Barbara Catania · Rastislav Královič
Jerzy Nawrocki · Giovanni Pighizzini
Roman Špánek (Eds.)

SOFSEM 2019: Theory and Practice of Computer Science

45th International Conference on Current Trends in Theory and Practice of Computer Science
Nový Smokovec, Slovakia, January 27–30, 2019

Proceedings of Student Research Forum

Editors

Barbara Catania
University of Genoa
Genoa, Genova, Italy

Rastislav Královič
Comenius University
Bratislava, Slovakia

Jerzy Nawrocki
Poznań University of Technology
Poznań, Poland

Giovanni Pighizzini
Università degli Studi di Milano
Milan, Italy

Roman Špánek
Technical University of Liberec
Liberec, Czech Republic

ISBN: 978-80-88720-22-5

Printed by OKAT PLUS s.r.o., 2019

Preface

This volume contains the papers selected for presentation at the Student Research Forum of SOFSEM 2019, the 45th International Conference on Current Trends in Theory and Practice of Computer Science, which was held on January 27-30, 2019 in Nový Smokovec, High Tatras, Slovakia.

SOFSEM (originally SOFtware SEMinar) is an annual international winter conference devoted to the theory and practice of computer science. Its aim is to present the latest developments in research for professionals from academia and industry, working in leading areas of computer science. While being a well-established and fully international conference, SOFSEM also maintains the best of its original Winter School aspects, such as high number of invited talks, in-depth coverage of selected research areas, and ample opportunities to discuss and exchange new ideas. SOFSEM 2019 was organized around the following three tracks:

- Foundations of Theoretical Computer Science (chair Giovanni Pighizzini)
- Foundations of Data Science and Engineering (chair Barbara Catania)
- Foundations of Software Engineering (chair Jerzy Nawrocki)

With these three tracks, SOFSEM 2019 covered the latest advances in both theoretical and applied research in leading areas of computer science.

An integral part of SOFSEM 2019 was the traditional Student Research Forum (chair Roman Špánek) organized with the aim to give students feedback on both the originality of their scientific results and on their work in progress. The papers presented at the Student Research Forum were published in separate local proceedings.

The SOFSEM 2019 Program Committee consisted of 70 international experts, representing the track areas with outstanding expertise. The committee stood before the task to assemble a scientific program for the SOFSEM audience by selecting from the 92 submissions entered in the EasyChair system in response to the call for papers. The submissions were carefully reviewed with approximately three reviews per paper, and thoroughly discussed. Following strict criteria of quality and originality, 35 papers have been accepted for presentation as regular research papers. Additionally, based on the recommendation of the chair of the Student Research Forum, 6 student papers have been accepted for presentation in the SOFSEM 2019 Student Research Forum.

SOFSEM 2019 added a new page to the tradition of SOFSEM dating back to 1974, which was possible due to the effort of many people. As editors of these proceedings, we are grateful to everyone who contributed to the scientific program of the conference. We thank all authors who have submitted their papers for consideration. Many thanks go to the Program Committee, and to all external referees, for their precise and detailed reviewing of the submissions. The work of

the PC was carried out using the EasyChair system, and we gratefully acknowledge this contribution. Special thanks are due to Roman Špánek for his expert preparation and handling of the Student Research Forum, and to the SOFSEM Steering Committee headed by Július Štuller, for its support throughout the preparation of the conference.

We are also indebted to the Organizing Committee led by Dana Pardubská.

Finally we want to thank the Slovak Society for Computer Science, and Faculty of Mathematics, Physics and Informatics of the Comenius University in Bratislava for their invaluable support.

January 2019

Barbara Catania,
Rastislav Královič,
Jerzy Nawrocki,
Giovanni Pighizzini,
Roman Špánek

Organization

Steering Committee

Barbara Catania	University of Genoa, Italy
Miroslaw Kutylowski	Wroclaw University of Technology, Poland
Tiziana Margaria-Steffen	University of Limerick, Ireland
Branislav Rovan	Comenius University, Bratislava, Slovakia
Petr Šaloun	Technical University of Ostrava, Czech Republic
Július Štuller (Chair)	Academy of Sciences, Prague, Czech Republic
Jan van Leeuwen	Utrecht University, The Netherlands

Program Chair

Rastislav Královič	Comenius University, Bratislava, Slovakia

Track Chairs

Barbara Catania	University of Genoa, Italy
Giovanni Pighizzini	University of Milano, Italy
Jerzy Nawrocki	Poznań University of Technology

Student Research Forum Chair

Roman Špánek	Technical University of Liberec, Czech Republic

Program Committee

Fabio Anselmi	Italian Institute of Technology
Ladjel Bellatreche	ISAE-ENSMA, Poitiers, France
Mária Bieliková	Slovak University of Technology in Bratislava
Stefan Biffl	Vienna University of Technology
Miklós Biró	Software Competence Center Hagenberg
Joan Boyar	University of Southern Denmark
Stephane Bressan	National University of Singapore
Francesco Buccafurri	University of Reggio Calabria, Italy

Davide Buscaldi	LIPN, Université Paris 13, Sorbonne Paris Cité
Barbara Catania	DIBRIS-University of Genoa, Italy
Alfredo Cuzzocrea	University of Trieste
Jurek Czyzowicz	Université du Québec en Outaouais
Adam Dabrowski	Poznan University of Technology
Johann Eder	Alpen Adria Universität Klagenfurt
Michele Flammini	Gran Sasso Science Institute & University of L'Aquila
Paola Flocchini	University of Ottawa
Pierre Fraigniaud	CNRS and University Paris Diderot
Johann Gamper	Free University of Bozen-Bolzano
Leszek Gasieniec	University of Liverpool
Pawel Gawrychowski	University of Wroclaw
Loukas Georgiadis	University of Ioannina
Giovanna Guerrini	DIBRIS - University of Genoa, Italy
Inge Li Gørtz	Technical University of Denmark
Yo-Sub Han	Yonsei University
Theo Härder	TU Kaiserslautern
Gabriel Istrate	West University of Timisoara and the eAustria Research Institute
Mirjana Ivanovic	University of Novi Sad, Faculty of Sciences, Department of Mathematics and Informatics
Johan Jeuring	Open Universiteit Nederland and Universiteit Utrecht
Jarkko Kari	University of Turku
Ralf Klasing	CNRS and University of Bordeaux
Dennis Komm	ETH Zurich, Department of Computer Science
Georgia Koutrika	Athena Research Center
Rastislav Královič	Comenius University, Bratislava, Slovakia
Orna Kupferman	Hebrew University
Martin Kutrib	Institut für Informatik, Universität Giessen
Martin Lange	University of Kassel
Julia Lawall	Inria/LIP6
Andrzej Lingas	Lund University
Lech Madeyski	Wroclaw University of Science and Technology
Yannis Manolopoulos	Open University of Cyprus
Tomáš Masopust	Palacký University, Olomouc
Elvira Mayordomo	Universidad de Zaragoza
Paolo Missier	Newcastle University
Nelma Moreira	University of Porto
Xavier Muñoz	Universitat Politècnica de Catalunya
Jerzy Nawrocki	Poznan University of Technology
Boris Novikov	St.-Pegersburg University
Mirosław Ochodek	Poznan University of Technology
Dana Pardubská	Comenius University, Bratislava, Slovakia

Andrea Pietracaprina	University of Padova
Giovanni Pighizzini	University of Milan
Andrei Popescu	Middlesex University London
Rajeev Raman	University of Leicester
Gunter Saake	University of Magdeburg
Philippe Schnoebelen	CNRS
Shinnosuke Seki	The University of Electro-Communications
Arseny Shur	Ural Federal University
Daniel Stefankovic	University of Rochester
Krzysztof Stencel	University of Warsaw
Ernest Teniente	Universitat Politècnica de Catalunya
Martin Theobald	University of Luxembourg
Panos Vassiliadis	University of Ioannina
Valentino Vranić	Slovak University of Technology in Bratislava
Dorothea Wagner	Karlsruhe Institute of Technology
Bruce Watson	Stellenbosch University
Abuzer Yakaryilmaz	University of Latvia
Tomoyuki Yamakami	University of Fukui
Christos Zaroliagis	Computer Technology Institute and Department of Computer Engineering & Informatics
Norbert Zeh	Dalhousie University
Wolf Zimmermann	Martin Luther University Halle-Wittenberg

Organization Chair

Dana Pardubská	Comenius University, Bratislava, Slovakia

Local Organizers

Vanda Hambálková	OKAT PLUS s.r.o.
Marek Nagy	Comenius University, Bratislava, Slovakia
Richard Ostertág	Comenius University, Bratislava, Slovakia
Oldřich Ulrych	Charles University, Prague, Czech Republic

Organizing Institutions

Slovak Society for Computer Science

Faculty of Mathematics, Physics, and Informatics, Comenius University in Bratislava, Slovakia

Additional Reviewers

Aceto, Luca
Aloisio, Alessandro
Badouel, Eric
Balkenius, Christian
Bampas, Evangelos
Barth, Lukas
Berg, Christian
Bilò, Davide
Böckenhauer, Hans-Joachim
Cho, Da-Jung
Czyzewski, Andrzej
Das, Shantanu
Demri, Stéphane
Dimokas, Nikos
Fabrega, Josep
Fiol, Miquel Angel
Forišek, Michal
Fribourg, Laurent
Gainutdinova, Aida
Georgiou, Konstantinos
Giannis, Konstantinos
Gualandi, Stefano
Guillon, Bruno
Hundeshagen, Norbert
Jajcayova, Tatiana
Kapoutsis, Christos
Kawachi, Akinori
Killick, Ryan
Kim, Hwee
Ko, Sang-Ki
Komusiewicz, Christian
Kowaluk, Mirosław
Krüger, Jacob
Kľuka, Ján
Li, Yang
Loff, Bruno
Luettgen, Gerald
Malcher, Andreas
Mchedlidze, Tamara
Meister, Andreas
Mráz, František
Noceti, Nicoletta
Pająk, Dominik
Papadopoulos, Charis
Plachetka, Tomáš
Prigioniero, Luca
Prūsis, Krišjānis
Rafailidis, Dimitrios
Raghavendra Rao, B. V.
Rástočný, Karol
Rogalewicz, Adam
Rovetta, Stefano
Salehi, Özlem
Salo, Ville
San Felice, Mário César
Schmidt, Paweł
Toft, Bjarne
Tomás, Ana Paula
Tsichlas, Kostas
Tzouramanis, Theodoros
Ueckerdt, Torsten
van Ee, Martijn
Villagra, Marcos
Vinci, Cosimo
Wehnert, Sabine
Wendlandt, Matthias
Wong, Tom
Zetzsche, Georg

Sponsors

Softec, spol. s r.o.

Table of Contents

Private-coin verification with magic coins

Maksims Dimitrijevs and Abuzer Yakaryılmaz

University of Latvia, Faculty of Computing
Raiņa bulvāris 19, Rīga, LV-1586, Latvia

University of Latvia, Center for Quantum Computer Science
Raiņa bulvāris 19, Rīga, LV-1586, Latvia

md09032@lu.lv, abuzer@lu.lv

Abstract. We systematically investigate the minimal resources for different types of probabilistic machines that can define uncountably many languages with bounded error. In this paper, we focus on private-coin interactive proof systems. We show that constant-space machines can verify uncountably many unary languages in quadratic time and binary languages in linear time. We also provide protocols for every language and we obtain logarithmic space-bound for unary languages and linear space-bound for binary languages.

1 Introduction

Probabilistic and quantum machines can recognize uncountably many languages with bounded error when using real number transitions [1, 3, 4, 12]. The idea is to encode an infinite sequence as a transition value and then determine its i-th bit by using some probabilistic or quantum experiments.

For quantum models [12], poly-time constant-space is enough for recognizing uncountably many languages. In exponential and double-exponential time, constant-space quantum machines can also verify any unary and binary languages, respectively. For probabilistic models, we have investigated sublogarithmic space two-way and realtime probabilistic Turing machines (PTMs) and probabilistic counter machines [3, 4]. Some highlighted results are as follows: poly-time double-logarithmic space PTMs can recognize uncountably many unary languages. For binary languages, the same result was obtained for any arbitrarily small non-constant space (but in exponential time). For realtime machines, we obtain logarithmic space for unary languages and double logarithmic space for binary languages. We refer the reader to [3, 4] for the other results and details.

In this paper, we turn our attention to the verification power of probabilistic machines and we investigate private-coin interactive proof systems (IPSs) for uncountably many languages and for all languages.

We show that constant-space PTMs can verify uncountably many languages in linear time. Then, we follow the same result also for unary languages in quadratic expected time. For this purpose, we also present new protocols for

two nonregular unary languages. After that, we focus on the verification of every language. We show that logarithmic-space PTMs can verify every unary language with bounded error in exponential time and linear-space PTMs can verify every binary language in double-exponential time.

After the background (Section 2), we present our results under three subsections (Section 3). In Section 3.1, we present our constant-space protocols for two unary and one binary languages that are used in Sections 3.2 and 3.3. In Section 3.2, we present our constant-space protocols verifying uncountably many languages. In Section 3.3, we present our logarithmic and linear space protocols for all unary and binary languages, respectively.

2 Background

We assume the reader is familiar with the basics of complexity theory and automata theory. Throughout the paper, Σ not containing symbols ¢ (the left end-marker) and \$ (the right end-marker) denotes the input alphabet, $\tilde{\Sigma}$ is the set $\Sigma \cup \{¢, \$\}$, Γ not containing symbol # (the blank symbol) denotes the work tape alphabet, $\tilde{\Gamma}$ is the set $\Gamma \cup \{\#\}$, Υ is the communication alphabet, and Σ^* is the set of all strings (including the empty string ε) defined over Σ. We order the elements of Σ^* lexicographically and then represent the i-th element by $\Sigma^*(i)$ where the first value $\Sigma^*(1)$ is the empty string. For any natural number i, $bin(i)$ denotes the unique binary representation and $(bin(i))^r$ denotes the reverse binary representation.

An interactive proof system (IPS) [9, 2] is composed of a prover (P) and a probabilistic verifier (V) and denoted as pair (P,V). The aim of the verifier is to make a decision on a given input by also using communication with the prover. The aim of the prover (assumed to have unlimited computational power) is to convince the verifier to make positive decision. Therefore, the verifier should be able to verify the correctness of the information (proof) provided by the prover since the prover may be cheating when the decision should be negative.

In this paper, we focus on memory-bounded verifiers [6] and so our verifiers are space–bounded probabilistic Turing machines. The verifier has two tapes: The read-only input tape and read/write work tape. The communication between the prover and verifier is done via a communication cell holding a single symbol. The prover can see only the given input and the symbols written on the communication cell by the verifier. The prover may know the program of the verifier but cannot know which probabilistic choices are done by the verifier. Since the outcomes of probabilistic choices are hidden from the prover, such IPS is named private-coin. If the probabilistic outcomes are not hidden (sent via the communication channel), then it is named public-coin, i.e. the prover can have complete information about the verifier during the computation. Public-coin IPS is also known as Arthur-Merlin games [2].

Due to communications with the prover, the program of the verifier with the possible communications is called protocol. A private-coin protocol is called one-way if the verifier always sends the same symbol to the prover. In this case,

we can assume that the prover provides a single string (possible infinite) and this string is consumed in every probabilistic branch.[1]

A space-bounded probabilistic verifier V is a space-bounded probabilistic Turing machine (PTM) extended with a communication cell. Formally, V is a 8-tuple

$$(S, \Sigma, \Gamma, \Upsilon, \delta, s_1, s_a, s_r),$$

where

1. S is the finite set of internal states composed by three disjoint sets of states, the set of reading states (S_r), the set of communicating states (S_c), and the set of halting states (S_h),
2. $s_1 \in S_r$ is the initial state,
3. $s_a \in S_h$ and $s_r \in S_h$ ($s_a \neq s_r$) are the accepting and rejecting states, respectively, and,
4. δ is the transition function composed by δ_c and δ_r that are responsible for the transitions when in a communicating state and in a reading state, respectively.

There is no transition from s_a or s_r since when V enters a halting state (s_a or s_r), the computation is terminated.

In a communicating state, say $s \in S_c$, the transition is very simple: V writes symbol $\tau_s \in \Upsilon$ on the communication cell (τ_s depends only on the current internal state) and the prover writes back a symbol, say $\tau \in \Upsilon$. Then, V switches to state $s' = \delta_c(s, \tau) \in S$.

When in a reading state, V behaves as an ordinary PTM:

$$\delta_r \colon S_r \times \tilde{\Sigma} \times \tilde{\Gamma} \times S \times \tilde{\Gamma} \times \{\leftarrow, \downarrow, \rightarrow\} \times \{\leftarrow, \downarrow, \rightarrow\} \to [0, 1].$$

That is, when V is in reading state $s \in S_r$, reads symbol $\sigma \in \tilde{\Sigma}$ on the input tape, and reads symbol $\gamma \in \tilde{\Gamma}$ on the work tape, it enters state $s' \in S$, writes $\gamma' \in \tilde{\Gamma}$ on the cell under the work tape head, and then the input tape head is updated with respect to $d \in \{\leftarrow, \downarrow, \rightarrow\}$ and the work tape head is updated with respect to $d' \in \{\leftarrow, \downarrow, \rightarrow\}$ with probability

$$\delta(s, \sigma, \gamma, s', \gamma', d, d'),$$

where "$\leftarrow$" ("$\downarrow$" and "$\rightarrow$") means the head is moved one cell to the left (the head does not move and the head is moved one cell to the right). To be a well-formed PTM, the following condition must be satisfied: For each triple $(s, \sigma, \gamma) \in S_r \times \tilde{\Sigma} \times \tilde{\Gamma}$,

$$\sum_{s' \in S, \gamma' \in \tilde{\Gamma}, d \in \{\leftarrow, \downarrow, \rightarrow\}, d' \in \{\leftarrow, \downarrow, \rightarrow\}} \delta(s, \sigma, \gamma, s', \gamma', d, d') = 1.$$

[1] It is also possible that this string (certificate) is placed on a separate one-way read-only tape (certificate tape) at the beginning of the computation and the verifier can read the certificate from this tape.

In other words, all outgoing transitions for the triple (s, σ, γ) have total probability of 1.

The computation starts in state s_1, and any given input, say $w \in \Sigma^*$, is placed as $\tilde{w} = ¢w\$$ on the input tape. It must be guaranteed that the input head never leaves $\tilde{w}$.

The space used by V on w is the number of all cells visited on the work tape during the computation with some non-zero probability. The verifier V is called to be $O(s(n))$ space bounded machine if it always uses $O(s(n))$ space on any input with length $n \geq 0$.

Any verifier without work tape is a constant-space verifier or two–way probabilistic finite state automaton verifier (2PFA verifier).

A two–way model is called sweeping if the direction of the input head can be changed only on the end-markers. If the input head is not allowed to move to the left, then the model is called "one–way".

Without communication, a verifier is a PTM. For a PTM, we simply remove the components related to communication in the formal definition (and so PTM does not implement any communicating transition). A constant-space PTM is a 2PFA [10, 7]. Its one-way version [11] is abbreviated as PFA.

A language $L \subseteq \Sigma^*$ is verifiable by a verifier V with error bound $\epsilon < \frac{1}{2}$ if there exists a prover P such that

1. any $x \in L$ is accepted by V with probability at least $1 - \epsilon$ by communicating with P, and,
2. any $x \notin L$ is always rejected by V with probability at least $1 - \epsilon$ when communicating with any possible prover (P*).

The first property is known as completeness and the second one known as soundness. Generally speaking, completeness means there is a proof for a true statement and soundness means no proof works for a false statement.

It is also said that there is an IPS (P,V) with error bound ϵ for language L. Remark that all the time and memory bounds are defined for the verifier since we are interested in the verification power of the verifier with limited resources.

The case when every member is accepted with probability 1 is also called as perfect completeness.

When there is no communication, then we use term recognition instead of verification.

We denote the set of integers $\mathbb{Z}$ and the set of positive integers $\mathbb{Z}^+$. The set $\mathcal{I} = \{I \mid I \subseteq \mathbb{Z}^+\}$ is the set of all subsets of positive integers and so it is an uncountable set like the set of real numbers ($\mathbb{R}$). The cardinality of $\mathbb{Z}$ or $\mathbb{Z}^+$ is $\aleph_0$ (countably many).

The membership of each positive integer for any $I \in \mathcal{I}$ can be represented as a binary probability value:

$$p_I = 0.x_1 01 x_2 01 x_3 01 \cdots x_i 01 \cdots , \quad x_i = 1 \leftrightarrow i \in I.$$

Similarly, the membership of each string for language $L \subseteq \Sigma^*$ is represented as a binary probability value:

$$p_L = 0.x_1 01 x_2 01 x_3 01 \cdots x_i 01 \cdots , \quad x_i = 1 \leftrightarrow \Sigma^*(i) \in L.$$

The coin landing on head with probability p_I (resp., p_L) is named as $\mathtt{coin}_I$ (resp., $\mathtt{coin}_L$).

3 Our results

We use a fact presented in our previous paper [3].

Fact 1 *[3] Let $x = x_1x_2x_3\cdots$ be an infinite binary sequence. If a biased coin lands on head with binary probability value $p = 0.x_101x_201x_301\cdots$, then the value x_k is determined correctly with probability at least $\frac{3}{4}$ after 64^k coin tosses, where x_k is guessed as the $(3k+3)$-th digit of the binary number representing the total number of heads after the whole coin tosses.*

3.1 Constant-space verification of nonregular languages

In this subsection, we present two nonregular unary languages and one nonregular binary language that can be verified by 2PFAs in quadratic and linear time, respectively. The protocols presented here will be also used in the next section.

Theorem 1. $\mathtt{USQUARE} = \{a^{m^2} \mid m > 0\}$ *is verifiable by a 2PFA in quadratic expected time with bounded error.*

Proof. The protocol is one-way and the verifier expects from the prover a string of the form

$$(a^m b)^m b$$

for the members of the language, where $m > 0$.

Let $w = a^n$ be the given input for $n > 3$ (the decisions on the shorter strings are given deterministically) and let y be the string provided by the prover. The verifier deterministically checks whether y is of the form

$$y = a^{m_1}ba^{m_2}b\cdots ba^{m_i}b\cdots \text{ or } y = a^{m_1}ba^{m_2}b\cdots ba^{m_t}bb$$

for some $t > 0$. If the verifier sees a defect on y, then the input is rejected.

In the remaining part, we assume that y is in one of these forms. At the beginning of the computation, the verifier places the input head on the left endmarker and splits computation in four paths with equal probabilities.

In the first path, the verifier reads w and y in parallel and checks whether y is finite, i.e.

$$y = a^{m_1}ba^{m_2}b\cdots ba^{m_t}bb,$$

and whether it satisfies the equality $n = \sum_{j=1}^{t} m_j$, where $t > 1$. If one of the checks fails, the input is rejected. Otherwise, it is accepted.

The second path is very similar to the first path and the following equality is checked:

$$n = \sum_{j=2}^{t} m_j + \sum_{j=1}^{t} 1,$$

i.e., the verifier skips a^{m_1} from y and counts b's instead. If the equality is satisfied, the input is accepted. Otherwise, it is rejected.

The computation in the first and second paths is deterministic (a single decision is given in each) and both paths terminate in linear time.

In the third path, the verifier tries to make the following consecutive comparisons:

$$m_1 = m_2, m_3 = m_4, \ldots, m_{2j-1} = m_{2j}, \ldots.$$

For each j, the verifier can easily determine whether $m_{2j-1} = m_{2j} < n$ by attempting to move the input head to the right by m_{2j-1} squares and then to the left by m_{2j} squares. If the right end-marker is visited ($m_{2j-1} \geq n$), or the left end-marker is visited earlier than expected ($m_{2j} > m_{2j-1}$) or is not visited ($m_{2j} < m_{2j-1}$), then the comparison is not successful and so the input is rejected. Otherwise, the comparison is successful and the verifier continues with a random walk (described below) before the next comparison, except that if the last comparison is successful, then the input is accepted without making the random walk.

The aim of the random walk is to determine whether the prover sends a finite string or not, i.e. the prover may cheat by sending the infinite string $(a^m b)^*$ for some $m < n$ which passes successfully all comparison tests described above.

The random walk starts by placing the input head on the first symbol of the input and terminates after hitting one of the end-markers. During the random walk, the verifier pauses the reading of the string y. It is a well known fact that this walk terminates in $O(n)$ expected number of steps and the probability of ending on the right (resp., the left) end-marker is $\frac{1}{n}$ (resp., $1 - \frac{1}{n}$).

If the walk ends on the left end-marker, then the verifier continues with the next comparison. If the walk ends on the right end-marker, the verifier checks whether the number of a's in the remaining part of y is less than n or not by reading whole input from right to left. If it is less than n, then the input is accepted. Otherwise (y contains more than n a's), the input is rejected. In any case, the computation is terminated with probability $\frac{1}{n}$ after the walk.

The fourth path is identical to the third path by shifting the comparing pairs: The verifier tries to make the following consecutive comparisons:

$$m_2 = m_3, m_4 = m_5, \ldots, m_{2j} = m_{2j+1}, \ldots.$$

Now, we can analyse the overall protocol. If $n = m^2$ for some $m > 1$, then the prover provides $y = (a^m b)^m b$ and the input is accepted in every path and so the overall accepting probability is 1. Thus, every member is accepted with probability 1. Moreover there will be at most m random walks and so the overall running time is $O(n\sqrt{n})$.

If the input is not a member, then the input is rejected in at least one of the paths. If it is rejected in the first or second path, then the overall rejecting probability is at least $\frac{1}{4}$. If it is rejected in the third or fourth paths, then the overall rejecting probability cannot be less than $\frac{3}{16}$ as explained below.

We assume that the input is not rejected in the first and second paths. Then, we know that y is finite, the number of a's in y is n, and y is composed by m_1

blocks. Since n is not a perfect square, there is at least one pair of consecutive blocks that have different number of a's. Hence, at least one of the comparisons will not be successful, and, the input will be rejected in one of these paths. Let l be the minimum index such that the comparison of the l-th pair is not successful (in the third or fourth path). Then, $l < \frac{\sqrt{n}}{2}$. (If not, y contains at least $2\left\lceil \frac{\sqrt{n}}{2} \right\rceil$ blocks and each of these blocks contains $m_1 = \lceil \sqrt{n} \rceil$ a's, and, this implies that y contains more than n a's.) Then, the maximum accepting probability in the corresponding path is bounded from above by

$$\sum_{i=1}^{l} \frac{1}{n}\left(1-\frac{1}{n}\right)^{i-1} = 1-\left(1-\frac{1}{n}\right)^{l} < 1-\left(1-\frac{1}{n}\right)^{\frac{\sqrt{n}}{2}} \leq \frac{1}{4}.$$

(Remember that $n > 3$.) Therefore, the rejecting probability in the third or fourth path is at least $\frac{3}{4}$, and so, the overall rejecting probability cannot be less than $\frac{1}{4} \cdot \frac{3}{4} = \frac{3}{16}$.

The maximum (expected) running time occurs when the prover sends the infinite $y = (a^m b)^*$ for some $m > 0$. In this case, the protocol is terminated in the third and fourth paths with probability 1 after $O(n)$ random walks, and so, the expected running time is quadratic in n, $O(n^2)$.

By repeating the protocol above many times, say $r > 1$, we obtain a new protocol such that any non-member is rejected with probability arbitrarily close to 1, i.e. $1 - \left(1 - \frac{3}{16}\right)^r$. □

Theorem 2. $\texttt{UPOWER64} = \{a^{64^m} \mid m > 0\}$ *is verifiable by a 2PFA with bounded error in quadratic expected time.*

Proof. See [5] for the proof. □

We continue with the verification of a binary nonregular language, a modified version of `DIMA` [3]: $\texttt{DIMA2} = \{0^{2^0}10^{2^1}10^{2^2}1\cdots10^{2^{3k-1}}11(0^{2^{3k}}1)^{2^{3k}} \mid k > 0\}$.

Theorem 3. `DIMA2` *is verifiable by a sweeping PFA in linear time with bounded error.*

Proof. See [5] for the proof. □

3.2 Constant-space verification of uncountably many languages

In this subsection, we present two constant-space protocols for verifying uncountably many unary and binary languages.

Theorem 4. *Bounded-error sweeping PFAs can verify uncountably many languages in linear time.*

Proof. Let w_k be the k-th shortest member of `DIMA2` for $k > 0$. For any $I \in \mathcal{I}$, we define the following language:

$$\texttt{DIMA2}(I) = \{w_k \mid k > 0 \text{ and } k \in I\}.$$

We describe a one-way protocol for $\texttt{DIMA2}(I)$. Let w be the given input. The verifier determines whether $w = w_k$ for some $k > 0$ by using the protocol for DIMA2 with high probability. If not, then the input is rejected. In the remaining part, we continue with

$$w = w_k = 0^{2^0}10^{2^1}10^{2^2}1 \cdots 10^{2^{3k-1}}11(0^{2^{3k}}1)^{2^{3k}}.$$

The verifier attempts to toss $\texttt{coin}_I$ 64^k times and in parallel processes the total number of heads for determining x_k in p_I. The verifier asks from the prover to send $0^{64^k}1$ and the prover sends $y = 0^n1$ (the input is deterministically rejected if y is not in this form).

The verifier splits into two paths with equal probabilities. In the first path, it easily determines whether $n = 64^k$ by passing over the input once (the number of 0's after symbols "11" is 64^k). If $n \neq 64^k$, then the input is rejected. Otherwise, the input is accepted.

The second path is responsible for coin-tosses and processing the total number of heads. The verifier performs n coin-tosses. For counting the heads, the verifier uses the part of w_k after symbols "11" as a read-only counter, which is composed of 8^k blocks of 0's and length of each block is 8^k. Let t be the total number of heads:

$$t = i \cdot 8^{k+1} + j \cdot 8^k + q = (8i + j)8^k + q,$$

where $i \geq 0$, $j \in \{0, \ldots, 7\}$, and $q < 8^k$. Due to Fact 1, x_k is the $(3k + 3)$-th digit of $bin(t)$ with probability at least $\frac{3}{4}$. In other words, x_k is guessed as 1 if $j \in \{4, \ldots, 7\}$, and as 0, otherwise. The verifier sets $j = 0$ at the beginning. Then, for each head, it reads a symbol 0 from the input and after 8^k heads it updates j as $(j + 1) \mod 8$. If the number of heads exceeds 64^k, then the input is rejected. If not, the decision given is parallel to the value of j: The input is accepted if $j \in \{4, \ldots, 7\}$ and rejected if $j \in \{0, \ldots, 3\}$.

The verifier operates in sweeping mode and each path terminates in linear time. If w is a member, then the input is accepted with probability at least $\frac{3}{4}$. If $w \notin \texttt{DIMA2}$, then it is rejected with high probability. If $w \in \texttt{DIMA2}$ but $w \notin \texttt{DIMA2}(I)$, then the input is rejected with probability at least $\frac{1}{2} \cdot \frac{3}{4} = \frac{3}{8}$. By repeating the protocol, the rejecting probability can get arbitrarily close to 1.

Since the cardinality of set $\{I \mid I \in \mathcal{I}\}$ is uncountable, there are uncountably many languages in $\{\texttt{DIMA2}(I) \mid I \in \mathcal{I}\}$, each of which is verified by a bounded-error linear-time sweeping PFA. □

Theorem 5. *2PFAs can verify uncountably many unary languages with bounded error in quadratic expected time.*

Proof. Here, we use all protocols given in the proofs of Theorems 1, 2, and 4.

Let w_k be the k-th shortest member of UPOWER64 for $k > 0$. For any $I \in \mathcal{I}$, we define language:

$$\texttt{UPOWER64}(I) = \{w_k \mid k > 0 \text{ and } k \in I\}.$$

We construct a verifier for this language.

Let w be the given input. By using the protocol given for UPOWER64, the verifier can determine whether $w \in$ UPOWER64 or not. If $w \notin$ UPOWER64, then the input is rejected with high probability.

In the following part, we assume that $w = w_k$ for some $k > 0$. The verifier asks from the prover the following string $y_k = (a^{8^k} b)^{8^k} b$. Let y be the string provided by the prover. The verifier splits into two paths with equal probabilities. In the first path, it checks whether $y = y_k$ by using the protocol for USQUARE. In the second path, the verifier assumes that $y = y_k$ and implements the part of the protocol for DIMA2(I), which is responsible for the coin tosses and for determining whether $k \in I$ or not by checking the total number of heads. The verifier reads w for 64^k coin tosses and y for determining the value x_k.

If the prover sends y_k, then the verifier correctly determines whether $k \in I$ or not with probability at least $\frac{3}{4}$ in the second path. If $w \in$ UPOWER64(I), the honest prover sends y_k and so the input is accepted with probability 1 in the first path and accepted with probability at least $\frac{3}{4}$ in the second path.

If $w \notin$ UPOWER64(I), then the input is rejected with probability at least $\frac{3}{16}$ if the prover does not send y_k in the first path, and rejected with probability at least $\frac{3}{4}$ in the second path if the prover does send y_k. Therefore, the overall rejecting probability is at least $\frac{3}{32}$.

Since each called protocol runs no more than quadratic expected time in $|w|$, the expected running time is quadratic. By repeating the protocol many times, we obtain a protocol with better success probability. □

3.3 Verification of all languages

We start with trivial results. Then, we present our protocols for all languages.

Theorem 6. *Any unary language is recognizable by a linear-space PTMs with bounded error.*

Proof. Let $\Sigma = \{a\}$ be our alphabet. For any unary language $L \subseteq \Sigma^*$, we design a PTM for L, say P_L, that uses $\mathtt{coin}_L$.

Let $w = a^n$ be the given input for $n \geq 0$. PTM P_L implements the procedure described in Fact 1 in a straightforward way and gives its decision accordingly, which will be correct with probability not less than $\frac{3}{4}$. The machine only uses linear-size binary counters to implement 64^k coin tosses and to count the number of heads (for unary L and $\mathtt{coin}_L$, $k = n + 1$). By repeating the same procedure, the success probability is increased arbitrarily close to 1. □

Remark 1. Let $L \subseteq \Sigma^*$ be a k-ary language for $k > 1$, where $\Sigma = \{a_1, \ldots, a_k\}$. For any given k-ary string $w \in \Sigma^*$, let x_l represent its membership bit in p_L. Then, by using the exactly the same algorithm given in the above proof, we can determine x_l correctly with high probability. However, l is exponential in $|w|$ and so the PTM uses exponential space.

Corollary 1. *Any k-ary ($k > 1$) language is recognizable by a exponential-space PTMs with bounded error.*

When interacting with a prover, we can reduce the above space bounds. For this purpose, we use the probabilistic fingerprint method: For comparing two s-bit numbers, say m and n, we can randomly pick a $(c \log s)$-bit prime number p for some positive integer c and compare $m' = m \mod p$ with $n' = n \mod p$. Being verifiable from the fact given below, this reduction works with high probability.

Fact 2 *Let $P_1(n)$ be the number of primes not exceeding $2^{\lceil log_2 n \rceil}$, $P_2(l, N', N'')$ be the number of primes not exceeding $2^{\lceil log_2 l \rceil}$ and dividing $|N' - N''|$, and $P_3(l, n)$ be the maximum of $P_2(l, N', N'')$ over all $N' < 2^n$, $N'' \leq 2^n$, $N' \neq N''$. Then, for any $\epsilon > 0$, there is a natural number c such that $\lim_{n\to\infty} \frac{P_3(cn,n)}{P_1(cn)} < \epsilon$. [8]*

Theorem 7. *Any unary language $L \subseteq \{a\}^*$ is verifiable by a log-space PTM with bounded error.*

Proof. The protocol is two-way. Let $w = a^n$ be the given input for $n > 0$. (The decision on the empty string is given deterministically.) Remember that the membership bit of a^n is x_{n+1} in p_L. Let $k = n + 1 \geq 2$. We pick a value of c (see Fact 2) satisfying the error bound $\frac{1}{8}$.

The protocol has three phases. In the first phase, there is no communication. The verifier picks two random $(c \cdot (4 \cdot \log k))$-bit prime numbers, say p_1 and p_2, and then, it calculates and stores $r_1 = 64^k \mod p_1$ in binary on the work tape. The verifier also prepares two binary counters C_1 and C_2 for storing the total number of coin tosses modulo p_1 and the total number of heads modulo p_2, respectively, and one "halting" counter $C_h = 0$.

In the second phase, the verifier asks from the prover to send $a^{64^k}b$. Once the verifier receives symbol b, the communication is ended in this phase. Therefore, we assume that the prover sends either the finite string $y = a^m b$ for some $m \geq 0$ or an infinite sequence of a's.

For each a received from the prover, the verifier reads the whole input and adds one to C_h with probability $\left(\frac{1}{64}\right)^k$. We call it a halting walk. If $C_h = 8$, the verifier terminates the computation and rejects the input. If the computation is not terminated, the verifier tosses $\mathtt{coin}_L$, sends the result to the prover, and increases C_1 by 1. If the result is heads, the verifier also increases C_2 by 1. When the second part is ended, the verifier checks whether previously calculated r_1 is equal to $r_1' = m \mod p_1$, stored on C_1. If they are not equal, then the input is rejected. In other words, if the prover does not send 64^k a's, then the verifier detects this with probability at least $\frac{7}{8}$.

Let r_2 be the binary value stored on C_2 and t be the total number of heads obtained in the second phase. In the third phase, the verifier asks from the prover to send $(bin(t))^r$ (the least significant bits are first). By using the input head, the verifier easily reads the $(3k + 3)$-th bit of $bin(t)$, say x_k', and also checks whether the length of $bin(t)$ does not exceed $(6k + 1)$. Meanwhile, the verifier also calculates $r_2' = t \mod p_2$. If the length of $bin(t)$ is greater than $6k + 1$ or $r_2 \neq r_2'$, then the input is rejected. In other words, if the prover does not

send $bin(t)$, then the verifier can catch it with probability $\frac{7}{8}$. At the end of third phase, the verifier accepts the input if x'_k is 1, and, rejects it if x'_k is 0.

According to Chebyshev's inequality, the value of C_h reaches 8 after more than $16 \cdot 64^k$ a's with probability

$$Pr[|X - E[X]| \geq 9] \leq \frac{(\frac{1}{64^k}) \cdot (1 - \frac{1}{64^k}) \cdot 16 \cdot 64^k}{9^2} < \frac{16}{81},$$

where $E[X]$ is expected value of C_h. This bound is important, since, for $m > 16 \cdot 64^k$, Fact 2 cannot guarantee the error bound $\frac{1}{8}$. (Remember that prime numbers p_1 and p_2 do not exceed $2^{c \cdot (4 \cdot \log k)} \geq 2^{c \cdot (\log(6 \cdot k + 4))}$ for $k \geq 2$.)

If w is not a member, then the accepting probability can be at most $\frac{209}{648}$.

- If $m \neq 64^k$, then the input is rejected with probability at least $\frac{7}{8} - \frac{16}{81}$, and so the accepting probability cannot be greater than $\frac{209}{648}$.
- Assume that $m = 64^k$. If the prover does not send $(bin(t))^r$, then the input is rejected with probability $\frac{7}{8}$, and so the accepting probability cannot be greater than $\frac{1}{8}$.
- Assume that $m = 64^k$ and the verifier sends $(bin(t))^r$, then the input is accepted with probability at most $\frac{1}{4}$.

The expected running time for the non-members is exponential in n due to the halting walks.

If w is a member, then the honest prover sends all information correctly, and the verifier guesses x_{n+1} correctly with probability at least $\frac{3}{4}$ if the computation is not terminated by halting walks. The probability of halting the computation (and rejecting the input) in the second phase is

$$Pr[|X - E[X]| \geq 7] \leq \frac{(\frac{1}{64^k}) \cdot (1 - \frac{1}{64^k}) \cdot 64^k}{7^2} < \frac{1}{49}.$$

Therefore, the verifier accepts w with probability at least $\frac{143}{196}$. The expected running time for members is also exponential in n.

The verifier uses $O(\log n)$ space and the success probability is increased by repeating the same algorithm. □

Due to Remark 1, we can follow the same result also for k-ary languages with exponential increase in time and space.

Corollary 2. *Any k-ary ($k > 1$) language $L \subseteq \{a_1, \ldots, a_k\}^*$ is verifiable by a linear-space PTM with bounded error.*

Acknowledgments

We thank to the anonymous reviewers for their helpful comments. Dimitrijevs is partially supported by University of Latvia projects AAP2016/B032 "Innovative information technologies" and ZD2018/20546 "For development of scientific activity of Faculty of Computing". Yakaryılmaz is partially supported by ERC Advanced Grant MQC.

References

1. Adleman, L.M., DeMarrais, J., Huang, M.D.A.: Quantum computability. SIAM Journal on Computing 26(5), 1524–1540 (1997)
2. Babai, L.: Trading group theory for randomness. In: STOC'85: Proceedings of the 17th Annual ACM Symposium on Theory of Computing. pp. 421–429 (1985)
3. Dimitrijevs, M., Yakaryılmaz, A.: Uncountable classical and quantum complexity classes. In: Eigth Workshop on Non-Classical Models for Automata and Applications. books@ocg.at, vol. 321, pp. 131–146. Austrian Computer Society (2016)
4. Dimitrijevs, M., Yakaryılmaz, A.: Uncountable realtime probabilistic classes. In: Descriptional Complexity of Formal Systems. LNCS, vol. 10316, pp. 102–113. Springer (2017)
5. Dimitrijevs, M., Yakaryılmaz, A.: Probabilistic verification of all languages. Tech. Rep. 1807.04735, arXiv (2018)
6. Dwork, C., Stockmeyer, L.: Finite state verifiers I: The power of interaction. Journal of the ACM 39(4), 800–828 (1992)
7. Freivalds, R.: Probabilistic two-way machines. In: Proceedings of the International Symposium on Mathematical Foundations of Computer Science. pp. 33–45 (1981)
8. Freivalds, R.: Space and reversal complexity of probabilistic one-way Turing machines. In: Conference on Fundamentals of Computation Theory. pp. 159–170 (1983)
9. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing 18(1), 186–208 (1989)
10. Kuklin, Y.I.: Two-way probabilistic automata. Avtomatika i vyčistitelnaja tekhnika 5, 36–36 (1973), (Russian)
11. Rabin, M.O.: Probabilistic automata. Information and Control 6, 230–243 (1963)
12. Say, A.C.C., Yakaryılmaz, A.: Magic coins are useful for small-space quantum machines. Quantum Information & Computation 17(11&12), 1027–1043 (2017)

Learning Restricted Regular Expressions with Interleaving

Chunmei Dong[1,2], Yeting Li[1,2], Xiaoying Mou[1,2], and Haiming Chen[1] *

[1] University of Chinese Academy of Sciences, Beijing, China
[2] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
{dongcm,liyt,mouxy,chm}@ios.ac.cn

Abstract. The advantages for the presence of an XML schema for XML documents are numerous. However, many XML documents in practice are not accompanied by a schema or by a valid schema. Relax NG is a popular and powerful schema language, which supports the unconstrained interleaving operator. Focusing on the inference of Relax NG, we propose a new subclass of regular expressions with interleaving and design a polynomial inference algorithm. Then we conducted a series of experiments based on large-scale real data and on three XML data corpora, and experimental results show that our subclass has a better practicality than previous ones, and the regular expressions inferred by our algorithm are more precise.

Keywords: schema inference · interleaving · regular expressions · Relax NG · XML documents

1 Introduction

As a main file format for data exchange, the eXtensible Markup Language (XML) has been widely used on the Web [2]. XML schemas define the structure constraints of XML documents. The advantages by the presence of an XML schema for XML documents are numerous, such as for data processing, automatic data integration, static analysis of transformations and so on [4, 12, 25, 28–30, 33]. However, many XML documents are not accompanied by a (valid) schema in practice. Research in 2013 showed that only 24.8% XML documents available on the Web were accompanied with corresponding schemas, of which the proportion of valid ones was only 8.9% [22]. Therefore, it has become an urgent problem to infer a suitable XML schema for given XML documents.

Document Type Definition (DTD), XML Schema Definition (XSD) and Relax NG are three popular XML schema languages. Among them Relax NG is more powerful than both DTD and XSD due to its expressive power [32]. Relax NG schemas support the interleaving operator and allow the interleaving to be mixed

* Work supported by the National Natural Science Foundation of China under Grants No. 61872339 and 61472405

with other operators, which can make the schemas succinct. Furthermore, the interleaving has been used in many applications. For example, it is used in the schema language ShEx for RDF [10, 35], and is necessary in solving many problems [15, 24, 27]. On the other hand, presently researches on interleaving are quite insufficient (for instance, see below). Therefore we concentrate on the study of interleaving, and focus on the inference of Relax NG in this paper. Notice that the results can also be applied to some other applications.

Actually, the major task of schema inference can be reduced to learning regular expressions from a set of given samples [6,8,18]. Gold proposed a classical language learning model (*learning in the limit or explanatory learning*) and pointed out that the class of regular expressions could not be identifiable from positive samples only [21]. Therefore, researches have focused on learning restricted subclasses of regular expressions [31].

For the inference of regular expressions, Bex et al. proposed two subclasses: *single occurrence regular expressions* (SOREs) and *chain regular expressions* (CHAREs), and gave their inference algorithms *RWR* and *CRX* [6, 7]. Freydenberger et al. gave two more efficient algorithms *Soa2Sore* and *Soa2Chare* for the above classes. Kim et al. developed an inference system using hedge grammars for learning Relax NG [23]. However, all of the above work are based on standard regular expressions, which do not support interleaving. Nevertheless, interleaving is vital since there may be no order constraint among siblings in data-centric applications [1], and it has been proved that regular expressions with interleaving are double exponentially more succinct than standard regular expressions [19].

With regard to the regular expressions with interleaving, Ghelli et al. proposed a restricted subclass called "conflict-free REs" supporting interleaving [20], where the subclass requires no symbol appears twice and repetition is only applied to single symbols, and no inference algorithm is provided. Ciucanu and Staworko proposed two subclasses called disjunctive multiplicity expressions (DME) and disjunction-free multiplicity expressions (ME) [9], which support unordered concatenation (a weaker form of interleaving), and disallow concatenation within siblings. The inference algorithm of DME is discussed in [13]. Peng et al. [34] proposed a subclass called the subset of regular expressions with interleaving (SIREs), and gave its inference algorithm based on the maximum independent set.

In this paper, to address the problem of inferencing Relax NG, we propose a restricted subclass of regular expressions with interleaving, named as Improved Subclass of Regular Expressions with Interleaving (ISIREs), and SIREs is a subclass of ISIREs. We also develop its learning algorithm. The main contributions of this paper are listed as follows.

- We propose a new subclass of regular expressions supporting interleaving, which helps the inference of RELAX NGs.
- We develop an inference algorithm InferISIRE to infer ISIREs and analyze its time complexity.
- We calculate the usage proportion of ISIREs and other popular subclasses based on the large-scale real data, and find that the proportion of ISIREs is

the highest, which indicates ISIREs have a better practicality. Then, based on three XML data corpora, we compare the inferred results of InferISIRE with other inference algorithms. Experimental results show that the regular expressions inferred by InferISIRE are more precise.

The rest of this paper is organized as follows. Section 2 presents the basic definitions. Section 3 gives the inference algorithm InferISIRE. Section 4 introduces the experiments. Conclusions are drawn in Section 5.

2 Preliminaries

Definition 1. ***Regular Expression with Interleaving****. Let Σ be a finite alphabet. A string is a finite sequence of symbols over Σ. The set of all finite strings over Σ is denoted by Σ^*. The empty string is denoted by ε. A regular expression with interleaving over Σ is defined inductively as follows: ε or $a \in \Sigma$ is a regular expression, for regular expressions E_1 and E_2, the disjunction $E_1|E_2$, the concatenation $E_1 \cdot E_2$, the interleaving $E_1\&E_2$, or the Kleene-Star E_1^* is also a regular expression. Usually we use E_1E_2 instead of $E_1 \cdot E_2$ for readability. The length of a regular expression E, denoted by $|E|$, is the total number of alphabet symbols and operators occurring in E. The language generated by E is defined as follows: $L(\emptyset) = \emptyset$; $L(\varepsilon) = \{\varepsilon\}$; $L(a) = \{a\}$; $L(E_1^*) = L(E)^*$; $L(E_1E_2) = L(E_1)L(E_2)$; $L(E_1|E_2) = L(E_1) \cup L(E_2)$; $L(E_1\&E_2) = L(E_1)\&L(E_2)$. $E^?$ and E^+ are used as abbreviations of $E|\varepsilon$ and EE^*, respectively.*

Let $u = au_0, v = bv_0$, where $a, b \in \Sigma$ and $u, u_0, v, v_0 \in \Sigma^*$. Then $u\&\varepsilon = \varepsilon\&u = \{u\}$; $u\&v = \{a(u_0\&v)\} \cup \{b(u\&v_0)\}$. For example, the string set generated by $a\&bc$ is $\{abc, bac, bca\}$.

Regular expressions with interleaving, in which each symbol occurs at most once, are called *ISOREs* extended from *SOREs* [6].

Definition 2. ***Single Occurrence Automaton (SOA)*** *[7] Let Σ be a finite alphabet, and let src and snk be distinct symbols that do not occur in Σ. A single occurrence automaton over Σ is a finite directed graph $G=(V, D)$ such that*

- *$src, snk \in V$, and $V \subseteq \Sigma \cup \{src, snk\}$;*
- *src has only outgoing edges, snk has only incoming edges and every node $v \in V$ lies on a path from src to snk.*

A *generalized single occurrence automaton (generalized SOA)* over Σ is defined as a directed graph in which each node $v \in V \backslash \{src, snk\}$ is an *ISORE* and all nodes are pairwise *ISOREs*.

For a given directed acyclic graph $G = (V, E)$ of an SOA, the level number of node v is the longest path from *src* denoted by $ln(v)$, and $ln(src)$ is 0 [16]. For a path v_1 to v_2 , if there exists node v such that $ln(v_1) < ln(v)$ and $ln(v) < ln(v_2)$, then $ln(v)$ is a skip level [16].

Partial Order Relation (POR) is a binary relation which is reflexive, dissymmetric and transitive. It is a collection of ordered pairs of elements over Σ. For

a string $s = s_1 s_2 \cdots s_n$, $s_i \prec s_j$ means s_i occurs before s_j. Using POR, we can construct the Constraint Set (CS) and Non-Constraint Set (NCS) for a set of given samples S. $POR(S)$ is denoted to represent the set of all partial orders obtained from each string $s \in S$. The formula is as follows.

- $CS(S) = \{< a_i, a_j > | < a_i, a_j > \in POR(S), and < a_j, a_i > \in POR(S)\}$;
- $NCS(S) = \{< a_i, a_j > | < a_i, a_j > \in POR(S), but < a_j, a_i > \notin POR(S)\}$.

Definition 3. ***Improved Subclass of Regular Expressions with Interleaving (ISIREs)*** *Let Σ be a finite alphabet and $a \in \Sigma$, the improved subclass of regular expressions with interleaving is defined by the following grammar: $S ::= TS|T$, $T ::= A\&T|A$, $A ::= \varepsilon|a|a^*|AA$.*

Moreover, we require that every symbol $a \in \Sigma$ can occur at most once in the regular expression that belongs to ISIREs.

By definition, it is obvious that ISIREs is a subclass of ISOREs and SIREs [34] is a subclass of ISIREs. For example, $E_1 = a\&b^*c\&de^?$ is an SIRE [34] and also an ISIRE, $E_2 = a^?(bc^?\&d^+e)f^*$ is an ISIRE but not an SIRE [34], and both E_1 and E_2 are ISOREs.

3 Learning Algorithm for ISIREs

In this section, we first introduce our learning algorithm InferISIRE, which infers an ISIRE for a set of given samples S. Then we analyse the time complexity of the algorithm. Finally, we show the learning process of InferISIRE through an example and compare it with other algorithms.

3.1 Algorithm Implementation and Analysis

The algorithm InferISIRE uses a number of subroutines, and we introduce some of them as follows.

- **cntOper(S)**. $cntOper(S)$ is a function to calculate unary operator of every symbol in a set of strings S. The function returns a dictionary of pair $(a : opt)$, where a is symbol in S and opt is an unary operator in $\{1, ?, *, +\}$, and the opt is computed according to the occurences of a in strings set S. For example, given the set of strings $S = \{aabd, abcd, bbcd\}$, $cntOper(s) = \{a : *; b : +; c :?; d : 1\}$.
- **Filter($mis, consist\_tr$)**. For a maximum independent set (MIS) mis and a non-constraint set $consist\_tr$, the function returns a subset of $consist\_tr$ consisting of the ordered pairs of elements over mis. That is, Filter(mis,$consist\_tr$) $=\{< x, y > | < x, y > \in consist\_tr$, and $x \in mis, y \in mis\}$;
- **contract($U, SubRE$)** [16]. For a nodes set U and a subexpression $SubRE$, the contract on an SOA modifies SOA such that all nodes of U are contracted to a single vertex and labeled $SubRE$ (corresponding edges are moved).

The pseudo-code of the inference algorithm InferISIRE is shown in Algorithm 1, which outputs an ISIRE for an input set of strings. We illustrate the main procedures of InferISIRE in the following.

1. Traverse the samples S to get the alphabet Σ, and get the dictionary *dictOper* with the function $cntOper(S)$. This step is shown in line 1.
2. We compute the non-constraint set *consist_tr* and constraint set *constraint_tr*, and construct an undirected graph G using symbols in Σ as nodes and ordered pairs in constraint set as edges. We then get the set *subGraphs* consisting of all maximal connected subgraph of G using function $connected\_subgraph(G)$. The lines 2-4 show the process.
3. In line 6 and line 7, for each maximal connected subgraph of G, namely every *subgraph* in the set *subGraphs*, we transform it to a subexpression with algorithm G2SubRE showed in Algorithm 2 (introduced later). We get the set R of all subexpressions.
4. We use 2T-INF [17] to construct the SOA $\mathcal{A}$ of S. Transform each subexpression $SubRE$ into set of nodes with function $SubRE2Nodes(SubRE)$, then contract the SOA $\mathcal{A}$ with function contract$(U, SubRE)$. After this procedure we actually turn the SOA into a generalized SOA. The steps are shown in lines 9-11.
5. Finally we calculate level number for every nodes of generalized SOA and find all skip levels, if there are more than one & node (node with & operator) with the same level number, or if a level is a skip level, then ? is appended to every chain factor on that level. The lines 12-20 show the process and return an ISIRE in the end.

Here we give a brief explanation of the algorithm G2SubRE, which transforms an undirected graph to a subexpression. For an undirected graph g, the algorithm finds the approximation of maximum independent set (MIS) of g with function $clique\_removal(g)$ [11], then adds it to list *allmis* and deletes the MIS and their related edges from g. The process is repeated until there exist no nodes in g. For each MIS *mis* in *allmis*, we establish a directed graph *dg* with its filtered subset Filter$(mis, consist_tr)$, then we compute the topological sort for all subgraphs of *dg* and add the result to T. Finally, the algorithm returns the subexpression whose corresponding counting operators can be read from *dictOper*.

Algorithm Analysis. Let m denotes the sum length of the input strings in S, and n denotes the number of alphabet symbols. It takes $O(m+n)$ to calculate all partial orders as well as constructing a graph. For a graph $G(V, E)$ with $|V| = n$ and $|E| = m$, it costs time $O(m+n)$ to find all maximal connected subgraphs. The time complexity of $clique\_removal()$ is $O(n^2+m)$. Thus for each subgraphs, computation of *allmis* costs time $O(n^3+m)$. Since the number of maximal connected subgraphs of a graph is finite, the computation of *allmis* for all subgraphs also costs time $O(n^3+m)$. Constructing SOA for S and generalizing SOA can be finished in time $O(m+n)$, and assigning level numbers and computing all skip levels will be finished in time $O(m+n)$. All nodes will be converted into specific chain factors of *ISIRE* in $O(n)$. Therefore, the time complexity of InferISIRE is $O(n^3+m)$.

Algorithm 1: InferISIRE

Input: A set of strings S
Output: An ISIRE

$\Sigma \leftarrow alphabet(S)$; $dictOper \leftarrow cntOper(S)$;
$consist\_tr = CS(S), constraint\_tr \leftarrow NCS(S)$;
$G \leftarrow Graph(\Sigma, constraint\_tr)$;
$subGraphs \leftarrow connected\_subgraph(G)$;
$R \leftarrow \emptyset$; $result \leftarrow \varepsilon$
foreach *subgraph in subGraphs* **do**
    $R.append(G2SubRE(subgraph, consist\_tr, dictOper))$;
Construct the $\mathcal{A} \leftarrow SOA(S)$ using *2T-INF* [17];
foreach *SubRE in R* **do**
    $U \leftarrow SubRE2Nodes(SubRE)$;
    $\mathcal{A} \leftarrow A.contract(U, SubRE)$;
$\mathcal{A}.constructLevelOrder()$;
for $i = 1$ *to* $ln(\mathcal{A}.snk) - 1$ **do**
    $B \leftarrow$ all nodes with level number i and &;
    foreach α *in* B **do**
        if $\mathcal{A}.isSkipLevel(i)$ *or* $|B| > 1$ **then**
            $result \leftarrow result \cdot \alpha^{?}$;
        else $result \leftarrow result \cdot \alpha$;
        ;
return *result*

Algorithm 2: G2SubRE

Input: An undirected graph g, a list $consist\_tr$, a map $dictOper$
Output: A regular expression r

$allmis \leftarrow \emptyset$, $T \leftarrow \emptyset$;
while $g.nodes() > 0$ **do**
    $mis \leftarrow clique\_removal(g)$;
    $g \leftarrow g - mis$;
    $allmis.append(mis)$;
foreach $mis \in allmis$ **do**
    $dg \leftarrow Digraph(mis, Filter(mis, consist\_tr))$;
    $T.append(topological\_sort(dg))$;
$r \leftarrow learner_{oper}(dictOper, T)$;
return r

3.2 A Learning Example

The Language Size. The Language Size (LS for short) is introduced in [5], which measures for the simplest deterministic expression that "best" describes sample S. Formally, the Language Size of regular expression is calculated as follows: $\mathcal{LS}(E) = \sum_{i=0}^{n} |L(E)^{=i}|$ where $n = 2|\Sigma| + 1$ and $|\Sigma|$ is the size of

alphabet of expression E, and $|L(E)^{=i}|$ denote the number of words in language $L(E)$ of length i. Then the regular expression with the smaller value of $\mathcal{LS}(E)$ overgeneralizes S the less.

The Minimum Description Length measure. To evaluate the preciseness of different learning algorithms, we employ the data encoding cost proposed in [3], which reflects the degree of generalization of a regular expression for a given set of samples S. The data encoding cost compares the size of S with the size of the language defined by an inferred expression E, $datacost(E, S) = \sum_{i=0}^{n}(2 * log_2 i + log_2(\binom{|L^{=i}(E)|}{|S^{=i}|}))$, where $n = 2|\Sigma| + 1$ as before, and $L^{=i}(E)$ is the subset of words in $L(E)$ that have length i. Suppose E_1 and E_2 are two generalizations of S. If $datacost(E_1, S) < datacost(E_2, S)$, then we say E_1 describes S better which overgeneralizes S less.

Let $S = \{aabcde, acdcfe, dbbcfe, adbcef\}$, we get $\Sigma = \{a, b, c, d, e, f\}$, $dictOper = \{a : *; b : *; c : +; d : 1; e : 1; f :?\}$, $CS(S) = \{< c, d >, < d, c >, < b, d >, < d, b >, < f, e >, < e, f >\}$, $NCS(S) = \{< c, f >, < a, c >, < a, e >, < b, f >, < b, c >, < d, e >, < a, d >, < a, f >, < c, e >, < b, e >, < a, b >, < d, f >\}$. By recognizing alphabet symbols and $CS(S)$ we get the undirected graph G showed in Fig. 1. We get three maximal connected subgraphs of G using function $connected\_subgraph(G)$, namely, $G_1 = (V_1 = \{a\}, E_1 = \{\})$, $G_2 = (V_2 = \{e, f\}, E_2 = \{(e, f)\})$, $G_3 = (V_3 = \{b, c, d\}, E_3 = \{(b, d), (d, c)\})$, and they are shown in Fig. 2 with different colors. Then we use G2SubRE to transform G_1, G_2, G_3 to subexpressions r_1, r_2, r_3. Here we introduce the transformation process of G_3. For graph G_3, we obtain MIS set $allmis = \{\{b, c\}, \{d\}\}$. Next, compute the topological sort for each MIS. Filter({b,c},NCS(S))=$\{< b, c >\}$, so we add bc to T. For $\{d\}$ there is only one node add d to T, at last $T = \{bc, d\}$. Then using $learner_{oper}(dictOper, T)$, we concatenate all elements of T with & and add unary operate for each symbol, and get the subexpression $r_3 = b^*c^+\&d$. Simiarily $r_1 = a^*$, $r_2 = e\&f^?$. We construct the SOA $\mathcal{A}$ using algorithm *2T-INF* [17], the SOA $\mathcal{A}$ is shown in Fig. 3. Then we contract SOA with r_1, r_2, r_3 and get generalized SOA presented in Fig. 4. We calculated level number for every nodes of generalized SOA and get the final inferred $result = a^*(b^*c^+\&d)(e\&f^?)$.

For this sample S, Table. 1 shows the learning result of $learner^+_{DME}$ (for DME) [13], conMiner (for SIRE) [34] and InferISIRE. The learning result of $learner^+_{DME}$ allows all symbols to appear in any order, but in the sample S, from $NCS(S)$ we can get all ordered pairs of symbols, i.e., symbol c always occurs before symbol f. Besides, in S, for all strings that symbol a occurs, a always occurs in the first position. Result of InferISIRE is consistent with this constrains while neither $learner^+_{DME}$ nor conMiner conform to this. Among the three algorithms, the result of InferISIRE has the smallest Language Size and *datacost*, which shows preciseness of the regular expressions inferred by InferISIRE.

Table 1: The Learning Results of Different Algorithms

Learning Method	Learning Result	LS	datacost
$learner^{+}_{DME}$	$a^*\&b^*\&c^+\&d\&e\&f^?$	$1.71 * 10^8$	108.67
conMiner	$a^*de\&b^*c^+f^?$	$1.79 * 10^5$	92.45
InferISIRE	$a^*(b^*c^+\&d)(e\&f^?)$	$4.86 * 10^3$	85.66

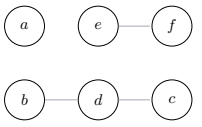


Fig. 1: Graph on Σ and $CS(S)$

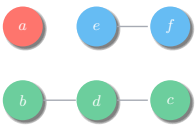


Fig. 2: Three maximal connected subgraphs

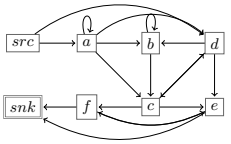


Fig. 3: SOA of sample

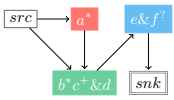


Fig. 4: Generalized SOA after contracting

4 Experiments

In this section, we first analyse the usage proportion of ISIREs and other subclasses based on large-scale real data. Then we compare our learning algorithm InferISIRE with other learning algorithms of different subclasses ($Learn^{+}_{DME}$ for DME [13], conMiner for SIRE [34])) and one XML tool (InstanceToSchema [14]). All experiments were conducted on a machine with Intel Core i5-5200U@2.20GHz, 4G memory. All codes are written in python 3.

4.1 Experiment of Practicality Analysis

We crawled 4,872 distinct Relax NG schema files from 254 websites, and 103 Github repositories utilizing Google Search Engine. We extracted 137,286 regular expressions from these files, and found that 38.45% expressions have the interleaving operator, which shows its widespread use in practice. Based on this 137,286 regular expressions, we investigated the usage proportion of ISIREs compared with other popular subclasses. The experimental result is shown in Fig.5. Note that SORE [7] and CHARE [16] are subclasses of standard regular expressions, and DME [9,13], SIRE [34] and ISIRE are subclasses supporting &. It is obvious from Fig. 5 that ISIRE has the highest proposition. The result shows ISIREs are more practical in real-world applications.

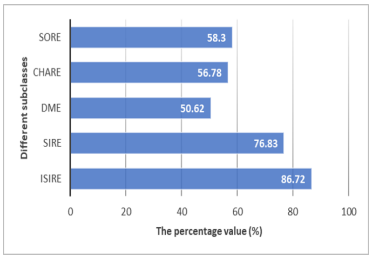


Fig. 5: Proportions of subclasses

4.2 Analysis on Learning Results of Different Algorithms

We downloaded three XML data corpora, including DBLP (computer science bibliography corpus), NASA (Datasets converted from legacy flat-file format into

XML), DocBook (a schema maintained by the DocBook Technical Committee of OASIS). Based on sets of strings extracted from these corpora, we compare the learning results of InferISIRE, the XML tool InstanceToSchema [14] (i2s for short), $learner^{+}_{DME}$ [13] and conMiner [34], which all support learning a regular expression with interleaving from XML data.

We first use the Language Size and *datacost* to measure the preciseness of expressions for a set of samples. However, during experiments on real data, we found that when the alphabet is large, the calculation of Language Size and *datacost* becomes extraordinarily time-consuming, since it needs a lot of tedious work for computing the number of subset words of $L(E)$. Given an expression $E = a^*b^?(c^*\&d^?)ef^+$ as an example, $n = 2 * 6 + 1 = 13$, actually it is tough to compute the number of words in $L(E)$ with length 13. Thus we introduce a simple definition called Combinatorial Cardinality to value the precision of expressions. Combinatorial Cardinality is limited to measure the precision of expressions with interleaving, satisfying that different expressions have the same symbols with the same unary operators $\{1, ?, *, +\}$. The smaller value of Combinatorial Cardinality, the more precise of an expression. It is defined as follows.

Definition 4. ***Combinatorial Cardinality*** *(CC) [26] Let Σ be the finite alphabet set. E_i is a regular expression with interleaving over Σ. $a, b \in \Sigma$ and $u, u', v, v' \in \Sigma^*$.*

- $CC(a) = CC(\varepsilon) = 1$, *where* $a \in \Sigma$;
- $CC(E^t) = CC(E)$, *where* $t \in rep$;
- $CC(E_1|E_2 \cdots |E_n) = CC(E_1) + CC(E_2) + \cdots + CC(E_n)$;
- $CC(E_1 \cdot E_2 \cdots E_n) = CC(E_1) \times CC(E_2) \times \cdots \times CC(E_n)$;
- $CC(u\&v) = CC(a \cdot (u'\&v)) + CC(b \cdot (u\&v'))$, *where* $u = au'$ *and* $v = bv'$;
- $CC((E_1|E_2|\cdots|E_m)\&(E'_1|E'_2|\cdots|E'_n)) = CC(\bigcup_{i=1j=1}^{i=mj=n} E_i\&E'_j)$.

Table 2: The Results on NASA

Method	LS	datacost	CC
i2s	$8.66 * 10^8$	$1.95 * 10^3$	$3.63 * 10^6$
$learner^{+}_{DME}$	$8.66 * 10^8$	$1.95 * 10^3$	$3.63 * 10^6$
conMiner	$1.42 * 10^8$	$1.67 * 10^3$	10
InferISIRE	$2.60 * 10^5$	$1.00 * 10^3$	2

Table 3: The Results on www(DBLP)

Method	LS	datacost	CC
i2s	$1.53 * 10^{18}$	$1.34 * 10^4$	$3.63 * 10^6$
$learner^{+}_{DME}$	$1.43 * 10^{15}$	$1.11 * 10^4$	2160
conMiner	$5.71 * 10^{12}$	$9.50 * 10^3$	360
InferISIRE	$2.35 * 10^{10}$	$6.47 * 10^3$	168

Table 4: The Results on phdthesis(DBLP)

Method	LS	datacost	CC
i2s	$5.89 * 10^{25}$	$5.63 * 10^3$	$8.72 * 10^{10}$
$learner^{+}_{DME}$	$7.04 * 10^{24}$	$5.27 * 10^3$	$3.27 * 10^7$
conMiner	$1.24 * 10^{18}$	$4.28 * 10^3$	$1.51 * 10^7$
InferISIRE	$4.07 * 10^{14}$	$4.04 * 10^3$	$8.32 * 10^5$

Table 5: The Results on DocBook

Method	LS	datacost	CC
i2s	$7.50 * 10^{12}$	$3.75 * 10^6$	$4.03 * 10^4$
$learner^{+}_{DME}$	$3.57 * 10^{11}$	$3.19 * 10^6$	$1.01 * 10^4$
conMiner	$8.43 * 10^7$	$1.50 * 10^6$	56
InferISIRE	$2.69 * 10^6$	$7.81 * 10^5$	6

Table 6: Results of Inference Using Different Methods on XML datasets

Dataset Name Sample Size	**Result of InstanceToSchema Result of learner$^{+}_{DME}$ Result of conMiner Result of InferISIRE**
NASA 2435	$a_1\&a_2^+\&a_3\&a_4^+\&a_5\&a_6^*\&a_7^*\&a_8\&a_9\&a_{10}^*$
	$a_1\&a_2^+\&a_3\&a_4^+\&a_5\&a_6^*\&a_7^*\&a_8\&a_9\&a_{10}^*$
	$a_1a_2^+a_3a_4^+a_5a_6^*a_7^*a_8a_9\&a_{10}^*$
	$a_1a_2^+a_3a_4^+a_5a_6^*(a_{10}^*\&a_7^*)a_8a_9$
table(DocBook) 1719728	$a_1\&a_2\&a_3^?\&a_4^*\&a_5^*\&a_7^*\&a_6^*\&a_8^?$
	$(a_6^*\|a_7^*)\&a_5^*\&a_4^*\&a_1\&a_2\&a_3^?\&a_8^?$
	$a_3^?\&a_4^*\&a_1a_2a_5^*a_7^*a_6^*a_8^?$
	$a_1a_2(a_3^?\&a_4^*\&a_5^*)a_7^*a_6^*a_8^?$
www(DBLP) 2000226	$a_7^?\&a_{10}^*\&a_1^*\&a_{17}^?\&a_2^*\&a_5^*\&a_{14}^*\&a_3^?\&a_6^?\&a_{15}^*$
	$(a_{10}^*\|a_{17}^?\|a_6^?)\&(a_5^*\|a_3^?\|a_7^?)\&(a_2^*\|a_1^*)\&a_{14}^*\&a_{15}^*$
	$a_6^?a_1^*a_{14}^*a_5^*a_3^?a_7^?a_{17}^?\&a_{15}^*\&a_2^*a_{10}^*$
	$a_1^*a_6^?(a_{14}^*a_5^*\&a_{15}^*\&a_2^*a_{10}^*a_3^?a_7^?a_{17}^?)$
phdthesis(DBLP) 63420	$a_7^*\&a_{10}^*\&a_{17}^+\&a_2^+\&a_{19}^*\&a_{14}\&a_{15}^?\&a_{16}^?\&a_{11}^?\&a_{12}^*\&a_9^?\&a_{21}^*\&a_{20}^?\&a_{13}^?$
	$(a_{11}^?\|a_9^?\|a_{19}^*)\&(a_{10}^*\|a_{20}^?\|a_{15}^?)\&a_7^*\&a_{16}^?\&a_{12}^*\&a_{13}^?\&a_2^+\&a_{14}\&a_{21}^*\&a_{17}^+$
	$a_{13}^?a_{20}^?\&a_{12}^*a_7^*\&a_{17}^+a_{16}^?\&a_2^+a_{14}a_9^?a_{21}^*a_{15}^?\&a_{11}^?a_{19}^*a_{10}^*$
	$a_2^+a_{14}(a_{19}^*\&a_{13}^?a_{20}^?\&a_{17}^+a_{16}^?a_{11}^?\&a_{21}^*a_{10}^*\&a_{12}^*a_9^?a_{15}^?a_7^*)$

The inferred regular expressions on these three XML corpora are shown in Table. 6. For every XML data set, the inferred regular expressions of InstanceToSchema [14], conMiner [34], $learner^{+}_{DME}$ [13] and InferISIRE are shown from top to bottom. To save space, we use the short names of words and the list of abbreviations is shown in `https://github.com/yetingli/sofsem2019`. We calculated the Language Size, *datacost* and *CC* for the inferred results of different methods on each data set, and they are presented in Table. 2-5 respectively. We find that all regular expressions inferred by InstanceToSchema are only symbols combined with the interleaving, not surprisingly, its result has a maximum Language Size, *datacost* and *CC* on each data set, which denotes overgeneralization of InstanceToSchema. Meanwhile, we find that the Language Size, *datacost* and *CC* values of regular expressions inferred by InferISIRE are the smallest on all the data sets, which indicates its better preciseness and the least generalization among these methods.

Taking the results on NASA as an example. Both InstanceToSchema and $learner^{+}_{DME}$ only use interleaving to connect all symbols, thus some sequential restrictions have been lost and learning results are overgeneralization. Furthermore, conMiner can not learn some consistent partial orders due to the limits in SIREs, that is the & operator can only appears at the outermost layer of an expression in SIRE. In the strings set extracted from nasa.xml, a_1 always occurs at the first position of strings and a_9 always occurs at the last position

of strings, the result of InferISIRE conforms to this constrain. But the result of conMiner allows a_{10} to appear before a_1 or after a_9, which makes the result overgeneralization for samples. To conclude, the regular expressions inferred by InferISIRE are more precise compared with other tools and methods.

5 Conclusion

In this paper, focusing on the inference of Relax NGs, we proposed a new subclass ISIREs of regular expressions with interleaving. Then we designed a polynomial inference algorithm InferISIRE to learn ISIREs. Based on large-scale real data, we calculated the usage proportion of ISIREs and other popular subclasses, and found that the proportion of ISIREs is the highest, which shows ISIREs have a better practicality. At last we compare the inferred results of InferISIRE with other inference algorithms on three XML data corpora, experimental results showed that the regular expressions inferred by InferISIRE are more precise.

References

1. Serge Abiteboul, Pierre Bourhis, and Victor Vianu. Highly Expressive Query Languages for Unordered Data Trees. *J. Theory Comput. Syst*, 57(4):927–966, 2015.
2. Serge Abiteboul, Peter Buneman, and Dan Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 1999.
3. Pieter W. Adriaans and Paul M. B. Vitányi. The power and perils of MDL. In *Proc. 12. ISIT*, pages 2216–2220, 2007.
4. Michael Benedikt, Wenfei Fan, and Floris Geerts. XPath Satisfiability in the Presence of DTDs. *J. ACM*, 55(2):1–79, 2008.
5. Geert Jan Bex, Wouter Gelade, Frank Neven, and Stijn Vansummeren. Learning Deterministic Regular Expressions for the Inference of Schemas from XML Data. *J. TWEB*, 4(4):14:1–14:32, 2010.
6. Geert Jan Bex, Frank Neven, Thomas Schwentick, and Karl Tuyls. Inference of Concise DTDs from XML Data. In *Proc. 32. VLDB*, pages 115–126, 2006.
7. Geert Jan Bex, Frank Neven, Thomas Schwentick, and Stijn Vansummeren. Inference of Concise Regular Expressions and DTDs. *J. ACM Trans. Database Syst*, 35(2):1–47, 2010.
8. Geert Jan Bex, Frank Neven, and Stijn Vansummeren. Inferring XML Schema Definitions from XML Data. In *Proc. 33. VLDB*, pages 998–1009, 2007.
9. Iovka Boneva, Radu Ciucanu, and Slawek Staworko. Simple Schemas for Unordered XML. In *Proc. 13. WebDB*, pages 13–18, 2013.
10. Iovka Boneva, José Emilio Labra Gayo, Eric G. Prud'hommeaux, and Slawek Staworko. Shape Expressions Schemas. *CoRR*, abs/1510.05555, 2015.
11. Ravi B. Boppana and Magnús M. Halldórsson. Approximating Maximum Independent Sets by Excluding Subgraphs. *J. BIT*, 32(2):180–196, 1992.
12. Dunren Che, Karl Aberer, and M. Tamer Özsu. Query Optimization in XML Structured-Document Databases. *J. VLDB*, 15(3):263–289, 2006.
13. Radu Ciucanu and Slawek Staworko. Learning Schemas for Unordered XML. In *Proc. 14. DBPL*, 2013.

14. Didier Demany. InstanceToSchema: a Relax NG Schema Generator from XML Instances. *http://www.xmloperator.net/i2s/*, 2005.
15. Jaroslav M. Fowkes and Charles A. Sutton. A Subsequence Interleaving Model for Sequential Pattern Mining. In *Proc. 22. KDD*, pages 835–844, 2016.
16. Dominik D Freydenberger and Timo Kötzing. Fast Learning of Restricted Regular Expressions and DTDs. *J. Theory Comput. Syst*, 57(4):1114–1158, 2015.
17. P. Garcia and E. Vidal. Inference of k-Testable Languages in the Strict Sense and Application to Syntactic Pattern Recognition. *J. IEEE Trans. Pattern Anal. Mach. Intell*, 12(9):920–925, 2002.
18. Minos N. Garofalakis, Aristides Gionis, Rajeev Rastogi, S. Seshadri, and Kyuseok Shim. XTRACT: Learning Document Type Descriptors from XML Document Collections. *J. Data Min. Knowl. Discov*, 7(1):23–56, 2003.
19. Wouter Gelade. Succinctness of Regular Expressions with Interleaving, Intersection and Counting. In *Proc. 33. MFCS*, pages 363–374, 2008.
20. Giorgio Ghelli, Dario Colazzo, and Carlo Sartiani. Efficient Inclusion for a Class of XML Types with Interleaving and Counting. In *Proc. 11. DBPL*, pages 231–245, 2007.
21. E Mark Gold. Language Identification in the Limit. *J. Information and Control*, 10(5):447–474, 1967.
22. Steven Grijzenhout and Maarten Marx. The Quality of the XML Web. *J. Web Sem*, 19:59–68, 2013.
23. Guen-Hae Kim, Sang-Ki Ko, and Yo-Sub Han. Inferring a Relax NG Schema from XML Documents. In *Proc. 10. LATA*, pages 400–411, 2016.
24. Kenrick Kin, Björn Hartmann, Tony DeRose, and Maneesh Agrawala. Proton: Multitouch Gestures as Regular Expressions. In *CHI Conference on Human Factors in Computing Systems*, pages 2885–2894, 2012.
25. Christoph Koch, Stefanie Scherzinger, Nicole Schweikardt, and Bernhard Stegmaier. Schema-Based Scheduling of Event Processors and Buffer Minimization for Queries on Structured Data Streams. In *Proc. 30. VLDB*, pages 228–239, 2004.
26. Yeting Li, Xiaolan Zhang, Han Xu, Xiaoying Mou, and Haiming Chen. Learning restricted regular expressions with interleaving from XML data. In *Proc. 37. ER*, pages 586–593, 2018.
27. Zheng Li and Tingjian Ge. PIE: Approximate Interleaving Event Matching over Sequences. In *Proc. 31. ICDE*, pages 747–758, 2015.
28. Ioana Manolescu, Daniela Florescu, and Donald Kossmann. Answering XML Queries on Heterogeneous Data Sources. In *Proc. 27. VLDB*, pages 241–250, 2001.
29. Wim Martens and Frank Neven. Typechecking Top-Down Uniform Unranked Tree Transducers. In *Proc. 9. ICDT*, pages 64–78, 2003.
30. Wim Martens and Frank Neven. Frontiers of Tractability for Typechecking Simple XML Transformations. In *Proc. 23. PODS*, pages 23–34, 2004.
31. Jun Ki Min, Jae Yong Ahn, and Chin Wan Chung. Efficient Extraction of Schemas for XML Documents. *J. Inf. Process. Lett*, 85(1):7–12, 2003.
32. Robert Paige and Robert Endre Tarjan. Three Partition Refinement Algorithms. *SIAM J. Comput.*, 16(6):973–989, 1987.
33. Yannis Papakonstantinou and Victor Vianu. DTD Inference for Views of XML Data. In *Proc. 19. PODS*, pages 35–46, 2000.
34. Feifei Peng and Haiming Chen. Discovering Restricted Regular Expressions with Interleaving. In *Proc. 17. APWeb*, pages 104–115. Springer, 2015.
35. Slawek Staworko, Iovka Boneva, José Emilio Labra Gayo, Samuel Hym, Eric G. Prud'hommeaux, and Harold R. Solbrig. Complexity and Expressiveness of ShEx for RDF. In *Proc. 18. ICDT*, pages 195–211, 2015.

Vehicle Data-Driven Air Quality Prediction By Using Kernel Density Estimation

Enes Esatbeyoglu [1], Oliver Cassebaum [1], Sandro Schulze [2] and Andreas Sass [1]

[1] Volkswagen AG, Wolfsburg, Germany
[2] Otto-von-Guericke University of Magdeburg, Germany

Abstract. Fixed roadside air monitoring stations measure air quality at a few locations in cities and generate hourly averages. Therefore, only a restricted statement about the spatial distribution of the pollutants is possible. A temporal and spatial prediction of air quality based on stationary measurements can also be extended and has already been sufficiently investigated. In this work we equipped a vehicle with additional sensors and measured the air quality, especially the NO_2 concentration, in real-time traffic. This allows us to collect more precise data, which also include spatial considerations due to our mobile measurement in real-time traffic. As we drove a selected route several times in succession, we were also able to investigate the temporal change of the measurement results on this route. This is necessary in order to enable a higher accuracy in the prediction phase. To this end, we first cleaned up the measured NO_2 concentration using specific vehicle parameters. Afterwards, we analyzed it using Kernel Density Estimation (KDE) in order to calculate a temporal and route-based forecasting based on probabilities for the measured NO_2 concentration. Results have shown that we can derive additional information (e.g. traffic light circuit) from specific vehicle parameters, which improve the probability temporal and route-based forecasting. Furthermore, by cross-validation we show that KDE is more suitable for temporal and route-based forecasting than the use of mean or median values. Therefore, in this work we enable an extended approach to precisely forecasting NO_2 concentration taking multiple parameters into account.

Keywords: Vehicle Data-Driven Approach, Kernel Density Estimation, Probabilistic Data, NO_2 Mobile Vehicle Measurement, Real-Time Traffic, Cross-Validation.

1 Introduction

It is known that official roadside monitoring stations measure air quality and generate hourly averages. The hourly limit value in European cities for NO_2 concentration, which may be exceeded 18 times a year, is 200 μgm^{-3} and the average annual limit value is 40 μgm^{-3} [1]. Studies exist in which the pollutants are statistically analyzed based on fixed locations or mobile laboratories [2, 3]. Mobile measurements with a

bicycle or vehicle are also evaluated to measure fine-grained air quality and other pollutants in real time [4, 5]. The statistical survey of the actual state and, in addition, a time and route-based forecast of air quality help to provide early warnings to the population and take actions before tolerance limits are exceeded.

In this respect, scientific approaches exist to make both, temporal as well as spatial forecasts. For instance, attempts on the basis of stationary measurement data were made to predict the NO_2 concentration using various methods in the field of machine learning [6, 7, 8 and 9]. Alternatively, different dispersion models based on mathematical or statistical approaches have been applied in order to investigate the spatial forecasting of pollutants [10, 11].

In order to extend these prediction models, we equipped our vehicle with NO_2 measuring technology. Therefore, our measurements are not limited locally, as we measure in the area. This allows us to collect more precise data, which also include spatial considerations due to our mobile measurement in real-time traffic. As we drive a selected route several times in succession and measure the NO_2 concentration, we can also investigate the temporal change of the measurement results on this route. This is necessary in order to enable a higher accuracy in the prediction phase.

To this end, we have connected the measuring technology to the measurement system of our vehicle, which also enabled us to log specific vehicle parameters from internal messages. Afterwards, we selected a route containing one traffic light and drove 17 times in a row. Thereby, we measured the NO_2 concentration in real-time traffic.

For this purpose, we first cleaned the measured data from disturbances such as measured exhaust plumes by using specific vehicle parameters. Subsequently, we used Kernel Density Estimation (KDE) to generate a temporal and route-based probabilistic forecasting of NO_2 concentration. Because our approach is vehicle data driven, we have derived the traffic light circuit from our velocity at the traffic light. This additional information helps us to improve the prediction especially in the area after the traffic light. Finally, we use methods of cross-validation to investigate the forecasting quality and compare KDE with statistical methods of averaging and median value formation in terms of their suitability.

The contributions of our work are:

1. We propose a lightweight, mobile measuring system that includes both, environmental as well as specific vehicle parameters, and thus, allows for more precise data collection and a higher accuracy in the prediction phase.
2. We propose a method of data cleaning for filtering exhaust plumes when measuring air quality with a vehicle.
3. We propose a methodology to generate a temporal and route-based probabilistic forecasting.
4. We show how traffic light information, which is indirectly determined via the velocity, can improve the prediction.
5. A comparison of the forecasting techniques used with statistical methods.

2 Background

In this Section, we describe in more detail the selected measurement technology and the specific vehicle parameters used in this work. Subsequently, we explain the driving route profile. Afterwards, we introduce the method of KDE as a probabilistic forecasting approach and the method of cross-validation to investigate the forecasting quality.

2.1 Measurement Technology used and specific Vehicle Parameters

We installed the measuring device "NO_2 / NO / NO_x Monitor Model 405 nm"[1] on the vehicle. The principle of measurement is a direct measurement of NO_2 in the concentration range 0-10,000 ppb with an accuracy of 2 ppb. This device is designated for NO_2 compliance monitoring in the United States but not in Europe [12]. Therefore, the measuring method does not comply with the official European directives and thus is not an officially licensed measuring technique. In comparison, the official measurement method for NO_2 concentration is based on the chemiluminescence method [1]. However, both methods mentioned have already been compared with each other, concluding that the data quality of our measuring device used is given [13].

We connected the measurement technology with the vehicle measurement system, so that specific vehicle parameters such as GPS position, velocity and adaptive cruise control (ACC) were also logged via a vehicle bus system, e.g. controller area network (CAN). With ACC, the distance to the front vehicle is measured uniformly by using a sensor. The smaller the value, the smaller the distance to the front vehicle.

2.2 Route Profile

With the described measuring technique we have driven a route 17 times in a row in order to make a statement about how the NO_2 concentration changes on the route for following vehicles. The investigated route has a length of 500 m, with a traffic light in the middle. First we analyzed the total route and then the section 150 m after the traffic light, because there are different levels of emissions from vehicles depending on the traffic light circuit (e.g. acceleration from standstill after red phases). It should also be mentioned that weather parameters (e.g. wind) that could have an influence on the experiment were considered constant.

2.3 Kernel Density Estimation

In order to analyze the measured data we have chosen Kernel Density Estimation's approach. This is a method to estimate the unknown probability density function of a random variable (in this case NO_2 concentration). The KDE is calculated by weighting the distances of all the data points. If there are more data nearby, the esti-

[1] https://twobtech.com/model-405-nm-nox-monitor.html

mate or the probability of existing data at that location is higher [14]. The kernel density estimator f is defined by [14]:

$$f(x) = \frac{1}{n*b}\sum_{j=1}^{n} K\left(\frac{x-x_j}{b}\right) \quad (1)$$

K: Kernel
n: Sample size
b: Bandwidth
x_j: Sample

Possible kernels that can be used to calculate the kernel density include the Gaussian Kernel, Cauchy Kernel and Epanechnikov Kernel. According to [15], the selection of the kernel function (compared to the appropriate selection of bandwidth b) is of secondary importance. This has a minor role on the final quality of the estimate. Therefore, the Gaussian Kernel is used in this work [14]:

$$K(x) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}x^2\right) \quad (2)$$

Various methods are proposed in the literature for estimating the optimum bandwidth (mathematically), such as distorted and undistorted cross-validation and plug-in estimators [16, 17]. Because the determination of the optimum bandwidth is not discussed in more detail in this work, we have simply assumed Silverman's rule of thumb (equation (3)) for the Gaussian Kernel in order to reduce the computational overhead [18]. Exemplary bandwidths are shown in Section 3.3.

2.4 Cross-Validation

Cross-validation methods are test methods of data analysis that are used, among other aspects, in data mining where the goal is prediction. There are several types of this method, such as simple, stratified or leave-one-out cross-validation [19, 20]. In this work, we use the process of simple cross-validation (Figure (1)).

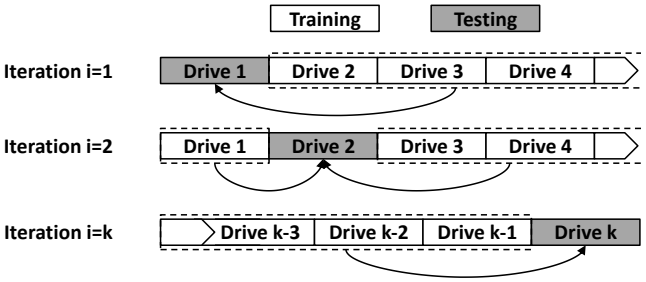


Fig. 1. Cross-Validation method used in this work

In k-fold cross-validation, the forecast results are evaluated by partitioning the original data set into k equally sized subsets $D_1,\ldots, D_k$ consisting of N elements. Fur-

thermore, a distinction is made between a training set and test set. k passes are started in which the i-th subset of D_i is used as the test set and the remaining k-1 subset of D_i is used as the training set. The total error rate is calculated as an average from the individual error rates of the k individual runs [19, 20, 21].

3 Methodology

In this Section, we first describe the data preparation and in particular a vehicle data-driven methodology which we use to clean data set from exhaust plumes of the front vehicle. Afterwards, we describe the bandwidth selection and its effect on distribution in more detail. Furthermore, we show how the velocity can be used to increase the accuracy of the calculated probability for a given section. Finally, we show an approach to assess the forecasting quality.

3.1 Data Preparation

First, we connected the measurement device to the vehicle's measuring system. All CAN bus channels stored on the measuring system have different transmission frequencies (asynchronous) because of different priorities on the CAN. This means, for example, that some channels are transmitted more frequently in a time unit than others. Consequently, all recorded signals must be normalized to the same equidistant time vector in order to be able to deduce the distance covered from the velocity.

In this work, we chose a linear interpolation based on the time vector with an increment of 0.2 s. In order to infer distant-equivalent signals from time-equivalent ones, asynchronous distance-based units were initially extracted from the velocity signal. Subsequently, we used the linear interpolation (based on the distance) with an increment of 1 m in order to calculate signals with an equivalent distance. Afterwards, we created a 17 x 500 matrix. The row size is the number of drives and the column size stands for the parameters recorded or interpolated per 1 m, which we consider in this work. These include specific vehicle parameters such as GPS position, ACC signal and velocity as well as the measured NO_2 concentration.

In addition, we have taken into account the delay time and the reaction time of our measurement device by shifting the measurement data of the NO_2 concentration by a certain time interval. This is particularly important to determine the location of the measured NO_2 concentration.

3.2 Data Cleansing

After we converted the data to a route-based equidistant form, we used the vehicle's ACC signal to clean up measured concentrations over the 500 m distance driven. In particular, we set values in the NO_2-matrix as *not a number* (NaN), where we drove too close to the front vehicle. We have not used a different type of interpolation for this area because this procedure can distort the results. With this approach, measured exhaust plumes of the front vehicle can be filtered because we want. This is necessary

to avoid distortion of the measured NO_2 concentration because we want to measure the air quality and not the exhaust plumes of front vehicles. The Figure (2) shows the mean measured NO_2 concentration for individual drives with and without ACC filtering.

It is shown that with this approach relatively high measured concentrations can be cleaned up from the data set. A disadvantage of this approach is that we assume that a front vehicle emits pollutants. However, it is also possible that the front vehicle drives emission-free (e.g. electric vehicles, coasting, etc.). As a result, for individual drives (2, 13 and 17) the approach causes an increase in the mean concentration.

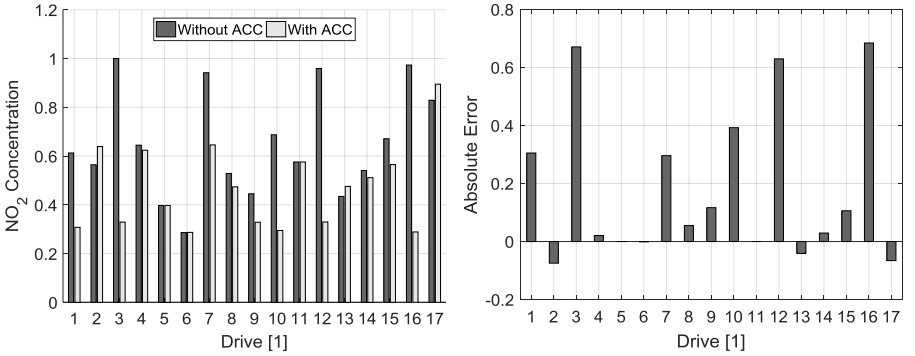


Fig. 2 Mean NO_2 concentration (standardized) over 500 m of individual drives with and without ACC filtering (left), difference between with and without ACC filtering (right)

3.3 Bandwidth Selection for the Gaussian Kernel

Figure (2) shows that the mean NO_2 concentration of successive drives deviate from one another without being able to detect patterns. For this reason, we first investigate the distribution of the measured mean NO_2 concentration. As already described in Section 2.3, the selection of the bandwidth b of the kernel density function has a higher priority. The distribution of the sample (here the mean NO_2 concentrations) by using the Gaussian Kernel exemplary for $b = 1$ is shown in Figure (3).

Alternative bandwidths ($b = 2$ and $b = 12$) can be seen in Figure (4). If the values of b are too small, random fluctuations are overrated and lead to noise in the graph ("under-smoothing"). In contrast, too high values of b lead to important details being ignored ("over-smoothing"). That is why we have taken Silverman's bandwidth to be optimal. We use this to investigate the distribution of NO_2 concentration per meter. It is calculated as follows:

$$b = \sqrt[5]{\left(\frac{4\sigma^5}{3n}\right)} \tag{3}$$

b: Bandwidth
σ: Standard deviation of the samples
n: Sample size

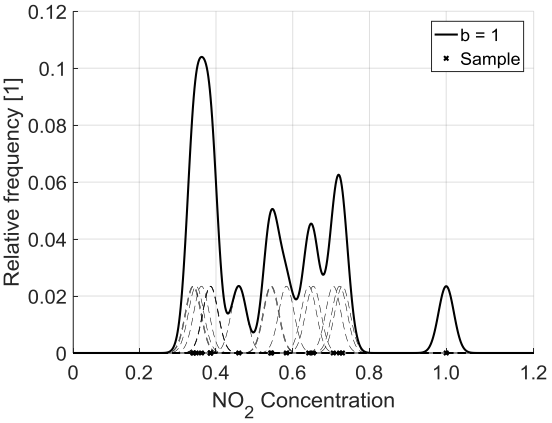


Fig. 3. KDE with Gaussian Kernel and $b = 1$.

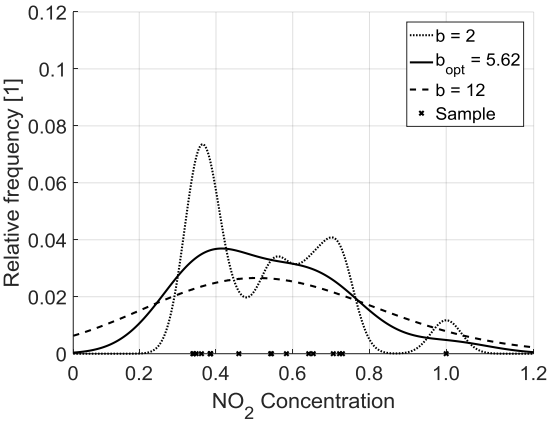


Fig. 4. KDE with different bandwidths. b_{opt} is calculated with equation (3)

3.4 Use of the Velocity as Additional Information

Because we have connected the measuring sensors with the measuring system of our vehicle, we have logged further specific vehicle parameters. Among other, the velocity v of our vehicle can vary over the distance depending on the traffic lights and the current traffic. We assume that we can deduce the switching of the traffic light from this information.

First, as long as the velocity (allowed 50 km/h) does not fall below 40 km/h on the complete route, we assume a green traffic light. If it was less than 7 km/h, we assume a red traffic light. Values that lay between the two limits have not been classified as they can be caused by different driving situations (e.g., heavy traffic, a car that turns left/right).

Table 1. Classification of velocities

	Number of drives	Class
v >= 40 km/h	4	1
v < 7 km/h	10	2
7 <= v < 40 km/h	3	3

In Table (1), we show an overview, summarizing the classification of velocities. By this classification of the drives depending on the velocity, we then reason about possible switching of the traffic light. The rationale of this approach is to have predictive information about the switching process of the traffic light through communication of traffic control systems with the vehicle. Subsequently, a precise probability of the distribution of the NO_2 concentration can be predicted in the area after the traffic light.

3.5 Forecasting Quality

We use the Root Mean Square Error (RMSE) to assess the forecasting quality after cross-validation. The RMSE indicates how well a forecast deviates on average from the real data. The larger the RMSE, the greater the deviation from the model. In the literature, the RMSE is used in many prognosis error evaluations, for example in regression-based and statistical methods [22, 23].
The RMSE is calculated as follows [23]:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(R_i - P_i)^2}{N}} \tag{4}$$

R: Real data (trainings data)
P: Predicted data (testing data)
N: Sample size

After dividing the data set into training and test data (method of cross-validation) we compare three approaches as training data. For KDE we first assess the NO_2 concentration (per meter), for which the probability (that this concentration occurs) is maximum. As a second and third approach we assess mean and median values (also per meter). Then we calculate the RMSE between the training data of the respective approach and the test data.

4 Results and Discussion

After we have explained the methodology of our approach in Section 3, we present and discuss the results in this Section. We no longer limit ourselves to the mean NO_2 concentrations shown in Figure (4). Rather, we look at the complete route and analyze the measured NO_2 concentrations per meter using the methodology of Kernel Density Estimation. Figure (5) shows the relative frequency distribution of the NO_2 concentra-

tion per meter as a display image with scaled colors. The darker the area, the higher the probability.

Our data reveal that up to the traffic light (at 250 m) the NO_2 concentration can be predicted with a relatively high probability. In particular, a range between the NO_2 concentration 0.1 and 0.3 can be assumed by constant traffic and meteorological conditions for this part of the course. Likewise, approx. 150 m after the traffic light (total from 400 m) an even range of the NO_2 concentration can be predicted with a high probability.

If the range between 250 m and 400 m is considered (the area between the two dashed lines in Figure (5), (6) and (7)), it is noticeable that the relative frequency distribution of the NO_2 concentration is over-smoothed. This shows that any NO_2 concentration can occur with nearly the same probability. As already described in Section 3.4, vehicles have different velocities at the traffic lights depending on the state (i.e., red or green). It is known that vehicles that do not accelerate, for example, generally emit less pollutant than vehicles that accelerate particularly from standstill. Due to different speeds and emissions (e.g. depending on acceleration) for 17 drives we measured different NO_2 concentrations in the range 0 to 0.8. Therefore, the distribution is stretched at or immediately after the traffic light. Therefore, in this area we can only make limited statements about the relative frequency distribution of the NO_2 concentration.

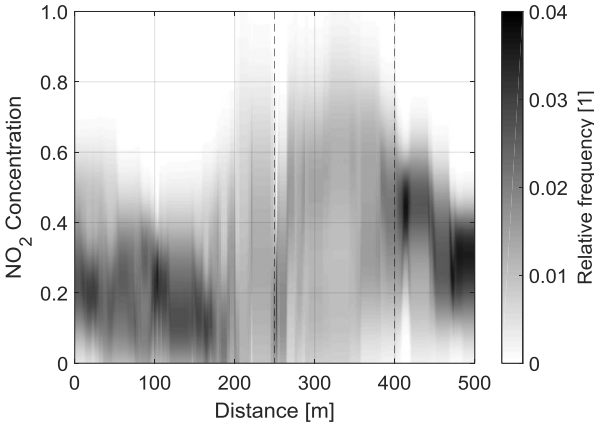


Fig. 5. Relative frequency distribution of all drives. The area between the two dashed lines shows the section immediately after the traffic lights.

In order to make a more precise statement about the distribution in the range between 250 m and 400 m, we compute this part of the course in dependence of the first two classes from Table (1). The first approach is that the velocity at each location of the road is according to Class 1 at least 40 km/h. To this end, we have calculated the difference between the relative frequency of all drives and only Class 1 in order to show the added value of this approach. Figure (6) shows that in particular in areas at or immediately after the traffic light an added value for the relative frequency distribu-

tion is observed. It is also shown that a lower NO_2 concentration in this section is to be expected for Class 1.

By contrast, the Figure (7) shows the added value of the relative frequency distribution of NO_2 concentration for drives of Class 2. It is shown that a higher probability for a higher range of NO_2 concentration can be observed in this case.

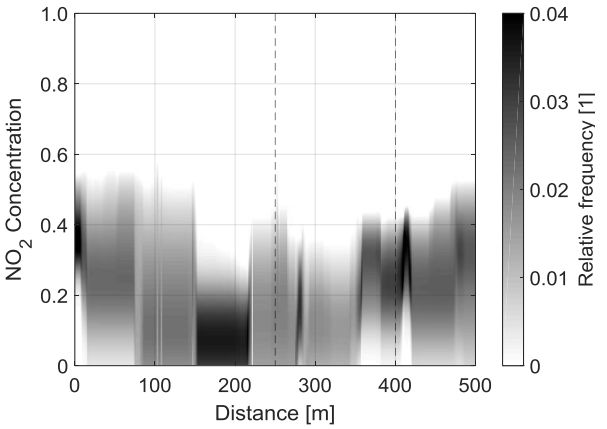


Fig. 6. Relative frequency distribution exclusively with drives with green traffic lights. The area between the two dashed lines shows the section immediately after the traffic lights.

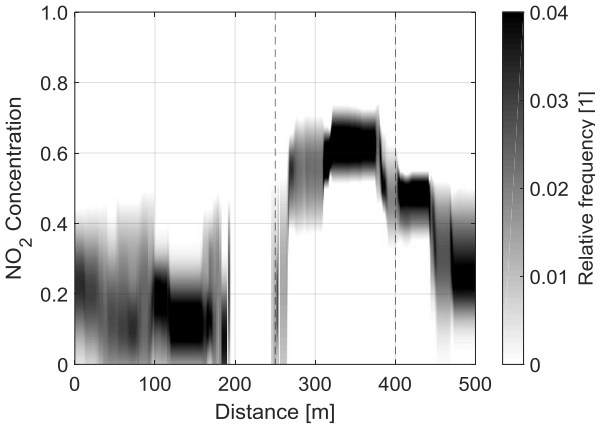


Fig. 7. Relative frequency distribution exclusively with drives with red traffic lights. The area between the two dashed lines shows the section immediately after the traffic lights.

Figures (6) and (7) have shown that additional information can increase the probability of the expected NO_2 distribution immediately after the traffic lights. Depending on the different traffic lights, there will be a different concentration range with a high probability.

In order to assess the forecasting quality in the area of 250 m to 400 m in particular by deriving traffic light information, we use cross-validation (Section 2.4) for three

approaches KDE, mean and median. Afterwards, we calculate the respective RMSE (Section 3.5). To this end, we first compare all drives of Class 1 and then of Class 2. Figure (8) shows for three approaches the median of the RMSE resulting from the cross-validation. It is shown that the KDE method is better suited for forecasting than statistical methods of averaging and median value formation, because the RMSE is smaller for both Class 1 and 2.

Furthermore, it is shown that for all approaches the RMSE is higher for Class 1. This can be explained by the comparatively smaller data size (number of drives in Class 1: 4 and Class 2: 10). Therefore, it can be assumed that a greater amount of training data would reduce the forecasting error.

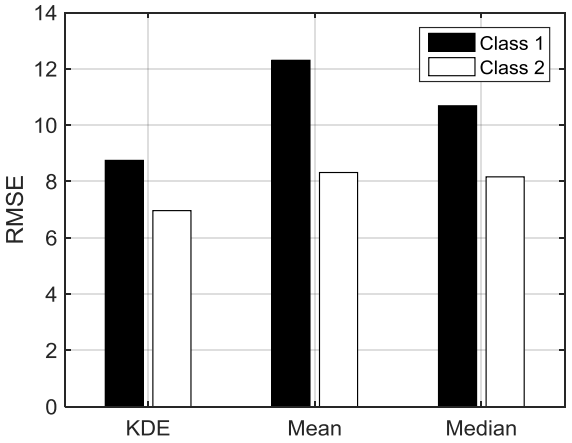


Fig. 8. Comparison of the forecasting quality of different approaches by calculating the RMSE

It should also be discussed to what extent the above experiment can be applied to more complex traffic situations. In this respect, for example, traffic jams can occur on any road segment depending on the traffic. If there is no traffic jam on a road segment, the traffic will flow better so that the measuring vehicles will have a higher velocity. However, if there is a traffic jam, this will result in a lower velocity for the measuring vehicles. Therefore, a classification similar to Table (1) can be used for the same road segment, considering the velocity of the measuring vehicle. Not only the traffic flow but also the number of vehicles per road segment would have to be taken into consideration. By using this information, the prediction quality could be improved in a similar way as in this work.

5 Conclusion and Future Work

In this work we enable an extended approach to precisely temporal and route-based forecasting of NO_2 concentration taking multiple parameters into account. To this end, we first showed which measurement technique was used to measure the NO_2 concentration on the vehicle. The mobile measurement allowed us to use both the

measurement data and the specific vehicle data to make a more accurate and precise statement about the NO_2 distribution. In particular, we used specific vehicle parameters (such as the distance sensor ACC) to remove the front vehicle's exhaust plumes from our database. Thus, we succeeded in presenting a method which we use to clean data set from exhaust plumes of the front vehicle.

To this end, we used the cleansed data to generate a temporal route-based probabilistic forecasting. For this we used kernel density functions, whereby the Gaussian approach formed the kernel and the rule of thumb according to Silverman the bandwidth selection. Because we measured different NO_2 concentration for each drive, this approach allowed us to identify which concentration would most likely result at which sections of the route.

In particular, we have found that the distribution of air quality after traffic lights is over-smoothed (taking into account all drives carried out). This has shown that any NO_2 concentration can occur with nearly the same probability. By taking other specific vehicle parameters (such as velocity) into account, we were able to derive traffic light information. This additional information enabled us to improve the accuracy of the calculated probability, especially in the area after traffic lights.

Afterwards, we assessed the forecasting quality using cross-validation methods and calculated the respective root mean square error. We have shown that the use of KDE results in lower error than the use of classical methods of averaging and median value formation.

The approach of the kernel density estimation can be extended by considering the time dependence of the route-based distribution. In this context, the question of how current or previous values should be weighted should be examined in particular. The current work can be expanded by equipping several vehicles with suitable NO_2 measuring technology in order to make a comprehensive statement about the quantity and quality of the measurements. Furthermore, the NO_2 concentration should also be measured at different weather conditions and times (especially rush hour) on a selected route in order to take into account the influences of these parameters.

References

1. Act on the Prevention of Harmful Effects on the Environment Caused by Air Pollution, Noise, Vibration and Similar Phenomena.
2. M. M. Rahman, B. Yeganeh, S. Clifford, L. D. Knibbs, L. Morawska, Development of a land use regression model for daily NO 2 and NO x concentrations in the Brisbane metropolitan area, Australia, Environmental Modelling & Software 95 (2017) 168-179. doi:10.1016/j.envsoft.2017.200 06.029.
3. K. Bashir Shaban, A. Kadri, E. Rezk, Urban Air Pollution Monitoring System With Forecasting Models, IEEE Sensors Journal 16 (8) (2016) 2598-2606. doi:10.1109/JSEN.2016.2514378.
4. B. Elen, J. Peters, M. Poppel, N. Bleux, J. Theunis, M. Reggente, A. Standaert, The Aeroflex: A Bicycle for Mobile Air Quality Measurements, Sensors 13 (12) (2012) 221-240. doi:10.3390/s130100221.
5. W. Liu, X. Li, Z. Chen, G. Zeng, T. Leon, J. Liang, G. Huang, Z. Gao, 210 S. Jiao, X. He, M. Lai, Land use regression models coupled with meteorology to model spatial and tem-

poral variability of NO2 and PM10 in Changsha, China, Atmospheric Environment 116 (2015) 272-280. doi:10.1016/j.atmosenv.2015.06.056.
6. S.I.V. Sousa, F.G. Martins, M.C.M. Alvim-Ferraz, M.C. Pereira, 2007. Multiple linear regression and artificial neural networks based on principal components to predict ozone concentrations. Environmental Modelling & Software 22, 97-103. doi: 10.1016/j.envsoft.2005.12.002
7. Y. Rybarczyk, R. Zalakeviciute. Machine learning approach to forecasting urban pollution. IEEE Ecuador Technical Chapters Meeting (ETCM), 2016, 1-6. doi: 10.1109/ETCM.2016.7750810
8. C. Yan, S. Xu, Y. Huang, Y. Huang, Z. Zhang. Two-phase Neural Network Model for Pollution Concentrations Forecasting. Fifth International Conference on Advanced Cloud and Big Data, 2017. doi: 10.1109/CBD.2017.73
9. M. M. Dedovic, I. Turkovic, T. Konjic, S. Avdakovic, N. Dautbasic. Forecasting PM10 concentrations using neural networks and system for improving air quality. XI International Symposium on Telecommunications (BIHTEL), October 24-26, 2016. doi: 10.1109/BIHTEL.2016.7775721
10. B. Owen, H. A. Edmunds, D. J. Carruthers, R. J. Singles, Prediction of total oxides of nitrogen and nitrogen dioxide concentrations in a large urban area using a new generation urban scale dispersion model with integral chemistry model, Atmospheric Environment 34 (3) (2000) 397-406.
11. S. Tunlathorntham, S. Thepanondh, Prediction of Ambient Nitrogen Dioxide Concentrations in the Vicinity of Industrial Complex Area, Thailand, Air, Soil and Water Research 10 (2017). doi: 10.1177/1178622117700906.
12. A. U. G. I. (GPO), J. Orme-Zavaleta, Office of Research and Development; Ambient Air Monitoring Reference and Equivalent Methods: Designation of One New Equivalent Method, Federal Register 82 (90).
13. S.Gilde, GAW Brief des DWD – Der CAPS-Monitor, ein neues Instrument zur Messung von Stickstoffdioxid in Umgebungslust, https://www.dwd.de/DE/forschung/atmosphaerenbeob/zusammensetzung_atmosphaere/hohenpeissenberg/download/gaw_briefe/gaw_brief_059_de_pdf.pdf?__blob=publicationFile
14. Y.C. Chen. A Tutorial on Kernel Density Estimation and Recent Advances. Department of Statistics. University of Washington. September 2017. doi: 10.1080/24709360.2017.1396742
15. A.C. Guidoum: Kernel Estimator and Bandwidth Selection for Density and its Derivatives. Department of Probabilities & Statistics. Faculty of Mathematics. University of Science and Technology Houari Boumediene. BP 32 El-Alia, U.S.T.H.B, Algeria. October 2015.
16. C.-Y. Chau, D. J. Henderson, C. F. Parmeter: Plug-in Bandwidth Selection for Kernel Density Estimation with Discrete Data. Econometrics 2015, 3, 199-214; ISSN 2225-1146 doi:10.3390/econometrics3020199
17. X. Wu: Robust Likelihood Cross-Validation for Kernel Density Estimation. Journal of Business & Economic Statistics (2018). doi: 10.1080/07350015.2018.1424633
18. B. W. Silverman. Density Estimation for Statistics and Data Analysis. Chapman and Hall, 1986.
19. R. Menard, M. Deshaies-Jacques, Evaluation of Analysis by Cross-Validation. Part I: Using Verification Metrics, Atmosphere 2018 (9) (2018) 86. doi:10.3390/atmos9030086.
20. P. Refaeilzadeh, L. Tang, H. Liu, Cross-validation (2009) 532-538.
21. T.-T. Wong, Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation, Pattern Recognition 48 (9) (2015) 2839-2846. doi:10.1016/j.patcog.2015.03.009.

22. Alho, J. and B. Spencer. (2005). Statistical Demography and Forecasting. Dordrecht, The Netherlands: Springer. W. Alonso and P. Starr (Eds.). The Politics of Numbers. New York: Russell Sage.
23. M. V. Shcherbakov, A. Brebels, N. L. Shcherbakova, A. P. Tyukov, T. A. Janovsky, V. A. Kamaev, A survey of forecast error measures, World Applied Sciences Journal 24 (2013) 171-176. doi: 10.5829/idosi.wasj.2013.24.itmies.80032.

Indoor Localization based on Advanced Point-Mass Filtering Method

Miroslav Opiela

Institute of Computer Science, P.J. Šafárik University, Faculty of Science
Jesenná 5, 041 54 Košice, Slovak Republic
miroslav.opiela@upjs.sk

Abstract. In this paper, Advanced point-mass (PM) method is utilized to estimate a user position in indoor environment. Noisy measurements of acceleration and walk bearing are acquired from common smartphone sensors. A posteriori stochastic system state is approximated using bayesian filtering when a step is detected. In Advanced PM filtering approach, a state space is discretized and represented by a floating regular grid or multigrid for multimodal probability density functions. The method incorporates efficient partial methods, e.g., boundary-based grid design and thrifty convolution which supports real-time positions estimations. The Advanced PM method application to two-dimensional single floor indoor localization is proposed and discussed. The position estimation accuracy is evaluated and compared with the grid-based filtering approach.

Keywords: Advanced Point-Mass Method· Indoor Localization · Bayesian Filtering · Grid Design

1 Introduction

User navigation in indoor environments introduces a few particular challenges compared to the outdoor navigation. Besides multiple floors building structure and visualization opportunities, the main difference is in the positioning method as satelite signal is not available indoors. No unique and widely adopted solution is established for estimating the current position of a user in the building.

Some approaches require additional infrastructure, its calibration and maintenance, e.g., so-called pseudolites to transmit signals detectable by GPS receivers [17], solutions based on Bluetooth Low Energy (BLE) devices [19], Ultra WideBand technologies [1] or existing WiFi access points [10]. On the contrary, a building independent technique called Pedestrian Dead Reckoning (PDR) exploits human kinematics. It incorporates processed noisy sensor measurements in format of detected steps and step headings into the relative user position estimation. A precise model of the building is required to obtain sufficient results. An initial user location is determined manually or automatically using markers (e.g., QR codes) or with external signal (e.g., GPS or BLE beacons). A current

position is computed from previous estimations whilst noisy measurements increase the accuracy error. The error is decreased due to the building structure as approaching walls or it may be reduced by additional data using sensor fusion. As an illustration, Chen et al. [5] proposed a solution where current relative position is produced by PDR approach and corrected by Wi-Fi localization and recognized landmarks.

The goal of this work is to confirm the applicability of the Advanced PM technique for indoor localization. The paper is organized as follows. Section 2 outlines the related work of PDR solutions covering various implementations of bayesian filtering. Section 3 introduces the proposed approach for user localization. In section 4, the general bayesian filtering technique is described and the positioning problem is discussed in terms of filtering. Advanced PM algorithm and its essential features are outlined and implementation details are discussed in section 5. In section 6, the estimation accuracy is evaluated and analyzed.

2 Related Work

Significant research focus is on Inertial Measurement Unit (IMU) which is a device widely used in aircrafts and spacecrafts. It contains the accelerometer, the gyroscope and often the magnetometer making it suitable for analyzing pedestrian gait [8]. Despite its availability as a low-cost device, a smartphone introduces new possibilities to widely adopted indoor navigation especially in crowded buildings with high level of users, e.g., airports. Smartphones with the same embedded sensors allow to elaborate the research using the same principles for positioning, even though the way of holding the device is different.

Bayesian filtering is an essential component to deal with uncertainty which is expressed by a probability distribution over random variables representing a system state. Different implementations of the filter [3] are adopted for the indoor localization application. Kalman filter represents the probability density function (pdf) by unimodal Gaussian distribution. Extended versions of the filter are applied on the localization problem to involve multimodal distribution and nonlinear filtering, e.g., Extended Kalman filter (EKF) and Unscented Kalman filter [9]. Kalman filters may be applied to other particular problems related to positioning, e.g., EKF application for step length model calibration [15].

Particle filter represents the probability density function by a set of particles. This discrete implementation of bayesian filtering is not restricted to Gaussian and linear problems. Hafner et al. [7] show that the Particle filter outperforms the Kalman filter with less accurate measurements.

Project HimLoc [14] utilize the Particle filter for fusion of WiFi fingerprinting location estimations, PDR estimations based on sensors and activity classification. Moder et al. [12] use the Particle filter for 2D localization. Moreover, the Kalman filter is used for the activity recognition and a floor detection.

In our previous work the grid-based filter is examined [6]. This discrete filter approximates the probability distribution by a finite set of points. A map is tessellated into a regular grid where typically center points of the grid cells

preserve the belief value and coordinates of these points are immutable unlike the particles in the Particle filter. Computation demands increase with the number of dimensions of the grid, i.e., number of random variables describing a system state.

Advanced point-mass (PM) filter is built on a point-mass filter introduced by Bucy and Senne in 1971 [4] as a method based on a numerical solution of the integral in the Bayesian recursive relations. Šimandl et al. [16] proposed a framework for the numerical approach to nonlinear state estimation with focus on reducing computational demands especially for multimodal pdf and the grid design. The aim of this work is to apply the proposed method with the satisfactory results in nonlinear state estimation on indoor localization problem and compare it with the grid-based filter. An accuracy improvement is expected as the Advanced PM filter may reduce the error caused by discretization of the space with adjustable grids (density of points) in every iteration.

3 Indoor Localization

Although the paper is dedicated to the Advanced PM method and Bayesian filtering, the outline of comprehensive indoor localization approach is introduced. Proposed indoor positioning method for smartphones involves five main components on input in order to estimate the user (device) current position:

- **Detected step** is identified using acceleration measurements. The accelerometer is the common sensor available almost in all smartphones. Step detection may be supported directly by the platform. Otherwise, approaches based on step phases recognition [6] or significant acceleration decrease detection [11] are sufficient with high accuracy.
- **Bearing** is acquired from magnetometer and/or gyroscope. A mobile device platform may provide the heading included in the rotation vector of the device. Bearing values are filtered to obtain more stable values.
- **Map model** is required for PDR to declare inaccessible places in the building. Floor plans are generated with a few centimeter precision and annotated with particular tools.
- **QR codes** enables absolute position detection. Strategically placed in the building, QR codes provide coordinates of the position with other relevant data. However, it may be more beneficial to incorporate other localization method for initial position (e.g., based on WiFi, Bluetooth). Compared to QR codes or manual detection by user, these methods require less efford from users but may be less precise in the positioning.
- **Floor detection** is based on barometer measurements [18]. Identification of a floor number is not trivial and require additional data. However, it is possible to detect the floor change with the barometer. Detection of transitions between floors along with the floor plans enable us to define a bayesian filtering state only with two position coordinates (x and y), i.e., a current position is estimated only within the same floor. The vertical coordinate (z) is not included and when the floor transition is detected, the Bayes filter

loads the map of the identified floor and is reset with the estimated position determined by the location of used elevators, stairs or other types of transitions. In case the barometer is not embedded in the smartphone, another approach to floor detection utilize Wi-Fi infrastructure [2].

Note that the step length is not available as input. Incorrect step length estimations may produce significant decrease of positioning accuracy. Autocorrection of the estimation is possible under certain circumstances. Nevertheless, some level of inaccuracy is expected and the indoor positioning system should be able to overcome underestimated or overestimated step lengths.

Noisy measurements are processed by a bayesian filtering method in order to approximate posterior pdf of the state. The place in building represented by coordinates of a point with the highest belief value is denoted as the current position estimation. Real-time indoor navigation application requires filtering computation with feasible computational demands. The posterior pdf calculation is expected to finish before next step is detected.

4 Bayesian Filtering

Imperfect sensors producing noisy measurements are not capable of indicating the accurate state (the user position in this use case). The uncertainty introduced by measurements is modeled by a Bayes filter which probabilistically estimate a dynamic system state recursively over time using these noisy measurements. Bayes filters represent the state at time t by random variables $\mathbf{x}_t$. Belief is a probability distribution over $\mathbf{x}_t$. The aim of the filter is to sequentially estimate the conditional pdf $p(\mathbf{x}_t|\mathbf{z}^t)$ of the state $\mathbf{x}_t$ given the sensor data as measurements $\mathbf{z}^t = [\mathbf{z}_0^\mathbf{T}, \mathbf{z}_1^\mathbf{T}, \ldots, \mathbf{z}_t^\mathbf{T}]^\mathbf{T}$ for the discrete-time stochastic system:

$$\mathbf{x}_{t+1} = \mathbf{f}_t(\mathbf{x}_t) + \mathbf{w}_t, \qquad t = 0, 1, 2, \ldots \tag{1}$$

$$\mathbf{z}_t = \mathbf{h}_t(\mathbf{x}_t) + \mathbf{v}_t, \qquad t = 0, 1, 2, \ldots \tag{2}$$

where $\mathbf{x}_t$ is a vector representing the system state. For indoor positioning problem $\mathbf{x}_t \in \mathbb{R}^2$ and single state is represented by x and y coordinates. More advanced approaches in terms of state description may include also third coordinate, rotation, acceleration and velocity values in the state vector. Measurements $\mathbf{z}^t$ are denoting sensor data. At time t, a step is detected and the corresponding bearing value is passed as $\mathbf{z}_t = \{\alpha_t\}, \alpha_t \in \mathbb{R}$.

Vector functions $\mathbf{f}_t : \mathbb{R}^2 \to \mathbb{R}^2$ and $\mathbf{h}_t : \mathbb{R}^2 \to \mathbb{R}$ are known and $\mathbf{w}_t \in \mathbb{R}^2$, $\mathbf{v}_t \in \mathbb{R}$ represent known mutually independent zero-mean state and measurement noise, respectively. State transition function in this solution for arbitrary point of state $\mathbf{x}_t$ is defined as

$$\mathbf{f}_t(x, y) = (x + L \times sin(\alpha_t), y - L \times cos(\alpha_t)) \tag{3}$$

The solution of the filtering problem is given by the Bayesian recursive relations:

$$p(\mathbf{x}_t|\mathbf{z}^t) = \frac{p(\mathbf{x}_t|\mathbf{z}^{t-1})p(\mathbf{z}_t|\mathbf{x}_t)}{\int p(\mathbf{x}_t|\mathbf{z}^{t-1})p(\mathbf{z}_t|\mathbf{x}_t)d\mathbf{x}_t} \tag{4}$$

$$p(\mathbf{x}_{t+1}|\mathbf{z}^t) = \int p(\mathbf{x}_t|\mathbf{z}^t)p(\mathbf{x}_{t+1}|\mathbf{x}_t)d\mathbf{x}_t \tag{5}$$

where $p(\mathbf{x}_0|\mathbf{z}^{-1}) = p(\mathbf{x}_0)$. State $\mathbf{x}_0$ is known with the belief value 1 for the declared initial position and 0 for all other positions.

Kalman filter, Particle filter and other filters described in related work section are implementations of Bayesian filtering that approximate belief distribution when assumption for optimal Bayesian solutions do not hold, i.e., the analytic solution is intractable [3].

5 Advanced Point-Mass Filtering

Advanced point-mass filter is the numerical approach to solve Bayesian recursive relations which is based on approximation of continuous state space by a grid of points. Similar to the grid-based filter, the belief values are computed only on these points. However, the floating grid used in Advanced PM enables the grid to transform and rotate according to the nonnegligible support of the pdf. In [16], the Advanced PM method is compared with Particle filter for nonlinear state estimation and the lower computational demands are achieved without a loss of accuracy.

5.1 Algorithm

Adapted algorithm for single floor indoor localization based on the algoritm introduced by Šimandl et al. [16] is proposed. A pdf $p(\mathbf{x}_m|\mathbf{z}^t)$ is represented by a grid of points $\Xi_m(N_m) = \{\xi_{m,i} : \xi_{m,i} \in \mathbb{R}^2, i = 1, \ldots, N_m\}$, by a set of volume masses for a grid cell with their centers in points $\mathcal{D}_m = \{\Delta\xi_{m,i}, i = 1, \ldots, N_m\}$ and by a set of belief values at the points $\mathcal{P}_{m|t} = \{P_{m|t,i} : P_{m|t,i} = p(\xi_{m,i}|\mathbf{z}^t), \xi_{m,i} \in \Xi_m(N_m)\}$.

Initialization Define an initial grid $\Xi_0(N_0)$ in $\mathbb{R}^2$ for the prior pdf $p(\mathbf{x}_0|\mathbf{z}^{-1})$: $\Xi_0(N_0) = \{\xi_{0,i}; i = 1, \ldots, N_0\}$ based on the initial position, volumes $\Delta\xi_{0,i}$ and set of pdf values $\mathcal{P}_{0,-1} = \{P_{0|-1,i}; i = 1, \ldots, N_0\}$. Then proceed for $t = 0, 1, 2, \ldots$

Step 1 - Filtering Compute values of the approximate filtering pdf at points of the grid $\Xi_t(N_t)$ using the equation (4), for $i = 1, \ldots, N_t$

$$P_{t|t,i} = c_t^{-1} P_{t|t-1,i} p_{\mathbf{v}_t}(\mathbf{z}_t - \mathbf{h}_t(\xi_{t,i}))$$

where $p_{\mathbf{v}_t}(\mathbf{z}_t - \mathbf{h}_t(\xi_{t,i}))$ denotes $p(\mathbf{z}_t|\mathbf{x}_t)$ from the equation (4) and the normalization constant is defined as $c_t^{-1} = \sum_{i=1}^{N_t} \Delta\xi_{t,i} P_{t|t-1,i} p_{\mathbf{v}_t}(\mathbf{z}_t - \mathbf{h}_t(\xi_{t,i}))$

Step 2 - Time update of grid Transform the grid $\Xi_t(N_t)$ to a grid $H_{t+1} = \{\eta_{t+1,i}; i = 1, \ldots, N_t\}$ by the system dynamics.

$$\eta_{t+1,i} = \mathbf{f}_t(\xi_{t,i})$$

where $\mathbf{f}_t$ denotes the state transition function defined in (3).

Step 3 - Grid redefinition Redefine the grid $H_{t+1}(N_t)$ to obtain a new grid $\Xi_{t+1}(N_{t+1})$ for $\mathbf{x}_{t+1}$ with the same structural properties as the original grid $\Xi_{t+1}(N_{t+1}) = \{\xi_{t+1,i}; i = 1, \ldots, N_{t+1}\}$. Compute $\mathcal{P}_{t+1}$. Note that the state noise $\mathbf{w}_t$ is incorporated in the pdf approximated by $\Xi_{t+1}(N_{t+1})$.

Step 4 - Prediction Compute values of the approximate predictive pdf at the new grid $\Xi_{t+1}(N_{t+1})$ using (5) for $j = 1, \ldots, N_{t+1}$.

$$P_{t+1|t,j} = \sum_{i=1}^{N_t} \Delta\xi_{t,i} P_{t|t,i} p_{\mathbf{w}_t}(\xi_{t+1,j} - \eta_{t+1,i})$$

where $p_{\mathbf{w}_t}(\xi_{t+1,j} - \eta_{t+1,i})$ denotes the transition pdf $p(\mathbf{x}_{t+1}|\mathbf{x}_t)$.

5.2 Method Features

The computational time for the Advanced PM filtering technique may be reduced applying particular methods for grid manipulation. An outline of the framework [16] is introduced:

- **Anticipative approach** includes computation of the predictive pdf support Ω_{t+1} for $p(\mathbf{x}_{t+1}|\mathbf{z}^t)$. Number of grid points and the volume mass is estimated without a loss of accuracy.
- **Boundary-based grid design** extends the anticipative approach and ensures a comprehensive approximation of state space by a grid for multimodal pdf's and non-Gaussian pdf's.
- **Thrifty convolution** derives significant points in transformed grid $\bar{H}_{t+1}$ for points $\xi_{t+1,j}$. Convolution is performed on nonrotated grids centered at the origin of the state space.
- **Multigrid representation** extends the method to support multimodal pdf's. Grid splitting and merging methods are introduced to obtain multiple grids which are handled independently.

5.3 Implementation Remarks

Walls and other obstacles contribute to the localization error reducement. It is decisive to examine a grid point accessibility during convolution in the prediction phase (Step 4). An accessible point corresponds to a place in building where it is possible to be situated, i.e., the position is not inside walls, obstacles or

beyond the walls. It is beneficial to tessellate the map into grid cells (in this case 33×33 centimeters) and label cells as accessible or not. A point is assigned to a cell to reduce computational demands. In some scenarios, it is required to examine accessibility also for places between two points, e.g., if thin enough wall is between the considered point at time t and the corresponding point at time $t+1$.

State noise $\mathbf{w}_t$ is zero-mean white noise with covariance matrix $cov(\mathbf{w}_t) = diag\{\sigma^2, \sigma^2\}$ where standard deviation $\sigma = 15cm$ for every iteration at time t.

State $\mathbf{x}_t$ does not involve heading information only coordinates. Bearing values determine the state transition function. Therefore, the measurement noise $\mathbf{v}_t$ was omitted, i.e., set to zero-mean and covariance matrix with zeros.

Other parameters were set as follows: $\gamma = 0.5$ for anticipative method, $a = 4$ for boundary-based grid design and $K = 3$ for multigrid design.

Reduction of points with negligible belief values is not performed in the evaluation. However, it may be unavoidable to reduce these belief values in complex large-scale scenario. The application impacts the number of points in convolution and therefore the procedure of resetting negligible values increases the speed of computation.

6 Evaluation

A set of experiments was performed in order to evaluate the proposed Advanced PM method and to demonstrate a sufficient quality of localization compared with the grid-based filtering introduced in [6]. The aim is to evaluate the localization accuracy of the filtering methods with a methodology similar to the one used for indoor localization competitions [13].

Similar or slightly better results are expected for the Advanced PM filter. The localization error is increased by multiple factors, e.g., noisy sensor measurements, incorrect step length estimation, space discretization. The grid-based filter computes the estimated position only on points denoting centers of grid cells. The Advanced PM method redefines grids at every iteration of filtering. Therefore, the distance between two points may be smaller compared to fixed positions of the grid-based filter. The discretization of the space may not contribute to the overall localization error to such an extent as with fixed grid points.

6.1 Evaluation Setup

The experiment took place in a faculty building on a single floor (see figure 1). A subject with a low-end hand-held tablet walked the 87 meters long predefined path with markers sticked to the floor. Markers denote coordinates where the quality of estimation is evaluated. The Euclidean distance between an estimated position and the real marker location determines the accuracy of the localization method. All sensor measurements are recorded for all experiments along with a camera record from another synchronized device which enables following investigation and comparison of proposed methods and their parameters under the same conditions.

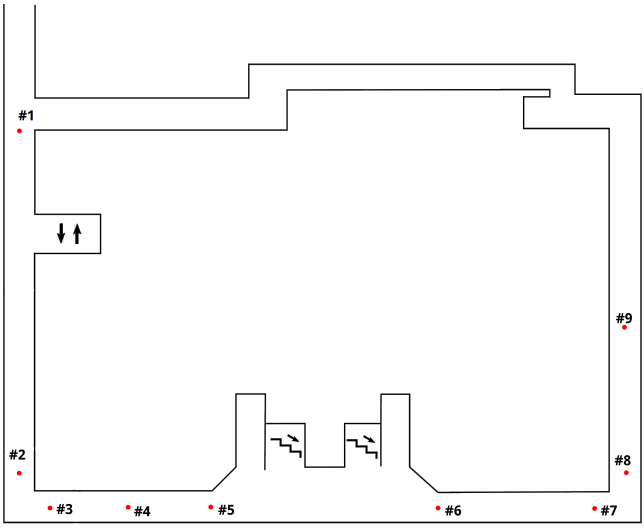


Fig. 1. A footprint of the faculty building where experiments were performed. Ordered red dots denotes markers where the localization accuracy is computed. Traveled distance of the path between marker #1 and #9 is 87 meters.

The real-time simulations of sensor readings were executed with different step lengths to analyze the robustness of the method. Approximate real step lengths were calculated from distance covered by twenty steps of every participant. Table 1 shows measured step lengths and selected step lengths rounded to the multiple of five for comparison of proposed methods. Best step lengths are selected based on rough observations in such a way that a set of estimated positions appear to create the smoothest path and have the lowest localization errors on markers for both methods. Note that recordings #4 and #7 were executed by the same participant.

Table 1. Approximate real step lengths for executed experiments.

Experiment	#1	#2	#3	#4	#5	#6	#7
Computed Step Length in cm	79.1	91.1	79.1	84.7	66.4	78.0	84.7
Selected Step Length in cm	80	90	80	85	65	80	85
Best Step Length in cm	90	85	85	90	80	80	90

6.2 Localization Accuracy

All recordings were analyzed in offline mode with different estimated step lengths between 70 and 95 centimeters for both methods. A few executed measurements

with markedly incorrect selected step length result to high errors. As an example, the underestimated step length produce less distance in simulation compared with real covered distance after a specified number of steps. Turning left between markers #7 and #8 may result to position estimation near stairs between markers #5 and #6 with the real position of participant on marker #9. These recordings for both methods were removed from the following results.

The average localization errors on the markers obtained from 70 walks (35 for each method) are listed in table 2 and visualized in figure 2. In the data, one can see the ability of the bayesian filters to utilize the changes in direction in order to reduce the positioning error. Markers #3 and #8 are a few meters after 90°-turnings and the localization error decreases on these markers for both methods. Walls as obstacles improve the positioning accuracy which supports the usage of PDR methods especially in buildings with narrow corridors with turnings.

Table 2. Average localization error on individual markers.

Method/Marker	#1	#2	#3	#4	#5	#6	#7	#8	#9
Grid-based	0.79	2.63	2.44	1.90	2.29	4.41	5.00	4.34	4.77
Advanced PM	0.80	2.45	2.28	1.70	2.02	3.71	4.31	3.55	3.42

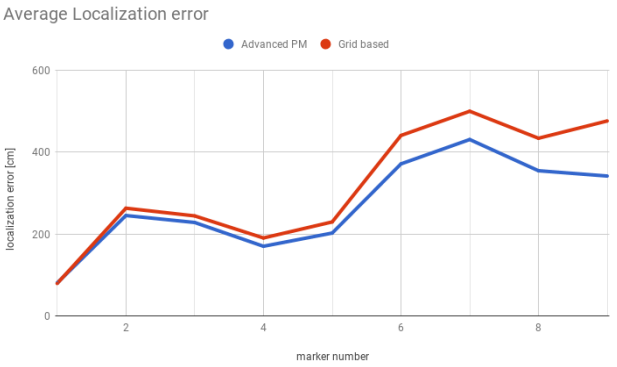


Fig. 2. The average localization error on individual markers. Blue line denoting Advanced PM method and red line for grid-based approach show comparable quality of the position estimation.

Both proposed methods have a non-zero value on the first marker which is caused by the way of computing an estimated position. A participant pressed the button when crossing the marker. A posterior belief density is calculated when a step is detected. The first estimated location after the identified marker

is labeled as the marker position. Therefore, the error value lower than the step length may be considered as negligible.

Table 3 and figure 3 show localization accuracy with the best possible step length estimation. Average error for individual markers are computed from a single best record for each test walk, i.e., seven measurements for each method. The impact of 90° turnings is obvious. Moreover, the error increases slowly on the segments between junctions compared with the underestimated or overestimated values of the step length.

Table 3. Average localization error with the optimal possible step length estimation.

Method/Marker	#1	#2	#3	#4	#5	#6	#7	#8	#9
Grid-based	0.79	1.25	1.29	1.22	1.15	2.13	1.96	0.80	1.71
Advanced PM	0.84	1.78	1.38	1.24	1.20	1.71	1.81	0.97	1.60

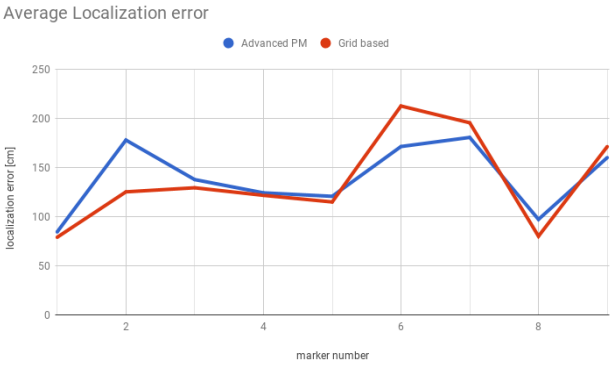


Fig. 3. Average localization error on individual markers with the optimal step length estimation.

Figure 4 demonstrates similar results of robustness for both methods. An average error is computed for all markers of a specified test walk. These errors are compared for different step lengths.

7 Conclusion

The Advanced PM filter achieves good results for nonlinear filtering. It reduces computational demands without a loss in the position estimation quality. The goal of this paper is to confirm the applicability of the Advanced PM filter for indoor localization. Bayesian filtering in general and Advanced PM filter as the

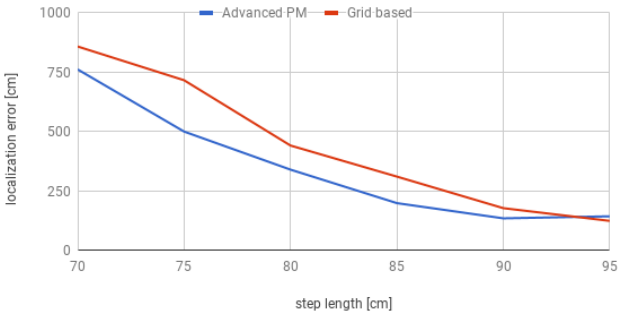


Fig. 4. Average localization error for individual step lengths estimations measured on the test walk #4.

implementation of Bayesian filtering are discussed and customized for the indoor localization problem. The evaluation supports the idea of Advanced PM method application in the field of positioning. The localization results show similar quality for the grid-based filter and the Advanced PM filter. However, better results were obtained for the Advanced PM filter in this evaluation scenario.

An opportunity to utilize all the features of the technique still remains unfulfilled. As an example, multigrid design with independently handled grids provides sufficient method for sensor fusion, e.g., incorporating Wi-Fi measurements.

Following evaluation experiments and analysis may include comparison with the Particle filter. The investigation may incorporate a sensor fusion case and more complex environment as well as method performance analysis with different parameters values, e.g., noise matrix.

Acknowledgement. The work presented in this paper was partially supported by the Slovak Grant Agency of the Ministry of Education and Academy of Science of the Slovak Republic under grant no. 1/0056/18 and by the Slovak Research and Development Agency under the contract No. APVV-15-0091.

References

1. Alarifi, A., Al-Salman, A., Alsaleh, M., Alnafessah, A., Al-Hadhrami, S., Al-Ammar, M., Al-Khalifa, H.: Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances. Sensors **16**(5), 707 (2016)
2. Alsehly, F., Arslan, T., Sevak, Z.: Indoor positioning with floor determination in multi story buildings. In: 2011 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2011. pp. 1–7. IEEE (sep 2011)
3. Arulampalam, M., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Transactions on Signal Processing **50**(2), 174–188 (2002)

4. Bucy, R.S., Senne, K.D.: Digital synthesis of non-linear filters. Automatica **7**(3), 287–298 (may 1971)
5. Chen, Z., Zou, H., Jiang, H., Zhu, Q., Soh, Y.C., Xie, L.: Fusion of WiFi, smartphone sensors and landmarks using the kalman filter for indoor localization. Sensors **15**(1), 715–732 (jan 2015)
6. Galčík, F., Opiela, M.: Grid-based indoor localization using smartphones. In: 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN). pp. 1–8. IEEE (oct 2016)
7. Hafner, P., Moder, T., Wieser, M., Bernoulli, T.: Evaluation of smartphone-based indoor positioning using different Bayes filters. In: 2013 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2013. pp. 1–10. IEEE (oct 2013)
8. Jiménez, A.R., Seco, F., Prieto, C., Guevara, J.: A comparison of pedestrian dead-reckoning algorithms using a low-cost MEMS IMU. In: WISP 2009 - 6th IEEE International Symposium on Intelligent Signal Processing - Proceedings. pp. 37–42. IEEE (aug 2009)
9. Julier, S.J., Uhlmann, J.K.: New extension of the Kalman filter to nonlinear systems. vol. 3068, p. 182. International Society for Optics and Photonics (jul 1997)
10. Ledlie, J., Park, J.g., Curtis, D., Cavalcante, A., Camara, L., Costa, A., Vieira, R.: Molé: a scalable, user-generated WiFi positioning engine. Journal of Location Based Services **6**(2), 55–80 (2012)
11. Link, J.Á.B., Smith, P., Viol, N., Wehrle, K.: FootPath: Accurate map-based indoor navigation using smartphones. In: 2011 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2011. pp. 1–8. IEEE (sep 2011)
12. Moder, T., Hafner, P., Wisiol, K., Wieser, M.: 3D indoor positioning with pedestrian dead reckoning and activity recognition based on Bayes filtering. In: IPIN 2014 - 2014 International Conference on Indoor Positioning and Indoor Navigation. pp. 717–720. IEEE (oct 2014)
13. Potortì, F., Park, S., Ruiz, A.R.J., Barsocchi, P., Girolami, M., Crivello, A., Lee, S.Y., Lim, J.H., Torres-Sospedra, J., Seco, F., Montoliu, R., Mendoza-Silva, G.M., Rubio, M.D.C.P., Losada-Gutiérrez, C., Espinosa, F., Macias-Guarasa, J.: Comparing the performance of indoor localization systems through the EvAAL framework. Sensors **17**(10), 2327 (oct 2017)
14. Radu, V., Marina, M.K.: HiMLoc: Indoor smartphone localization via activity aware pedestrian dead reckoning with selective crowdsourced WiFi fingerprinting. In: 2013 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2013. pp. 1–10. IEEE (oct 2013)
15. Renaudin, V., Demeule, V., Ortiz, M.: Adaptative pedestrian displacement estimation with a smartphone. In: 2013 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2013. pp. 1–9. IEEE (oct 2013)
16. Šimandl, M., Královec, J., Söderström, T.: Advanced point-mass method for nonlinear state estimation. Automatica **42**(7), 1133–1145 (2006)
17. Wang, J.: Pseudolite applications in positioning and navigation: Progress and problems. Journal of Global Positioning Systems **1**(1), 48–56 (2002)
18. Xia, H., Wang, X., Qiao, Y., Jian, J., Chang, Y.: Using multiple barometers to detect the floor location of smart phones with built-in barometric sensors for indoor positioning. Sensors **15**(4), 7857–7877 (mar 2015)
19. Yang, J., Wang, Z., Zhang, X.: An iBeacon-based Indoor Positioning Systems for Hospitals. International Journal of Smart Home **9**(7), 161–168 (2015)

Unary Patterns of Size Four with Morphic Permutations

Kamellia Reshadi

Institut für Informatik, Christian-Albrechts-Universität zu Kiel.
kre@informatik.uni-kiel.de

Abstract. We investigate the avoidability of unary patterns of size of four with morphic permutations. More precisely, we show that, for the positive integers i, j, k, the sizes of the alphabets over which a pattern $x\pi^i(x)\pi^j(x)\pi^k(x)$ is avoidable are an interval of the integers (where x is a word variable and π is a function variable with values in the set of all morphic permutations of the respective alphabets). We also show how to compute a good approximation of this interval. This continues the work of [Manea et al., 2015], where a complete characterisation of the avoidability of cubic patterns with permutations was given.

1 Introduction

The avoidability of patterns in infinite words is an old area of interest with a first systematic study going back to Thue [5,6]. In these initial papers it was shown that there exist a binary infinite morphic word and a ternary infinite morphic word that avoid cubes and squares, respectively. That is, these infinite words do not contain instances of the patterns xxx and xx, respectively. The most important classical results on avoidability are surveyed in several chapters of [3]; see, e.g., Chapters 2 and 7 of [3] and the references therein.

In this article, we are studying the avoidability of repetitions in a generalised setting. Namely, we are interested in the avoidability of unary patterns with functional dependencies between variables. We are considering patterns like $x\pi^i(x)\pi^j(x)\pi^k(x)$, where x is a word variable while π is function variable, which can be replaced by bijective morphisms only. The instances of such patterns over an alphabet Σ are obtained by replacing x with a concrete word, and π by a morphic permutation of Σ. For example, an instance of the pattern $x\pi(x)x\pi(x)$ over $\Sigma = \{a, b\}$ is the word $uvuv$ such that $|u| = |v|$, and v is the image of u under any permutation on the alphabet. Considering the permutation $a \to b$, and $b \to a$, then $aba|bab|aba|bab$ is an instance of $x\pi(x)x\pi(x)$.

In this setting, we continue the work of [1] and [4] as follows. In [4], a complete characterisation of the avoidability of cubic patterns with permutations $x\pi^i(x)\pi^j(x)$ was given. Furthermore, in [1] it was shown that there exists a ternary word that avoids all patterns $\pi_{i_1}(x)\dots\pi_{i_r}(x)$ where $r \geq 4$, x a word variable over some alphabet Σ, with $|x| \geq 2$ and $|\Sigma| \geq 3$, and the π_{i_j} function variables that may be replaced by anti-/morphic permutations of Σ. However,

this result only holds when the length of x is restricted to be at least 2. Also, in an extension [2] of the aforementioned paper [1], it was shown that all patterns $\pi^{i_1}(x)\dots\pi^{i_n}(x)$ with $n \geq 4$ under morphic permutations are avoidable in alphabets of size $2, 3$, and 4, but there exist patterns which are unavoidable in alphabets of size 5. We extend these results by showing how to determine exactly, for a given unary pattern $\mathcal{P}$ of size four with permutations, which are the alphabets in which it is avoidable.

The main result of our paper is that given $i, j, k \geq 0$, we show how to compute the value σ such that the pattern $x\pi^i(x)\pi^j(x)\pi^k(x)$ is unavoidable in alphabets of size at least $\sigma + 1$ and avoidable in alphabets of size $2, 3, 4, \dots, \sigma - 1$. The avoidability of this pattern in alphabets of size σ has to be analysed individually for some i, j, k. Acoordingly, we show that for each pattern there exists an interval (whose left end is 2 and right end is defined based on the respective parameters) such that over each alphabet whose size is in the respective interval, there exists an infinite word that does not contain instances of the given pattern. This shows that the main result of [4] holds in the more general case of unary patterns of size four. However, the technicalities we develop here are much more involved.

The structure of the paper is as follows: we first give a series of basic definitions and preliminary results. Then we define the aforementioned parameters, and show how to use them to compute, for a given pattern p, the value σ such that p is unavoidable over alphabets with $m > \sigma$ letters. Finally, we show the dual of the previous result: for alphabets with at most $\sigma - 1$ symbols the pattern p is avoidable. Due to space constraints, most of the repetitive technicalities of this paper (e.g., a list of infinite words avoiding certain patterns) are left out. However, the main part contains the major ideas needed to obtain the results we state.

2 Preliminaries

We define $\Sigma_k = \{0, \dots, k-1\}$ to be an alphabet with k letters; the empty word is denoted by ε. For words u and w, we say that u is a prefix (resp. suffix) of w, if there exists a word v such that $w = uv$ (resp. $w = vu$). If $f : \Sigma_k \to \Sigma_k$ is a permutation, we say that the order of f, denoted $\mathbf{ord}(f)$, is the minimum value $m > 0$ such that f^m is the identity. If $a \in \Sigma_k$ is a letter, the order of a with respect to f, denoted $\mathbf{ord}_f(a)$, is the minimum number m such that $f^m(a) = a$. A function $f : \Sigma_k^* \to \Sigma_k^*$ is a morphism if $f(xy) = f(x)f(y)$ for all words x, y; f is a morphic permutation if the restriction of f to Σ_k is a permutation of Σ_k.

A pattern with functional dependencies is a term over (word) variables and function variables (where concatenation is an implicit functional constant). For example, $x\pi(y)\pi(\pi(x))y$ is a pattern involving the variables x and y and the function variable π. An instance of a pattern p in Σ_k is the result of substituting uniformly every variable by a word in Σ_k^+ and every function variable by a function over Σ_k^*. A pattern is avoidable in Σ_k if there is an infinite word over Σ_k that does not contain any instance of the pattern.

In this paper, we consider only unary patterns (i.e., containing only one variable) with morphic permutations, that is, all function variables are unary and are substituted by morphic permutations only.

The infinite Thue-Morse word t is defined as $t = \lim_{n\to\infty} \phi_t^n(0)$, for the morphism $\phi_t : \Sigma_2^* \to \Sigma_2^*$ where $\phi_t(0) = 01$ and $\phi_t(1) = 10$. It is well-known (see [3]) that the word t avoids the patterns xxx (cubes) and $xyxyx$ (overlaps).

The infinite ternary Thue word h is defined as $h = \lim_{n\to\infty} \phi_h^n(0)$, for the morphism $\phi_h : \Sigma_3^* \to \Sigma_3^*$ where $\phi_h(0) = 012$, $\phi_h(1) = 02$ and $\phi_h(2) = 1$. The infinite word h avoids the pattern xx (squares).

This paper is related to the study of the avoidability of cubic patterns with permutations from [4]. In the respective paper for a given pattern $x\pi^i(x)\pi^j(x)$ the authors defined the following four values: $\alpha_1 = \inf\{t : t \nmid |i-j|, t \nmid i, t \nmid j\}$, $\alpha_2 = \inf\{t : t \mid |i-j|, t \nmid i, t \nmid j\}, \alpha_3 = \inf\{t : t \mid i, t \nmid j\}, \alpha_4 = \inf\{t : t \nmid i, t \mid j\}$. Further, for $k = \min\{\max\{\alpha_1, \alpha_2\}, \max\{\alpha_1, \alpha_3\}, \max\{\alpha_1, \alpha_4\}\}$, it was shown that $x\pi^i(x)\pi^j(x)$ is unavoidable in Σ_m, for $m \geq k$, and avoidable in Σ_m, for $4 \leq m < k$. The avoidability of $x\pi^i(x)\pi^j(x)$ in Σ_2 and Σ_3 was separately investigated, and a complete characterisation of the alphabets over which a pattern $x\pi^i(x)\pi^j(x)$ is avoidable was obtained.

The reader is referred to [3,4,2] for further details. All computer programs referenced in this paper can be found at `http://media.informatik.uni-kiel.de/zs/patterns.zip`.

3 Avoidability of patterns under permutations

In this section we try to identify an upper bound on the size of the alphabets Σ_m in which a pattern $x\pi^i(x)\pi^j(x)\pi^k(x)$, with $i, j, k \geq 0$ is unavoidable, when π is substituted by a morphic permutation.

In the pattern $x\pi^i(x)\pi^j(x)\pi^k(x)$, the factors x, $\pi^i(x)$, $\pi^j(x)$, or $\pi^k(x)$ are called x-items in the following. Our analysis is based on the relation between the possible images of the four x-items occurring in a pattern, following the ideas of [4]. For instance, we want to check whether in a possible image of our pattern, all four x-items can be mapped to a different word, or whether the second and the last x-items can be mapped to the same word, etc.

To achieve this, we define in Table 1 the parameters α_a, with $1 \leq a \leq 14$. Intuitively, they allow us to define, for a pattern $x\pi^i(x)\pi^j(x)\pi^k(x)$, which are the alphabets Σ_m in which we can model certain (in-)equality relationships between the images of the x-items. For example, in alphabets Σ_m with $m \geq \alpha_1$ we can assign values to x and π such that the images of every two of $\pi^i(x)$, $\pi^j(x)$, and $\pi^k(x)$ are different (and this property does not hold in alphabets with less than α_1 letters). Also, in Σ_m with $m \geq \alpha_2$ we can assign values to x and π such that the images of x and $\pi^i(x)$ are equal to some word, while the images of $\pi^j(x)$ and $\pi^k(x)$ are assigned to two other distinct words (also different between them; again, this property does not hold in smaller alphabets). To simplify, we use a simple digit-representation for any of these cases, defined in the last column of Table 1. In this representation of each α_a, we assign different digits to the

x-items that can be mapped to different words in alphabets of size at least α_a. For example, we use the representation 0123 for the case defined through α_1 and 0012 for the case defined by α_2. In general, when considering an α_a, we assign a 4-digit representation to the pattern $x\pi^i(x)\pi^j(x)\pi^k(x)$ in the following manner: we start with 0, and then put a 0 on all of the remaining three positions corresponding to an x-item $\pi^t(x)$ to such that α_a divides t. We then put a 1 on the the leftmost empty position. If the x-item on the respective position is $\pi^r(x)$, we put 1 on all empty positions s such that α_a divides $(r-s)$, and so on.

Please note that the actual values the parameters α_a, with $1 \leq a \leq 14$, take depend on the pattern $x\pi^i(x)\pi^j(x)\pi^k(x)$, and, more precisely, on i, j, k. Thus, for different patterns we will have different parameters.

$\alpha_1 = \inf\{t : t \nmid i, t \nmid j, t \nmid k, t \nmid \lvert i-j\rvert, t \nmid \lvert i-k\rvert, t \nmid \lvert j-k\rvert\}$	0123
$\alpha_2 = \inf\{t : t \mid i, t \nmid j, t \nmid k, t \nmid \lvert j-k\rvert\}$	0012
$\alpha_3 = \inf\{t : t \nmid i, t \mid j, t \nmid k, t \nmid \lvert i-k\rvert\}$	0102
$\alpha_4 = \inf\{t : t \nmid i, t \nmid j, t \mid \lvert i-k\rvert\}$	0121
$\alpha_5 = \inf\{t : t \nmid i, t \nmid j, t \nmid \lvert i-j\rvert, t \nmid \lvert i-k\rvert, t \mid \lvert j-k\rvert\}$	0122
$\alpha_6 = \inf\{t : t \mid i, t \mid j, t \nmid k\}$	0001
$\alpha_7 = \inf\{t : t \mid i, t \nmid j, t \mid k\}$	0010
$\alpha_8 = \inf\{t : t \nmid i, t \mid j, t \mid k\}$	0100
$\alpha_9 = \inf\{t : t \nmid i, t \mid \lvert i-j\rvert, t \mid \lvert i-k\rvert\}$	0111
$\alpha_{10} = \inf\{t : t \mid i, t \nmid j, t \mid \lvert j-k\rvert\}$	0011
$\alpha_{11} = \inf\{t : t \nmid i, t \mid j, t \mid \lvert i-k\rvert\}$	0101
$\alpha_{12} = \inf\{t : t \nmid i, t \mid k, t \mid \lvert i-j\rvert\}$	0110
$\alpha_{13} = \inf\{t : t \nmid i, t \nmid k, t \mid \lvert i-j\rvert\}$	0112
$\alpha_{14} = \inf\{t : t \nmid i, t \nmid j, t \mid \lvert i-j\rvert\}$	0120

Table 1. Definition of the values α_a, with $1 \leq a \leq 14$.

Recall that $\inf \varnothing = \infty$, so the value of some α_as may be infinite. However, note that the set $\{t : t \nmid i, t \nmid j, t \nmid k, t \nmid \lvert i-j\rvert, t \nmid \lvert i-k\rvert, t \nmid \lvert j-k\rvert\}$ defining α_1 is always non-empty, and also that $\alpha_1 > 3$. Indeed, at least two of i, j, k have the same parity, so α_1 should not divide 2. Similarly, out of $0, i, j, k$ at least two have the same reminder modulo 3, so α_1 should also not divide 3. Let $K = \{\alpha_1, \alpha_2, \ldots, \alpha_{14}\}$.

For a pattern $x\pi^i(x)\pi^j(x)\pi^k(x)$, we say that one of the numbers α_a (and its corresponding representation) models an instance $uf^i(u)f^j(u)f^k(u)$ of the pattern in the case when two of the factors $u, f^i(u), f^j(u), f^k(u)$ are equal if and only if the digits associated to the respective factors in the representation of α_a are equal. An infinite word w over some alphabet Σ avoids a set $S \subseteq K$ if w contains no instance of the pattern $x\pi^i(x)\pi^j(x)\pi^k(x)$ that is modelled by the parameters of S; note that when we discuss about words avoiding a set of parameters, we implicitly assume that the pattern $x\pi^i(x)\pi^j(x)\pi^k(x)$ is fixed.

Before showing our first results, we need several new notations.

Let w_1 and w_2 be the digit representation of some α_ℓ, and α_p respectively, with $\ell, p \geq 1$, we say that w_1 is a swapped form of w_2 if there exists a position

$i \leq 4$ such that $w_1[i] = w_2[i+1]$, and $w_1[i+1] = w_2[i]$, and $w_1[j] = w_2[j]$ for all $j \notin \{i, i+1\}$. For instance, 0012 and 0102 are swapped forms of each others.

Let α be the digit representation of $x\pi^i(x)\pi^j(x)\pi^k(x)$. We say that α has a prefix square if it starts with 00, while the other two digits are 1 and 2; this is the case for $0012 = \alpha_2$. Furthermore, a digit representation has a suffix square if it ends with 22 and the two other digits are 0 and 1; this is the case for $0122 = \alpha_5$. We say that α has a gapped square, if it is 0102, where the 0s form the gapped square, or if it is 0121, where the 1s form the gapped square. We say that α contains a cube if it is 0001 or 0111. We say α has two squares if it is 0011. Finally, α contains gapped cubes if it is 0010 or 0100.

Now based on these relations, we define the following collections of sets. The idea behind all these collections is to generate sets of parameters α_as that cannot be avoided and have a minimal cardinality. No matter what will be added to these sets, they will preserve their unavoidability, while erasing something from them will make them avoidable. To obtain these collections we used a computer program and randomly generated some unavoidable sets of parameters of size five. Using the similarities between the instances modelled by these sets, defined in terms of (gapped) squares and cubes occurring in their digit representation, we developed an algorithm to generate more sets of patterns.

Let $\mathcal{S}_1$ be the collection of sets (each with five elements) that contain α_1 and:

- one of the α_as whose representation has a prefix or a suffix square, but no gapped cube. That is: α_2 or α_5.
- one of the α_as that has a gapped square, but does not have two gapped squares. These are α_3 or α_4.
- one of the α_as that contains cubes or two squares: α_6 or α_9 or α_{10}.
- one of the α_as that contains gapped cubes: α_7 or α_8.

For example, one possible set from $\mathcal{S}_1$ is $\{\alpha_1, \alpha_2, \alpha_4, \alpha_6, \alpha_7\}$. Note that more sets like this one can be constructed using this scheme, and we should consider all of them, but because of lack of the space, we do not list all the examples here.

We also have the restriction that if the representations of the squares and gapped squares of a set from $\mathcal{S}_1$ are not swapped form of each other, then the elements of $\mathcal{S}_1$ representing cubes or gapped cubes should have the same digit on all positions of equal digits from the representations of squares and gapped squares. For example, in the case of 0012 we have that the first and second position contain the same digit and for 0121 we have that the second and the last position contain the same digit, so our gapped cube should be 0010 meaning that the first, second and the last position should contain the same digits.

Let $\mathcal{S}_2$ be the collection of sets (with five elements) that contain α_1 and:

- one of the α_as of the set $\{\alpha_2, \alpha_3, \alpha_4\}$, and
- one of the α_as of the set $\{\alpha_6, \alpha_7, \alpha_9\}$, and
- both α_as that contain a square in the middle of the word (α_{12} and α_{13}).

Moreover, we have the restriction that if we choose α_2 then α_7 should be added to the set. For example, one possible set from $\mathcal{S}_2$ is $\{\alpha_1, \alpha_2, \alpha_7, \alpha_{12}, \alpha_{13}\}$.

Let $\mathcal{S}_3$ be the collection of sets (with five elements) that contain α_1 and α_{10} (the only α_a that has two square factors) as well as:

- one of the α_as whose representation has a prefix or a suffix square, but no gapped cube. That is: α_2 or α_5.
- one of the α_as whose representation has a gapped square, but does not have two gapped squares. That is: α_3 or α_4.
- one of the α_as whose representation contains gapped cubes: α_7 or α_8.

We also have the restriction that if the representation of the squares and gapped squares of a set from $\mathcal{S}_3$ are not swapped form of each other, then its elements representing cubes or gapped cubes should have the same digit on all positions of equal digits from the representations of squares and gapped squares. For example, one possible set from $\mathcal{S}_3$ is $\{\alpha_1, \alpha_2, \alpha_4, \alpha_7, \alpha_{10}\}$.

Let $\mathcal{S}_4$ be the collection of sets that contain α_1, α_2, α_7, and:

- one of the α_as whose representations contain cubes (α_6 or α_9), or two square (α_{10}) and
- one of the α_as for whose representation only the first and last digits are equal, and they are different from all other digits (α_{14}).

One such set is, for example, $\{\alpha_1, \alpha_2, \alpha_7, \alpha_{10}, \alpha_{14}\}$.

Let $\mathcal{S}_5$ be the collection of sets that contain α_1, α_{12}, α_{13}, and α_{14}, as well as one of the α_as whose representation contains cubes (α_6 or α_9). One example is $\{\alpha_1, \alpha_9, \alpha_{12}, \alpha_{13}, \alpha_{14}\}$.

Let $\mathcal{S}_6$ be the collection of sets that contain α_1, α_{10}, α_{13}, α_{14}, and

- one of the α_as whose representation contains a cube (α_6 or α_9), and
- one of the α_as whose representation contains a gapped cube (α_7 or α_8), and
- one of the α_as whose representation contains a gapped square (α_3 or α_4).

Here we have the restriction that if we have in a set of $\mathcal{S}_6$ the element α_7, whose representation has on the first, the second and the last positions the same digit, we should add α_4 whose second and last digit are the same. Furthermore, the presence of both α_4 and α_8 in a set is not permitted. For example, one possible such set is $\{\alpha_1, \alpha_4, \alpha_6, \alpha_7, \alpha_9, \alpha_{10}, \alpha_{13}, \alpha_{14}\}$.

Let $\mathcal{S}_7$ be the collection of sets that contain α_1, α_{12}, α_{13}, and one of the α_as of the set $\{\alpha_2, \alpha_5\}$, and one of the α_as of the set $\{\alpha_3, \alpha_4\}$, and one of the α_as of the set $\{\alpha_7, \alpha_8\}$. Here we have this restriction that the combination of $\{\alpha_2, \alpha_4\}$ and $\{\alpha_2, \alpha_7\}$ is not permitted and if the α_as of a set from $\mathcal{S}_7$ whose representation contain squares and gapped squares are not swapped form of each other, then its elements representing cubes or gapped cubes should have the same digit on all positions of equal digits from the representations of α_as that contain squares and gapped squares. For example, one possible such set $\{\alpha_1, \alpha_3, \alpha_5, \alpha_8, \alpha_{12}, \alpha_{13}\}$.

Let $\mathcal{S}_8$ be the collection of sets (with six elements) that contain α_1 and all elements of the set $\{\alpha_3, \alpha_5, \alpha_7, \alpha_{14}\}$, and one of the α_as of the set $\{\alpha_6, \alpha_9\}$. One example is $\{\alpha_1, \alpha_3, \alpha_5, \alpha_7, \alpha_9, \alpha_{14}\}$.

Let $\mathcal{S}_9$ be the collection of sets (with seven elements) that contain α_1 and , α_{10}:

- one or two elements of the set $\{\alpha_2, \alpha_5\}$, and
- all elements of the set $\{\alpha_7, \alpha_{14}\}$ or $\{\alpha_8, \alpha_{14}\}$ or one or two elements of the set $\{\alpha_{12}, \alpha_{13}, \alpha_{14}\}$,.
- one of the α_as of the set $\{\alpha_3, \alpha_4\}$, and
- one of the α_as of the set $\{\alpha_6, \alpha_9\}$, and
- one of the α_as whose representation has two gapped squares α_{11}.

Here we have the restriction that if two elements of the set $\{\alpha_2, \alpha_5, \alpha_{10}\}$ were selected, one element of the set $\{\alpha_{12}, \alpha_{13}, \alpha_{14}\}$ should also be chosen, and the other way around. Furthermore, if we choose α_2 or α_5 or α_{10} as the α_as with squares in a set, then α_3, α_4, and α_{14} should be selected as gapped squares, respectively, and in the first two conditions, α_{10} and (α_{12} or α_{13}), and in the last condition, (α_3 or α_4) and (α_7 or α_8 or α_{12} or α_{13}) should be added to the set. The other restriction is that, if we have the numbers α_7, α_{14}, then α_3, and if we have the numbers α_4, α_{14}, then (α_8 or α_{12}) should be chosen as the gapped squares in a set. In the end, the union of the sets $\{\alpha_6, \alpha_{10}, \alpha_{13}, \alpha_{14}\}$, and $\{\alpha_{12}, \alpha_{13}\}$ is not allowed, and the following sets: $\{\alpha_1, \alpha_3, \alpha_6, \alpha_8, \alpha_{10}, \alpha_{11}, \alpha_{14}\}$, $\{\alpha_1, \alpha_4, \alpha_5, \alpha_6, \alpha_{10}, \alpha_{12}, \alpha_{14}\}$, $\{\alpha_1, \alpha_4, \alpha_6, \alpha_7, \alpha_{10}, \alpha_{11}, \alpha_{14}\}$ are also exceptions that should not be considered. A set fulfilling all the above is $\{\alpha_1, \alpha_2, \alpha_3, \alpha_6, \alpha_{10}, \alpha_{11}, \alpha_{13}\}$.

Let $\mathcal{S}_{10}$ be the collection of sets (with eight elements) that contain α_1 and (only) one of the following sets:

- α_3 and one of the sets $\{\alpha_5, \alpha_6, \alpha_{10}, \alpha_{11}, \alpha_{13}, \alpha_{14}\}$ or $\{\alpha_5, \alpha_9, \alpha_{10}, \alpha_{11}, \alpha_{13}, \alpha_{14}\}$,
- $\{\alpha_2, \alpha_4, \alpha_{13}\}$ and one of the sets $\{\alpha_6, \alpha_{10}, \alpha_{11}, \alpha_{14}\}$ or $\{\alpha_9, \alpha_{10}, \alpha_{11}, \alpha_{14}\}$.

For example, one possible such set is $\{\alpha_1, \alpha_3, \alpha_5, \alpha_6, \alpha_{10}, \alpha_{11}, \alpha_{13}, \alpha_{14}\}$.

While the choice of these classes may seem, in a sense, arbitrary, we tried to group together in the same class sets of parameters, according to the common combinatorial features of the elements defining them. The way we obtained these sets is by computer exploration. The idea behind the definition is to generate unavoidable sets of parameters α_a which have a minimal cardinality. We basically started with the sets of size 5, and tried to extend them one element at a time in order to obtain unavoidable sets. As such, we ensured that removing some element from them leads to an avoidable set of parameters, while, no matter what element we add to them preserves their unavoidability. The reason to start with sets of 5 parameters is given in the next lemma.

Lemma 1 *Let $K' \subset K$ be any subset of size at most 4 of K. There exists an infinite word w such that w does not contain 4-powers and if w contains an instance of the pattern $x\pi^i(x)\pi^j(x)\pi^k(x)$ then it can not be modelled by any tuples of the set of patterns K'.*

The main result of this section is the following theorem.

Theorem 1. *Given positive integers i, j, k such that $i \neq j \neq k \neq i$, consider the pattern $p = x\pi^i(x)\pi^j(x)\pi^k(x)$. Let $\sigma = \min\{\max(S) \mid S = S_\ell$ for some $\ell = 1, \ldots, 10\}$. Then $\sigma \geq 4$ and p is unavoidable in Σ_m, for all $m > \sigma$.*

Proof. Because $m \geq \alpha_1$, we have that for every word $u \in \Sigma_m^+$ there exists a morphic permutation f such that every two words of $u, f^i(u), f^j(u), f^k(u)$ are different. Indeed, we take f to be a permutation such that the orbit of $u[1]$ is a cycle of length α_1, which means that the first letters of $u, f^i(u), f^j(u)$ and $f^k(u)$ are pairwise different. Similarly, the fact that $m \geq \alpha_2$ (when $\alpha_2 \neq \infty$) means that for every $u \in \Sigma_m^+$ there exists a morphic permutation f such that $f^k(u) \neq u = f^i(u) \neq f^j(u) \neq f^k(u)$. In this case, we take f to be a permutation such that $\mathbf{ord}_f(u[1]) = \alpha_2$. We can derive similar observations for all the α_a parameters involved in the definition of σ.

One can check with the aid of a computer, by a backtracking algorithm, that if $m \geq \max(S) + 1$, when $S = S_\ell$ for some $\ell = 1, \ldots, 10$, then p is unavoidable in Σ_m. Our computer program tries to construct a word as long as possible by always adding a letter to the current word (obtained by backtracking). This letter is chosen in all possible ways from the letters contained in the word already, or it may also be a new letter, and we just check whether it creates an instance of the pattern as a suffix of the word. Generally, we were not able to check if an arbitrary instance of the pattern is created, due to the complexity of checking all permutations as possible image of π. But, in most of the cases we need to check, we got the result even when we explicitly allowed π to be only a cycle. In the remaining cases, we needed to allow π to act as the identity on a symbol, and as a cycle on the rest of the alphabet. This latter case, which was still easy to check, is the reason why we got that p is only avoidable over alphabets of size at least $\sigma + 1$ and not already over an alphabet of size σ.

For instance, looking at $\mathcal{S}_1$, using a computer program that explores all the possibilities by backtracking we obtained that if $m \geq \max\{\alpha_1, \alpha_2, \alpha_4, \alpha_6, \alpha_7\}$, which is at its turn greater or equal to 4 as $\alpha_1 > 3$, the longest word that does not contain an instance of this pattern, even when constraining π to be a cycle, has length 36, and it is 010210210210033001133001133001133000 (adding new letters to this word does not lead to a longer one). On the other hand, we found arbitrarily long words that contain instances of the pattern modelled by $\alpha_1, \alpha_2, \alpha_3, \alpha_6, \alpha_{10}, \alpha_{11}, \alpha_{12}$ when we allow π to be replaced only by cycles. However, if we allow π to be more general (i.e., only fix one symbol of the alphabet and be a cycle on the rest), we obtain that there are no infinite words that avoid the pattern in this case. So, over alphabets of size $m \geq \max\{\alpha_1, \alpha_2, \alpha_3, \alpha_6, \alpha_{10}, \alpha_{11}, \alpha_{12}\} + 1$ the pattern is unavoidable.

The longest words that do not contain their instances, as well as words for all the other cases, can be easily found by backtracking.

Note that by the results (Theorem 4 and Theorem 6) of [2] it also follows that $\sigma + 1 \geq 5$. □

4 Algorithm to generate avoidable cases

In Lemma 1, we proved that given the pattern $x\pi^i(x)\pi^j(x)\pi^k(x)$, for each i, j, and k, we can compute an upper bound on the minimum size of an alphabet over which the pattern is unavoidable. Now to show that this is the minimum

cardinality over which the pattern of size four is unavoidable, we proceed as follows.

Let K_a be the class that contains all nonempty sets of α_a parameters S' such that S' does not include any set $S = S_\ell$ for some $\ell = 1, \ldots, 10$. In other words, K_a contains all nonempty strict subsets of the sets $S = S_\ell$ for some $\ell = 1, \ldots, 10$ as well as any other sets of parameters that do not include any of the sets $S = S_\ell$ for some $\ell = 1, \ldots, 10$. We already know that all subsets of the sets $S = S_\ell$ for some $\ell = 1, \ldots, 10$ are avoidable. Also, all supersets of the sets $S = S_\ell$ for some $\ell = 1, \ldots, 10$ are unavoidable (as the sets S already are unavoidable), so we try to show that all the other sets of parameters are avoidable. However, K_a has about 1400 sets of patterns, so checking each of them is hard to be done by pen and paper.

Fortunately, there is an observation we can exploit at this point: all subsets of an avoidable set of parameters is avoidable as well. For instance, if the set $\{\alpha_1, \alpha_2, \alpha_5, \alpha_6, \alpha_8, \alpha_{14}\}$ can be avoided by a word $\mathbf{w}$, then the set $\{\alpha_1, \alpha_2, \alpha_5\}$ can also be avoided by $\mathbf{w}$. Thus, we can look for the sets of parameters with maximal cardinality that belong to K_a and are avoidable. Clearly, the entire K is unavoidable. However, $K \setminus \{\alpha_1\}$ can be shown to be avoidable. Our approach is implemented in the following algorithmic scheme.

Algorithm 1 Algorithm to generate avoidable cases

1: Let $n = 10$. Using the sets $\mathcal{S}_i, (1 \leq i \leq 10)$, generate all sets of α_as of cardinality n, that have no unavoidable sets of patterns as subset; show that they are avoidable;
2: For all n from 9 down to 4, generate all sets of cardinality n that have no unavoidable sets of patterns as subset; these sets should not be subsets of the avoidable sets of α_as of cardinality $n + 1$ (to avoid generating repetitive avoidable sets of cases generated in the past step); show that they are avoidable.

The following theorem states which sets of α_as can be avoided, according to the algorithm above, concluding thus our approach. It is worth noting that the search space was drastically reduced by our aproach.

Theorem 2. *For each of the following sets there exists an infinite word over an alphabet of size at most* 5*, such that if this word contains an instance of*

$x\pi^i(x)\pi^j(x)\pi^k(x)$ then this instance can not be modelled by an element of the set.

$\{\alpha_2,\alpha_3,\alpha_4,\alpha_5,\alpha_6,\alpha_7,\alpha_8,\alpha_9,\alpha_{10},\alpha_{11},\alpha_{12},\alpha_{13},\alpha_{14}\}$,
$\{\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5,\alpha_6,\alpha_9,\alpha_{10},\alpha_{11},\alpha_{14}\}$, $\{\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5,\alpha_6,\alpha_9,\alpha_{10},\alpha_{12},\alpha_{14}\}$,
$\{\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5,\alpha_6,\alpha_9,\alpha_{10},\alpha_{13},\alpha_{14}\}$, $\{\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5,\alpha_6,\alpha_9,\alpha_{11},\alpha_{12},\alpha_{14}\}$,
$\{\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5,\alpha_6,\alpha_9,\alpha_{11},\alpha_{13},\alpha_{14}\}$, $\{\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5,\alpha_7,\alpha_8,\alpha_{11},\alpha_{12},\alpha_{14}\}$,
$\{\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5,\alpha_7,\alpha_8,\alpha_{11},\alpha_{13},\alpha_{14}\}$, $\{\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5,\alpha_{10},\alpha_{11},\alpha_{12},\alpha_{13},\alpha_{14}\}$,
$\{\alpha_1,\alpha_2,\alpha_4,\alpha_6,\alpha_8,\alpha_9,\alpha_{10},\alpha_{12}\}$, $\{\alpha_1,\alpha_2,\alpha_4,\alpha_6,\alpha_9,\alpha_{10},\alpha_{11},\alpha_{12}\}$,
$\{\alpha_1,\alpha_2,\alpha_4,\alpha_6,\alpha_8,\alpha_9,\alpha_{11},\alpha_{12},\alpha_{14}\}$, $\{\alpha_1,\alpha_2,\alpha_4,\alpha_8,\alpha_{10},\alpha_{11},\alpha_{12},\alpha_{13}\}$,
$\{\alpha_1,\alpha_2,\alpha_4,\alpha_8,\alpha_{10},\alpha_{12},\alpha_{13},\alpha_{14}\}$, $\{\alpha_1,\alpha_2,\alpha_4,\alpha_8,\alpha_{11},\alpha_{12},\alpha_{13},\alpha_{14}\}$,
$\{\alpha_1,\alpha_2,\alpha_4,\alpha_6,\alpha_8,\alpha_9,\alpha_{10},\alpha_{11},\alpha_{13},\alpha_{14}\}$, $\{\alpha_1,\alpha_2,\alpha_5,\alpha_6,\alpha_7,\alpha_8,\alpha_9,\alpha_{10},\alpha_{11},\alpha_{12},\alpha_{13},\alpha_{14}\}$
$\{\alpha_1,\alpha_3,\alpha_4,\alpha_6,\alpha_7,\alpha_8,\alpha_9,\alpha_{10},\alpha_{11},\alpha_{12},\alpha_{14}\}$, $\{\alpha_1,\alpha_3,\alpha_4,\alpha_6,\alpha_7,\alpha_8,\alpha_9,\alpha_{10},\alpha_{11},\alpha_{13}\}$,
$\{\alpha_1,\alpha_3,\alpha_4,\alpha_6,\alpha_7,\alpha_8,\alpha_9,\alpha_{11},\alpha_{13},\alpha_{14}\}$, $\{\alpha_1,\alpha_2,\alpha_4,\alpha_6,\alpha_8,\alpha_9,\alpha_{10},\alpha_{11},\alpha_{13}\}$,
$\{\alpha_1,\alpha_3,\alpha_4,\alpha_6,\alpha_9,\alpha_{10},\alpha_{11},\alpha_{13},\alpha_{14}\}$, $\{\alpha_1,\alpha_3,\alpha_4,\alpha_7,\alpha_8,\alpha_{10},\alpha_{11},\alpha_{12},\alpha_{13}\}$,
$\{\alpha_1,\alpha_3,\alpha_4,\alpha_7,\alpha_8,\alpha_{10},\alpha_{11},\alpha_{13},\alpha_{14}\}$, $\{\alpha_1,\alpha_3,\alpha_4,\alpha_7,\alpha_8,\alpha_{11},\alpha_{12},\alpha_{13},\alpha_{14}\}$,
$\{\alpha_1,\alpha_3,\alpha_4,\alpha_7,\alpha_8,\alpha_{10},\alpha_{12},\alpha_{13},\alpha_{14}\}\{\alpha_1,\alpha_3,\alpha_5,\alpha_6,\alpha_7,\alpha_9,\alpha_{10},\alpha_{12}\}$,
$\{\alpha_1,\alpha_3,\alpha_5,\alpha_6,\alpha_9,\alpha_{10},\alpha_{11},\alpha_{12}\}$, $\{\alpha_1,\alpha_3,\alpha_5,\alpha_7,\alpha_{10},\alpha_{11},\alpha_{12},\alpha_{13}\}$,
$\{\alpha_1,\alpha_3,\alpha_5,\alpha_7,\alpha_{10},\alpha_{12},\alpha_{13},\alpha_{14}\}$, $\{\alpha_1,\alpha_3,\alpha_5,\alpha_7,\alpha_{11},\alpha_{12},\alpha_{13},\alpha_{14}\}$,
$\{\alpha_1,\alpha_3,\alpha_5,\alpha_6,\alpha_7,\alpha_9,\alpha_{10},\alpha_{11},\alpha_{13},\alpha_{14}\}$, $\{\alpha_1,\alpha_3,\alpha_5,\alpha_6,\alpha_7,\alpha_9,\alpha_{11},\alpha_{12},\alpha_{14}\}$
$\{\alpha_1,\alpha_3,\alpha_7,\alpha_{10},\alpha_{11},\alpha_{12},\alpha_{13},\alpha_{14}\}$, $\{\alpha_1,\alpha_4,\alpha_6,\alpha_8,\alpha_9,\alpha_{10},\alpha_{12},\alpha_{14}\}$,
$\{\alpha_1,\alpha_4,\alpha_6,\alpha_8,\alpha_9,\alpha_{10},\alpha_{13},\alpha_{14}\}$, $\{\alpha_1,\alpha_4,\alpha_8,\alpha_{10},\alpha_{11},\alpha_{12},\alpha_{13},\alpha_{14}\}$

Proof. We only show the statement for the set $T = \{\alpha_2,\alpha_3,\alpha_4,\alpha_5,\alpha_6,\alpha_7,\alpha_8,\alpha_9,\alpha_{10},\alpha_{11},\alpha_{12},\alpha_{13},\alpha_{14}\}$. The other cases can be proved in a similar fashion.

Let $h_\alpha = \alpha(h)$, where $\alpha : \Sigma_3^* \to \Sigma_5^*$ is the morphism defined by
$0 \to 0123041203410234$, $1 \to 0132403124302134$, $2 \to 0123402134201324$.

We show that if h_α contains an instance of the pattern $x\pi^i(x)\pi^j(x)\pi^k(x)$ then this instance can not be modelled by any tuple of the set T. Assume, for the sake of contradiction, that h_α contains a factor of the form $uf^i(u)f^j(u)f^k(u)$ which can be modelled by any of the $\alpha_a \in T$ (with f morphic permutation). The maximum length of a factor of h_α that does not contain a full image of any letter of the ternary Thue word under α is 30. Using a computer program we checked that h_α has no factor of the form $uf^i(u)f^j(u)f^k(u)$ with $|u| < 31$ which can be modelled by any of the $\alpha_a \in T$. Further, if u is a word of length ≥ 31, each of the factors $u, f^i(u), f^j(u), f^k(u)$ contains a full image of a letter of h. If all these factors contain only the image of 1 then we get a contradiction, as it would mean that h contains a square (either 11 or a longer square whose image covers $uf^i(u)f^j(u)f^k(u)$). If one of them contains the image of 0 or 2 we proceed as follows. Note that the letters 0 in the image of 0 under α and the letters 2 in the image of 2 occur repeatedly four times, with 3 symbols between them. So, in one of $u, f^i(u), f^j(u), f^k(u)$, we will have either the image of 0 under α, or the image of 2 under α, and, consequently, four occurrences of 0, with

3 other symbols between two consecutive 0s, or, respectively, four occurrences of 2, with three other symbols between two consecutive 2s. Consequently, the four occurrences of 0 or 2 should be aligned to four occurrences of another symbol, when considering the alignment of the factors $u, f^i(u), f^j(u), f^k(u)$. Thus, if at least one of the f^i, f^j, or f^k is not the identity, in $uf^i(u)f^j(u)f^k(u)$, based on the repetition of the letters 0 in the image of 0, we should have one of the following alignments: 0123041203410234 aligned with 0412034102340132 (a contradiction, because this would mean that there is a function mapping 3 to both 2 and 3); 01230412034102340 aligned with 04120341023401234 (a contradiction, because this would mean that there is a function mapping 0 to both 0 and 4); 0123041203410234 aligned with 3240312430213401 (a contradiction, because this would mean that there is a function mapping 1 to both 1 and 4); 0123041203410234012 aligned with 2340213420132401230 (a contradiction, because this would mean that there is a function mapping 0 to both 2 and 3); 012304120341023401324031243 02 aligned with 234021342013240123041203410 23 (a contradiction, because this would mean that there is a function mapping 3 to both 1 and 2); 01230412034102340 aligned with 23402134201324013 (a contradiction, because this would mean that there is a function mapping 3 to both 0 and 1); 01230412034102340 aligned with 21342013240123041 (a contradiction, because this would mean that there is a function mapping 4 to both 0 and 4). We can apply the same reasoning for the alignments based on the repetition of the letters 2 in the image of 2, and get again only contradictions. Therefore, no instance of the pattern is contained in h_α. This concludes our proof. □

We can now show the main theorem of this paper:

Theorem 3. *Given a pattern $p = x\pi^i(x)\pi^j(x)\pi^k(x)$ we can determine effectively the value σ, such that p is avoidable in Σ_m for $m \leq \sigma - 1$ and unavoidable in Σ_m for $m \geq \sigma + 1$.*

Proof. By [1,2], we get that all the unary patterns of size 4 with permutations are avoidable in Σ_m for $m \in \{2, 3, 4\}$. If $i = j$ or $j = k$ then all the instances of the pattern contain squares, so the pattern is avoidable in Σ_m for all $m \geq 3$. If $i = k$, then the pattern is avoidable in Σ_m, for all $m \geq 3$, according to the results of [4], where it is shown that $\pi^i(x)\pi^j(x)\pi^i(x)$ is avoidable in such alphabets.

Let us thus assume that $i \neq j$, $i \neq k$, and $j \neq k$ (which is also the setting of Theorem 1). We compute the parameters α_a, with $1 \leq a \leq 14$, for the given pattern. Then, we consider the sets S_i, with $1 \leq i \leq 10$, and compute $\sigma = \min\{\max(S) \mid S = S_\ell \text{ for some } \ell = 1, \ldots, 10\}$. By Theorem 1 we get that $x\pi^i(x)\pi^j(x)\pi^k(x)$ is unavoidable in Σ_m, for $m \geq \sigma + 1$. Let now $S' = S_\ell$ for some $\ell = 1, \ldots, 10$ be a set such that $\max(S') = \sigma$. Assume that there exists $\ell \geq 5$ such that $x\pi^i(x)\pi^j(x)\pi^k(x)$ is unavoidable in Σ_ℓ and $\ell < \sigma$. Let A_0 be the set containing all α_a parameters which are at most ℓ, or, in other words, let A_0 be the maximal subset (with respect to inclusion) of K such that if $\alpha \in A_0$ then $\alpha \leq \ell$. Clearly, A_0 is either a strict subset of a set $S = S_\ell$ for some $\ell = 1, \ldots, 10$ or A_0 is incomparable to any of the sets of S. It cannot include any set $S'' = S_\ell$ for some $\ell = 1, \ldots, 10$ as then $\max(S'') < \max(S') = \min\{\max(S) \mid S = S_\ell \text{ for}$

some $\ell = 1, \ldots, 10\}$, a contradiction. Thus A_0 is included in one of the sets from the statement of Theorem 2. Consequently, there exists an infinite word w over a five letter alphabet that avoids A_0. In fact, w avoids A_0 over all alphabets Σ_m such that the instances of p over Σ_m correspond only to α_as contained in A_0. This means that w avoids A_0 in Σ_m with $5 \leq m \leq \ell$. So, p is avoidable in Σ_ℓ, a contradiction.

In conclusion, the pattern $p = x\pi^i(x)\pi^j(x)\pi^k(x)$ is avoidable in Σ_m when $2 \leq m < \sigma$. This concludes our proof. □

We get the next corollary, by taking, in the setting of the previous theorem, $\delta = \sigma$, if $x\pi^i(x)\pi^j(x)\pi^k(x)$ is avoidable in Σ_σ, or $\delta = \sigma - 1$, otherwise.

Corollary 1. *Given a pattern $p = x\pi^i(x)\pi^j(x)\pi^k(x)$ there exists δ a natural number or $+\infty$, such that p is avoidable in Σ_m for $m \in \{2, 3, \ldots, \delta\}$ and unavoidable in $\mathbb{N} \setminus \{2, 3, \ldots, \delta\}$.*

5 Conclusions

We have shown how to compute, given a pattern $x\pi^i(x)\pi^j(x)\pi^k(x)$, a rather precise approximation of the size of the alphabets where this pattern is avoidable. More importantly, we show that the sizes of these alphabets form an interval of integers. Our results extend the results of [4] and [2]. The method we used is to explore the number theoretic connections between i, j, and k, in relation to the possible orders the permutation π may have. This approach follows the initial ideas of [4], but requires a much more careful and deeper analysis. Essentially, while the relations between i and j in a cubic pattern $x\pi^i(x)\pi^j(x)$ can be modelled with four parameters only, in the case of $x\pi^i(x)\pi^j(x)\pi^k(x)$ we have 14 such parameters. Exhaustively analysing all the possible relations between these parameters, as it was done in [4], would take too long, so we devised a less complex way of exploring them. We basically see, on the one hand, which minimal combinations (in the sense of cardinality) of such parameters lead to the conclusion that the pattern is unavoidable in alphabets of large enough size, while also looking for the maximal combinations of the parameters that lead to the conclusion that the pattern is avoidable in alphabets of small enough size. This approach produced a rather large, but still tractable, case analysis.

In order to extend our results to arbitrarily long unary patterns with permutations, we expect that a valid approach would still be based on defining similar sets of parameters and exploring their combinatorial properties. However, it is to be expected that a direct generalization of the ideas above would lead to a number of parameters which grows exponentially with the length of the pattern, hence to a way too complex exploration in the end.

References

1. Currie, J.D., Manea, F., Nowotka, D.: Unary patterns with permutations. In: Proc. DLT 2015. LNCS, vol. 9168, pp. 191–202. Springer (2015)
2. Currie, J.D., Manea, F., Nowotka, D., Reshadi, K.: Unary patterns with permutations. to appear in Theor. Comput. Sci. (2018), `https://media.informatik.uni-kiel.de/zs/unary-patterns-permutations.pdf`
3. Lothaire, M.: Combinatorics on Words. Cambridge University Press (1997)
4. Manea, F., Müller, M., Nowotka, D.: Cubic patterns with permutations. J. Comput. Syst. Sci. 81(7), 1298–1310 (2015)
5. Thue, A.: Über unendliche Zeichenreihen. Norske Vid. Skrifter I. Mat.-Nat. Kl., Christiania 7, 1–22 (1906)
6. Thue, A.: Über die gegenseitige Lage gleicher Teile gewisser Zeichenreihen. Norske Vid. Skrifter I. Mat.-Nat. Kl., Christiania 1, 1–67 (1912)

Gait-Based Authentication Using MEMS Sensors in Smartphones

Steven Wessels and Dustin van der Haar

Academy of Computer Science and Software Engineering, University of Johannesburg, Gauteng, South Africa
215029263@student.uj.ac.za, dvanderhaar@uj.ac.za

Abstract. The technology found in smart devices continues to improve and become increasingly complex. Sensors that are commonly found in such devices can be used for more than merely enriching UX as they can also be used measure the user's unique behaviour. By utilizing these sensors to measure the user's characteristic locomotion, we can begin to create a basis for a non-obtrusive, continuous authentication system. The proposed system is a gait authentication system that uses the accelerometer and gyroscope in a user's device for measurement. The system is tested on a data set consisting of gait sequences collected in realistic settings. A novel algorithm for locating the optimal threshold value for peak detection is presented. The system is ultimately capable of performing its goal. The EER achieved by the accelerometer was 7.04% while the gyroscope performed poorly in comparison with an EER of 14.71%.

Keywords: Biometrics · Gait · Signal Processing · Machine Learning · Mobile Data · Microelectromechanical Sensors · Security · Data applications

1 Introduction

Modern smart devices come equipped with a plethora of sensors that are used to assist in providing an immersive user experience, whether it be a gyroscope giving the ability to simulate a steering wheel in a racing game or the GPS sensor helping a navigational app. Additionally, these sensors can also be used to monitor the user's behavioural characteristics. By collating information surrounding usage behaviour, and identifying patterns within the behavioural data, we can create a biometric model for the user that can be employed to identify and authenticate that user. A biometric that can be revealed by leveraging a smart device's gyroscope and accelerometer data is that of a person's gait. Using gait as an authentication mechanism is far less established in comparison to fingerprint or facial biometrics, and the advantages and disadvantages of using gait as biometric will be expounded on later in the paper. However, by utilizing gait as a means of authentication in conjunction with other traditional authentication methods, there is the possibility of building authentication systems that are completely unobtrusive and more robust against malicious intent. The purpose of

this paper is to ascertain if a gait-based authentication system that leverages the microelectromechanical gyroscope and accelerometer sensors on a smartphone is a feasible authentication solution.

The remainder of the paper will be structured as follows: Section 2 will include a brief discussion on the topics of gait and biometrics in general. A literature review of previous related work will also be included. Section 3 will provide the details of the proposed system and its constituent components. Section 4 explains the methodology used by the proposed system and how will be evaluated. Section 5 discusses the results found by the system. Section 6 concludes the paper by discussing the future direction of this research.

2 Problem Background

2.1 An Overview of Authentication and Biometrics

Authentication is a one-to-one process that allows for the verification of a person's identity. Authentication can be broadly categorized into three classes:

- Knowledge-based: What a person knows - e.g. Passwords, PIN numbers, usernames.
- Token-based: What a person has - e.g. ID cards, security tokens.
- Inherence-based: What the person is or does - Fingerprint, facial features, behavioural patterns.

A critical flaw of knowledge and token-based authentication is that they cannot guarantee non-repudiation. Passwords can be forgotten by its user or recovered by malicious parties, and security tokens can be stolen, but something that is inherently unique about a person is difficult to replicate. Biometrics is the scientific field concerned with the forming of metrics pertaining to human characteristics. The prevalence of biometric systems in modern society is due to the unique and often permanent nature of human characteristics, and how they can be measured to implement large-scale identity management systems [1]. Biometrics can be divided into two categories: Physiological and Behavioural. A physiological biometric is measured directly from the human body. Common examples of physiological biometrics are fingerprints, hand geometry, iris scans, etc. A physiological biometric system requires explicit user interaction and requires special dedicated hardware to collect data [6]. However, behavioural biometrics, which are the identifiable and measurable patterns in human activity, are considerably more transparent and unobtrusive to measure as all data collection is done implicitly, sometimes without the person even being aware of it taking place [7]. Behavioural biometric systems cannot yet match the level of performance that physiological biometric systems can achieve, and currently large-scale deployments of behavioural biometric systems are yet to become universal.

2.2 Gait as a Biometric

There are advantages and disadvantages to using any biometric, and the choice of what biometric is best suited for an application is dependent on the context of that application. Jain et al. identified seven criterion which can be used to assess how suitable a biometric might be in any application (1999): Universality, Uniqueness, Permanence, Measurability, Performance, Acceptability and Circumvention [8].

Boyd and Little defined gait as the coordinated and cyclic combination of movements that result in human locomotion (2005) [4]. The uniqueness of a person's gait lies in the combination of its cyclicity and coordination. An advantage of using gait as a biometric is that it can be measured unobtrusively and without a person having to alter their natural behaviour [5]. Additionally, circumvention of the gait biometric is challenging because imitating a person's gait is very difficult. Regarding permanence and reliability, gait is not the best biometric as a person's gait may be affected by a variety of factors [3][4]:

- Biological factors: Weight, limb length, posture.
- External factors: Footwear, load bearing, ground surface characteristics.
- Transient factors: Emotional state, physical state.

The measurement of gait can also be problematic, leading to the implementation of authentication systems that would be too complex for real-world applications. Some of these problems will be highlighted in the literature review to follow.

2.3 Previous Work

Possibly the earliest research conducted in gait recognition via computer vision was done by Niyogi and Adelson in 1994. The authors concluded that a human's characteristic motion could be recognised and tracked by analysing spatiotemporal images [13]. In 2005, Boyd and Little conducted research that evaluated the validity of using gait as a biometric and suggested that although much progress had been made in the decade prior, still more research was required in uncontrolled environments before gait could become a widely used biometric [4]. More recently, Lee, Tan and Lim published a comprehensive review on vision-based gait recognition. They showed that there were two main categories of gait feature extraction: model-based and model-free. Model-based approaches tended to be less sensitive to view and scale variations but are less accurate at locating joint positions. Although there had been significant advances into gait recognition, many challenges still exist into its usability in real world scenarios [14]. While much work has been done concerning the measuring of gait using machine vision and floor sensors, the rest of this literature review will be devoted to discussing research into using wearable or mobile sensors to measure gait. One of the pioneering works of using wearable sensors to measure gait was published in 2005. Mantyjarvi et al. developed a biometric system for users of portable devices that

used pattern recognition algorithms on accelerometer data to identify subjects. Test data was collected at three different walking speeds from each subject. The best EER (Equal Error Rate) achieved was 7%. In 2007, Gafurov, Snekkenes and Bours designed a gait authentication and identification system using a wearable accelerometer [9]. Each subject had the sensor placed in their pocket and was instructed to walk for approximately 20 meters. The data collected was evaluated using four different methods, absolute distance, correlation, histograms, and higher order moments. The best performing method was with the absolute distance metric which yielded an EER of 7.3%. Additionally, the system's identification feature was tested with the subjects wearing a 4kg backpack, to see how the measurable gait of the subject was affected. The EER rose to 9.3%, however, the authors concluded that this was not a significant enough drop in performance to cause concern. A few years later, Derawi, Bours and Holien proposed a similar wearable accelerometer gait authentication system that aimed to improve gait-cycle detection with a simplified approach (2010) [11]. This simplified approach included pre-processing, cycle detection and recognition analysis being applied to the sensors signal. Ultimately, the authors managed to achieve an EER of 5.7% and develop an algorithm that was more rich and stable than any that had come before it. A long-time problem for gait measurement from wearable sensors is the location of those sensors. In an effort in minimising the sensitivity of gait authentication systems to sensor placement, Zhong and Dheng published an article where they proposed a solution to address the variance in results that sensor placement causes (2014). Their solution included the use of gyroscope readings in addition to accelerometer readings to derive relationships between the two sensors so that the effect of sensor rotation would be irrelevant to the feature computation. After the I-vector extraction method had been applied an EER of 5.6% was achieved.

3 Exprimental Setup

3.1 Sensors

The device used to collect data is a Nokia 3 which runs an Android operating system (OS). All Android Nokia's are equipped with a Bosch BMI160, a small low power inertial measurement unit that houses a gyroscope and an accelerometer. Accelerometers are electromagnetic devices that measure non-gravitational linear acceleration. Gyroscopes are devices that measure angular velocity. The BMI160 outputs accelerometer triaxial data at a rate of between 12.5 - 1600 Hz depending on the Android OS scheduler, with an output data accuracy magnitude of error of $\pm 1\%$ [15]. The gyroscope data is output at a rate of between 25 - 3200 Hz also with output data accuracy magnitude of error of of $\pm 1\%$.

3.2 Data Collection

A primary data set was created across multiple days in various places to simulate real world conditions. The walking environments in which data collection took

place included a flat corridor, a tiled corridor with stairs, and a carpeted room. The data set consisted of 20 total subjects aged between 18 and 69 years of age. The smart device is placed in the subject's pocket during data collection, and the subjects are asked to walk in a natural manner around their environment. The data collection period is timed so that a data sample of approximately 30 seconds is collected. Two samples were acquired from each subject. All data collection was done via the Android application, SensorLab. During each subject's test, the application records data from all the devices sensors and outputs the recorded collection session in .csv (comma separated values) files located in the same directory for convenience.

3.3 Python Mathamatical and Machine Learning Libraries

SciPy is an open source library for Python development that contains modules indented for applications in mathematics, statistics, data science, and engineering. The proposed model utilized the interpolation tools for a cubic spline, signal processing tools for a lowpass filter, and the statistics module to calculate the time-domain features of gait cycles.

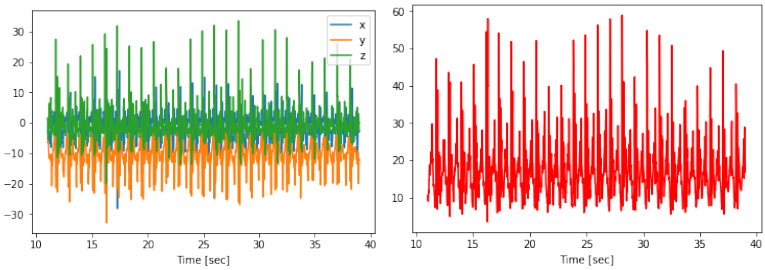
NumPy is a comprehensive mathematical library for Python that, like SciPy, is open source. Arguably, its largest contribution to Python development is the NumPy array data structure [19] which was used throughout the implementation of the proposed model. Additionally, many other of its submodules constituted the proposed model, including the fast Fourier transform module that was used to calculate frequency-domain features.

SciKitLearn is a Python library that focuses on machine learning applications and was designed to be interoperable with NumPy and SciPy [20]. The Support Vector Machine (SVM) classifier it provides was used to perform the classification task in the proposed model. Additionally, SciKitLearn contains a metrics module that can be used to evaluate the results of the classification task.

4 Model

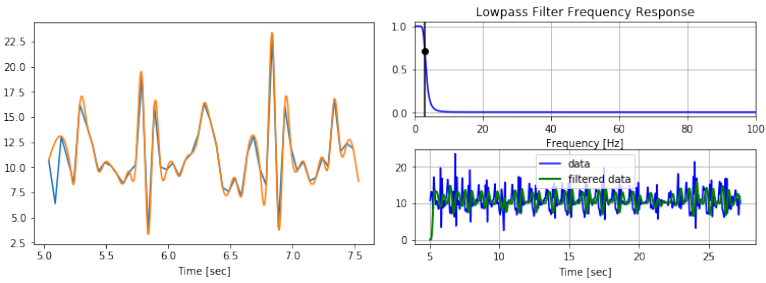
4.1 Pre-processing

The purpose of pre-processing a data set is to negate the effect of incomplete or inconsistent data. By examining the data output from the accelerometer, unnecessary data collected during the time the subject is positioning the cell phone can be observed. This data is not required for determining the subjects gait patterns and must be cleaned before any pattern recognition algorithms can be applied. Due to the highly variant environmental conditions under which data collection occurred, the pre-processing of for the proposed system had to be sophisticated and robust. The following section will outline the pre-processing steps in detail.



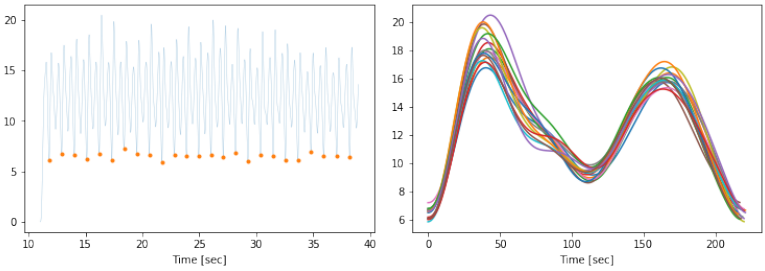
(a) (b)

Fig. 1: Graphical representation of example data output from the device's accelerometer a) represents raw accelerometer data from the sensor b) The signal with the first and last 50 milliseconds removed, and magnitude calculated and plotted.



(a) (b)

Fig. 2: a) Cubic spline applied to the magnitude of raw triaxial signals b) The Butterworth lowpass filter applied to previously splined data



(a) (b)

Fig. 3: a) Detected peaks. b) The overlapped gait cycles traces of segmented data

Magnitude Calculation The triaxial signals recorded from the device are highly sensitive to the orientation and positioning of its sensors. The orientation

of the sensors may change during data collection. To eliminate the orientation sensitivity, the magnitude of the triaxial signal is computed using the following formula [16]:

$$a_{mag} = \sqrt{a_x^2 + a_x^2 + a_x^2} \tag{1}$$

Where a_x, a_y, and a_z are the linear accelerations measured along the x, y, and z axes respectively.

Cubic Spline Interpolation Due to the sensor collection app running on top of an Android OS, data collection will not occur at a fixed sample rate. The irregular sample rate occurs because the OS must give other processes time to run on the phone's CPU. The signal data must be resampled and interpolated so that a fixed sample rate can be used. It is important to set a high enough resampling frequency so that the dynamics of a gait cycle may be captured with sufficient detail to promote accurate feature extraction [17]. The magnitude of the signal data was interpolated using a cubic spline and resampled at 50Hz (200 samples per second).

Butterworth Lowpass Filter A common problem when collecting inertial sensor signal data is that the sensors are sensitive and collect "noisy" data. Noisy data manifests itself in the form of abnormally high and sharp peaks. To remove noisy data, a Butterworth low pass filter was applied with a cutoff threshold of $3Hz$.

Optimal Threshold Calculation and Gait Cycle Segmentation To extract features for every gait cycle, the signal must be segmented into individual cycles. The first step to doing this is to locate either the minimum or maximum peaks; this model uses the minimum peaks. Most peak detection algorithms require a parameter that specifies the minimum distance between peaks to be identified, and a threshold value under which each peak value should fall (if minimums are being detected). The threshold cannot be a set to a static value, as there are many variances between gait features, such as average peak height, between subjects. The algorithm used to calculate this threshold value finds the smallest value that minimizes the standard deviation between the peaks and is presented in Algorithm 1.

4.2 Selection

Gait Cycle Average Trace After segmentation, the traces were calculated and stored in a data structure. Traces whose time was either too long or too short to be considered a realistic gait cycle where thrown out to further clean the data set. The remaining traces would individually each have features extracted from them. The feature extraction process is explained in the following section of the paper.

Algorithm 1 Optimal Threshold Calculation

Input: Time series gait data $s(t)$, Range of allowed threshold, τ_{min} and τ_{max}
Output: Optimal peak threshold τ_{opt}

```
σ_close = 1000
τ_opt = 0.65
for τ between τ_min, τ_max do
    index = peakIndexes(s(t), τ, d_min)
    for i in index do
        δ_index = index[i + 1] − index[i]
    end for
    normalize(δ_index)
    σ_test = stdDeviation(δ_index)
    if σ_test < σ_close then
        σ_test = σ_close
        τ_opt = τ
    end if
end for
return τ_opt
```

4.3 Feature Extraction

Feature extraction provides a set of values with which classification tasks can be performed. Extracting features that as a set will provide a holistic description of a gait cycle is indispensable to efficiently and accurately recognize subjects. The features extracted from the gait cycle traces can be used to distinguish between different users thus providing a basis for gait authentication. The feature set for each gait cycle contains eight time-domain based features, which are computationally inexpensive, and two frequency-domain based, which has a higher computational cost due to the calculations of Fourier transforms. Table 1 outlines the features extracted from each gait cycle.

4.4 Enrolment

A feature vector containing the extracted features is generated for each gait cycle trace. At least five feature vectors must be generated from each sensor data for a subject to qualify for enrolment. These feature vectors generated from accelerometer and gyroscopic data are stored in separate .csv files and marked with an identification number for the enrolled subject. The identification number will act as a class label for the supervised learning task to be performed. The .csv files containing enrolled data will be used as training data sets for the classification algorithm.

4.5 Classification

Support Vector Machine Classification Scheme Support Vector Machines are a supervised, binary, machine learning model that can be used for classification. Initially, a feature space, which is an n-dimensional vector space created

Table 1: Features extracted per gait cycle trace for user authentication

Feature	Description	Formula	Domain
Maximum Amplitude	The largest value found in the signal data	$s_{max} = max\{s(t)\}$	Time
Minimum Amplitude	The smallest value found in the signal data	$s_{min} = min\{s(t)\}$	Time
Mean	The average value of the signal data	$\overline{x} = \frac{1}{N}\sum s(t)$	Time
Variance	The expected square deviance of a signal value from its mean	$\sigma^2 = \frac{1}{N}\sum(s(t) - \overline{x})^2$	Time
Kurtosis	The measure of the signal curves' tailedness	$K = \frac{m_4}{\sigma^2}$	Time
Skewness	The asymmetry of a distribution. Pearson's definition was used	$S = \frac{m_3}{\sqrt{m_2^3}}$	Time
Peak-To-Peak Time	Time to complete an entire gait cycle	$t = t_i - t_{i-1}$	Time
Absolute Latency to Amplitude Ratio	The absolute value of signal latency over maximum signal value	$ALAR = \lvert\frac{t_{s_{max}}}{s_{max}}\rvert$	Time
Energy	The sum of the squared discrete fast Fourier transforms component magnitudes	$E_f = \frac{1}{K}\sum_{k=1}^{K} A(k)^2$	Frequency
Entropy	The periodicity of the acceleration signal	$S = -\sum_{k=1}^{K} p(k)\log_2(p(k))$	Frequency

by the n specified features of the data that is to be classified, is formed [21]. An SVM performs its classification task by constructing a hyperplane, or a set of hyperplanes to try separate classes by as much distance as possible. The generalization error of the classifier is reduced when the margin between the hyperplane and the nearest training data point is as large as possible. SVMs utilize support vectors, the data point that lie closest to the hyperplane, to locate the optimal decision surface and specify the decision function [22]. SVM classifiers generally perform well in high dimensional spaces and tends to yield good results on smaller data sets, making it the ideal classifier on the model's data set. The

model's SVM classifier used a radial basis function kernel and a C value of 1. SVMs are not scale invariant, so all data that constituted the feature vectors were scaled to obtain meaningful results [20].

5 Results

5.1 Performance Results

Authentication can be characterized as a binary classification problem and the typical way of visualizing the performance of an authentication system is by plotting a receiver operating characteristic (ROC) curve for the classifier [16]. A ROC curve is a graphical representation of the output of a binary classifier as its threshold value changes. Ideally, the ROC curve will pass though the point (0, 1) at the top left corner of the plot, as this is the point where the true positive rate has been completely maximised and the false positive rate is completely minimised. This point closest to (0, 1) defines the systems equal error rate (EER), which is an important metric for evaluating a binary classification system. The EER signifies the point where the false acceptance rate (FAR), the percentage of imposter data samples that will likely be wrong authenticated by a system, equals the false rejection rate (FRR) is the percentage that a valid data sample will be wrongly denied authentication by a system [1]. The smaller the EER the better the performance of the system. Other metrics that will be used to evaluate the performance of the model are accuracy, precision and recall. The accuracy score of a system containing n samples can be calculated by:

$$acc(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{n_{samples}}^{i=1} 1(\hat{y}_i = y_i) \tag{2}$$

Given the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) we can calculate the scores for precision, recall and the f1-score:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

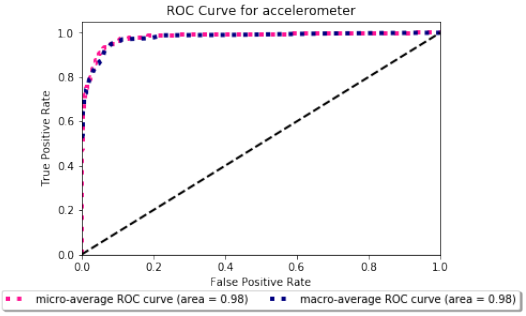
$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$F1\_score = \frac{2(Precision \times Recall)}{Precision + Recall} \tag{5}$$

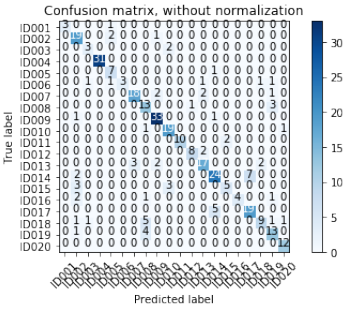
An additional mechanism for visualizing the performance of a classifier, particularly supervised ones, is a confusion matrix. A confusion matrix is defined by a matrix C, where $C_{i,j}$ is set to the number of data points that belong to class i but have been predicted to be in group j [19]. A confusion matrix allows for mislabelled classes to be easily identified. Finally, a classification report for each class was generated. The additional metrics presented here are an F1 score, the weighted mean of precision and recall, and support, the number of occurrences of a class in target values. All metrics were calculated using the one-vs-rest classification strategy.

5.2 Discussion

The results were generated using the one-vs-the-rest classification strategy, which fits one classifier per class. For validation, a testing and training split of 50-50 was used. The accelerometer achieved a satisfactory accuracy of 78.49% and EER of 7.04%. A precision score of 80.44% confirms that the model has a good exactness and and minimizes the number of false positives. The accelerometer's recall, however, did not fair as well with a score of 74.48%, a measure which is also reflected by f1 score achieved, 75.58%. The gyroscope archived an accuracy of 53.26% with an EER of 14.71%. The precision score attained by the gyroscope, 57.54%, was greater than that of it's recall score, 52.31%, as was the case with the accelerometer.

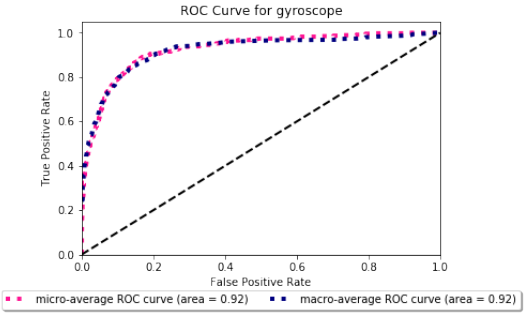


(a)

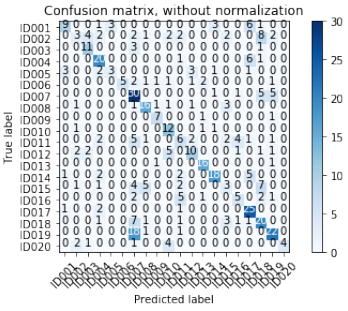


(b)

Fig. 4: a) ROC curve for the accelerometer. b) Confusion matrix for the accelerometer



(a)



(b)

Fig. 5: a) ROC curve for the gyroscope. b) Confusion matrix for the gyroscope

Table 2: Summary of Results.

Metric	Accelerometer	Gyroscope
Equal Error Rate	7.04%	14.71%
Accuracy	78.49%	53.26%
Precision	80.44%	57.54%
Recall	74.48%	52.31%
F1-Score	75.58%	51.78%

The accelerometer clearly outperformed the gyroscope across all the evaluated metrics, possibly due to the filtering step in pre-processing being insufficient to effectively reduce the noisy data picked up by the gyroscope. The performance of the gyroscope would most likely end up being a liability in a real-world authentication system by increasing the occurrences of false rejection and false acceptance cases. In the confusion matrices, it can be seen that classes ID001, ID003, ID005, ID006 and ID016 severely affect the overall performance of the model. A possible reason for the poor metric scores for these classes is that they both yielded a low number of feature vectors during enrolment, indicating the raw inertial signal data that was captured for both classes was not of sufficient length.

6 Conclusion and Future Work

The proposed system has the goal of providing a means to authenticate smartphone users according to their gait as measured by these smartphone sensors. Although the prototype implementation has proven gait based inertial sensor authentication to be possible, the solution has many improvements to make. Firstly, the pre-processing of gyroscopic data requires fine tuning. It may be possible to combine both accelerometer and gyroscope data using a Kalman filter to create an authentication module to replace the proposed systems gyroscope authentication model. Extracting additional features such as stride length and joint angle to increase the size of the feature vector may improve the decision boundary for the classifier. Also, the accelerometer pre-processing can be refined by segmenting each gait cycles trace into regions of interest and having feature extraction performed at this level. Additionally, other classification algorithms such as Naïve-Bayers or Random Forest could be utilized to potentially improve classification accuracy.

References

1. Jain, A. Ross, A (2008). Introduction to Biometrics. New York: Springer-Verlag New York, pp.1-2, 8-9.
2. Maghsoudi, J. and Tappert, C. (2016). A Behavioral Biometrics User Authentication Study Using Motion Data from Android Smartphones. 2016 European Intelligence and Security Informatics Conference (EISIC).

3. Zhong, Y. and Deng, Y. (2014). Sensor Orientation Invariant Mobile Gait Biometrics. IEEE International Joint Conference on Biometrics.
4. Boyd, J. and Little, J. (2005). Biometric Gait Recognition. Advanced Studies in Biometrics, pp.19-42.
5. Mason, J., Traoré, I. and Woungang, I. (2016). Gait Biometric Recognition. Machine Learning Techniques for Gait Biometric Recognition, pp.9-35.
6. Buriro, A. (2017). Behavioral Biometrics for Smartphone User Authentication. Ph.D. University of Trento, pp.15-21
7. Yampolskiy, R. and Govindaraju, V. (2008). Behavioural biometrics: a survey and classification. International Journal of Biometrics, 1(1), pp.81.
8. A. K. Jain, R. Bolle, and S. Pankant (1999). Biometrics: Personal Identification in Networked Society. Kluwer Academic Publishers, pp.16
9. Gafurov, D., Snekkenes, E. and Bours, P. (2007). Gait Authentication and Identification Using Wearable Accelerometer Sensor. 2007 IEEE Workshop on Automatic Identification Advanced Technologies.pp.1-9
10. Sebastijan, S. and Damjan, Z. (2009). Gait Identification Using Cumulants of Accelerometer Data. Proceedings of the 2nd WSEAS International Conference on Sensors, and Signals and Visualization, Imaging and Simulation and Materials Science, 2, pp.94-99.
11. Derawi, M., Bours, P. and Holien, K. (2010). Improved Cycle Detection for Accelerometer Based Gait Authentication. 2010 Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing.pp3-7
12. Mantyjarvi, J., Lindholm, M., Vildjiounaite, E., Makela, S. and Ailisto, H. (2005). Identifying Users of Portable Devices from Gait Pattern with Accelerometers. Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. 2.pp.1-7
13. Niyogi, S. and Adelson, E. (1994). Analyzing gait with spatiotemporal surfaces. Proceedings of 1994 IEEE Workshop on Motion of Non-rigid and Articulated Objects.
14. Lee, C., Tan, A. and Lim, K. (2017). Review on Vision-Based Gait Recognition: Representations, Classification Schemes and Data sets. American Journal of Applied Sciences, 14(2), pp.252-266.
15. Boch Sensortech. "BMI160 A Small, low power inertial measurement unit". BST-BMI160-DS000-07. Datasheet. (2005)
16. Zhao, Y. and Zhou, S. (2017). Wearable Device-Based Gait Recognition Using Angle Embedded Gait Dynamic Images and a Convolutional Neural Network. Sensors, 17(12), pp.478.
17. Kupryjanow, A., Maziewski, P., Kaszuba, K. and Czyżewski, A. (2009). Accelerometer signal pre-processing influence on human activity recognition. pp.1 - 6.
18. Ehatisham-ul-Haq, M., Azam, M., Loo, J., Shuang, K., Islam, S., Naeem, U. and Amin, Y. (2017). Authentication of Smartphone Users Based on Activity Recognition and Mobile Sensing. Sensors, 17(9), pp.2043.
19. Johansson, R. (2015). Numerical Python. Berkeley, CA: Apress, pp.30 - 31.
20. Pedregosa, F., Varoquaux, G., Gramfort, A. and Michel, V. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, pp.2825-2830.
21. Fradkin, D. and Muchnik, I. (2000). Support Vector Machines for Classification. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pp.1 - 9.
22. Robert, B. (2009). An Idiot's Guide to Support Vector Machines (SVMs). Available at http://www.cs.ucf.edu/courses/cap6412/fall2009/paper

Author Index