92	1.98	8.73	6.66	7.03	0.95	0.80	1.93	3.03
12000	6.22	4.86	3.94	0.68	4.98	-0.79	10.82	1.57	-0.44	1.28	3.31
Average	19.72	11.75	10.10	7.58	6.58	6.55	5.05	5.03	1.53	1.30	

Figure 6: A per activity comparison of the F1-scores achieved by the end-to-end model and the model trained using transfer learning. Each cell shows the difference between the F1-score achieved but the transfer learning model and the score achieved by the end-to-end model. A positive value means that the transfer learning model performed better, while a negative one suggest better performance by the end-to-end model.

[5] N. Y. Hammerla, S. Halloran, and T. Plötz, "Deep, convolutional, and recurrent models for human activity recognition using wearables," *arXiv preprint arXiv:1604.08880*, 2016.

[6] Y. Guan and T. Plötz, "Ensembles of deep lstm learners for activity recognition using wearables," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 2, pp. 1–28, 2017.
https://doi.org/10.1145/3090076.

[7] F. J. Ordóñez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.
https://doi.org/10.3390/s16010115.

[8] A. Calatroni, D. Roggen, and G. Tröster, "Automatic transfer of activity recognition capabilities between body-worn motion sensors: Training newcomers to recognize locomotion," in *Eighth international conference on networked sensing systems (INSS'11)*. Eighth International Conference on Networked Sensing Systems (INSS'11), 2011.

[9] M. Kurz, G. Hölzl, A. Ferscha, A. Calatroni, D. Roggen, and G. Tröster, "Real-time transfer and evaluation of activity recognition capabilities in an opportunistic system," *machine learning*, vol. 1, no. 7, p. 8, 2011.

[10] S. Inoue and X. Pan, "Supervised and unsupervised transfer learning for activity recognition from simple in-home sensors," in *Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2016, pp. 20–27.
https://doi.org/10.1145/2994374.2994400.

[11] D. Cook, K. D. Feuz, and N. C. Krishnan, "Transfer learning for activity recognition: A survey," *Knowledge and information systems*, vol. 36, no. 3, pp. 537–556, 2013.
https://doi.org/10.1007/s10115-013-0665-3.

[12] F. J. O. Morales and D. Roggen, "Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations," in *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, 2016, pp. 92–99.
https://doi.org/10.1145/2971763.2971764.

[13] J. Wang, V. W. Zheng, Y. Chen, and M. Huang, "Deep transfer learning for cross-domain activity recognition," in *proceedings of the 3rd International Conference on Crowd Science and Engineering*, 2018, pp. 1–8.
https://doi.org/10.1145/3265689.3265705.

[14] M. Gjoreski, S. Kalabakov, M. Luštrek, M. Gams, and H. Gjoreski, "Cross-dataset deep transfer learning for activity recognition," in *Adjunct Proceedings of the 2019 ACM International Joint Conference on*

Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers, 2019, pp. 714–718.
https://doi.org/10.1145/3341162.3344865.

[15] M. Gjoreski, V. Janko, G. Slapničar, M. Mlakar, N. Reščič, J. Bizjak, V. Drobnič, M. Marinko, N. Mlakar, M. Luštrek *et al.*, "Classical and deep learning methods for recognizing human activities and modes of transportation with smartphone sensors," *Information Fusion*, vol. 62, pp. 47–62, 2020.
https://doi.org/10.1016/j.inffus.2020.04.004.

[16] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha *et al.*, "Collecting complex activity datasets in highly rich networked sensor environments," in *2010 Seventh international conference on networked sensing systems (INSS)*. IEEE, 2010, pp. 233–240.
https://doi.org/10.1109/INSS.2010.5573462.

[17] H. Gjoreski, B. Kaluža, M. Gams, R. Milić, and M. Luštrek, "Context-based ensemble method for human energy expenditure estimation," *Applied Soft Computing*, vol. 37, pp. 960–970, 2015.
https://doi.org/10.1016/j.asoc.2015.05.001.

[18] S. Kozina, H. Gjoreski, M. Gams, and M. Luštrek, "Three-layer activity recognition combining domain knowledge and meta-classification," *Journal of Medical and Biological Engineering*, vol. 33, no. 4, pp. 406–414, 2013.
http://dx.doi.org/10.5405/jmbe.1321.

[19] V. Janko, M. Gjoreski, G. Slapničar, M. Mlakar, N. Reščič, J. Bizjak, V. Drobnič, M. Marinko, N. Mlakar, M. Gams *et al.*, "Winning the sussex-huawei locomotion-transportation recognition challenge," in *Human Activity Sensing*. Springer, 2019, pp. 233–250.
https://doi.org/10.1007/978-3-030-13001-5_15.

[20] X. Su, H. Tong, and P. Ji, "Activity recognition with smartphone sensors," *Tsinghua science and technology*, vol. 19, no. 3, pp. 235–249, 2014.
https://doi.org/10.1109/TST.2014.6838194.

[21] A. Hoelzemann and K. Van Laerhoven, "Digging deeper: towards a better understanding of transfer learning for human activity recognition," in *Proceedings of the 2020 International Symposium on Wearable Computers*, 2020, pp. 50–54.
https://doi.org/10.1145/3410531.3414311.

Routing Algorithm in Networks on the Globe

Sujit Kumar Bose
S.N. Bose National Centre for Basic Sciences, Kolkata, West Bengal, India
E-mail: sujitkbose1@gmail.com

Technical paper

Keywords: router network algorithm, geodesic, link cost, waiting time in queueing model

Received: June 16, 2020

Packet switching of data in networks is done by either the distance-vector or the link-state routing protocols. These protocols use the Bellman-Ford and the Dijkstra's algorithms respectively for the least cost path from a source base station to a destination station. For inter-network transmission, the path-vector routing protocol is in use. With progress of time, the network topologies are becoming huge in size, requiring large demand on book keeping of routing tables and transmission of the data packets dynamically to several other stations of the network by broadcast, increasing the load on the network. Here, assuming the router stations to be terrestrially located with links along the ground, a large network is assumed to lie on a spherical surface, and so the shortest geodesic path from source to destination becomes a great circular arc. For fast transmission, the cost of a link to a node is multicast to its neighboring nodes only for selection of the path lying as close as possible to the geodesic line between the source and the destination. As the arrival and dispatch of data packets at a nodal station occurs randomly, the cost of a link is estimated in this paper by the waiting time of a queueing process. This process at a router station is thus modeled by the Markovian $M/M/c$ model, where c is the number of servers at the router station. If other commercial fixed charge is involved for the use of a link, then that can be included in the total cost of a link. Finally, a method of search of a mobile destination is also presented using sphericity of the network. Algorithms for the near geodesic path, costs of links as waiting times and destination search in mobile environment are clearly presented.

Povzetek: Predstavljen je algoritem povezovanja v globalnih omrežjih.

1 Introduction

Global digital data transmission networks have become huge in size with passage of time, with ever changing topology. A data network architecture essentially consists of router ground stations as nodes connected by radio/microwave transmitters or preferably by fiberoptic cables along the ground/sea bed that form the links of the network. The essential function of the network is to deliver data packets without loss from a source node to a destination node of the network. The transmission may take place by arbitrary paths, but a fixed consistent path without loops, is preferable.

Routing algorithms at present, are mainly of three types: *distance-vector routing, link-state routing, and path-vector routing*. Essential features of these algorithms are as follows. In *distance-vector routing* each router periodically multicasts its knowledge to its neighbors covering the whole network. The procedure is dynamic and kept in the form of an updated table in each router. A table contains an assigned cost of each link of the router and the least-cost path between any two nodes calculated by the Bellman-Ford algorithm (Bellman [3], Ford [11], Cormen et. al. [9]). This information enables dispatch of data packets from source node to its destination node along the least-cost path. RIP (Routing Information Protocol) and IGRP (Interior Gateway Routing Protocol) are protocols of this type (Cisco Systems [7] and [8]), that are widely used as a protocol in the TCP/IP environment to route packets between gateways of the internet. The *link-state protocol* was designed to overcome some of the short comings of the distance-vector protocol. In the link-state procedure, initially a router defines the cost of each link and broadcast its information to all other nodes of the network. Data transmission from source to destination along the least-cost path is determined by Dijkstra's algorithm (Bose [4]). However, if a change occurs in the routing table of some node on the path, then all the intermediate nodes are notified accordingly. OSPF (*Open Shortest Path First*) is an elaborate algorithm that carries out this basic feature of link-state routing (Cisco Systems [6]. The distance-vector and link-state routing protocols are suitable for intra-domain autonomous systems. the former tends to become unsuitable for large number of hops to destination node, while the latter needs huge amount of resources to calculate routing tables, creating heavy traffic of router information. Path-vector routing on the other hand is useful for inter-domain routing of autonomous systems. It is similar to the distance-vector routing protocol, but differs in the sense that a node called the speaker node of each autonomous system acts on behalf of the entire system, creating its own routing table, and advertising it to its neighboring speaker nodes in the neighboring autonomous systems. In this way the destination addresses (and not the costs of links) together with the path descriptions to each node of the destinations is multicast to reach the final destination node. The path selection in a domain is based on routing metric, consisting of information like bandwidth, network delay, hop count, path cost etc. BGP (*Border Gateway Protocol*) belongs to this category.

The three types of routing protocols require voluminous book keeping at a router and transmission of information to other routers dynamically. The layers of protocol at a router in present day mobile environment is described in detail in Dutta and Schulzrinne [10]. A careful mathematical analysis of routing algorithms is given by Busch and Tirthapura [5]. A dynamic source routing algorithm in such Mobile Ad hoc Network (MANET) is presented in Jayakumar and Chellappan [12] using a new link cache structure maintaining source transparent route. An alternative algorithm is presented by Meghanathan [16] that uses a strategy of energy-efficient broadcast route discovery in the network density and mobility to determine stable routes of transmission. Other approaches such as consensus based networks and multicast pipelined network coding have been suggested in recent years by Arellano-Vazquez et. al. [2] and Li et. al. [13]. On the other hand Li et. al. [14] have presented routing schemes for optimal congestion control for multipath networks having non-congested packet lossess. An earlier investigation by Manvi et. al. [15] suggests a Mobile Agent based Routing (MAR) scheme with the objecives similar to RIP that uses a more flexible, adaptable and distributed mechanism. The present paper addresses the difficulties of data transmission from a geodetic point of view, by presenting algorithms for transmission dynamically, selecting router stations as close to the (shortest) geodesic path between the source and destination nodes on the spherical globe. As different group of data packets arrive at a nodal router through possibly multiple links of the network for forward dispatch, the traffic through the router is modeled as a Markovian $M/M/c$ queueing model, where c is the number of servers employed at the router. The waiting time at such a queue is well known (Bose [4]), and all that is required is to multicast it periodically to the nearest neighboring nodes. From a given node, the next hop is made to that nearest neighbor node, which points towards the destination node, provided the waiting time of that node is not prohibitive. In this way the destination node is reached in quick time, with minimal book keeping of waiting times at a node along the path. The waiting times at the nodes play the role of costs of links used in the standard protocols described earlier. Any fixed charge of a link, as in the case of undersea fiber optic cables may be added to the link cost, if necessary.

2 Selection of linking nodes

A network of stations on the globe is considered, as shown in figure $1\,(a)$. The source and the destination nodes are respectively considered as A_1 and A_n, with respective latitude and longitude $(\phi_1,\,\lambda_1)$ and $(\phi_n,\,\lambda_n)$, which determine the geographical position of the two nodes. The node $A_i(\phi_i,\,\lambda_i)$ on the desired path has a nearest neighbor $A_j(\phi_j,\,\lambda_j)$ that makes the angle $\chi_k = \angle A_kA_iA_n$ the least possible with a favorable waiting time W_k, where $A_k(\phi_k,\,\lambda_k)$ is any node in the neighborhood of A_i. These nodes are shown separately in figure $1(b)$. In the spherical triangle A_iA_kN, where N is the north pole of the earth, $\angle A_kNA_i$ = difference of latitude of A_k and $A_i = \lambda_k - \lambda_i$, and arc A_iN = colatitude of $A_i = \pi/2 - \phi_i$, arc rA_kN =colatitude of $A_k = \pi/2 - \phi_k$. Let $r := \pi/2 - \phi_i,\ (0 \le r \le \pi)$, and $\alpha := \angle NA_iA_n$, then from the spherical triangle NA_iA_k (Abramowitz and Stegun [4], p. 79)

$$\frac{\sin(\alpha - \chi_k)}{\cos\phi_k} = \frac{\sin(\lambda_k - \lambda_i)}{\sin r} \tag{4}$$

and

$$\cos r := \sin\phi_i\,\sin\phi_k + \cos\phi_i\,\cos\phi_k\,\cos(\lambda_k - \lambda_i) \tag{5}$$

Eq. (4) yields

$$\alpha - \chi_k = arcsin\left[\frac{\cos\phi_k\,\sin(\lambda_k - \lambda_i)}{\sin r}\right] \tag{6}$$

where $\sin r = \sqrt{1-\cos^2 r}$ and can be determined from Eq. (5). Similarly in the spherical triangle NA_iA_n, let $a :=$ arcA_iA_n $(0 \le a \le \pi)$, then since $\angle A_iNA_n = \lambda_n - \lambda_i$ and arc $NA_n = \pi/2 - \phi_n$,

$$\alpha = arcsin\left[\frac{\cos\phi_n\,\sin(\lambda_n - \lambda_i)}{\sin a}\right] \tag{7}$$

where

$$\cos a = \sin\phi_i\,\sin\phi_n + \cos\phi_i\,\cos\phi_n\,\cos(\lambda_n - \lambda_i) \tag{8}$$

so that $\sin a = \sqrt{1-\cos^2 a}$. Using Eq. (7), Eq. (6) yields the angle χ_k in terms of the latitudes and longitudes $(\phi_k,\,\lambda_k),\ (\phi_i,\,\lambda_i),\ (\phi_n,\,\lambda_n))$ of the nodes A_k, A_i and A_n respectively. If as before, p is the priority assigned to $|\chi_k|$, the goal is to minimize the objective function

$$z_k = p\,|\chi_k| + (1-p)\,\frac{W_k}{W_{max}} \tag{9}$$

where W_k is the waiting time at A_k, normalized by maximum allowable time W_{max}.

3 Waiting time at a node

A router at a node k is assumed to consist of c number of servers. The data packets of information arrive at the router

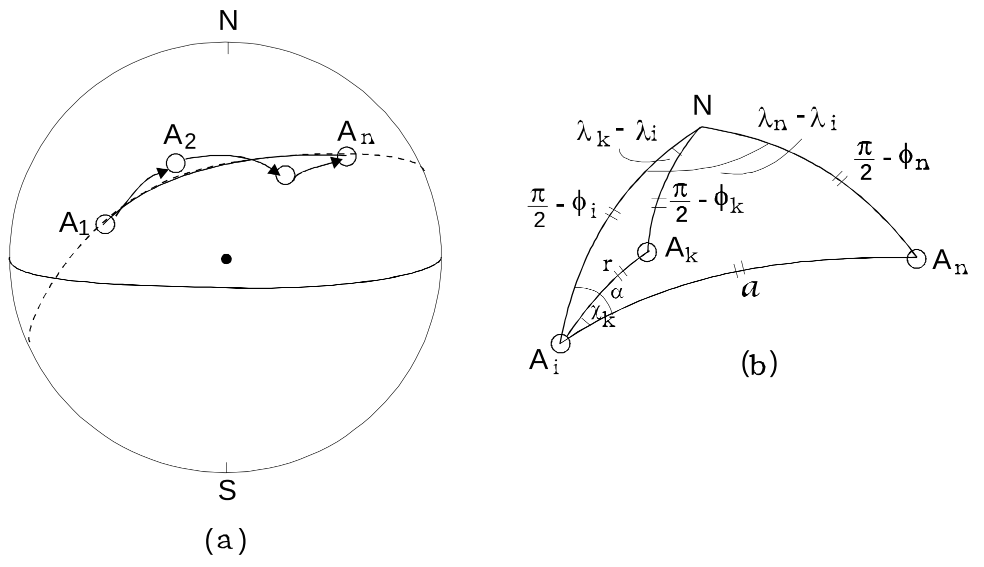


Figure 1: (a) Path $A_1A_2 \cdots A_n$ on spherical earth. (b) Configuration of nodes A_i, A_k, andA_n.

at random times, and processed by one of the servers, requiring random serving time and forwarded through a desired link. The serving times are random as the servers of the router may have different specifications and the processing time of a packet for screening by different layers of the router protocol may be different. Thus data forwarding from this point of view is a queueing system that can be modeled by the Markovian $M/M/c$ model (Bose [4], pp. 237-239). According to the model, if λ_k = average number of data packets arriving at the router through different links, and μ_k = average service rate of data packets of one server of the router, then the waiting time W_k is given by

$$W_k = L_k/\lambda_k \tag{10}$$

where L_k is the queue length given by the expression

$$L_k = \frac{\rho}{(1-\rho)^2}\frac{(c\rho)^c}{c!} \Big/ \Big[\sum_{n=0}^{c}\frac{(c\rho)^n}{n!} + \frac{c^c}{c!}\frac{\rho^{c+1}}{1-\rho}\Big] \tag{11}$$

In Eq. (11), $\rho = \lambda_k/c\mu_k$ is the traffic intensity at the router, and must be such that $\rho < 1$ for traffic flow. Otherwise if $\rho \geq 1$, the queue will grow blocking the router altogether.

It is to be noted that the waiting time at a router station will vary during the course of a day, requiring periodic upgradation and shared accordingly with the nodes of the network.

4 The algorithms

The method developed in sections 2 leads to the following pseudo-code for fast transmission of data through terrestrial networks.

Algorithm 1. Fast Data Transmission Path

```
1. Input: $\phi 1, \lambda 1, \phi n, \lambda n$;   \\ Latitude, Longitude of Source and Destination.
          $p, W_{max}$   \ \ Priority of deviation from geodesic path, and maximum permitted waiting time.

2. Output: $\phi[\ ], \lambda[\ ]$   \\ Latitude, Longitude of intermediate nodes $i = 2, 3, \cdots, n-1$.

3. $i \leftarrow 1$
   $\phi[1] \leftarrow \phi 1$;  $\lambda[1] \leftarrow \lambda 1$

4. $imax \leftarrow$ Number of stations in the neighbourhood of node $i$.
   $\cos a \leftarrow \sin\phi[i] \sin\phi n + \cos\phi[i] \cos\phi n \cos(\lambda n - \lambda[i])$;  $\sin a \leftarrow \sqrt{1-\cos^2 a}$

5. for $k \leftarrow 1$ to $imax$
     $\phi[k]; \lambda[k] \leftarrow$ (Latitude, Longitude) of stations in the neighbourhood of node $i$ (must be known).
     $\cos r \leftarrow \sin\phi[i] \sin\phi[k] + \cos\phi[i] \cos\phi[k] \cos(\lambda[k] - \lambda[i])$
     $\sin r \leftarrow \sqrt{1-\cos^2 r}$
     $\chi[k] \leftarrow |\arcsin\{\cos\phi n\ \sin(\lambda n - \lambda[i]/\sin a\}$
             $- \arcsin\{\cos\phi[k]\ \sin(\lambda[k] - \lambda[i])/\sin r\}|$
       \\ angle of node $k$ with respect to node $n$.
     $W[k] \leftarrow$ Waiting time at base station $k$ (From Algorithm 2).
     if $(W[k] > W_{max})$ exit \\ Blocked Link.
     $z[k] \leftarrow p\,\chi[k] + (1-p)\,W[k]/Wmax$ \ \ Objective function to be minimised.
```

```
end for
6. for $k \leftarrow 1$ to $imax - 1$    \\ Sort angles
                                   $\chi[k]$ to avoid ties.
    for $l \leftarrow k + 1$ to $imax$
    if $(\chi[k] > \chi[l])$ then
        temp $\leftarrow \chi[k]$; $\chi[l] \leftarrow \chi[k]$; $\chi[l] \leftarrow$temp
      end if
   end for
    end for
7. $kmin \leftarrow 1$
   for $k \leftarrow 2$ to $imax$
       if $(z[kmin] > z[k])$   $kmin \leftarrow k$
     end for
8. $i \leftarrow kmin$   \\   Next hop to node.
9. if $(i = n - 1)$   stop
10. Go To Step 4.
11. end
```

The waiting time pseudo-code following Eqs. (10) and (11) required in Algorithm 1 is:

Algorithm 2. Waiting Time following $M/M/c$ Queueing Model

```
1. Input: $\lambda,\ \mu,\ c$   \\   $\lambda$ = Average number of arrival of data packets at a Station,
              $\mu$ = Average Service Rate of data packets by one Server.
              $c$ = Number of Servers in parallel at a Station.
2. Output: $W[\ ]$   \\   Waiting Time at the Station.
3. $\rho \leftarrow \lambda/(c\mu)$   \\   Traffic Intensity at the Station.
    if $(\rho \geq 1)$ return   \\   Station is blocked.
4. $p_0 \leftarrow 0$
      for $n \leftarrow 0$ to $c$
          $p_0 \leftarrow p_0 + (c\rho)^n/n!$
       end for
    $p_0 \leftarrow 1/\left(p_0 + \frac{c^c}{c!}\frac{\rho^{c+1}}{1-\rho}\right)$
    $W[\ ] \leftarrow \frac{\rho}{(1-\rho)^2}\frac{(c\rho)^c}{c!}\frac{p_0}{\lambda}$
5. return
6. end
```

5 Locating destination node

In case the destination node is not known precisely, as in a mobile environment, its location can be determined by adopting a parsimonious method of flooding the search in a restricted zone of terrestrial links whose pole is the source node A_1. In the method, it is assumed that the mobile device possesses some *unique label*, named here as "TARGET_DEVICE". With A_1 as a pole, the spherical surface is first divided in to "latitudinal" strips of width 1^o, the angle subtended at the center of the sphere. The search is carried out starting from the innermost strip outwards, until the destination node of the "TARGET_DEVICE" is detected. For further restricting the search area, the strips are bound on the two sides by "longitudinal" great circles of vertical angle $\pi/8$. This means that the polar region is divided in to eight equal sectors.

With A_1 as pole, a node A_k having latitude-longitude $(\phi_k,\ \lambda_k)$ in the searching zone is suppose bounded by of inner and outer circles of "colatitudes" $(\psi_0,\ \psi_1)$, then it can be shown from elementary considerations that

$$\phi_1 + \psi_0 \leq \phi_k \leq \phi_1 + \psi_1 \tag{12}$$

where ϕ_1 is the latitude of A_1. For restricting the "longitudinal" boundary of the search zone, let the geographical meridian through A_1 be taken as the reference circle. If χ be the vertical angle at A_1 subtended by the node A_k with the reference circle, and ψ_k its "colatitude", then as as in section 2, the angle χ is given by the equations

$$\sin\chi = \frac{\sin(\lambda_k - \lambda_1)}{\sin\psi_k} \times \cos\phi_k \tag{13}$$

where

$$\cos\psi_k{=}\sin\phi_1\sin\phi_k + \cos\phi_1\cos\phi_k\cos(\lambda_k - \lambda_1) \tag{14}$$

The angle χ must then lie in the octant of search. The method described above leads to the following:

Algorithm 3. Location of Destination Node

```
1. Input: $\phi1,\ \lambda1$;   \\ Latitude, Longitude of Source Node.
2. Output: $\phi n, \lambda n$;   \\ Latitude, Longitude of Destination Node.
3. $\psi0 \leftarrow 0$
4. for $i \leftarrow 1$ to 180
   $\psi_1 = \leftarrow i * \pi/180$
   for $j \leftarrow 1$ to 8
   $\chi_0 \leftarrow 0$; $\chi_1 = \leftarrow \chi_0 + \pi/8$
   for $k \leftarrow 1$ to $kmax$;   \\ $kmax$ is max. number of nodes in the search zone.
   $\phi[k]$; $\lambda[k] \leftarrow$ Latitude, Longitude of the stations in $k$th search zone.
   $\cos(\psi k) \leftarrow \sin\phi_1\ \sin\phi[k]$
              $+ \cos\phi_1\ \cos\phi[k]\ \cos(\lambda[k] - \lambda 1)$
   $\sin(\psi k) \leftarrow \sqrt{1 - \cos^2(\psi k)}$
   $\chi \leftarrow \arcsin(\sin(\lambda[k] - \lambda 1) * \cos(\phi[k])/\sin\psi k)$
```

```
    while ($(\psi_0 \leq \phi[k] - \phi 1 \leq \psi_1$ &&
                                    $\chi_0 \leq \chi \leq \chi_1)$ do
    { if (DEVICE_LABEL ==
                          "TARGET_DEVICE")
    then  \\ TARGET_DEVICE                        located.
      $\phi n = \phi[k]; \lambda n = \lambda[k]$
      stop
      end if }
    end for
    $\chi_0 =\leftarrow \chi_1, \chi_1 \leftarrow \chi_0 + \pi/8$
    end for
    end for
5. end
```

6 Conclusion and future scope

Data transmission networks require large tables at the router stations for lossless steady transmission. Moreover, the information at a router node is dynamically shared by other nodes as well. In present day dynamic networks, the information sharing tends to increase by huge amounts. To mitigate the work load of the routers, a data transmission algorithm is presented here, in which transmission takes place from source to destination dynamically along a path as close as possible to the terrestrial geodesic joining the two nodes, taking in to account the costs of the intervening links. The cost of a link at a connecting node is estimated here by the waiting time at the node router according to the queueing $M/M/c$ model. All that is required is to share the information of the waiting times periodically among all the nodes of the network. The algorithm however does not preclude use of other methods of estimating the costs of the different links. An improvement in this direction could be to find a model that incorporates sudden unsteady bursts observed in actual data transmission in fiber cables, and raising the tail of the Markov queueing model. Finally, an algorithm is presented for searching the destination node, if it is unknown, such as in the case when the device is in a mobile environment.

7 Acknowledgement

The author is thankful to the S.N. Bose National Center for Basic Sciences, Kolkata for providing necessary facilities for undertaking this research. Special thanks also go to Dr. Sagar De for useful discussions and inputs.

8 References

References

[1] M. Abramowitz and I.A. Stegun (1972), Handbook of Mathematical Functions,Dover Publications, New York.

[2] M. Arellano-Vazquez, M. Benitez-Perez, J. Ortega-Arjono (2015), A consensus routing algorithm for mobile distributed systems, Int. J. of Distributed Sensor Networks, **11**, https://doi.org/10.1155/2015/510707.

[3] R.E. Bellman (1958), On a routing problem, Quart. Appl. Math. **16**, 87-90.

[4] S.K. Bose (2012), Operations Research Methods, Narosa Publishing, New Delhi.

[5] C. Busch, S. Tirthapura (2005), Analysis of Link Reversal Routing Algorithms, SIAM J. Comput. **35**, 305-326, https://doi.org/10.1137/S0097539704443598.

[6] Cisco Systems (2011), IOS IP Routing, RIP Configuration Guide.

[7] Cisco Systems (2005), An Introduction to IGRP.

[8] Cisco Systems (2011), IOS IP Routing, OSPF Command Reference.

[9] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein (2009), Introduction to Algorithms, McGraw-Hill, 651-655.

[10] A. Dutta, H. Schulzrine (2014), Mobility Protocols and Handover Optimization, John Wiley, New York.

[11] L.R. Ford, Network Flow Theory (1956), RAND Corporation paper P-923, Santa Monica, California.

[12] C. Jayakumar, C. Chellappan, Optimized on demand routing protocol of mobile ad hoc network, Informatica, **17** (2006), 481-502.

[13] P. Li, S. Guo, S. Yu, A.V. Vasilkos (2014), Reliable Multicast with Pipelined Network Coding using Opportunistic Feeding and Routing, IEEE Transactions on Parallel and Distributed Systems, **25** (2014), 3264-3273, https://doi.org/10.1109/TDPS.2013.2297105.

[14] S. Li, W. Sun, Y. Zhang, Y. Chen, Optimal congestion control and routing for multipath networks with random losses, Informatica, **26** (2015), 313-334, http://dx.doi.org/10.15388/Informaitica. 2015.50.

[15] Manvi, S. Sunilkumar, P. Venkataram, An agent-based best effort routing technique for load balancing, Informatica, **17** (2006), 407-426.

[16] N. Meghanathan, Performance comparison study of multicast routing protocols for mobile ad hoc networks under default flooding and density and mobility aware energy-efficient (DMEF) broadcast strategies, Informatica, **35** (2011), 165-184.

Research on Campus Network Equipment Environment Monitoring Based on Internet of Things

Jianwei Liu
Fujian Preschool Education College, Fuzhou, Fujian 350007, China
E-mail: liujianwei139@fjys.edu.cn

Student paper

Keywords: internet of things, campus network environment, equipment monitoring, ZigBee

Received: May 20, 2021

The working environment of equipment has a great impact on the performance of equipment; therefore, it needs to be monitored and managed effectively. Based on the Internet of things (IoT), this study analyzed the monitoring of the campus network equipment environment. A monitoring system was designed, the design methods of software and hardware were analyzed, and the system was tested. The test results showed that the sensor used in the system had high accuracy, the temperature error was ± 0.5 ℃, the humidity error was within 0.5% RH, the data transmission performance was good, the transmission delay was between 8 ms and 9 ms, and the average packet loss rate was 0.7%. The test results verify the effectiveness of the designed system, and the system can be popularized and applied in the actual campus network.

Povzetek: Predstavljen je sistem za nadzor opreme v fakultetnem okolju z IoT.

1 Introduction

With the development of technology, the way of information acquisition has developed from manual to automatic. With the application of technologies such as radio frequency identification (RFID) and sensors [1], the Internet of things (IoT) has also developed rapidly [2]. IoT refers to the interconnection between things based on computer technology to form an intelligent network [3], through which data transmission [4], sharing and storage [5] are carried out, thus providing reliable intelligent management [6], including online monitoring, real-time positioning, remote alarm and other functions. IoT creates a new measurable, quantifiable world [7]. IoT has been widely valued by all countries in the world and has good applications in fields such as smart grid [8], industry [9], and medicine [10]. Marques et al. [11] proposed an IoT-based indoor air quality system, which combined Arduino, ESP8266, and XBee technologies. The system could be accessed through the web and mobile applications, and doctors could access the data to support the medical judgment. Jaco et al. [12] designed a vehicle positioning system combining RFID technology with a global system for mobile communication (GSM) technology, which obtained a reading range of about 31 cm in the low-frequency communication range. After the field test, they found that the system could realize the positioning and tracking of vehicles. Ray et al. [13] proposed a thermal comfort index measurement system based on IoT, which intelligently integrated sensors and cloud services. They found that the system realized heterogeneous network communication and low power consumption, showing good performance. Zhu et al. [14] designed a smart home control system based on IoT, which could realize remote query and control of home applications. The system had the characteristics of reliable transmission and intelligent processing. The test showed that the system could run stably. With the popularity of the campus network, the equipment and scale in the network environment have been gradually expanded. The failure of the equipment will cause network paralysis, affecting the normal study and work of teachers and students. The working environment of equipment has a direct impact on the working state of equipment, such as temperature, humidity, etc. In the current campus network equipment management, manual duty is usually used, which is difficult to timely and comprehensively monitor the equipment. Therefore, based on the IoT, this study designed a campus network equipment environment monitoring system, which realized the collection and monitoring of equipment environment data and the intelligence and informatization of network equipment environment management through ZigBee and sensor technology. This work makes some contributions to improve the service quality of the campus network.

2 Design of environment monitoring system for campus network equipment based on IoT

2.1 Overall structure of the system

A complete IoT mainly includes three parts: (1) perception layer: it takes equipment such as a single-chip microcomputer as the terminal node, carries various sensors, and configures the communication module to realize the communication between nodes, and it provides data to the upper later via the hardware interface; (2) network layer: it uploads the data obtained by the hardware interface to the application center through communication technology or transmission control protocol/internet protocol (TCP/IP) port; (3) application layer: it integrates technologies such as database and uses the interface provided by the lower layer to realize the control of the terminal node.

In the design of the system, the environmental parameters of the equipment were collected through temperature, humidity, and smoke sensors, and the ZigBee network was used for realizing the data transmission. Through the RJ232 interface, the information collected by the sensor was transmitted to the upper computer. Finally, in the application layer, the software system was developed by VB to realize the real-time monitoring of the equipment environment. The overall structure of the system is shown in Figure 1.

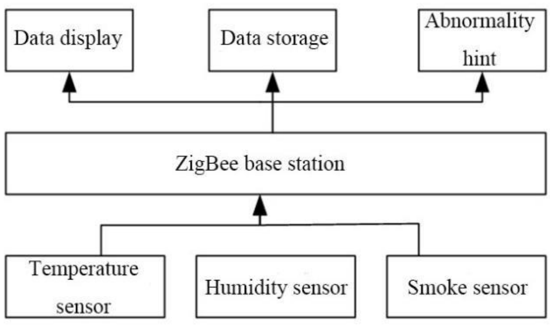


Figure 1: Monitoring system of campus network equipment environment.

2.2 System hardware design

The TM35 temperature sensor was used, with a working voltage of 2.7 - 5.5 V and an error of ± 2 ℃. When the ambient temperature was 25 ℃, the output voltage of the temperature sensor was 250 MVC, the output proportional coefficient was 10 mV/℃, and the measurement range was - 10 ℃ - 100 ℃. The IH-3605 humidity sensor was used, with a relative humidity of 0% - 100% and a corresponding output direct voltage of 0.8 - 4 V. When the humidity sensor worked under 5 V, the power consumption was only 200 μA, and the working temperature was - 40 ℃ - 85 ℃. It had high precision, strong stability, and good chemical corrosion resistance; thus, it could provide stable data in equipment environmental monitoring. QJB gas monitoring alarm was used. When the detected gas appeared in the air, the platinum electrode would change, which would be converted into relay signal output. It had an anti-interference performance as it was less affected by temperature and humidity changes. Moreover, it had high sensitivity.

MSP430F5638 single-chip microcomputer was used as the processor, including two universal serial communication interfaces, four 16-bit timers, and a microcontroller with multiple input/output (I/O) pins. It had a working voltage of 1.8-3.6 V, low power consumption, long battery life, and fast response to wake-up; thus, it could adapt to long-time working situations.

In the choice of wireless communication protocol, there are four kinds of wireless communication protocol, as shown in Table 1.

	Working frequency band	Transmission speed	Power consumption	Effective range
802.11 protocol	2.4 GHz	11 Mbps	100 mW	100 m
Ultra wide band (UWB) communication	3.1 GHz - 10.6 GHz	480 Mbps	1 mW	10 m
Bluetooth protocol	2.4 GHz	1-3 Mbps	1-100 mW	10-100 m
ZigBee protocol	868/915 MHz, 2.4 GHz	20, 40, 250 Kbps	1-3 mW	70 m

Table 1: Comparison of communication protocols.

It was seen from Table 1 that the power consumption of 802.11 protocol and Bluetooth protocol was large, while the power consumption of UWB communication was small, but its effective range was only 10 m, which could not meet the communication distance requirements of the monitoring system. Therefore, this paper selected ZigBee protocol as the communication protocol and used CC2530 single-chip microcomputer produced by T1 company to send the collected data. The single-chip microcomputer consisted of an 8051 single-chip microcomputer and a radio-frequency (RF) transceiver and had a communication distance of 1000 m and a storage space of 256 KB. It could realize long-distance communication, with stable networking performance and low price.

The battery part used two No. 5 batteries as the main power supply, which was easy to replace. The gateway part used the OMAP3730 chip produced by T1 company, which integrated two subsystems. The digital signal processing (DSP) subsystem had a highest working frequency of 800 MHz and a maximum processing capacity of 6400 MIPS. The ARM subsystem used the ARM Coretex-A8 processor and had a highest frequency of 1000 MHz, which had reliable performance in data processing, receiving, and sending.

2.3 Design of system software

The development environment of the system was IAR Embedded Workbench for MCS-51 7.51A, and the programming language was C language. The sensor node collected the environmental parameters through the sensor and transmitted them to the upper computer through the wireless communication module to monitor various parameters. Before the node worked, the clock frequency of the CC2530 single-chip microcomputer was set to ensure the power supply stability of the voltage regulator, and the port of the CC2530 single-chip microcomputer was initialized to run the corresponding program. After the node initialization, different sensors were allowed to join the ZigBee network, collect the corresponding data, and send them to the monitoring center. In this study, the Niagara software was used as the development environment of the upper computer for secondary development to realize the real-time monitoring of the equipment environment.

3 System test

3.1 Sensor accuracy test

In environmental monitoring, temperature and humidity sensors have high requirements for accuracy, while smoke sensors only have two states, "yes (1)" and "no (0)", which is relatively simple. Therefore, this study mainly tested the accuracy of temperature and humidity sensors, collected five groups of data at different times and places, and compared the temperature and humidity collected by the sensors with the actual temperature and humidity. The results are shown in Table 2.

Number		Temperature/℃	Humidity/% RH
1	The sensor in this study	26.89	30.12
	Actual situation	27.31	30.29
2	The sensor in this study	25.33	28.96
	Actual situation	25.12	29.03
3	The sensor in this study	28.78	27.64
	Actual situation	28.56	27.33
4	The sensor in this study	23.46	34.12
	Actual situation	23.57	34.29
5	The sensor in this study	25.66	35.09
	Actual situation	26.07	34.87

Table 2: Test results of sensor accuracy.

It was seen from Table 2 that the errors between the sensors used in this study and the actual result were very small in the comparison of the five groups of data, the temperature error was within ± 0.5 ℃, and the humidity error was not more than 0.5% RH, indicating that the sensors with high accuracy could meet the usage requirement of the equipment environmental monitoring system and provide accurate and reliable data for the subsequent environmental monitoring.

3.2 Network performance test

The time delay of the transmission of the single-hop data between two nodes was tested. After the node joined the network, the coordinator sent a request to the third party and recorded the time as 1. Then, the node received the request, processed it immediately, and returned the packet. The third-party captured the packet again and recorded the time of the process as 2. The time difference between 1 and 2 was the transmission delay. Under different distances, the transmission delay of data is shown in Table 3.

Distance/m	Time delay/ms
10	8
20	8
30	9
40	8
50	9
60	9
70	8
80	8

Table 3: Data transmission delay.

It was seen from Table 3 that the data transmission time was between 8-9 ms in the single-hop range, which showed that the designed system had high transmission speed and good stability in the process of data transmission and could meet the needs of environmental monitoring of campus network equipment.

The packet loss rate of the network was tested. One gateway node, six sensor nodes, and six routing nodes were used. The collected data were forwarded to the gateway through the routing node. The test results are shown in Figure 2.

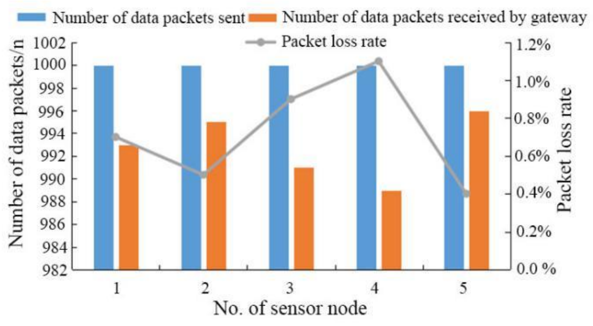


Figure 2: Test results of packet loss rate.

It was seen from Figure 2 that every sensor node sent 1000 packets, and the number of packets received by the gateway node was more than 900, indicating that the network had relatively stable performance in the communication process. The packet loss rates of the five nodes were 0.7%, 0.5%, 0.9%, 1.1%, and 0.4%, respectively, and the average packet loss rate was 0.7%, which verified the communication stability of the designed system.

4 Discussion

With the development of the Internet and other technologies and because of the advantages such as low power consumption, strong flexibility, and low cost, IoT has been more and more widely used in various fields of society [15], such as intelligent label [16], environmental monitoring [17], and intelligent control [18]. An intelligent label means labeling subjects with different technologies to distinguish them, for example, the bar code on commodities. Environmental monitoring means collecting and monitoring the information of subjects with technologies such as sensors, for example, monitoring of urban exhaust pollution. Intelligent control means controlling subjects through technologies such as sensors [19], for example, control of watering time and frequency of greenhouse. As the social demand for IoT increases, the research on IoT will be more extensive and in-depth.

Based on IoT, this study analyzed the monitoring of the campus network equipment environment, designed a monitoring system, and realized the real-time monitoring and judgment of the equipment working environment. First of all, the two sensors used in this study had good performance in accuracy, the error of the temperature sensor was ± 0.5 ℃, and the error of the humidity sensor was ± 0.5% RH, which could meet the needs of the system in environmental monitoring. The test results of network performance showed that the time delay was between 8-9 ms in the process of data transmission, the transmission speed was high, the packet loss rate was low, and the average packet loss rate was 0.7%, which showed that the system had high stability and could meet the needs in practical application.

Although this article has achieved some results in the monitoring of campus network equipment environment, there are still some shortcomings. In future work, the author will:

(1) further improve the stability of data communication;
(2) study the energy consumption of the system to ensure that the nodes can work permanently,
(3) search the module that can expand the transmission distance to improve transmission efficiency.

5 Conclusion

Based on IoT, this study designed an environment monitoring system of campus network equipment and tested it. The results showed that the system had high sensor precision, good network performance, high transmission speed, and low packet loss rate, which could be further promoted and applied in practice. This work is beneficial to the real-time monitoring and management of the campus network equipment environment.

References

[1] Al-Fuqaha A, Guizani M, Mohammadi M, Aledhari M, Ayyash M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*, 17, pp. 2347-2376. https://doi.org/10.1109/COMST.2015.2444095.

[2] Li S, Li D X, Zhao S. (2015). The internet of things: a survey. *Information Systems Frontiers*, 17, pp. 243-259. https://doi.org/10.1016/j.comnet.2010.05.010.

[3] Singh K J, Kapoor D S. (2017). Create Your Own Internet of Things: A survey of IoT platforms. *IEEE Consumer Electronics Magazine*, 6, pp. 57-68. https://doi.org/10.1109/MCE.2016.2640718

[4] King J, Awad AI. (2016). A distributed security mechanism for resource-constrained IoT devices. *Informatica*, 40, pp. 133-143.

[5] Bali M, Tari A, Almutawakel A, Kazar O. (2020). Smart Design for Resources Allocation in IoT Application Service Based on Multi-agent System and DCSP. *Informatica*, 44, pp. 373-386. https://doi.org/10.31449/inf.v44i3.2962

[6] Whitmore A, Agarwal A, Xu L D. (2015). The Internet of Things—A survey of topics and trends. *Information Systems Frontiers*, 17, pp. 261-274. https://doi.org/10.1007/s10796-014-9489-2.

[7] Tran HA, Tran DQ, Nguyen G, Ha QT, Tong V, Mellouk A. (2018). SHIOT: A novel SDN-based framework for the heterogeneous Internet of Things. *Informatica*, 42(3). https://doi.org/10.31449/inf.v42i3.2245

[8] Viswanath S K, Yuen C, Tushar W, Li WT, Wen CK, Hu K, Chen C, Liu X. (2016). System Design of Internet-of-Things for Residential Smart Grid. *IEEE Wireless Communications*, 23, pp. 90-98.

[9] Trappey A J C, Trappey C V, Govindarajan U H, Chuang AC, Sun J. (2016). A review of essential standards and patent landscapes for the Internet of Things: A key enabler for Industry 4.0. *Advanced Engineering Informatics*, 33, pp. 208-229. https://doi.org/10.1016/j.aei.2016.11.007.

[10] Islam S, Kwak D, Kabir M H, Hossain M, Kwak KS. (2015). The Internet of Things for Health Care: A

Comprehensive Survey. *IEEE Access*, 3, pp. 678-708. https://doi.org/10.1109/ACCESS.2015.2437951.

[11] Marques G, Pitarma R. (2016). An Indoor Monitoring System for Ambient Assisted Living Based on Internet of Things Architecture. *International Journal of Environmental Research and Public Health*, 13, pp. 1152-. https://doi.org/10.3390/ijerph13111152.

[12] Jaco P, Reza M. (2016). Accurate Vehicle Location System Using RFID, an Internet of Things Approach. *Sensors*, 16, pp. 825. https://doi.org/10.3390/s16060825.

[13] Ray PP. (2016). Internet of Things Cloud Enabled MISSENARD Index Measurement for Indoor Occupants. *Measurement*, 92, pp. 157-165. https://doi.org/10.1016/j.measurement.2016.06.014

[14] Zhu J, Jia X, Mei X Q. (2015). Smart Home Control System Based on Internet of Things. *Applied Mechanics and Materials*, 738-739, pp. 233-237.

[15] Ma H, Liu L, Zhou A, Zhao D. (2017). On Networking of Internet of Things: Explorations and Challenges. *IEEE Internet of Things Journal*, 3, pp. 441-452. https://doi.org/10.1109/JIOT.2015.2493082.

[16] Marktscheffel T, Gottschlich W, Popp W, Werli P, Fink SD, Bilzhause A, de Meer H. (2016). QR code based mutual authentication protocol for Internet of Things. *Workshop on IoT-SoS: Internet of Things Smart Objects & Services*, pp. 1-6. https://doi.org/10.1109/WoWMoM.2016.7523562.

[17] Kokkonis G, Psannis K E, Roumeliotis M, Schonfeld D. (2017). Real-time wireless multisensory smart surveillance with 3D-HEVC streams for internet-of-things (IoT). *The Journal of Supercomputing*, 73, pp. 1044-1062. https://doi.org/10.1007/s11227-016-1769-9.

[18] Francisco-Pastor F P, García-Chamizo JM, Nieto-Hidalgo M, Mora-Pascual J, Mora-Martínez J. (2016). Developing Ubiquitous Sensor Network Platform Using Internet of Things: Application in Precision Agriculture. *Sensors*, 16, pp. 1141. https://doi.org/10.3390/s16071141.

[19] Kawakami T, Ishi Y, Yoshihisa T, Teranishi Y. (2015). A Churn Resilience Technique on P2P Sensor Data Stream Delivery System Using Distributed Hashing. *Informatica: An International Journal of Computing and Informatics*, 39, pp. 355-363. https://doi.org/10.1109/3PGCIC.2014.114

Research on the Efficiency of Intelligent Algorithm for English Speech Recognition and Sentence Translation

Gang Zhang
School of Foreign Languages and Literature, Nanjing Tech University, Nanjing 211816, Jiangsu, China
E-mail: zg@njtech.edu.cn

Student paper

Keywords: intelligent algorithm, speech recognition, machine translation, long short term memory

Received: May 25, 2021

Machine translation has been gradually widely used to improve the efficiency of English translation. This paper briefly introduced the English speech recognition algorithm based on the back-propagation (BP) neural network algorithm and the machine translation algorithm based on the long short-term memory-recurrent neural network (LSTM-RNN) algorithm. Then, the machine translation algorithm was simulated and compared with BP-RNN and RNN-RNN algorithms. The results showed that the BP neural network algorithm had a lower word error rate and shorter recognition time compared with the manual recognition approach; the LSTM-RNN-based machine translation algorithm had the lowest error rate for the translation of speech recognition results, and the translation gained the highest rating in the evaluation of ten professional translators.

Povzetek: Predstavljena je analiza več pristopov umetne inteligence za prepoznavanje govora in prevajanje.

1 Introduction

As internationalization progresses faster and faster, economic and cultural exchanges between different countries have become more and more frequent. However, different countries have different languages, and even within the same country, there are languages with different accents in different regions, making the variety of languages further increase [1].

The variety of languages, while guaranteeing cultural diversity, can cause significant communication barriers in practice, especially in today's environment of deepening internationalization [2]. Under normal circumstances, the cost of learning multiple languages is enormous, so to ensure smooth communication, a common language is often chosen. English is currently one of the most common languages. Although the variety of language learning has been reduced from multiple to one, the cost of learning is equally high, and it is difficult to reach the level of free communication [3]. In the wave of globalization, it would be sufficient for face-to-face daily communication, but in formal situations and when a large amount of information needs to be exchanged, it is difficult for a single human translator to meet the increasing demand for language translation. Simultaneous interpretation, for example, requires high on the attention of translators; thus, they often cannot work for long hours. Therefore, a translation tool is needed to replace manual translation. Machine translation uses computers and Chinese-English thesaurus to perform batch translation based on, but it is too rigid. The emergence of intelligent algorithms has effectively promoted the efficiency and quality of machine translation. Luong [4] proposed an effective technique to solve the problem that traditional neural network machine translation can not accurately translate rare words. The experimental results showed that this method had a better translation effect compared with neural network machine translation without applying this technique. Lee et al. [5] used a character-level convolution network for machine translation. The character-level convolution network encoder outperformed the subword-level encoder in all of the multilingual experiments. Choi et al. [6] proposed the use of nonlinear word-packet representation of source statements to contextualize embedding vectors in neural machine translation. The experiment found that the proposed approaches of contextualization and symbolization significantly improved the translation quality of neural machine translation systems.

2 English speech recognition

The advancement of computer technology and the emergence of intelligent algorithms have effectively improved the efficiency and quality of machine translation. However, computers have neither vision nor hearing; therefore, machine translation of English requires the input of the text to be translated, which is relatively cumbersome [7]. Inputting text to be translated by voice is easier than text input; moreover, the input English voice can be translated, which means that it can achieve long periods of non-manual simultaneous interpretation. Before translating the English input by voice, it is necessary to first recognize the voice and convert the audio to text characters, and then translate the text. The commonly used algorithms for recognition of speech are dynamic time warping method [8], hidden Markov method, and neural network method.

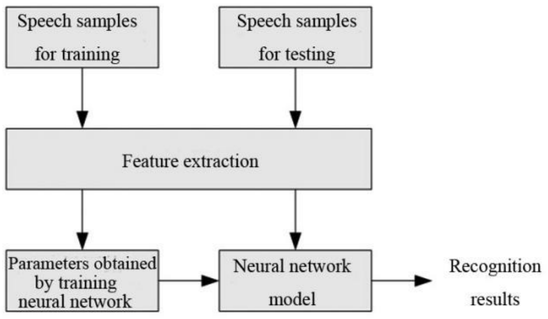

Speech samples for training
Speech samples for testing
Feature extraction
Parameters obtained by training neural network
Neural network model
Recognition results

Figure 1: The principle of speech recognition based on neural network.

This study uses the neural network method to recognize English speeches. The neural network method is an imitation of the human brain and animal nervous system in reality; thus, it also has the ability of autonomous learning, i.e., it can effectively adapt to the randomness of pronunciation and excavate the corresponding hidden rules between pronunciation and text, as shown in Figure 1. Both training speech samples and testing speech samples require feature extraction first. This study extracts the features of speech samples using the Mel cepstrum coefficient method. After that, the selected neural network model is trained using the training speech samples to obtain the super-parameters in the neural network. The trained super-parameters are then fixed in the selected neural network, and the testing speech samples are input into the trained neural network model after feature extraction to obtain recognition results.

There are various types of neural network models that can be used in the speech recognition process in Figure 1, and this paper adopts the widely used BP neural network [9]. During training, the samples that have undergone feature extraction are input into the BP neural network, and the multi-layer forward calculation of the extracted features of the speech is performed in the hidden layer using the activation function. The results obtained after the layer-by-layer calculation are compared with the results corresponding to the training samples, and the super-parameters in the hidden layer are adjusted in the reverse direction according to the difference between the results. Then, the forward calculation is carried out layer by layer again, and the calculated results are compared with the actual results to reversely adjust the parameters. The steps are repeated until the difference between the calculated results and the actual results is reduced to the set threshold. The speech samples used in the test are input into the trained neural network model after feature extraction, and the recognition results are output after calculation.

3 Translation of English sentences

After speech recognition of the neural network algorithm, English speech is converted into English texts. The traditional machine translation translates English word by word with the Chinese English thesaurus. This translation method is simple in principle and has relatively high efficiency. It is more suitable for an essay composed of short sentences. However, this translation method is less effective for long texts composed of long sentences. On the one hand, due to the differences in grammar between Chinese and English, word-by-word translation will lead to grammar chaos and even completely opposite semantics; on the other hand, some auxiliary words that have no specific meanings in English sentences will also be translated, and word-by-word translation will affect the coherence of the translation. In order to solve the above problems and further improve the efficiency and quality of translation, this study chooses neural network algorithms for machine translation. A neural network algorithm is an imitation of a biological neural system. With the help of the parallel corpus of big data and the computational performance of computers, we can get the corresponding hiding rules between Chinese and English. As the parallel corpus of big data is used for training, hiding rules obtained contain the grammatical rules to a certain extent.

3.1 Recurrent neural network-based machine translation

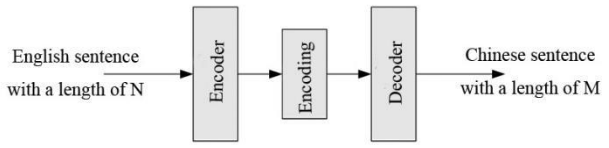

English sentence with a length of N
Encoder
Encoding
Decoder
Chinese sentence with a length of M

Figure 2: The basic structure of neural network-based machine translation.

The basic structure of conducting English machine translation with neural network algorithms is shown in Figure 2, including an encoder and a decoder [10]. The reason for using this structure is as follows. When translating English into Chinese, the length of the character sequence of English and Chinese cannot be the same; thus, it is necessary to transform an English sentence with a length of N to a code with a required length and decode the code with the decor to obtain a Chinese sentence with a length of M. In the above process, both the encoder and decoder realize the encoding and decoding of the sequence through neural network algorithms. As the length of sentences to be translated is different and the meaning of words will be affected by their sequence in the sentence, it is not suitable to use the traditional BP neural network. Recurrent neural network (RNN) [11] has memory function in training, and the result of the current moment will be affected by the result of the previous moment, which is very consistent with the characteristic that the word sequence will affect the meaning of words in languages; therefore, it is applied to the coding and decoding of machine translation. First, an English sentence is set as $\boldsymbol{\varepsilon} = \{\varepsilon_0, \varepsilon_1, \varepsilon_2, \cdots, \varepsilon_n\}$, where ε_n represents the $n+1$-th word in sentence $\boldsymbol{\varepsilon}$; then, RNN propagates the input data forward in the hidden layer according to the following formula:

$$\begin{cases} \boldsymbol{x}_t = e(\varepsilon_t) \\ \boldsymbol{a}_t = \boldsymbol{\omega} h_{t-1} + \boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{b} \\ h_t = tanh(\boldsymbol{a}_t) \\ t = 0,1,2,3,4,\cdots,n \end{cases}, \tag{1}$$

where $\boldsymbol{x}_t$ stands for the word vector after inputting ε_t into the vector transformation function $e(\cdot)$, t stands for time step (every time step corresponds to the input moment of a word in the sequence), h_{t-1}, h_t are the hidden states of words vectors with a word order of $t-1$ and t, $\boldsymbol{a}_t$ is the output at time t, $\boldsymbol{\omega}$ and $\boldsymbol{U}$ are the hidden state at time $t-1$ and the weight matrix of the word vector of ε_t input at time t respectively, and $\boldsymbol{b}$ is a bias term.

After the forward operation of equation (1) in the encoder, every word (ε_t) in the English sentence obtains the corresponding coding vector $\boldsymbol{a}_t$, and the coding vector of every word is affected by the former word vector, ensuring the influence of the word order in the coding vector of the whole sentence on word meaning. Then, the coding vector is decoded to obtain the corresponding Chinese translation. The decoding is carried out in the decoder. In order to ensure the influence of word order on word meaning, the coding vector is decoded by RNN. The forward calculation formula in the hidden layer is:

$$\begin{cases} \mathbf{y}_{t-1} = e(Z_{t-1}) \\ z_t = \tanh(\boldsymbol{\omega} z_{t-1} + \mathbf{U}\mathbf{y}_{t-1} + \mathbf{b}) \\ \hat{\mathbf{y}}_t = \mathrm{softmax}(\mathbf{d} + \mathbf{V} z_t) \\ Z_t = \arg\max \hat{\mathbf{y}}_t \end{cases}, \tag{2}$$

where Z_t is a word with an order of t in the target sentence, y_{t-1} is the vector of a word with an order of $t-1$ in the target sentence, $\boldsymbol{d}$ is a bias term, $\boldsymbol{V}$ is a weight matrix, $\hat{\boldsymbol{y}}_t\hat{\boldsymbol{y}}_t$ is the probability distribution of different characters in the translation [12], and z_t is the hidden state in the decoder.

3.2 Improving machine translation with long short term memory

The above is the introduction of the RNN-based machine translation algorithm. This machine translation algorithm adopted the basic structure of encoder-decoder. In the encoder, the English text is coded by RNN in the encoder to get the intermediate vector, and the intermediate vector is decoded by RNN in the decoder to get the Chinese translation. However, the encoder faces the problem of gradient explosion when using RNN to encode the English text, which makes the algorithm inefficient and less accurate in the training and use process. Therefore, this study used LSTM instead of RNN to encode the English text. LSTM [13] is also a kind of recurrent neural network algorithm. Compared with the traditional RNN, LSTM introduces the structural units of input gate, forgetting gate, and output gate to simulate the phenomena of deep impression and forgetting in the process of human brain memory, thus reducing the unimportant parts in the English text, highlighting the key points, reducing the amount of computation while enhancing the accuracy. The calculation formula of LSTM in the encoder is as follows:

$$\begin{cases} f_t = \sigma(b_f + U_f x_t + W_f h_{t-1}) \\ s_t = f_t s_{t-1} + g_t \sigma(b + U x_t + W h_{t-1}) \\ g_t = \sigma(b_g + U_g x_t + W_g h_{t-1}) \\ h_t = tanh(s_t) q_t \\ q_t = \sigma(b_q + U_q x_t + W_q h_{t-1}) \end{cases}, \tag{3}$$

where f_t is the output of the forgetting gate, b_f, U_f, and W_f are the bias term, input term weight, and forgetting gate weight in forgetting gate [14], s_t is the output of the cycle gate, b , U, and W are the bias term, input term weight, and cycle gate weight in the cycle gate weight, g_t is an external input gate unit, b_g, U_g, and W_g are the bias term, the weight of the input term, and the weight of the input gate in the input gate respectively, q_t is the output gate unit, b_q, U_q, W_q are the output gate bias term, input term weight, and output gate weight.

In the subsequent decoder, RNN is still used to decode the output vector of the encoder, but in actual use, the encoder of the machine translation algorithm compresses the information contained in the whole sentence to be translated in a vector when encoding English using LSTM, resulting in partial loss of information in the compressed vector. The larger the length of English text to be translated is, the more serious the loss is. Therefore, when decoding the LSTM-encoded vector using RNN, its effectiveness decreases with the increase of the length of the original text. In order to remedy the above defects, an attention mechanism [15] is added to the decoder to make the decoder interact with the original text in the decoding process to alleviate the information loss. The calculation formula of attention is as follows:

$$\begin{cases} \alpha_t = attention(z_t, h_t) \\ \sum_t \alpha_t = 1 \\ c = \sum_t \alpha_t h_t \end{cases}, \tag{4}$$

where α_t is the weight of the matching degree between the query vector and the key vector in the value vector (the query vector is the hidden state in the decoder, and the key vector is the hidden state in the encoder), and c is the attention weight. The attention calculation formula compares the hidden states in the decoder with the hidden states of the original text in the encoder to obtain the attention weights of the hidden states in the decoder, and the attention weights participate in the decoding process of the decoder afterwards to calculate the probability distribution of the translated text within equation (2). The probability distribution formula after transformation is:

$$\hat{\boldsymbol{y}}_t = \mathrm{softmax}(\mathbf{c} + \mathbf{V} z_t + \boldsymbol{W}_c c). \tag{5}$$

4 Experimental analysis

4.1 Experimental environment

The experiment was carried out on a laboratory server, configured with Windows 7 system, I7 processor, and 16 G memory.

4.2 Experimental data

This study used the English speech data set from the UCI machine learning database. The speakers in the data set covered most of the age groups from 12 to 70. Ten thousand sentences with clean, clear, and standard pronunciation were selected, 9000 sentences were randomly selected as the training samples, and the remaining 1000 sentences were used as as the test samples. The experimenters read the sentences aloud. The speech characteristic parameters of the sentences were collected. The sampling rate was set as 16 kHz, and the 16-bit coding was used. Some of the sentences are as follows.

① It's a nice day today;

② How can I get to the airport, please;

③ What's the price of this product……

4.3 Experimental project

(1) Training and testing of the speech recognition algorithm: the BP algorithm-based speech recognition algorithm was trained by the training set. In the BP algorithm, there were five hidden layers containing the relu activation function and 1024 hidden layer nodes in each layer, and the maximum number of iterations during training was 500. The test set was used for testing.

(2) Training and testing of the machine translation algorithm: in the machine translation algorithm, the encoder used LSTM containing the attention mechanism. There were four hidden layers, and the number of nodes in each layer was 1024. The decoder used RNN with two hidden layers, and the size of the hidden layer was consistent with LSTM. Two machine translation algorithms are used for comparison, one used BP neural network as the encoder, and the other used a RNN as the encoder. The decoder of both algorithms used RNN. The test set was used for testing.

4.4 Evaluation criterion

In this paper, the machine translation algorithm was evaluated by the word error rate after recognition by the speech recognition algorithm. The calculation formula is:

$$WER = \frac{X+Y+Z}{P} * 100\%, \quad (6)$$

where X is the number of error words substituted, Y is the number of error words deleted, Z is the number of error words inserted, and P is the number of all words in the test set. However, the corresponding word order differs between the original text and the translated text because of the difference in grammar. Therefore, when evaluating the machine translation algorithm, not only the word error rate but also the overall translation level of the sentence should be considered. Therefore, ten professional translators were invited to evaluate the translation obtained after machine translation, and the score is based on the expression content and grammatical structure of the translated text. The total score was 100 points, and the average score of the ten translators was taken as the final result.

4.5 Experimental results

The accuracy of English speech recognition algorithms will greatly affect the quality of machine translation It was seen from Table 1 that the word error rate of speech recognition by manual recognition was 6.34%, and it took 35 minutes; the word error rate of speech recognition by BP neural network was 1.33%, and it takes one minute. The comparison of the results of two speech recognition methods showed that BP neural network had a lower word error rate and consumed less recognition time than manual recognition. The reason for the above result is as follows. Computers were more efficient in computational efficiency, and when faced with a large number of speech sounds in the test set, people cannot maintain their attention for a long time, resulting in a higher word error rate and longer recognition time; however, BP neural network used computers for speech recognition, which was not only computationally efficient but also did not have the disadvantage of inattention.

	Word error rate/%	Time consumed in recognition
Manual recognition	6.34	35 min
BP neural network recognition	1.33	1 min

Table 1: Word error rate and recognition time of artificial recognition and BP neural network recognition.

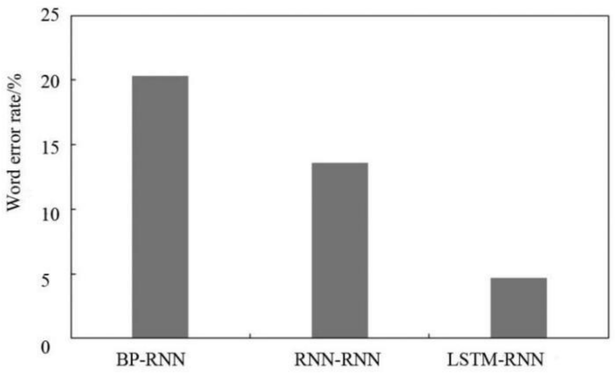


Figure 3: The word error rates of three machine translation algorithms in translating English speech recognition results.

The word error rates of the translations of English speech recognition results by the three machine translation algorithms are shown in Figure 3. The word error rate of the machine translation algorithm using BP neural network for encoder and RNN for decoder was 20.3%; the word error rate of of the machine translation algorithm with RNN for encoder and RNN for decoder was 13.6%; the word error rate of the machine translation algorithm with LSTM for encoder and RNN for decoder and attention mechanism was 4.7%. It was seen from Figure 3 that the LSTM-RNN-based machine translation had the lowest word error rate, while the BP-RNN-based machine translation had the highest word error rate. BP neural networks encoded English directly word by word, and although it can also explore the hidden laws, it cannot accurately describe the influence of word order on word meaning, which leads to an increase in word error rate; RNN took into account the influence of the previous moment in the forward calculation, i.e., the influence of the previous word in this study; in the LSTM-RNN-based machine translation, LSTM, as a variant of RNN, can also summarize the influence of word order, while the forgetting gate, input gate, and output gate units introduced by it can filter the unimportant words among them and improve the accuracy, and the attention mechanism introduced by RNN in the decoder also made the decoding focus more on to the main information.

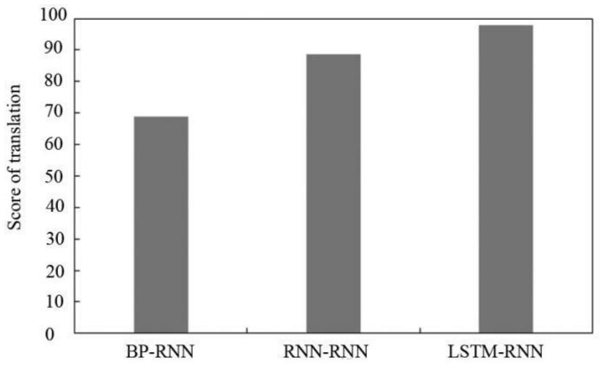


Figure 4: Translation scores of three machine translation algorithms for English speech recognition results.

For the machine translation algorithm, the criterion of whether the translation performance is excellent or not includes the grammatical goodness of the translation as a whole, in addition to the judgment of whether the words of the translation are accurate or not. Therefore, ten professional translators were invited to manually translate the English sentences in the test set and evaluated and scored the machine translation. The results are shown in Figure 4. The translation scores were 68.9 using the BP-RNN algorithm, 88.7 using the RNN-RNN algorithm, and 97.8 using the LSTM-RNN algorithm. It was seen from Figure 4 that the translation obtained by using the BP-RNN algorithm for machine translation had the lowest rating, and the translation obtained by using the LSTM-RNN algorithm for machine translation had the highest rating. The reason was similar to the above. The BP-RNN algorithm did not consider the influence of word order, resulting in a final translation that was similar to word-by-word translation, with no problem in overall comprehension but difficult to read smoothly; the RNN-RNN algorithm RNN-RNN considered the influence of word order when encoding English through RNN, so the translation was relatively more smooth and had a higher score; the LSTM-RNN algorithm also considered the influence of word order coding because LSTM is a kind of RNN, and the LSTM algorithm in the encoder and the attention mechanism in the decoder effectively highlighted the key points and improved the fluency of translation while reducing the translation computation.

5 Conclusion

In this study, LSTM network was used as the encoding algorithm of the encoder, RNN was used as the decoding algorithm of the decoder, and the attention mechanism was introduced into the decoder. The results are as follows: (1) BP neural network algorithm was used to recognize English speech, and the recognition results were used in machine translation; compared with human recognition, BP neural network had a lower word error rate and less recognition time; (2) the LSTM-RNN algorithm had the lowest word error rate for English speech recognition, the RNN-RNN-based machine translation algorithm had a higher word error rate, and the BP-RNN-based machine translation algorithm had the highest word error rate; (3) the LSTM-RNN-based machine translation algorithm had the lowest rating in the evaluation of professional translators, followed by the RNN-RNN machine translation algorithm and the BP-RNN machine translation algorithm.

References

[1] Graham Y, Baldwin T, Moffat A, Zobel J (2015). Can machine translation systems be evaluated by the crowd alone. *Natural Language Engineering*, 23, pp. 1-28. https://doi.org/10.1017/S1351324915000339.

[2] Wang Y, Li J, Gong Y (2015). Small-footprint high-performance deep neural network-based speech recognition using split-VQ. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4984-4988. https://doi.org/10.1109/ICASSP.2015.7178919.

[3] Wu C, Karanasou P, Gales M J F, et al (2016). Stimulated Deep Neural Network for Speech Recognition. *INTERSPEECH*, pp. 400-404. https://doi.org/10.21437/Interspeech.2016-580.

[4] Luong M, Sutskever I, Le Q, Vinyals O, Zaremba W (2015). Addressing the Rare Word Problem in Neural Machine Translation. *Bulletin of University of Agricultural Sciences and Veterinary Medicine Cluj-Napoca. Veterinary Medicine*, 27, pp. 82-86. https://doi.org/10.3115/v1/P15-1002.

[5] Lee J, Cho K, Hofmann T (2017). Fully Character-Level Neural Machine Translation without Explicit Segmentation. *Transactions of the Association for Computational Linguistics*, 5, pp. 365-378. https://doi.org/10.1162/tacl_a_00067.

[6] Choi H, Cho K, Bengio Y (2017). Context-Dependent Word Representation for Neural Machine Translation. *Computer Speech & Language*, 45, pp. 149-160. https://doi.org/10.1016/j.csl.2017.01.007.

[7] Chorowski J, Bahdanau D, Serdyuk D, Cho K, Bengio Y (2015). Attention-Based Models for Speech Recognition. *Computer Science*, 10, pp. 429-439.

[8] Miao YJ, Gowayyed M, Metze F (2015). EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. *2015 IEEE Workshop on Automatic Speech Recognition and*

Understanding (ASRU), pp. 167-174. https://doi.org/10.1109/ASRU.2015.7404790.

[9] Schwarz A, Huemmer C, Maas R, Kellermann W (2015). Spatial diffuseness features for DNN-based speech recognition in noisy and reverberant environments. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4380-4384. https://doi.org/10.1109/ICASSP.2015.7178798.

[10] Kipyatkova I (2017). Experimenting with Hybrid TDNN/HMM Acoustic Models for Russian Speech Recognition. https://doi.org/10.1007/978-3-319-66429-3_35.

[11] Yoshioka T, Karita S, Nakatani T (2015). Far-field speech recognition using CNN-DNN-HMM with convolution in time. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4360-4364. https://doi.org/10.1109/ICASSP.2015.7178794.

[12] Wang Y, Bao F, Zhang H, Gao G (2017). Research on Mongolian Speech Recognition Based on FSMN. https://doi.org/10.1007/978-3-319-73618-1_21.

[13] Alam MJ, Gupta V, Kenny P, Dumouchel P (2015). Speech recognition in reverberant and noisy environments employing multiple feature extractors and i-vector speaker adaptation. *Eurasip Journal on Advances in Signal Processing*, 2015, pp. 50.

[14] Brayda L, Wellekens C, Omologo M (2015). N-best parallel maximum likelihood beamformers for robust speech recognition. *2006 14th European Signal Processing Conference*, pp. 1-4.

[15] Hammami N, Bedda M, Nadir F (2012). The second-order derivatives of MFCC for improving spoken Arabic digits recognition using Tree distributions approximation model and HMMs. *International Conference on Communications and Information Technology*, pp. 1-5. https://doi.org/10.1109/ICCITechnol.2012.6285769.

Three Methods for Energy-Efficient Context Recognition

Vito Janko
Jožef Stefan Institute, Ljubljana, Slovenia
E-mail: vito.janko@ijs.si

Thesis summary

Keywords: context recognition, optimization, energy efficiency, Markov chains, duty-cycling, decision trees

Received: April 13, 2021

Context recognition is a process where (usually wearable) sensors are used to determine the context (location, activity, etc.) of users wearing them. A major problems of such context-recognition systems is the high energy cost of collecting and processing sensor data. This paper summarizes a doctoral thesis that focuses on solving this problem by proposing a general methodology for increasing the energy-efficiency of context-recognition systems. The thesis proposes and combines three different methods that can adapt a system's sensing settings based on the last recognized context and last seen sensor readings.

Povzetek: V doktorski disertaciji so predstavljene tri različne metode za povečevanje energijske učinkovitosti nosljivih senzorjev.

1 Introduction

Widespread accessibility of wearable sensing devices opens many possibilities for tracking the users who wear them. Possible applications range from measuring their exercise patterns and checking on their health, to giving them location-specific recommendations. In this work we will use the term context-recognition for all these tasks.

A common problem when using such context-recognition systems is their impact on the battery life of the sensing device. It is easy to imagine that an application that monitors users' habits using all the sensors in a smartphone (accelerometer, GPS, Wi-Fi etc.) will quickly drain the phone's battery, making it useless in practice. While many methods for reducing the energy-consumption of a context-recognition system already exist, most of them are specialized. They work either in a specific domain or can only optimize the energy consumption of specific sensors [1]. Adapting these methods to another domain can be laborious and may require a lot of expert knowledge and experimentation. In addition, many methods proposed in the related work are "static", i.e., they always use the same sensing setting instead of switching between them based of the current need [2].

It would be highly beneficial to have a general methodology that can prescribe rules for dynamically adapting sensing settings of any context-recognition system to make it more energy-efficient – without having any expert knowledge about the domain. One such methodology was proposed in our doctoral thesis [3], and is summarized in this paper.

2 Methodology

The presented methodology consists of three different methods for decreasing the energy-consumption of a context recognition system, and then combining them in three different ways. The first two methods adapt sensor settings based on the last recognized context, while the last adapts them based on the values of features used by machine-learning models for context recognition.

2.1 Setting to Context Assignment (SCA)

Assume that the context recognition system is classifying subsequent instances using some setting *s*. A setting can be what sampling frequency is used, the list of active sensors, duty-cycle lengths, etc. The setting used changes whenever the context changes, for example, when context c_1 is detected we might use setting s_4, but when context c_2 is detected we might use setting s_1. This opens the problem of finding an ideal assignment of which setting to use for each context.

Trying every combination is infeasible, due to a) the large number of assignment combinations and b) the long time needed to evaluate each combination. We thus proposed a more efficient search scheme. First, we created a mathematical model based on Markov chains that can predict the performance of any single assignment. The only inputs needed for this are the performances of using each setting individually, and the transition matrix between different contexts. Second, we used multi-objective optimization (specifically the NSGA-II [4] algorithm) – with the two objectives being the energy consumption and the classification accuracy – to efficiently search for Pareto-optimal assignments.

2.2 Duty-Cycle to Context Assignment (DCA)

The second method works essentially the same way as the first: a mathematical model is first built to model assignments and the NSGA-II algorithm is used to search for the best ones.

The main difference is that the DCA method is specialized and can only use duty-cycle lengths as the settings. This allows for the construction of a mathematical model that is more accurate and needs only the performance of one setting (no duty-cycling) as the input. This specialization is justified as effectively every context recognition system can use duty-cycling to effectively reduce its energy consumption.

2.3 Cost-Sensitive Decision Trees (CS-DT)

The first two methods change settings based on the last recognized context. Such adaptation might be considered too "coarse", especially if the number of contexts is low. To remedy this, we introduced the third method that can adapt based on the feature values instead. This is done by using cost-sensitive decision trees (CS-DT) [5]. These trees work similarly to regular decision trees, with the exception that when building them we also consider the cost of each feature and not only its utility. In our thesis we show how to adapt the CS-DT methodology for context recognition. When traversing such a tree, the setting (usually which sensors are used) is dynamically changing based on the tree node reached – thus implicitly adapting to feature values.

2.4 Method combinations

The three methods can be combined in different ways to further increase the energy efficiency. We proposed the SCA + DCA, SCA + CS-DT and the SCA + DCA + CS-DT combinations.

The combination of all three methods is made by first generating different CS-DTs, and then using them as settings for the SCA method. After an assignment of which CS-DT is used for each context is made, we use the DCA method to determine the duty-cycle lengths for different contexts.

3 Results

Our methodology was tested on four real-life datasets (publicly available SHL and Opportunity, and our own Commodity12 and E-Gibalec) and on a family of artificial datasets. While many solutions were generated, we here list one sample solution for each dataset to serve as a reference for the efficiency of our methodology. These results were taken from the best performing method combination.

For the SHL dataset we were able to use only 5% of the available data (by using lower frequency, duty-cycling and using a sensor subset) and in exchange sacrifice only 5% of the accuracy. For the Opportunity dataset we were able to find a solution that uses 4 sensors on average (instead of the original 30) and in exchange loses a minimal amount (0.01 points) of F-score. For the Commodity12 dataset we analysed four different users with somewhat different habits. Averaging the results of all four we decreased the smartphone's energy consumption from 123 mA to 29 mA. In exchange we lost less than 1 percentage point of accuracy in all cases. Finally, for the E-Gibalec dataset, energy was reduced from 46 mA to 24 mA at the cost of less than 1 percentage point of accuracy.

The methods outperformed two state-of-the-art methods from the related work – both when compared to our methods used individually, but especially when our methods were combined.

Comparison between different methods is shown in Figure 1. It shows different trade-offs between the energy consumption and classification accuracy, and how combining different methods improves the quality of the found trade-offs.

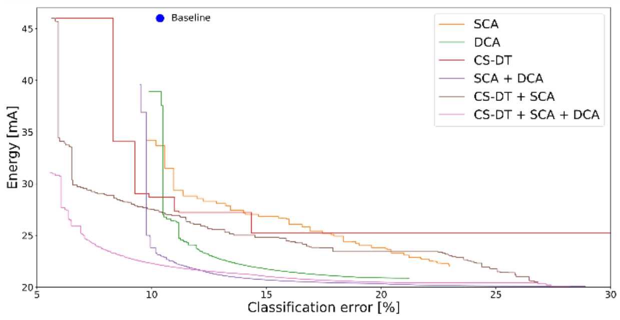


Figure 1: Comparison of different methods and their combinations on the E-Gibalec dataset.

4 Conclusion

The presented methods are very general and should be applicable to most context recognition systems. They require no expert knowledge (with the exception of selecting possible settings) and almost no hand-picked parameters. The method implementation is openly available in the form of a Python library [6] and we hope it could be of use to designers of context-recognition systems.

References

[1] Wang, Yi, et al. "A framework of energy efficient mobile sensing for automatic user state recognition." Proceedings of the 7th international conference on Mobile systems, applications, and services. 2009.

[2] Khan, Aftab, et al. "Optimising sampling rates for accelerometer-based human activity recognition." *Pattern Recognition Letters* 73 (2016): 33-40. https://doi.org/10.1016/j.patrec.2016.01.001

[3] Janko, Vito. Adapting sensor settings for energy-efficient context recognition. Diss. Ph. D. thesis, Jožef Stefan International Postgraduate School, 2020.

[4] Lomax, Susan, and Sunil Vadera. "A survey of cost-sensitive decision tree induction algorithms." *ACM Computing Surveys (CSUR)* 45.2 (2013): 1-35. https://doi.org/10.1145/2431211.2431215

[5] Deb, Kalyanmoy, et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II." *IEEE transactions on evolutionary computation* 6.2 (2002): 182-197 https://doi.org/10.1109/4235.996017

[6] EECR, https://pypi.org/project/eecr/, Last accessed: 08-03-2021

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 900 staff, has 700 researchers, about 250 of whom are postgraduates, around 500 of whom have doctorates (Ph.D.), and around 200 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of **Slove**nia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

From the Jožef Stefan Institute, the Technology park “Ljubljana” has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

Part of the Institute was reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project was developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park is a shareholding company hosting an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Higher Education, Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of the Economy, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel.:+386 1 4773 900, Fax.:+386 1 251 93 85
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Public relations: Polona Strnad

INFORMATICA

AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS

INVITATION, COOPERATION

Submissions and Refereeing

Please register as an author and submit a manuscript at: http://www.informatica.si. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible from typing errors to global philosophical disagreements. The chosen editor will send the author the obtained reviews. If the paper is accepted, the editor will also send an email to the managing editor. The executive board will inform the author that the paper has been accepted, and the author will send the paper to the managing editor. The paper will be published within one year of receipt of email with the text in Informatica MS Word format or Informatica LaTeX format and figures in .eps format. Style and examples of papers can be obtained from http://www.informatica.si. Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the managing editor.

SUBSCRIPTION

Please, complete the order form and send it to Dr. Drago Torkar, Informatica, Institut Jožef Stefan, Jamova 39, 1000 Ljubljana, Slovenia. E-mail: drago.torkar@ijs.si

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than twentyseven years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering intelligent systems in the European computer science, informatics and cognitive community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica web edition is free of charge and accessible at http://www.informatica.si.
Informatica print edition is free of charge for major scientific, educational and governmental institutions. Others should subscribe.

Informatica WWW:

http://www.informatica.si/

Referees from 2008 on:

A. Abraham, S. Abraham, R. Accornero, A. Adhikari, R. Ahmad, G. Alvarez, N. Anciaux, R. Arora, I. Awan, J. Azimi, C. Badica, Z. Balogh, S. Banerjee, G. Barbier, A. Baruzzo, B. Batagelj, T. Beaubouef, N. Beaulieu, M. ter Beek, P. Bellavista, K. Bilal, S. Bishop, J. Bodlaj, M. Bohanec, D. Bolme, Z. Bonikowski, B. Bošković, M. Botta, P. Brazdil, J. Brest, J. Brichau, A. Brodnik, D. Brown, I. Bruha, M. Bruynooghe, W. Buntine, D.D. Burdescu, J. Buys, X. Cai, Y. Cai, J.C. Cano, T. Cao, J.-V. Capella-Hernández, N. Carver, M. Cavazza, R. Ceylan, A. Chebotko, I. Chekalov, J. Chen, L.-M. Cheng, G. Chiola, Y.-C. Chiou, I. Chorbev, S.R. Choudhary, S.S.M. Chow, K.R. Chowdhury, V. Christlein, W. Chu, L. Chung, M. Ciglarič, J.-N. Colin, V. Cortellessa, J. Cui, P. Cui, Z. Cui, D. Cutting, A. Cuzzocrea, V. Cvjetkovic, J. Cypryjanski, L. Čehovin, D. Čerepnalkoski, I. Ćosić, G. Daniele, G. Danoy, M. Dash, S. Datt, A. Datta, M.-Y. Day, F. Debili, C.J. Debono, J. Dedič, P. Degano, A. Dekdouk, H. Demirel, B. Demoen, S. Dendamrongvit, T. Deng, A. Derezinska, J. Dezert, G. Dias, I. Dimitrovski, S. Dobrišek, Q. Dou, J. Doumen, E. Dovgan, B. Dragovich, D. Drajic, O. Drbohlav, M. Drole, J. Dujmović, O. Ebers, J. Eder, S. Elaluf-Calderwood, E. Engström, U. riza Erturk, A. Farago, C. Fei, L. Feng, Y.X. Feng, B. Filipič, I. Fister, I. Fister Jr., D. Fišer, A. Flores, V.A. Fomichov, S. Forli, A. Freitas, J. Fridrich, S. Friedman, C. Fu, X. Fu, T. Fujimoto, G. Fung, S. Gabrielli, D. Galindo, A. Gambarara, M. Gams, M. Ganzha, J. Garbajosa, R. Gennari, G. Georgeson, N. Gligorić, S. Goel, G.H. Gonnet, D.S. Goodsell, S. Gordillo, J. Gore, M. Grčar, M. Grgurović, D. Grosse, Z.-H. Guan, D. Gubiani, M. Guid, C. Guo, B. Gupta, M. Gusev, M. Hahsler, Z. Haiping, A. Hameed, C. Hamzaçebi, Q.-L. Han, H. Hanping, T. Härder, J.N. Hatzopoulos, S. Hazelhurst, K. Hempstalk, J.M.G. Hidalgo, J. Hodgson, M. Holbl, M.P. Hong, G. Howells, M. Hu, J. Hyvärinen, D. Ienco, B. Ionescu, R. Irfan, N. Jaisankar, D. Jakobović, K. Jassem, I. Jawhar, Y. Jia, T. Jin, I. Jureta, Đ. Juričić, S. K, S. Kalajdziski, Y. Kalantidis, B. Kaluža, D. Kanellopoulos, R. Kapoor, D. Karapetyan, A. Kassler, D.S. Katz, A. Kaveh, S.U. Khan, M. Khattak, V. Khomenko, E.S. Khorasani, I. Kitanovski, D. Kocev, J. Kocijan, J. Kollár, A. Kontostathis, P. Korošec, A. Koschmider, D. Košir, J. Kovač, A. Krajnc, M. Krevs, J. Krogstie, P. Krsek, M. Kubat, M. Kukar, A. Kulis, A.P.S. Kumar, H. Kwaśnicka, W.K. Lai, C.-S. Laih, K.-Y. Lam, N. Landwehr, J. Lanir, A. Lavrov, M. Layouni, G. Leban, A. Lee, Y.-C. Lee, U. Legat, A. Leonardis, G. Li, G.-Z. Li, J. Li, X. Li, X. Li, Y. Li, Y. Li, S. Lian, L. Liao, C. Lim, J.-C. Lin, H. Liu, J. Liu, P. Liu, X. Liu, X. Liu, F. Logist, S. Loskovska, H. Lu, Z. Lu, X. Luo, M. Luštrek, I.V. Lyustig, S.A. Madani, M. Mahoney, S.U.R. Malik, Y. Marinakis, D. Marinčič, J. Marques-Silva, A. Martin, D. Marwede, M. Matijašević, T. Matsui, L. McMillan, A. McPherson, A. McPherson, Z. Meng, M.C. Mihaescu, V. Milea, N. Min-Allah, E. Minisci, V. Mišić, A.-H. Mogos, P. Mohapatra, D.D. Monica, A. Montanari, A. Moroni, J. Mosegaard, M. Moškon, L. de M. Mourelle, H. Moustafa, M. Možina, M. Mrak, Y. Mu, J. Mula, D. Nagamalai, M. Di Natale, A. Navarra, P. Navrat, N. Nedjah, R. Nejabati, W. Ng, Z. Ni, E.S. Nielsen, O. Nouali, F. Novak, B. Novikov, P. Nurmi, D. Obrul, B. Oliboni, X. Pan, M. Pančur, W. Pang, G. Papa, M. Paprzycki, M. Paralič, B.-K. Park, P. Patel, T.B. Pedersen, Z. Peng, R.G. Pensa, J. Perš, D. Petcu, B. Petelin, M. Petkovšek, D. Pevec, M. Pičulin, R. Piltaver, E. Pirogova, V. Podpečan, M. Polo, V. Pomponiu, E. Popescu, D. Poshyvanyk, B. Potočnik, R.J. Povinelli, S.R.M. Prasanna, K. Pripužić, G. Puppis, H. Qian, Y. Qian, L. Qiao, C. Qin, J. Que, J.-J. Quisquater, C. Rafe, S. Rahimi, V. Rajkovič, D. Raković, J. Ramaekers, J. Ramon, R. Ravnik, Y. Reddy, W. Reimche, H. Rezankova, D. Rispoli, B. Ristevski, B. Robič, J.A. Rodriguez-Aguilar, P. Rohatgi, W. Rossak, I. Rožanc, J. Rupnik, S.B. Sadkhan, K. Saeed, M. Saeki, K.S.M. Sahari, C. Sakharwade, E. Sakkopoulos, P. Sala, M.H. Samadzadeh, J.S. Sandhu, P. Scaglioso, V. Schau, W. Schempp, J. Seberry, A. Senanayake, M. Senobari, T.C. Seong, S. Shamala, c. shi, Z. Shi, L. Shiguo, N. Shilov, Z.-E.H. Slimane, F. Smith, H. Sneed, P. Sokolowski, T. Song, A. Soppera, A. Sorniotti, M. Stajdohar, L. Stanescu, D. Strnad, X. Sun, L. Šajn, R. Šenkeřík, M.R. Šikonja, J. Šilc, I. Škrjanc, T. Štajner, B. Šter, V. Štruc, H. Takizawa, C. Talcott, N. Tomasev, D. Torkar, S. Torrente, M. Trampuš, C. Tranoris, K. Trojacanec, M. Tschierschke, F. De Turck, J. Twycross, N. Tziritas, W. Vanhoof, P. Vateekul, L.A. Vese, A. Visconti, B. Vlaovič, V. Vojisavljević, M. Vozalis, P. Vračar, V. Vranić, C.-H. Wang, H. Wang, H. Wang, H. Wang, S. Wang, X.-F. Wang, X. Wang, Y. Wang, A. Wasilewska, S. Wenzel, V. Wickramasinghe, J. Wong, S. Wrobel, K. Wrona, B. Wu, L. Xiang, Y. Xiang, D. Xiao, F. Xie, L. Xie, Z. Xing, H. Yang, X. Yang, N.Y. Yen, C. Yong-Sheng, J.J. You, G. Yu, X. Zabulis, A. Zainal, A. Zamuda, M. Zand, Z. Zhang, Z. Zhao, D. Zheng, J. Zheng, X. Zheng, Z.-H. Zhou, F. Zhuang, A. Zimmermann, M.J. Zuo, B. Zupan, M. Zuqiang, B. Žalik, J. Žižka,

Informatica

An International Journal of Computing and Informatics

Web edition of Informatica may be accessed at: http://www.informatica.si.

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Litostrojska cesta 54, 1000 Ljubljana, Slovenia.
The subscription rate for 2021 (Volume 45) is
– 60 EUR for institutions,
– 30 EUR for individuals, and
– 15 EUR for students
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

Typesetting: Borut, Peter and Jaša Žnidar; borut.znidar@gmail.com.
Printing: ABO grafika d.o.o., Ob železnici 16, 1000 Ljubljana.

Orders may be placed by email (drago.torkar@ijs.si), telephone (+386 1 477 3900) or fax (+386 1 251 93 85). The payment should be made to our bank account no.: 02083-0013014662 at NLB d.d., 1520 Ljubljana, Trg republike 2, Slovenija, IBAN no.: SI56020830013014662, SWIFT Code: LJBASI2X.

Informatica is published by Slovene Society Informatika (president Niko Schlamberger) in cooperation with the following societies (and contact persons):
Slovene Society for Pattern Recognition (Vitomir Štruc)
Slovenian Artificial Intelligence Society (Sašo Džeroski)
Cognitive Science Society (Olga Markič)
Slovenian Society of Mathematicians, Physicists and Astronomers (Dragan Mihailović)
Automatic Control Society of Slovenia (Giovanni Godena)
Slovenian Association of Technical and Natural Sciences / Engineering Academy of Slovenia (Mark Pleško)
ACM Slovenia (Nikolaj Zimic)

Informatica is financially supported by the Slovenian research agency from the Call for co-financing of scientific periodical publications.

Informatica is surveyed by: ACM Digital Library, Citeseer, COBISS, Compendex, Computer & Information Systems Abstracts, Computer Database, Computer Science Index, Current Mathematical Publications, DBLP Computer Science Bibliography, Directory of Open Access Journals, InfoTrac OneFile, Inspec, Linguistic and Language Behaviour Abstracts, Mathematical Reviews, MatSciNet, MatSci on SilverPlatter, Scopus, Zentralblatt Math

Volume 45 Number 2 June 2021 **ISSN 0350-5596**

Informatica

An International Journal of Computing and Informatics

Informatica **45** (2021) Number 2, pp. 177–317