Acta Universitatis Sapientiae

Informatica

Volume 5, Number 1, 2013

Sapientia Hungarian University of Transylvania
Scientia Publishing House

Contents

Acta Univ. Sapientiae, Informatica, **5,** 1 (2013) 5–15

Online data clustering algorithms in an RTLS system

Tamás NÉMETH
University of Szeged
Institute of Informatics
email: tnemeth@inf.u-szeged.hu

Sándor NAGY
Fraunhofer Institut for Integrated Circuits, Locating and Communication Systems Department
email:
sandor.nagy@iis.fraunhofer.de

Csanád IMREH
University of Szeged
Institute of Informatics
email: cimreh@inf.u-szeged.hu

Abstract. This paper proposes and evaluates solutions for an online clustering problem and gives a mathematical model for it. The problem at hand occurs often in the fusion of data streams for example in real time locating systems. The goal is to gather as much incoming information from several sources as possible but also minimize the delay before the next processing step can be executed. The key characteristic is that the data is available in a bursty fashion, in the special case of an RTLS according to the locating cycles. After an introduction of the background a general mathematical model for the problem is given, and then two basic algorithms referred to as NWT and CWT are analyzed by the method of competitive analysis. Each turning out to deliver an optimal solution under different constraints. Then an experimental evaluation follows based on a simulation involving the CWT and the algorithm referred to as VWT. The later is giving a configuration free solution for the problem.

Computing Classification System 1998: F.1.2
Mathematics Subject Classification 2010: 68W27
Key words and phrases: online algorithms, clustering, locating systems

1 Introduction

The problem for this paper emerged in the real time locating system developed by the Fraunhofer IIS, but it can be generalized. We will do so in the second section by giving a mathematical model for it. To draw a context around the problem here we explain where it originates from, and why the specific constraints are posed.

Real time location systems (RTLS) promise to deliver high precision positions of objects with manifold applications in transport and logistics, ambient assisted living, or emergency mission support and also in the entertainment and sports like the Chip-in-the-Ball technologies. The mentioned locating system of the Fraunhofer IIS is a radio based system working with a local locating infrastructure using Angle of Arrival (AoA) and Round-Trip-Time (RTT) measurements to determine the position of objects (see [2] and [3] for a more detailed description of this RTLS system). In this system the objects are equipped with tags based on the in-house-developed Wireless Smart Item platform (WISMIT). These tags periodically broadcast a radio signal to the infrastructure nodes denoting the beginning of a locating cycle. We will refer to this as a burst. The radio signal carries also user data, aside from tag identification information: an incremental number identifying the burst, referred to as burst-id later on, is also included. A set of infrastructure nodes in proximity of the tag consisting of WISMIT anchors capable of RTT measurements and of Goniometers capable of both RTT and AoA measurements receive this broadcast and initiate location data acquisition for this tag. The measured parameters form a so called burst data set which is at first only available distributed on the infrastructure nodes. After the measurement is done the individual parts of the burst data set, the data elements are sent over an Ethernet connection to the central positioning server through the TCP/IP network stack utilizing UDP as a transport protocol. UDP packets were chosen in contrast to a TCP stream because of its lower overhead and characteristics: if a datagram was lost, no additional time is spent for detecting the loss and retransmitting it. There is also no guaranty for order-reserving delivery.

After receiving sufficiently enough data, ideally the whole burst data set, the positioning server can determine the position of the object. For this the raw location data is going through the following algorithmical steps in the server:

1. Raw data filter
2. Clustering

3. Burst filter
4. Position calculation
5. Position filter

Most of the steps are meant for reducing measurement noise and errors.

The focus of this paper lies on the second step: the clustering component. It is in charge for gathering as much of the burst data set as possible and forward it as quickly as possible to the burst filtering and position calculation. For this it has to work in an online manner by clustering the data collected by the infrastructure nodes. It has also to deal with transmission errors or disturbances like the temporary coverage of an infrastructure node, and network packet losses and with variable network and processing delays as well. Therefore the clustering algorithm can not simply wait for the arrival of all elements of the burst data set, it needs a more sophisticated technique. Thus the goal of the algorithm is to decide when to pass the collected data to the position calculation. There are two contradicting objectives which should be satisfied. First, the positioning server should receive all the available data from the infrastructure nodes. Moreover the system is not allowed to wait a long time for the incoming data, since the delay decreases the relevance of the determined position. A fairly-good position is better than a position too late. We present an integrated objective function which considers these goals, and defines this clustering problem as a maximization problem. Since collecting the data and the position calculation can be done parallel and independently for several tags, we suppose that there is only one tag present and the infrastructure nodes collect data only from this tag. We note that this problem is similar to the online data acknowledgment problem, where the goal is to determine the sending time of the acknowledgments (see [1] and [4]). One of the presented algorithms uses ideas used in the field of data acknowledgment problems.

As next we present the mathematical model of the problem, giving the objective function and we define competitiveness. Typically, the quality of an online algorithm is judged using competitive analysis. This method will be used in Section 3 to analyze the efficiency of our algorithms. We will present there two analyses: one without constraints on the data arrival times, and one with constraints to better describe the system behavior and deliver stronger results.

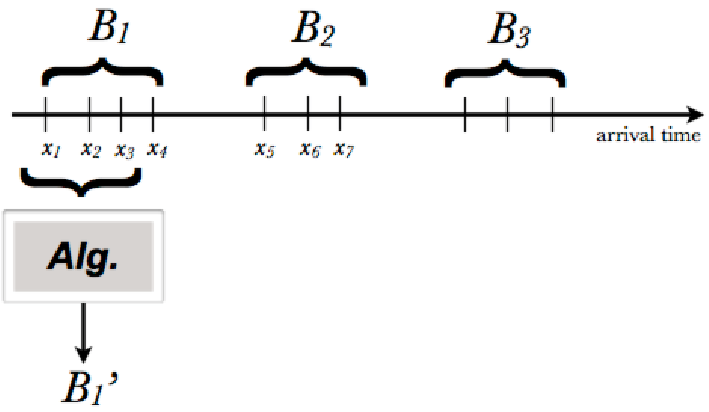


Figure 1: The clustering of the input signals

2 The mathematical model of the clustering problem

The input of this clustering problem is a list X: $x_1, \ldots, x_n$ of collected data elements. We will use the following notation: $\mathtt{Burst}(x_i)$ denotes the burst id which identifies to which locating cycle the data belongs to. $\mathtt{Node}(x_i)$ gives the infrastructure node which collected the data. $\mathtt{Rcpt}(x_i)$ is the reception time of the data. The difference between the reception times of the last and the first data elements of a burst is called the length of the burst.

We note that in the application the data has further attributes also (e.g.: AoA values, RTT values) which are substantial for the position calculation, but are not used in the clustering algorithm. $B_j = \{x_i | \mathtt{Burst}(x_i) = j\}$ is the burst data set, $B'_j \subseteq B_j$ is the subset of the burst data set which is sent into the positioning. The point of time when the algorithm decides to send B'_j to the positioning is denoted by p_j ($j = 1, \ldots, b$, where b is the number of bursts). These sets are presented on Figure 1.

If the infrastructure nodes have different technical properties (position, accuracy) then the collected data can have different importance. Therefore we assign a weight $w_k > 0$ to the infrastructure node k which describes the importance of the data collected by the node. Without loosing generality we can assume that $w_1 = \min_{k=1}^{m} w_k$ where m is the number of infrastructure nodes.

To evaluate the algorithms first we have to define an objective function

which measures their efficiency. We have to take into account both objectives of the algorithm: loose minimal amount of data, minimize the delay of starting the positioning part. To define the objective function we use the following notations. For each positioning time p_j and infrastructure node k let $e_{jk} = 1$ if node k sends data into the positioning calculation, otherwise let $e_{jk} = 0$. Let

$$r_j = \max_{i=1}^{n}\{Rcpt(x_i)|x_i \in B'_j\}.$$

Now we can define the following objective function, which we have to maximize:

$$f = \sum_{j=1}^{b}\sum_{k=1}^{m} e_{jk}w_k - c\sum_{j=1}^{b}(p_j - r_j).$$

The first part considers the total importance of the data which are sent to the positioning server, the second part is the sum of the unnecessary latencies multiplied by c, the cost of the unity latency. We assume $c > 0$.

An online algorithm for a maximization problem is ρ-competitive if the optimal gain is never more than ρ times the gain of the online algorithm.

In a more formal way: for an arbitrary online algorithm $\mathcal{A}$ and an input sequence X the gain of the solution given by $\mathcal{A}$ is denoted by $\mathcal{A}(X)$. Moreover for an input sequence X let $OPT(X)$ denote the gain of the optimal offline solution. Then an online algorithm $\mathcal{A}$ is called ρ-competitive if $OPT(X) \leq \rho \cdot \mathcal{A}(X)$ for any input sequence X.

We note that it would also be possible to define the problem as a minimization problem using an objective function which is the sum of the weighted number of the lost data elements and the latencies. But in this case the optimal offline algorithm has cost 0. This would make it impossible to use competitive analysis to study our algorithm. Therefore we decided to develop the maximization version of the objective function.

3 Competitive analysis

3.1 Analysis without constraints

First we do not introduce any further constraints on the input sequence X. For this case we show that no constant competitive algorithm exists as the following lower bound dictates.

Theorem 1 *There exists no algorithm which has smaller competitive ratio than* $\frac{1}{w_1}\sum_{k=1}^{m} w_k \geq m$.

Proof. Consider the following sequence. Let the first data element with burst id 1, and infrastructure node 1 arrive at time 0. If the online algorithm chooses $p_1 \geq \frac{w_1}{c}$, then the sequence is ended, and the online gain is non-positive, the offline algorithm can use $p_1^{opt} = 0$, and its gain is w_1. Therefore in this case the algorithm is not competitive. Now suppose that $p_1 < \frac{w_1}{c}$. Then $m-1$ further data elements arrive each of them having burst id 1 and infrastructure node ids $2, \ldots, m$ at time $\frac{w_1}{c}$. In this case the online algorithm has a gain of at most w_1 and the optimal offline algorithm uses $p_1^{opt} = \frac{w_1}{c}$, and is $\sum_{k=1}^{m} w_k$. □

3.1.1 No Waiting Time Algorithm

The previous lower bound shows that in the worst case data arrives after sending the data to the positioning server. Therefore the online algorithm has no reason to wait for further data after the arrival of the first. The No Waiting Time Algorithm (NWT in short) follows this idea, it sends the first data element for each burst id into the positioning server immediately after it arrives. The competitive ratio of this algorithm is determined below.

Theorem 2 *The competitive ratio of NWT is* $\frac{1}{w_1}\sum_{k=1}^{m} w_k$.

Proof. Consider an arbitrary sequence which contains b bursts. For each burst the algorithm gains the value of the first data element, therefore its gain is at least $b \cdot w_1$. On the other hand the optimal gain cannot be more than $b \cdot \sum_{k=1}^{m} w_k$. □

Corollary 3 *NWT achieves the smallest possible competitive ratio in the general model.*

3.2 Latency limited analysis

In this subsection we consider restricted inputs better modeling the system functions. We suppose that the difference among the arrivals of the data elements with the same burst id (the length of the burst) cannot be greater than

a given value d. In this subsection we will assume that $\frac{w_1}{c} > d$, otherwise the lower bound proof of the general case also works in this case and we obtain that NWT is an algorithm with the smallest possible competitive ratio.

Theorem 4 *There exists no algorithm which has smaller competitive ratio than* $\min\left\{\frac{w_1}{w_1 - c \cdot d}, \frac{1}{w_1}\sum_{k=1}^{n} w_k\right\}$.

Proof. Consider the following sequence. Let the first data element with burst id 1, and infrastructure node 1 arrive at time 0. If the online algorithm chooses $p_1 \geq d$, then the sequence is ended, and the online gain is $w_1 - c \cdot p_1 \leq w_1 - c \cdot d$, the offline algorithm can use $p_1^{opt} = 0$, and its gain is w_1. Therefore in this case the algorithm is not better than $\frac{w_1}{w_1 - c \cdot d}$-competitive. Now suppose that $p_1 < d$. Then $m - 1$ further data elements arrive each having burst id 1 and infrastructure node ids $2, \ldots, m$ at time d. In this case the online algorithm has a total gain of at most w_1 and the optimal offline algorithm uses $p_1^{opt} = d$, and its gain is $\sum_{k=1}^{m} w_k$. □

The Constant Waiting Time algorithm (CWT) waits time d after the arrival of the first data element for each burst id before sending the data to the location server.

Theorem 5 *The competitive ratio of CWT is* $\frac{w_1}{w_1 - c \cdot d}$.

Proof. Consider an arbitrary sequence X which contains b bursts. By the limited latency constraint it follows that for each burst id the algorithm collects all data. Therefore its total gain is at least $G - b \cdot c \cdot d$, where G is the sum of the gains of all the incoming data. On the other hand the optimal gain cannot be more than G, and $G \geq b \cdot w_1$. Therefore the competitive ratio of CWT is at most

$$\frac{G}{G - b \cdot c \cdot d} \leq \frac{w_1}{w_1 - c \cdot d}.$$

□

Corollary 6 *If* $\frac{w_1}{w_1 - c \cdot d} \leq \frac{1}{w_1}\sum_{k=1}^{m} w_i$ *then CWT achieves the smallest possible competitive ratio with the latency constraint, otherwise NWT achieves the smallest possible competitive ratio.*

4 Experimental evaluation

As it is shown in Section 3 some very simple algorithm can achieve the best possible competitive ratio but its efficiency depends on the knowledge of the parameter d. In this section we first introduce a more sophisticated algorithm trying to get knowledge about d, and later present an empirical analysis which compares the algorithms in the average case.

4.1 Variable Waiting Time Algorithm

In this algorithm each burst id j has a starting time denoted by $S(j)$ which is the reception time of the first data element having burst id j. Furthermore each burst has a dataset $DS(j)$ which contains the collected data for the burst. (This will be the set B'_j after releasing it.) The algorithm uses a waiting time variable t which is the time while the algorithm waits data of burst j. The algorithm tries to learn the best value of this variable. Algorithms which use similar parameter learning idea are presented for the online data acknowledgment problem and for the scheduling problem with rejection in [5] and [6]. The algorithm also uses two other parameters DEC and INC, the first gives a lower bound on the possible change of the variable W in the descending direction, INC is a security distance which gets added additionally if the value of W has to be increased. MEM is a parameter which determines how long the algorithm should keep record about recent bursts.

Algorithm VWT uses the following rules to send data to the positioning server:

- If the actual time is $S(j)+W$ and $DS(j)$ is not closed then the algorithm closes set $DS(j)$. It notes that its closing time is $S(j)+W$, and it sends the data to the positioning server. Furthermore let

$$PD(j) = \max\{0, \min_{a=0}^{MEM-1}\{p_{j-a} - r_{j-a}\} - INC\}$$

  be the minimal possible decrease amount for W, respecting also INC security time, so that the data of the last MEM bursts would have been left complete. So if $PD(j) \geq DEC$, then W is decreased by $PD(j)$.

- If $DS(j)$ collects the data at time t from all of the infrastructure nodes the algorithm closes set $DS(j)$. It notes that its closing time is t, and it sends the data to the positioning server. Note that this might happen

only in the case when $t \leq S(j) + W$. A possible decrease of W can also happen analogue to the first point.

The algorithm uses the following rules to handle the received data x_i.

- If an x_i which belongs to burst j arrives at time t and $DS(j)$ is not closed, then it extends it with the new data element, and checks whether it is the last missing infrastructure node.
- If $DS(j)$ is already closed with closing time at r_j then $W := t - S(j) + INC$.

4.2 Empirical analysis

To analyse the algorithms in average case we used the simulation tool developed at the Fraunhofer Institute. In this simulation tool a virtual object with a tag is moving on a configurable trajectory. It generates RTT and AoA data for 3 Goniometers (collecting both RTT and AoA data) and 8 anchors (collecting only RTT data). The following properties of the system are determined randomly in our simulation:

- each data element sent by the virtual infrastructure nodes is lost with a given probability (it is defined separately for the RTT and AoA data)
- the length of the time span between two bursts and between the data elements of a burst are both normally distributed.

We generated the following 6 inputs. In tests A and B the number of bursts was 68 and 262 and during these bursts 952 and 3668 data elements arrived, the average length of the bursts was 2.04 and 2.6. In tests C and D we studied slightly shorter bursts, the average length was 0.99 and 1.06. In test C we used 131 bursts with 1800 data elements, in test D 138 bursts was used with 1932 data elements. Finally in tests E and F much longer bursts were used with average length 34.07 and 35.9. Test E was smaller it contained 127 bursts with 1742 incoming data elements, test F was large it contained 879 bursts with 12079 data elements.

We considered 3 constant waiting time algorithms for the solution of the problem. CWT(1) used a smaller constant which is close to the average burst lengths of C and D, CWT(2) used a larger constant which was still smaller then the average length in tests E and F, and $CWT(3)$ used a constant which was close to the average lengths in tests E and F. In Table 1 we collected the ratios which were received by dividing the gain of the algorithm by the optimal

gain. We used the values $w_i = 0.07$ for each i and $c = 0.01$ in the objective function.

	Test A	Test B	Test C	Test D	Test E	Test F
CWT(1)	0.930	0.883	0.967	0.964	0.142	0.194
CWT(2)	0.730	0.736	0.715	0.721	0.794	0.759
CWT(3)	0.660	0.666	0.643	0.651	0.976	0.936
VWT	0.976	0.985	0.978	0.984	0.985	0.914

Table 1: The test results

Based on the above results we can draw the following conclusions:

- It is clear that the efficiency of the CWT algorithms depends highly on the average length of the bursts. CTW(1) gives good results for tests A, B, C, D but has extremely poor performance on tests E and F. CWT(2) and CWT(3) gave good results on tests E and F but their results were weak on the other tests. This means that these algorithms can be used only in the cases where we have some a priori information about the data. But we note that in these cases they work well: their performance is better than 0.9.
- The VWT algorithm performs also well with unknown average burst-lengths, it delivers good results in each testcase. The minimal ratio of the algorithms to the optimal gain is 0.914, the average ratio is 0.97.

5 Summary and open questions

In this paper we defined an online optimization model for the clustering problem which appears in real time location systems. We presented optimal online algorithms in the sense that they achieve the smallest possible competitive ratio and a more sophisticated algorithm which is designed to learn the average length of the bursts. We showed by an empirical analysis that this learning algorithm is useful if we have no a priori information about the data.

There are some further interesting questions related to our problem. It would be interesting to study other, more difficult objective functions. Furthermore, in this model we supposed that at the arrival of the data element we receive the id of the burst where the data element was sent. In some application we do not receive this information, it would be interesting to study this scenario as well.

Acknowledgement

This research was partially supported by the TÁMOP-4.2.1/B-09/1/KONV-2010-0005 program of the Hungarian National Development Agency. Cs. Imreh was supported by the Alexander von Humboldt Foundation, completed most of this work at the Humboldt University at Berlin, and is grateful to Professor Suzanne Albers for the kind hospitality. Tamás Németh completed some of this work while visiting the Fraunhofer Institute in Nuremberg, and is grateful for the kind hospitality.

References

[1] S. Albers,H. Bals, Dynamic TCP acknowledgement: Penalizing long delays, *SIAM J. Discrete Math.* **19,** 4 (2005) 938–951. ⇒7

[2] M. Brugger, T. Christ, F. Kemeth, S. Nagy, M. Schaefer, M. M. Pietrzyk, The FMCW technology-based indoor localization system, *Proc. Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS)*, Helsinki, Finnland, 2010, pp. 1–6. ⇒6

[3] M. Brugger, F. Kemeth, Locating rate adaptation by evaluating movement specific parameters, *Proc. 2010 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, Anaheim, USA, 2010, pp. 127–133. ⇒6

[4] D. R. Dooly, S. A. Goldman, S. D. Scott, On-line analysis of the TCP acknowledgment delay problem, *J. ACM* **48,** 2 (2001) 243–273. ⇒7

[5] Cs. Imreh, T. Németh, Parameter learning algorithm for the online data acknowledgment problem, *Optim. Methods Softw.*, **26,** 3 (2011) 397–404. ⇒12

[6] T. Németh, Cs. Imreh, Parameter learning online algorithm for multiprocessor scheduling with rejection, *Acta Cybernet.*, **19,** 1 (2009) 125–133. ⇒12

Received: March 8, 2013 • Revised: May 14, 2013

Acta Univ. Sapientiae, Informatica, **5,** 1 (2013) 16–34

Efficient computing of n-dimensional simultaneous Diophantine approximation problems

Attila KOVÁCS
Eötvös Loránd University
Faculty of Informatics
email:
attila.kovacs@compalg.inf.elte.hu

Norbert TIHANYI
Eötvös Loránd University
Faculty of Informatics
email:
ntihanyi@compalg.inf.elte.hu

Abstract. In this paper we consider two algorithmic problems of simultaneous Diophantine approximations. The first algorithm produces a full solution set for approximating an irrational number with rationals with common denominators from a given interval. The second one aims at finding as many simultaneous solutions as possible in a given time unit. All the presented algorithms are implemented, tested and the PariGP version made publicly available.

1 Introduction

1.1 The problem statement

Rational approximation, or alternatively, Diophantine approximation is very important in many fields of mathematics and computer science. Archimedes approximated the irrational number π with $22/7$. Long before Archimedes, ancient astronomers in Egypt, Babylonia, India and China used rational approximations. While the work of John Wallis (1616–1703) and Christiaan Huygens (1629–1695) established the field of continued fractions, it began to blossom

Computing Classification System 1998: G.2.0
Mathematics Subject Classification 2010: 68R01, 11J68
Key words and phrases: Diophantine approximation

when Leonhard Euler (1707–1783), Johann Heinrich Lambert (1728–1777) and Joseph Louis Lagrange (1736–1813) embraced the topic. In the 1840s, Joseph Liouville (1809–1882) obtained an important result on general algebraic numbers: if α is an irrational algebraic number of degree $n > 0$ over the rational numbers, then there exists a constant $c(\alpha) > 0$ such that

$$\left|\alpha - \frac{p}{q}\right| > \frac{c(\alpha)}{q^n}$$

holds for all integers p and $q > 0$. This result allowed him to produce the first proven examples of transcendental numbers. In 1891 Adolf Hurwitz (1859–1919) proved that for each irrational α infinitely many pairs (p, q) of integers satisfy

$$\left|\alpha - \frac{p}{q}\right| < \frac{1}{q^2\sqrt{5}},$$

but there are some irrational numbers β for which at most finitely many pairs satisfy

$$\left|\beta - \frac{p}{q}\right| < \frac{1}{q^{2+\gamma}\sqrt{5+\mu}}$$

no matter how small the positive increments γ and μ are.

The idea can be generalized to simultaneous approximation. Simultaneous diophantine approximation originally means that for given real numbers $\alpha_1, \alpha_2, \ldots, \alpha_n$ find $p_1, p_2, \ldots, p_n, q \in \mathbb{Z}$ such that

$$\left|\alpha_i - \frac{p_i}{q}\right|$$

is "small" for all i, and q is "not too large".

For a given real α let us denote the nearest integer distance function by $\|.\|$, that is, $\|\alpha\| = \min\{|\alpha - j|, j \in \mathbb{Z}\}$. Then, simultaneous approximation can be interpreted as minimizing

$$\max\{\|q\alpha_1\|, \ldots, \|q\alpha_n\|\}.$$

In 1842 Peter Gustav L. Dirichlet (1805–1859) showed that there exist simultaneous Diophantine approximations with absolute error bound $q^{-(1+1/n)}$. To be more precise, he showed that there are infinitely many approximations satisfying

$$|q \cdot \alpha_i - p_i| < \frac{1}{q^{1/n}} \tag{1}$$

for all $1 \le i \le n$. Unfortunately, no polynomial algorithm is known for the simultaneous Diophantine approximation problem. However, due to the L^3 algorithm of Lenstra, Lenstra and Lovász, if $\alpha_1, \alpha_2, \ldots, \alpha_n$ are irrationals and $0 < \varepsilon < 1$ then there is a polynomial time algorithm to compute integers $p_1, p_2, \ldots, p_n, q \in \mathbb{Z}$ such that

$$1 \le q \le 2^{n(n+1)/4}\varepsilon^{-n} \text{ and } |q \cdot \alpha_i - p_i| < \varepsilon$$

for all $1 \le i \le n$ (see [10]).

Lagarias [7, 8] presented many results concerning the best simultaneous approximations. Szekeres and T. Sós [12] analyzed the signatures of the best approximation vectors. Kim et al. [4] discussed rational approximations to pairs of irrational numbers which are linearly independent over the rationals and applications to the theory of dynamical systems. Armknecht et al. [1] used the inhomogeneous simultaneous approximation problem for designing cryptographic schemes. Lagarias [9] discussed the computational complexity of Diophantine approximation problems, which, depending on the specification, varies from polynomial-time to $\mathcal{NP}$-complete. Frank and Tardos [2] developed a general method in combinatorial optimization using simultaneous Diophantine approximations which could transform some polynomial time algorithms into strongly polynomial.

In this paper we focus on two algorithmic problems. Consider the set of irrationals $\Upsilon = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$. Let $\varepsilon > 0$ be real and $1 \le a \le b$ be natural numbers. Furthermore, let us define the set

$$\Omega(\Upsilon, \varepsilon, a, b) = \{k \in \mathbb{N} : a \le k \le b, \|k\alpha_i\| < \varepsilon \text{ for all } \alpha_i \in \Upsilon\}. \tag{2}$$

For given Υ, ε and a, b

1. determine all the elements of $\Omega(\Upsilon, \varepsilon, a, b)$,
2. determine as many elements of $\Omega(\Upsilon, \varepsilon, a, b)$ as possible in a given time unit

efficiently. We refer to the first problem as the "all-elements simultaneous Diophantine approximation problem". In case of $|\Upsilon| = n \ge 1$ we call it an n-dimensional simultaneous approximation. The second problem is referred to as the "approximating as many elements as possible" problem.

Challenges:

1. Determine all elements of

$$\Omega\left(\{\sqrt{2}\}, 10^{-17}, 10^{20}, 10^{21}\right). \tag{3}$$

2. Determine as many elements of

$$\Omega\left(\left\{\frac{\log(p)}{\log(2)}, p \text{ prime}, 3 \le p \le 19\right\}, 10^{-2}, 1, 10^{18}\right) \tag{4}$$

as possible in a given time unit.

1.2 The continued fraction approach

It is well-known that continued fractions are one of the most effective tools of rational approximation to a real number. *Simple continued fractions* are expressions of the form

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cdots}}$$

where a_i are integer numbers with $a_1, a_2, \ldots > 0$. It is called *finite* if it terminates, and *infinite* otherwise. These continued fractions are usually represented in bracket form $[a_0, a_1, \ldots, a_m, \ldots]$, i.e.

$$C_0 = [a_0] = a_0, \quad C_1 = [a_0, a_1] = a_0 + \frac{1}{a_1}, \quad C_2 = [a_0, a_1, a_2] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2}}, \quad \ldots$$

where C_m are called *convergents*. Clearly, the convergents C_m represent some rational numbers p_m/q_m. An infinite continued fraction $[a_0, a_1, a_2, \ldots]$ is called convergent if its sequence of convergents C_m converges in the usual sense, i.e. the limit

$$\alpha = \lim_{m\to\infty} C_m = \lim_{m\to\infty} [a_0, a_1, \ldots, a_m]$$

exists. In this case we say that the continued fraction represents the real number α. The simple continued fraction expansion of $\alpha \in \mathbb{R}$ is infinite if and only if α is irrational. The convergents C_m are the best rational approximations in the following sense:

Lemma 1 *No better rational approximation exists to the irrational number* α *with smaller denominator than the convergents* $C_m = p_m/q_m$.

Example 2 *The simple continued fraction approximation for* $\sqrt{2}$ *is* $[1, 2, 2, \ldots]$, *the sequence of the convergents is*

$$1, \frac{3}{2}, \frac{7}{5}, \frac{17}{12}, \frac{41}{29}, \frac{99}{70}, \frac{239}{169}, \frac{577}{408}, \frac{1393}{985}, \frac{3363}{2378}, \frac{8119}{5741}, \ldots$$

Among all fractions with denominator at most 29, the fraction 41/29 is the closest to $\sqrt{2}$, among all fractions with denominator at most 70, the fraction 99/70 is closest to $\sqrt{2}$, and so on.

Every convergent is a best rational approximation, but these are not all of the best rational approximations. Fractions of the form

$$\frac{p_{m-1} + jp_m}{q_{m-1} + jq_m} \quad (1 \leq j \leq a_{m+2} - 1),$$

are called *intermediate convergents* or *semi-convergents*. To get every rational approximation between two consecutive p_m/q_m and p_{m+1}/q_{m+1}, we have to calculate the intermediate convergents.

Example 2 (cont.) *The missing intermediate convergents of Example 2 are*

$$\frac{4}{3}, \frac{10}{7}, \frac{24}{17}, \frac{58}{41}, \frac{140}{99}, \frac{338}{239}, \frac{816}{577}, \frac{1970}{1393}, \frac{4756}{3363}, \ldots$$

The approximations $|\alpha - p/q|$ above are also known as "best rational approximations of the first kind". However, sometimes we are interested in the approximations $|\alpha \cdot q - p|$. This is called the *approximation of a second kind.*

Lemma 3 [3] *A rational number p/q, which is not an integer, is a convergent of a real number α if and only if it is a best approximation of the second kind of α.*

In 1997 Clark Kimberling proved the following result regarding intermediate convergents [5]:

Lemma 4 *The best lower (upper) approximates to a positive irrational number α are the even-indexed (odd-indexed) intermediate convergents.*

Example 2 (cont.) *In order to generate many integers q that satisfy*

$$\|q \cdot \sqrt{2}\| < 10^{-5} \tag{5}$$

one can apply the theory of continued fractions, especially convergents. If q_m is the first integer that satisfies $\|q_m \cdot \sqrt{2}\| < 10^{-5}$ in the continued fraction expansion of $\sqrt{2}$, then all convergents with denominator larger than q_m will satisfy equation (5).

Example 5 *Consider Challenge* 1 *stated in* (3). *There are only* 3 *convergents of* $\sqrt{2}$ *where* $10^{20} < q_m < 10^{21}$. *They are*

$$\frac{233806732499933208099}{165326326037771920630}, \quad \frac{564459384575477049359}{399133058537705128729}, \quad \frac{1362725501650887306817}{963592443113182178088}.$$

With intermediate convergents we get 2 *more solutions. Hence, with the theory of continued fractions we are able to find only* 5 *appropriate integers. One may ask how many elements are in the set* Ω *in* (3)*?*

Hermann Weyl (1855–1955) and Waclaw Sierpiński (1882–1969) proved in 1910 that if $\alpha \in \mathbb{R} \setminus \mathbb{Q}$ then $\alpha, 2\alpha, 3\alpha, \ldots \pmod 1$ is uniformly distributed on the unit interval. From this theorem it immediately follows that there are approximately $2(b-a)\varepsilon$ appropriate integers in the $[a, b]$ interval. In Challenge 1 we expect $2(10^{21} - 10^{20}) \cdot 10^{-17} = 18000\ (\pm 1)$ integers. This is by several orders of magnitude more than what we were able to obtain by continued fractions.

1.3 The Lenstra–Lenstra–Lovász approach

We have seen in the previous section that Challenge 1 is unsolvable with the theory of continued fractions. Challenge 2 is a 7-dimensional simultaneous approximation problem and is even more beyond the potentials of continued fractions. Although there is not known polynomial-time algorithm that is able to solve the Dirichlet type simultaneous Diophantine approximation problem, there exists an algorithm that can be useful for *similar* problems. The Lenstra–Lenstra–Lovász basis reduction algorithm (L^3) is a polynomial-time algorithm that finds a reduced basis in a lattice [10]. The algorithm can be applied to solve simultaneous Diophantine approximation *with an extra condition.*

Lemma 6 *There exists a polynomial-time algorithm for the given irrationals* $\alpha_1, \alpha_2, \ldots, \alpha_n$ *and* $0 < \varepsilon < 1$ *that can compute the integers* $p_1, \ldots, p_n$ *and* q *such that*

$$\left|\alpha_i - \frac{p_i}{q}\right| < \frac{\varepsilon}{q} \tag{6}$$

and

$$0 < q \leq \beta^{n(n+1)/4} \varepsilon^{-n}$$

hold for all $1 \leq i \leq n$, *where* β *is an appropriate reduction parameter.*

The extra condition is the bound $0 < q \leq \beta^{n(n+1)/4} \varepsilon^{-n}$.

In one-dimension the L^3 algorithm provides exactly the continued fraction approach discussed in the previous section, hence L^3 is not an effective tool for answering Challenge 1. And what about the multidimensional case like Challenge 2?

Let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be irrational numbers and let us approximate them with rationals admitting an $\varepsilon > 0$ error. Let $X = \beta^{n(n+1)/4}\varepsilon^{-n}$ and let the matrix A be the following:

$$A = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ \alpha_1 X & X & 0 & \ldots & 0 \\ \alpha_2 X & 0 & X & \ldots & 0 \\ \vdots & & & & \vdots \\ \alpha_n X & 0 & 0 & \ldots & X \end{bmatrix}.$$

Applying the L^3 algorithm for A, the first column of the resulting matrix contains the vector $[q, p_1, p_2, p_3, \ldots, p_n]^T$ which satisfies (6).

Let us see how the L^3 algorithm works in dimension 7. Let $\alpha_i = \frac{\log(p_{i+1})}{\log(2)}$ where p_i denotes the i-th prime for $1 \leq i \leq 7$, and let $\varepsilon = 0.01$. We are looking for an integer $q \leq 2^{14} \cdot 100^7$ that satisfies $\|q \cdot \alpha_i\| < \varepsilon$ for all i. Applying the L^3 algorithm we got $q = 1325886000944418$. It is easy to verify that $\|q\,\alpha_i\| < 0.01$ holds for all $1 \leq i \leq 7$.

The L^3 algorithm can also be applied in higher dimensions, however, there are some cases where the algorithm can not be used efficiently. The real drawback of the method for our purposes is that it is inappropriate for finding all or many different solutions q in an *arbitrary interval*. We note that sometimes one can find a few more solutions with a different choice of β (but not much more).

It can be concluded that the apparatus of the continued fractions and the L^3 algorithm is not appropriate for solving Challenge 1 and Challenge 2 problems. In this paper we present new methods that can be used to solve these kinds of problems efficiently. All the algorithms presented in this paper were implemented and tested in PARI/GP 2.5.3 with an extension of GNU MP 5.0.1. The experimenting environment was an Intel® Core i5-2450M with Sandy Bridge architecture. The code can be downloaded from the project homepage[1].

[1] http://www.riemann-siegel.com/

2 Approximation in the one-dimensional case

2.1 "All-elements" approximation

In this section we present how to calculate all the elements of $\Omega(\Upsilon, \varepsilon, a, b)$ where $\Upsilon = \{\alpha\}$.

For a given Ω let $k : \{1, 2, \ldots, |\Omega|\} \to \Omega$ monotonically increasing, so k_i denotes the ith integer in Ω. Let us define the set

$$\Delta_\Omega = \{k_{n+1} - k_n : \ 1 \le n \le |\Omega| - 1\}.$$

The set Δ_Ω contains all possible step-sizes between two consecutive k_i's.

Theorem 7 $|\Delta_\Omega| \le 3$.

Proof. The proof has two parts. In the first step we construct all the possible three elements of Δ_Ω and in the second step we show that there is no more. For the given irrational α and an arbitrary $m \in \mathbb{N}$ let

$$\langle m \rangle = \begin{cases} \|\alpha m\| & \text{if } \alpha m - \|\alpha m\| \in \mathbb{N}, \\ -\|\alpha m\| & \text{if } \alpha m + \|\alpha m\| \in \mathbb{N}. \end{cases}$$

Let us furthermore define the following open intervals:

$$A = (-2\varepsilon, -\varepsilon), \ B = (-\varepsilon, 0), \ C = (0, \varepsilon), \ D = (\varepsilon, 2\varepsilon). \tag{7}$$

Let m_1 be the smallest positive integer that satisfies $\langle m_1 \rangle \in C \cup D$, let m_2 be the the smallest positive integer that satisfies $\langle m_2 \rangle \in A \cup B$ and let $m_3 = m_1 + m_2$.

The first part of the proof is to show that there is always at least one integer (m_1, m_2 or m_3) that adding to an arbitrary $k_i \in \Omega$ always produces a new integer $k_j \in \Omega$. Clearly, $\langle k_i \rangle \in B \cup C$ for all k_i. Let us see the following cases:

$\langle k_i \rangle \in B$:

If $\langle m_1 \rangle \in C$, $\langle m_2 \rangle \in A \cup B$ then $\langle k_i + m_1 \rangle \in B \cup C$.

If $\langle m_1 \rangle \in D$, $\langle m_2 \rangle \in A$ and $\langle m_1 + m_2 \rangle \in C$ then $\langle k_i + (m_1 + m_2) \rangle \in B \cup C$.

If $\langle m_1 \rangle \in D$, $\langle m_2 \rangle \in A$ and $\langle m_1 + m_2 \rangle \in B$ then $\langle k_i + (m_1 + m_2) \rangle \in A \cup B$.
If $\langle k_i + (m_1 + m_2) \rangle \in A$ then $\langle k_i + (m_1 + m_2) - m_2 \rangle \in B \cup C$.

If $\langle m_1 \rangle \in D$, $\langle m_2 \rangle \in B$ and $\langle m_1 + m_2 \rangle \in C$ then $\langle k_i + (m_1 + m_2) \rangle \in B \cup C$.

If $\langle m_1 \rangle \in D$, $\langle m_2 \rangle \in B$ and $\langle m_1 + m_2 \rangle \in D$ then $\langle k_i + (m_1 + m_2) \rangle \in C \cup D$.
If $\langle k_i + (m_1 + m_2) \rangle \in D$ then $\langle k_i + (m_1 + m_2) - m_1 \rangle \in B \cup C$.

$\langle k_i \rangle \in C$:

If $\langle m_1 \rangle \in C \cup D$, $\langle m_2 \rangle \in B$ then $\langle k_i + m_2 \rangle \in B \cup C$.

If $\langle m_1 \rangle \in C$, $\langle m_2 \rangle \in A$ and $\langle m_1 + m_2 \rangle \in B$ then $\langle k_i + (m_1 + m_2) \rangle \in B \cup C$.

If $\langle m_1 \rangle \in C$, $\langle m_2 \rangle \in A$ and $\langle m_1 + m_2 \rangle \in A$ then $\langle k_i + (m_1 + m_2) \rangle \in A \cup B$.
If $\langle k_i + (m_1 + m_2) \rangle \in A$ then $\langle k_i + (m_1 + m_2) - m_2 \rangle \in B \cup C$.

If $\langle m_1 \rangle \in D$, $\langle m_2 \rangle \in A$ and $\langle m_1 + m_2 \rangle \in B$ then $\langle k_i + (m_1 + m_2) \rangle \in B \cup C$.

If $\langle m_1 \rangle \in D$, $\langle m_2 \rangle \in A$ and $\langle m_1 + m_2 \rangle \in C$ then $\langle k_i + (m_1 + m_2) \rangle \in C \cup D$.
If $\langle k_i + (m_1 + m_2) \rangle \in D$ then $\langle k_i + (m_1 + m_2) - m_1 \rangle \in B \cup C$.

Let now $X = \Delta_\Omega \setminus \{m_1, m_2, m_3\}$. We claim that $X = \varnothing$. Suppose otherwise, and let j be the smallest index with $m = k_{j+1} - k_j \in X$. Clearly, $\langle m \rangle \in A \cup B \cup C \cup D$. We can observe as well that for all $m \in \mathbb{N}$, $k_i \in \Omega$, $\langle k_i + m \rangle \in B \cup C$ implies $\langle m \rangle \in A \cup B \cup C \cup D$. Then it is easy to see that

- $j > 1$, and k_i's are integer linear combinations of m_1 and m_2 for all $i \leq j$,
- $m_1, m_2 < m < m_1 + m_2$,
- $\langle m \rangle \in A \cup D$.

If $\langle m \rangle \in A$ then $\langle m - m_2 \rangle \in B \cup C$, which contradicts the mimimality of j. In the same way, if $\langle m \rangle \in D$ then $\langle m - m_1 \rangle \in B \cup C$, which is a contradiction again. Hence, such an m does not exist. The proof is complete. □

Finding the integers m_1 and m_2 can be done very effectively with the theory of intermediate convergents. It was already discussed that intermediate convergents of an irrational α always produce the best upper and lower approximations to α, so m_1 and m_2 must be intermediate convergents.

Example 5 (cont.) *Applying the* ***FindMMM*** *algorithm (Algorithm 1) we have the values*
$m_1 = 59341817924539925$,
$m_2 = 24580185800219268$,
$m_3 = 83922003724759193$.
After the precalculation of m_1 *and* m_2 *it is very easy to compute every* k_i *between* 10^{20} *and* 10^{21}. *First we have to find an intermediate convergent between* 10^{20} *and* 10^{21}. *It can be done in polynomial time with the theory of continued fractions (e.g:* 233806732499933208099*). After that we can add, subtract* m_1, m_2 *or* m_3 *until we reach the bounds of the interval. The Weyl equidistribution theorem predicts* 18000 *integers that solve* (3). *Applying* ***Challenge 1 Solver***

algorithm (Algorithm 2) we found exactly $18\,000$ *integers. The precalculation and the computation of all* k_i *values took only* 31 *ms.*

Algorithm 1 FindMMM

Description:

The algorithm is based on Theorem 7. The algorithm finds the smallest m_1, m_2 and m_3 integers such that $0 < \langle m_1 \rangle < 2\varepsilon$, $-2\varepsilon < \langle m_2 \rangle < 0$. The output of the algorithm is $\Delta_\Omega = \{m_1, m_2, m_1 + m_2\}$. The main **while** loop in this algorithm (from line 5 to 15) goes through all intermediate convergents to find m_1 and m_2. The theory of intermediate convergents ensures that $m_1, m_2 \in q_i$ where q_i is the i^{th} intermediate convergent. When m_1 and m_2 are found the **while** loop terminates and the algorithm returns m_1,m_2 and $m_1 + m_2$ in ascending order.

Precondition: $\alpha \in \mathbb{R} \setminus \mathbb{Q}, \alpha > \varepsilon > 0$.

```
1:  procedure FindMMM(α, ε)
2:      i ← 0
3:      m1 ← 0
4:      m2 ← 0
5:      while m1 = 0 or m2 = 0 do
6:          i ← i + 1
7:          qi ← i^th intermediate convergents of α
8:          k ← Frac(qi · α)                    ▷ Fractional part of qi · α
9:          if m1 = 0 and k < 2ε then
10:             m1 ← qi
11:         end if
12:         if m2 = 0 and k > 1 − 2ε then
13:             m2 ← qi
14:         end if
15:     end while
16:     Return(min(m1, m2), max(m1, m2), m1 + m2)
17: end procedure
```

Algorithm 2 Challenge 1 Solver

Description:

The algorithm solves Challenge 1 (see (3)). Line 5 calls the FindMMM algorithm to determine Δ_Ω. With the theory of continued fractions line 6 finds an integer $k \in \Omega$. In the first **while** loop (lines $9-18$) the appropriate m_i is subtracted from k to generate a new integer $k_i \in \Omega$. The process is repeated until the lower bound A of the interval is reached. In the second **while** loop (lines $20-29$) the appropriate m_i is added to k generating $k_i \in \Omega$. The process is repeated until the upper bound of the interval B is reached. This method produces all the $18\,000$ integers that satisfy Challenge 1.

```
 1: x ← √2
 2: ε ← 10^−17
 3: A ← 10^20
 4: B ← 10^21
 5: ν ← FindMMM(x, ε)
 6: k ← Find q_x in the interval [A,B] where FRAC(q_x · x) < ε
 7: ktemp ← k
 8: PRINT(k)
 9: while k > A do
10:     for i = 1 → 3 do
11:         ok ← FRAC((k − ν[i]) · x)
12:         if (ok < ε) or (ok > 1 − ε) then k ← k − ν[i]
13:             if k > A then PRINT(k)
14:             end if
15:             break                                  ▷ Leave the for loop
16:         end if
17:     end for
18: end while
19: k ← ktemp
20: while k < B do
21:     for i = 1 → 3 do
22:         ok ← FRAC((k + ν[i]) · x)
23:         if (ok < ε) or (ok > 1 − ε) then k ← k + ν[i]
24:             if k < B then PRINT(k)
25:             end if
26:             break                                  ▷ Leave the for loop
27:         end if
28:     end for
29: end while
```

2.2 "Many elements" approximation

In some cases it is not necessary to find all the k_i elements of Ω, rather it is enough to find as much as possible within a given time unit. Then, the following procedure works:

Find the smallest integer x that satisfies $0 < \langle x \rangle < \varepsilon$ and find the smallest integer y that satisfies $-\varepsilon < \langle y \rangle < 0$. Using the notations (7) it is easy to see that if $\langle k_i \rangle \in B$ and $\langle x \rangle \in C$ then $\langle k_i + x \rangle \in B \cup C$. In the same way, if $\langle k_i \rangle \in C$ and $\langle y \rangle \in B$ then $\langle k_i + y \rangle \in B \cup C$. Only with these two integers it is always possible to produce a subset of Ω.

Example 5 (cont.) *If we want to determine just "many" elements of* Ω*, the previous method generates* 12945 *integers within* 15 *ms.*

3 Approximations for the multi-dimensional case

3.1 "Many elements" approximation

Calculating all-elements of Ω seems to be hard in higher dimensions. However, we can generalize our one dimensional method to find "many" $q \in \Omega$ integers recursively. The algorithm is based on the following lemma:

Lemma 8 *Let the irrationals* $\alpha_1, \alpha_2, \ldots, \alpha_n$ *and the real* $\varepsilon > 0$ *be given. Then there is a set* Γ_n *with* 2^n *elements with the following property: if* $q \in \Omega$ *then* $q + \gamma \in \Omega$ *for some* $\gamma \in \Gamma_n$*.*

Proof. Let $q \in \Omega$ be given. Let us define an n-dimensional binary vector b associated with q in the following way:

$$b_i = \begin{cases} 1 & \text{if } q\alpha_i - \|q\alpha_i\| \in \mathbb{N}, \\ 0 & \text{if } q\alpha_i + \|q\alpha_i\| \in \mathbb{N} \ . \end{cases} \tag{8}$$

Let Γ_n be the set for which

1. $\gamma \in \Gamma_n$ implies $\|q\alpha_i\| < \varepsilon$ for all $1 \le i \le n$,
2. all the associated binary representations by (8) are different.

Then, for a given $q \in \Omega$ there exists a $\gamma \in \Gamma_n$ such that $q + \gamma \in \Omega$, e.g. when their associated binary representations are (1's binary) complements. Clearly, $|\Gamma_n| = 2^n$. The proof is finished. □

356205059916	3487229338057	3565485794412	3921690854328	4576624903864
5800642344603	7056176493393	7134432949748	7490638009664	9054007777845
10977867347721	11591199235356	11889764427290	12324225943561	15811455281618
16900850847425	17257055907341	18046611831809	18152923635291	18647375728749
18725632185104	19081837245020	19380402436954	19814863953225	20686645377416
20960788735295	21721870790627	22050020503416	22945888231366	23302093291282
23957027340818	25181044781557	25537249841473	25822432016319	27522513135230
27878718195146	28790615274715	29102735635885	29703499532825	31938492285330
32712306129043	35925048224463	38160204774654	39113548572900	39113712370586
40202944138707	41949469020031	42383930536302	42438100688898	44262882026577
44423200184969	44619087086493	44917652278427	47693582148371	51437938314147
51794143374063	52092708565997	56669333469861	58494114807540	59583346575661
60430542368111	62415805661868	62714370853802	63070575913718	65007167271975
65305732463909	65383988920264	66992266768046	67564975317859	68559097897151
68871218258321	75394965654965	76540726639349	76718061327901	76975188155620
79615221701227	79971426761143	81850378251418	82152413158738	84031364649013
84387569708929	87953055503341	88607825755191	88686082211546	91522002658677
91562625996499	92173311549603	95131573151835	95443693513005	96098463764855
98618802489892	98697058946247	100932215496438	103273520052425	104419444834495
105152471542700	107689663000211	112821979923728	119085139131729	121320131884234
140045764069338	140401969129254	143970916284590	147101940562731	151379836476975
153024924062435	156512153400492	161661323056483	164002791410156	170838495370284
175415120274148	179814246691774	183383193847110	189974502767900	204205735548863
208621878496649	208998700144938	261026707423816	266621541210731	269101092800260
269457297860176	299828135635546	305300628267360	320949406326919	331272339625104
382947603204990	408250989141648	616256074389738		

Table 1: The result of the precalculation for solving Challenge 2

Remark 9 *Computing the appropriate* $\gamma \in \Gamma_n$ *for a given* $q \in \Omega$ *is not necessarily unique.*

Corollary 10 *Remember the first dimension case: For all* $m \in \mathbb{N}$, $q \in \Omega$, $\langle q + m \rangle \in B \cup C$ *implies that* $\langle m \rangle \in A \cup B \cup C \cup D$. *We can generalize this to higher dimensions. Let* $q \in \Omega$ *and* $m \in \mathbb{N}$ *be given. Then* $q + m \in \Omega$ *implies* $\|m \cdot \alpha_i\| \in A \cup B \cup C \cup D$ *for all* $1 \leq i \leq n$.

Unfortunately, the precalculation of the 2^n integers is in general computationally expensive. However, there are several tricks based upon Lemma 8 that can be applied to make the generation more efficient.

Example 5 (cont.) *In Challenge* 2 *the precalculation of the* $2^7 = 128$ *integers took approximately* 6.14 *sec on our architecture. Table 1 shows the result. Applying the* ***Challenge*** 2 ***Solver*** *we were able to produce* 120852 *integers in* Ω *within* 26.8 *sec.*

Algorithm 3 Challange 2 Solver

Description:

The algorithm answers Challenge 2 (see (4)). Line 5 calls the PRECALC algorithm in order to determine the 2^n integers. The **while** loop generates a new integer in Ω using the precalculated ones. The method produces 120 852 integers that satisfy Challenge 2.

```
1:  n ← 7
2:  X ← log(p)/log(2), p prime, 3 ≤ p ≤ 19
3:  ε ← 0.01
4:  B ← 10^18
5:  ν ← PRECALC(n, ε, X, 2^12)
6:  k ← 0
7:  while k < B do
8:      for i = 1 → length(ν) do
9:          t ← TRUE
10:         for j = 1 → n do
11:             ok ← FRAC((k + ν[i]) · X[j])
12:             if (ok > ε) and (ok < 1 − ε) then
13:                 t ← FALSE
14:                 break                          ▷ Leave the for loop
15:             end if
16:         end for
17:         if t = TRUE then
18:             k ← k + ν[i]
19:             if k < B then
20:                 PRINT(k)
21:             end if
22:             break
23:         end if
24:     end for
25: end while
```

Algorithm 4 Reduce

Description: The algorithm reduces the generation time of Γ_n in the **Precalc** algorithm with adding new elements to K. In this algorithm K is a list of integers and X is a set of irrationals such that $\|K[i] \cdot X[j]\| < \varepsilon$ for all i and for all $j < n$. The main part of the algorithm is the for loop (lines $4-9$). Each element of K is subtracted (added) from (to) every element of K and the new integer k_i that satisfies $\|k_i \cdot X[j]\| < \varepsilon$ for all $j < n$ are appended to K.

Precondition: K: set of integers,$n \in \mathbb{N}$, $\varepsilon > 0$, X: set of irrationals

```
1:  procedure REDUCE(K, n, ε, X)
2:      SORT(K)                          ▷ Sorting, every element ouccurs only once
3:      M ← dynamic array()
4:      for i = 1 → length(K) do
5:          for j = 1 → length(K) do
6:              APPEND(M, abs(K[i] − K[j]))     ▷ append abs(K[i] − K[j]) to M
7:              APPEND(M, abs(K[i] + K[j]))
8:          end for
9:      end for
10:     SORT(M)
11:     for i = 1 → length(M) do
12:         t ← TRUE
13:         for j = 1 → n do
14:             t ← t and (FRAC(M[i]·X[j]) < 2ε or FRAC(M[i]·X[j]) > 1−2ε)
15:         end for
16:         if t = FALSE then
17:             DELETE(M[i])                ▷ Delete the i^th element of M
18:         end if
19:     end for
20:     APPEND(K, M)                        ▷ Append array M to K
21:     SORT(K)
22:     if K[1] = 0 then
23:         DELETE(K[1])                    ▷ Delete the zero value from K
24:     end if
25:     RETURN(K)
26: end procedure
```

Algorithm 5 Precalc

Description: The algorithm is based on **Lemma** 8. It generates Γ_n, a subset of Δ_Ω. In dimension n the set Γ_n contains exactly 2^n elements. Initially (line 2), the FindMMM algorithm is used. In higher dimensions ($2, 3, \ldots$ up to m) the algorithm produces many integers from Δ_Ω by which Γ_n can be generated. M is a matrix with i rows. The i^{th} row contains the binary representation of i. (Note: the size of M is changing depending on the dimension.) To produce as many integers as possible the **Reduce** algorithm is used (see lines 10, 11). If β goes to infinity then Δ_Ω should contain allmost all possible step-sizes, not just some. To solve Challenge 2, we set $\beta = 2^{12}$. With this choice of β the algorithm is able to generate the appropriate Γ_n up to 10 dimensions. For higher dimensions bigger β is needed.

```
 1: procedure PRECALC(m, ε, X, β)
 2:     T ← FINDMMM(X[1], ε)                          ▷ T is a dynamic array
 3:     for n = 2 → m do
 4:         T2 ← dynamic array()
 5:         N ← 0, T3 ← 0     ▷ N, T3 are arrays with 2^n elements, every element is 0
 6:         M ← 2^n × n matrix, the i^th row contains the binary representation of i
 7:         k ← 0, tmp ← 0, l ← 0, number ← 0
 8:         while TRUE do
 9:             if l = 2^n and number > β then
10:                 REDUCE(T2, n, ε, X)
11:                 REDUCE(T2, n, ε, X)
12:                 T ← T2
13:                 break                             ▷ Leave the while loop
14:             end if
15:             for i = 1 → length(T) do
16:                 t ← TRUE
17:                 for j = 1 → n − 1 do
18:                     ok ← FRAC((k + T[i]) · X[j])
19:                     if ok > ε and ok < 1 − ε then
20:                         t ← FALSE
21:                         break                     ▷ Leave the for loop
22:                     end if
23:                     if t = TRUE then
24:                         k ← k + T[i]
25:                         break                     ▷ Leave the for loop
26:                     end if
27:                 end for
28:             end for
```

Algorithm 6 Precalc (contd.)

```
            number ← number + 1
            t ← TRUE
            for j = 1 → n do
                t ← t and (FRAC(k · X[j]) < ε or FRAC(k · X[j]) > 1 − ε)
            end for
            if t = FALSE then
                next
            end if
            t ← FALSE
            for i = 1 → length(T2) do
                if T2[i] = k − tmp then
                    t ← TRUE
                end if
            end for
            if t = FALSE then
                APPEND(T2, k − tmp)              ▷ append k − tmp to the array T2
            end if
            tmp ← k
            for i = 1 → 2^n do
                t ← TRUE
                for j = 1 → n do
                    if M[i, j] = 0 then
                        t ← t and (FRAC(k · X[j]) < ε)
                    else
                        t ← t and (FRAC(k · X[j]) > 1 − ε)
                    end if
                end for
                if t and N[i] = 0 then
                    N[i] ← 1
                    l ← l + 1
                    T3[l] ← k
                    if l = 2^n and n = m then
                        break(2)                 ▷ Leave while loop
                    end if
                end if
            end for
        end while
    end for
    RETURN(T3)
end procedure
```

4 Practical use of our methods

The real power of the presented methods is the ability to use them in a distributed way.

There are several fields of mathematics where the techniques shown in this paper can be applied. We used our methods in order to find high peak values of the Riemann-zeta function effectively. It is computationaly hard to find real t values where $|\zeta(1/2+it)|$ is high (see [11]). In 2004 Tadej Kotnik observed that large values of $|\zeta(1/2+it)|$ are expected when $t = \frac{2k\pi}{\log 2}$, where $k\frac{\log(p_i)}{\log(2)}$ are close to an integer for all primes $p_i > 2$ [6]. The methods shown in this paper can be used to find thousands of candidates within a few minutes where high values of $|\zeta(1/2+it)|$ are expected. We plan to continue our research in this direction.

5 Acknowledgement

The authors would like to thank Prof. Dr. Antal Járai for his very helpful comments, suggestions and to the anonymous reviewers for many constructive comments. The research of the first author was partially supported by the European Union and co-financed by the European Social Fund (ELTE TÁMOP-4.2.2/B-10/1-2010-0030).

References

[1] F. Armknecht, C. Elsner, M. Schmidt, Using the Inhomogeneous Simultaneous Approximation Problem for Cryptographic Design. *AFRICACRYPT*, 2011, pp. 242–259. ⇒18

[2] A. Frank, É. Tardos, An application of simultaneous Diophantine approximation in combinatorial optimization, *Combinatorica*, **7,** 1 (1987) 49–66. ⇒18

[3] A. Y. Khinchin, *Continued Fractions*, Translated from the third (1961) Russian edition, Reprint of the 1964 translation, Dover, Mineola, NY, 1997. ⇒20

[4] Sh. Kim, S. Östlund, Simultaneous rational approximations in the study of dynamical systems, *Phys. Rev. A*, **34,** 4 (1986) 3426–3434. ⇒18

[5] C. Kimberling, Best lower and upper approximates to irrational numbers, *Elem. Math.*, **52,** 3 (1997) 122–126. ⇒20

[6] T. Kotnik, Computational Estimation of the order of $\zeta(1/2+it)$, *Math. Comp.*, **73,** 246 (2004) 949–956. ⇒33

[7] J. C. Lagarias, Best simultaneous Diophantine approximations I., Growth rates of best approximation denominators, *Trans. Am. Math. Soc.*, **272,** 2 (1982) 545–554. ⇒18

[8] J. C. Lagarias, Best simultaneous Diophantine approximations II., Behavior of consecutive best approximations, *Pacific J. Math.*, **102,** 1 (1982) 61–88. ⇒18
[9] J. C. Lagarias, The computational complexity of simultaneous Diophantine approximation problems, *SIAM J. Computing* **14,** 1 (1985) 196–209. ⇒18
[10] A. K. Lenstra, H. W. Lenstra Jr., L. Lovász, Factoring polynomials with rational coefficients, *Math. Ann.*, **261,** 4 (1982) 515–534. ⇒18, 21
[11] A. M. Odlyzko, The 10^{20}-th zero of the Riemann zeta function and 175 million of its neighbors, 1992 (unpublished) ⇒33
[12] V. T. Sós, G. Szekeres, Rational approximation vectors, *Acta Arithm.*, **49,** 3 (1988) 255–261. ⇒18

Received: April 10, 2013• Revised: June 8, 2013

Acta Univ. Sapientiae, Informatica, **5,** 1 (2013) 35–52

Finding suitable paths for the elliptic curve primality proving algorithm

Antal JÁRAI
Eötvös Loránd University
Faculty of Infomatics
email: ajarai@moon.inf.elte.hu

Gyöngyvér KISS
Eötvös Loránd University
Faculty of Infomatics
email: kissgyongyver@gmail.com

Abstract. An important part of the Elliptic Curve Primality Proving algorithm consists of finding a sequence of elliptic curves with appropriate properties. In this paper we consider a strategy to search for an improved sequence, as part of an implementation (implemented in Magma 2.19) to obtain improved heuristics and compare it to an implementation which does not use such heuristics, namely to a built-in Magma function.

1 Introduction

Although mathematicians have been interested in prime numbers since ancient times, there is still no general, deterministic, unconditional, practical, polynomial time algorithm for primality proving. If we are willing to drop some of these adjectives, the situation becomes different. There exist tests of Lucas-Lehmer type that can certify primes of very large size but only of a special form. The Miller-Rabin test has a version that is practical and runs in polynomial time but only provides primality proofs conditional on a generalized version of the Riemann hypothesis; the variant commonly used only produces *probable primes*, in the sense that with small probability a composite number will pass the tests. The now famous AKS test [1], on the other hand, is deterministic and proves primality in polynomial time, but has yet to be proven practical; for an improved randomized version see Bernstein [3].

Computing Classification System 1998: F.2.1, G.4
Mathematics Subject Classification 2010: 11Y11, 11A51, 14H52, 97R20
Key words and phrases: ECPP, elliptic curves, primality proving, Magma

Somewhere in between there are two algorithms that can prove primality in situations of practical importance (primes of hundreds or several thousands of decimal digits), of which the complexity analysis shows sub-exponential dependency on the size of the prime, but for which polynomial time bounds have not been proven. The significance of such primality tests has increased with the widespread use of primes for cryptographic purposes.

This paper aims to describe an implementation of one of the two successful practical tests for primality proving, *ECPP* see [2], written in Magma, a high performance software system. The test is based on elliptic curve arithmetic, by looking at heuristics for an optimal choice of parameters and next step in the recursion and to compare it to an implementation without heuristics, which is a part of Magma.

In what follows, we will always assume that n is the input of our algorithm, for which we want to construct a primality proof; also, we assume that n is a probable prime in the sense that it has passed some compositeness tests, and that it is free of small divisors. In particular, $\gcd(n, 6) = 1$. However of course, we do not *assume* that n is prime.

2 Elliptic curves

The main objective in the Elliptic Curve Primality Proving (*ECPP* for short) algorithm, which will be described in detail in the next section, is to construct a sequence of integers $n_0, n_1, \ldots, n_k$ that will be proved prime in reversed order, ending at $n_0 = n$. When the proof is completed, these numbers n_i will be (divisors of) orders of groups of points of elliptic curves over finite fields, as they are defined modulo n_{i-1}. However, during the construction we can not use yet that n_{i-1} is prime, and this means that we will have to be careful in defining elliptic curves modulo n, and their arithmetic; see [8].

Definition 1 *The projective plane modulo* n*, denoted* $\mathbb{P}^2(\mathbb{Z}/n\mathbb{Z})$*, for a positive integer* n*, consists of equivalence classes* $(x : y : z)$ *of triples* $(x, y, z) \in (\mathbb{Z}/n\mathbb{Z})^3$ *satisfying* $\gcd(x, y, z, n) = 1$*, under the equivalence* $(x, y, z) \sim (\lambda x, \lambda y, \lambda z)$ *for any* $\lambda \in (\mathbb{Z}/n\mathbb{Z})^*$.

Definition 2 *Let* n *be an integer with* $\gcd(n, 6) = 1$*. An elliptic curve* E *modulo* n *is a pair* $(a, b) \in (\mathbb{Z}/n\mathbb{Z})^2$ *for which* $\gcd(4a^3 + 27b^2, n) = 1$*. The set of points* $E[\mathbb{Z}/n\mathbb{Z}]$ *on an elliptic curve* E *modulo* n *consists of* $(x : y : z) \in \mathbb{P}^2(\mathbb{Z}/n\mathbb{Z})$ *for which*

$$y^2z = x^3 + axz^2 + bz^3.$$

Definition 3 *Let* n *be an integer with* $\gcd(n, 6) = 1$*, and* $a \in \mathbb{Z}/n\mathbb{Z}$*. Define* $V = V[\mathbb{Z}/n\mathbb{Z}]$ *as the set of all* $(x : y : 1) \in \mathbb{P}^2(\mathbb{Z}/n\mathbb{Z})$ *together with* $O = (0 : 1 : 0) \in \mathbb{P}^2(\mathbb{Z}/n\mathbb{Z})$*. Given* (V, a)*, the partial addition algorithm computes for any pair* $P = (x_p : y_p : z_p)$*,* $Q = (x_q : y_q : z_q) \in V$ *either an element* $R = (x_r, y_r, z_r) \in V$ *called the sum* $P + Q$ *of* P *and* Q*, or a non-trivial divisor* d *of* n*, as follows.*

(1) If $x_p = x_q$ *and* $y_p = -y_q$*, then output* $R = (0 : 1 : 0)$*.*

(2) If $x_p \neq x_q$ *and* $y_p = -y_q$*, then let* $\nu = x_p - x_q$*, otherwise let* $\nu = y_p + y_q$*; then use the extended Euclidean algorithm to compute* $s, t \in \mathbb{Z}/n\mathbb{Z}$ *such that* $s\nu + tn = d = \gcd(\nu, n)$*. If* $d > 1$ *then output* d*.*

(3) Let $\lambda = s(y_p - y_q)$ *if* $x_p \neq x_q$ *and* $\lambda = s(3x_p^2 + a)$ *if* $x_p = x_q$*. Output* $R = (\lambda^2 - x_p - x_q : \lambda(\lambda^2 - 2x_p - x_q) + y_p : 1)$*.*

Remark 1 *If* $n = p$ *is prime, the set* $E[\mathbb{Z}/p\mathbb{Z}]$ *forms an Abelian group for any elliptic curve* $E = E_{a,b}$*, with unit element* O*. In this case, the partial addition algorithm, which will now always produce a sum of two points on* E*, is equivalent to the usual addition algorithm.*

Moreover, it can be shown that for a prime divisor p *of arbitrary* n *coprime to* 6*, the sum* R *produced by the partial addition algorithm for any two points* P, Q *on an elliptic curve* $E_{a,b}$ *modulo* n*, has the property that* R_p *(obtained by reducing the coordinates of* R *modulo* p*) is the sum of (the similarly defined) points* P_p *and* Q_p *in the group* $E_{\bar{a},\bar{b}}[\mathbb{Z}/p\mathbb{Z}]$*, where* $\bar{a} \equiv a \bmod p$*, and* $\bar{b} \equiv b \bmod p$*.*

Using the partial addition algorithm repeatedly, it is of course possible to obtain a partial multiplication algorithm, which computes either $k \cdot P$ or finds a divisor of n, for any positive integer k, given any $P \in V$ and any a as before. However, there are various ways to speed up this computation of $k \cdot P$, using partial doubling, and the fact that it is not necessary to keep track of the y-coordinate.

In the next sections we will occasionally be sloppy, and write about the sum and multiples of points on elliptic curves modulo n; we mean the result of application of the partial addition and multiplication algorithms, which in exceptional cases means that a divisor of n is found, rather than a point.

3 Elliptic Curve Primality Proving

We give an outline of the Elliptic Curve Primality Proving algorithm; some of the necessary definitions and details will be given in the subsequent sections. The algorithm is based on the following theorem.

Theorem 2 *Let* $n_0 \in \mathbb{N}$ *with* $\gcd(6, n_0) = 1$. *Let* E *be an elliptic curve modulo* n_0, *and let* $m, n_1 \in \mathbb{N}$ *with* $n_1 \mid m$. *Suppose that for every prime factor* q *of* n_1 *there exists* $P \in E$ *such that* $mP = 0_E$ *and* $\frac{m}{q}P \neq 0_E$. *Then for all prime factors* p *of* n_0 *holds* $\#E[\mathbb{Z}/p\mathbb{Z}] \equiv 0 \bmod n_1$.

Corollary 3 *Suppose that the hypotheses of Theorem 2 are satisfied. Then:*

$$n_1 > (n_0^{\frac{1}{4}} + 1)^2 \quad \Rightarrow \quad n_0 \text{ is prime.}$$

Note that the requirement is that n_1 exceeds a bound slightly larger than $\sqrt{n_0}$.

Essential in the proof of the Corollary is the Theorem of Hasse, stating that the number of points on any elliptic curve modulo a prime p equals $p+1-t$ for some integer t with $|t| \leq 2\sqrt{p}$. Theorem 2 easily follows from the observation that, modulo any prime divisor p of n_0 the conditions imply that $\#E[\mathbb{Z}/p\mathbb{Z}]$ can not be a proper divisor of n_1.

Starting point for the application of *ECPP*, will always be a probable prime $n_0 = n$; it is assumed that n will be free of small prime factors (in particular 2 and 3), and that n has passed certain compositeness tests (of Miller-Rabin type). This will make it very likely that n is indeed prime; the objective is to prove that.

Given such an integer n, the basic Elliptic Curve Primality Proving algorithm proceeds roughly in these three stages:

(D) starting with $n_0 = n$, find a sequence of probable primes $n_0, n_1, \ldots, n_k$, such that n_{i+1} divides the order of some elliptic curve modulo n_i, such that $n_{i+1} > (\sqrt[4]{n_i} + 1)^2$, and such that n_k is so small that primality can be verified by easy inspection (or trial division).

(F) For each of the integers n_i with $i = 0, 1, \ldots, k-1$, construct an elliptic curve E_i of order a multiple of n_{i+1} modulo n_i, together with a point P_i of order n_{i+1} on the curve modulo n_i.

(P) Verify that the conditions of Theorem 2 hold for the given probable primes n_i, curves E_i and points P_i, for $i = k-1, k-2, \ldots, 0$.

The original idea came from Goldwasser and Kilian, who designed such an algorithm, which uses random elliptic curves over the integers n_i, computes the order of them and factors the orders. Computing the order of a random elliptic curve over n_i is very cumbersome. It is yet a faster way to determine the curve order first and construct a curve with such order. Besides, we get two elliptic curves for each integers, that increases the possibility of success. m order has to be selected from the algebraic integer of an imaginary quadratic field $\mathbb{Q}(\sqrt{D})$. D is a *negative fundamental discriminant*, and as such, it has certain properties: $D \equiv 0 \pmod 4$, or $D \equiv 1 \pmod 4$, for every k (> 1) D/k^2 is not a fundamental discriminant, $D \leq 0$ (from practical point of view we use $D \leq -7$). Moreover D must be appropriate for n, which means: $(D|n) = 1$, where $(D|n)$ is the Jacobi symbol and there exist such $x, y \in \mathbb{Z}$ for which

$$4n = (2x + yD)^2 - y^2D. \tag{1}$$

An appropriate D provides two possible orders: $m = |\nu \pm 1|^2$, where

$$\nu = x + y\frac{D + \sqrt{D}}{2}.$$

If (1) is valid, with the help of an x_0 root of the *Hilbert polynomial* $\pmod n$ we get two elliptic curves with order $m = |\nu \pm 1|^2$. Refer to [2]

We will refer to this algorithm as *ECPP* in the rest of the paper.

3.1 Downrun

The *ECPP* algorithm consists of two parts. The first part of the algorithm will be called recursively with input n_i. The main objective is to find n_{i+1}. This is what happens at level i:

(D) select a pair D, m of negative discriminant D and integer $m = m_{i+1}$ such that m is the product of small primes and a probable prime n_{i+1} that exceeds $(\sqrt[4]{n_i} + 1)^2$.

In practice this is what happens:

(D_0) Prepare a list of primes up to some bound $s = s_i$, as well as list of negative fundamental discriminants up to a bound $d = d_i$ that factor completely in a product of primes from the prime list, together with their full prime factorization.

(D_1) Select one discriminant D in the list, find the reduction of the binary quadratic form $Ax^2 + Bxy + Cy^2$ of discriminant D, where $A = n$, $B^2 \equiv$

$-D \bmod n$, and $C = (B^2 + D)/(4n)$. This requires the modular square root of $-D$ modulo n, which is obtained as a product of the square roots of the prime factors of $-D$. If this provides ν with $|\nu|^2 = n$, then $m_1 = |\nu - 1|^2$, $m_2 = |\nu + 1|^2$.

(D_2) From the two pairs D, m_1, D, m_2 found in the previous step, for which a probably prime q_1 dividing $m_1 = |\nu - 1|^2$ can be found such that m_1/q_1 is the product of small primes only, similarly for m_2, one is selected.

(D_3) Let n_{i+1} be the probable prime $q = q_1$ or q_2, for which $m = m_1$ or m_2, according to the selection in the previous step, m/q is the product of small primes, if that satisfies the conditions, otherwise select another D from the list.

Several comments are in order.

Usually a 'master-list' of primes up to some bound B is prepared in advance; the bound s_i (and hence the list) in step (D_0) may depend on i (the level of the recursion arrived at), but should be at most B. Similarly for the list of discriminants, and the bound d_i. This means that step (D_0) will mainly consist of the selection of sub-lists, from precompiled lists that are computed once for all n up to a fixed size N. In Step (D_2) the probable factorization of possible curve order m_i has to be found; one uses a smoothness bound $b = b_i$, that is, all prime factors smaller than b are removed (and considered small).

Note that backtracking may be necessary: it is possible that at some level no D provides a new n_i!

The output of the first phase of the algorithm will consist of a triple (n_i, D_i, m_i) for $i = 0$ to $i = k - 1$ such that m_i is the product of small primes and a probable prime n_{i+1} that exceeds $(\sqrt[4]{n_i} + 1)^2$.

3.2 Finding elliptic curves

The second phase is executed after the recursive call of the first part, which results in a list of n_i, D_i, m_i such triples, where n_i is the input of the recursive step produces n_{i+1}. This phase in the primality testing algorithm can be done as follows. Again, we describe the steps to be taken at level i.

(F) Find elliptic curves E_i and points P_i on $E_i[\mathbb{Z}/n_{i-1}\mathbb{Z}]$ with the property that if n_{i-1} is prime, then the order of P_i is n_i.

This is done as follows.

(F_0) Compute an auxiliary polynomial $G_i \in \mathbb{Z}[x]$; see the comments below.

(F_1) Find a root j_i of G_i mod n_{i-1} in $\mathbb{Z}/n_{i-1}\mathbb{Z}$, as well as an integer t_i such that the Jacobi symbol $\left(\frac{t_i}{n_{i-1}}\right)$ equals -1.

(F_2) Define elliptic curves E_i' and E_i'' by

$$E_i' : \quad y^2 = x^3 + 3kx + 2k$$

and

$$E_i'' : \quad y^2 = x^3 + 3kt_i^2x + 2kt_i^3,$$

where $k = \dfrac{j}{1728 - j}$.

(F_3) Find (for example by randomly choosing) a point on E_i' or E_i'' that has order n_i, if n_{i-1} is prime.

Remark 4 *The auxiliary polynomial* G_i *is the Hilbert (or Weber) polynomial or a variant of this. The two elliptic curves are the twists of the elliptic curve with* j*-invariant* j_i*. We refer to [2] and [8] for more details, as this part of the algorithm plays no major role in what follows.*

4 Magma

Magma [4] is a large computer algebra system, with high-performance computations in number theory as one of its specializations, including very advanced integer factorization and primality proving algorithms. Since its launch (London, August 1993) a large body of intrinsic functions (implemented in the C language), have been supplemented by packages developed on top of this, making use of the Pascal-like user language and the programming environment that is provided.

4.1 Magma-ECPP

Magma has a built-in primality test, which uses a combination of the Miller-Rabin compositness test, and *ECPP*. It can be invoked by

`IsPrime(n:` *parameter*`) : RngIntElt` $\mapsto$ `BoolElt`

function. By default, this function proves primality using *ECPP* (after a quick test to throw out composites), but it is possible to set the optional Boolean parameter `Proof` to FALSE, in which case the function only uses the probabilistic

Miller-Rabin test, with the default number of bases (20). In the rest of the paper we refer to the function in Magma v2.19 (December 2012) as Magma-ECPP. We would like to compare our *ECPP* implementation, Modified-ECPP to this function.

This function is based on F. Morain's implementation of *ECPP* in the C language, and works more or less according to the description above as one can tell from the verbose printing (although the details of the source code are hidden from the user). It has a list of base discriminants in a file, likely fully factored and ordered by the value of their field h, where it loops through in each iteration. In the ith iteration it first applies a trial division sieve on n_i with bound `Pmax`, then on each discriminant, it performs (D) until it either finds n_{i+1} or runs out of discriminants. In the case of success it goes into the $(i+1)$th iteration with the new input; in the second case it loops through the same set of discriminants, but applies stronger factoring methods to each of the numbers m_i. It applies trial division, Pollard's ρ and Pollard's $p-1$ in the first round and if (D) is not successful, supposedly ECM is used too. If the second round still yields no n_{i+1}, it has to backtrack to input n_{i-1}. As after each iteration the information on which discriminants it succeeded is stored, on backtracking it starts from the next discriminant in the list.

4.2 Modified-ECPP

In the Modified-ECPP version, one does not abort executing the ith iteration as soon as a good discriminant is found, but only after a certain range of discriminants are scanned, thus we could gain a *range* of $n_{i+1,j}$'s instead of a single one in each (recursive) call of step (D), from which the input n_{i+1} for the next call is to be selected. Choosing one n_{i+1} out of the range of possibilities can be done based on criteria depending on certain properties of the numbers.

4.2.1 Theoretical observations

The following observation plays an important role in the running time analysis: if we are able to find all prime factors of curve cardinalities m_{i+1} up to a bound b_i in the ith iteration, the probability that one such curve cardinality m_{i+1} leads to a new node will be the probability that the second largest prime factor of m_{i+1} is less then b_i; this is approximately

$$e^{\gamma}\frac{\log b_i}{\log n_i}.$$

It is then reasonable to suppose that, if we have e_i such curve orders m_{i+1}, the number of new nodes has a probability distribution with average approximately

$$\lambda_i = e^{\gamma} \frac{\log b_i}{\log n_i} e_i.$$

Refer to [7].

For each negative discriminant $D \leq -7$ the probability of success is

$$\frac{1}{2h(D)},$$

where $h(D)$ is the ideal class number of $\mathbb{Q}(\sqrt{D})$. As each successful case results in 2 m_{i+1}'s, we expect

$$e_i \approx \sum_D \frac{1}{h(D)}.$$

Refer to [2].

4.2.2 The tree structure

The process of *ECPP* may be envisaged as choosing a path through a (directed) tree of which the nodes represent probable primes. (Note that strictly speaking the graph is not a tree, as it is possible, but not really likely that different paths lead to the same node!) The root of this tree is n_0, the leaves correspond to probable primes that are small enough to be recognized as primes by some direct method. The aim is to find a relatively short and "easy" path from the root n_0 to a leaf n_k as fast as possible; in particular, one would like to avoid computing too many nodes explicitly.

By the latter we mean that we store certain information with the nodes that we compute explicitly: for possible descendants $n_{i,j}$, $j = 1, \ldots, t_i$ of n_i, we store some information to base our choice of n_{i+1} on it. (For the rest of the paper with n_{i+1} we mean the selected $n_{i,j}$.) This includes the value $n_{i,j}$, as well as $s_{i,j}$, $d_{i,j}$ and $b_{i,j}$, (respectively: the smoothness bound for the discriminants, the bound on the size of the discriminants and the smoothness bound on the curve order), the level i in the tree and a parameter measuring the *suitability*. The choice of node n_{i+1} is based on this *suitability* parameter, which is determined when the node is produced. The value of the *suitability* is initialized to the value of the field $d_{i,j}$ stored with the nodes. Backtracking is unfavorable as, besides the useless work decreasing the level in the tree, the size of the primes in the nodes is likely to increase, therefore there is a fixed, global penalty value

p added to the *suitability* of each node when a new level starts up, except for the nodes of the new level.

The idea is that the computations for smaller discriminants will be cheaper, and hence the algorithm will be completed faster. The field $d_{i,j}$ that gives the initial value of *suitability* of the node, is determined based on the function λ, searching for the minimal power of $d_{i,j} = \log(n_{i,j})^{\delta_{i,j}}$ allowing the value of λ to exceed a certain bound given $s_{i,j}$, $b_{i,j}$. The power $\delta_{i,j}$ is stored as the initial value of *suitability*. The nodes are stored in an array, and a certain penalty p is added, if necessary. The value of p depends on how hard we want to punish backtrack steps. Order by *suitability* and select the smallest one and call algorithm (T) recursively with that node as input. If (T) is successful the new nodes are added to the array and the sorting process starts again. If there is no new node found the value of *suitability* and the field $d_{i,j}$ of the selected node is increased. The power $d_{i,j}$ can reach a given bound where the node falls out from the array of possible nodes. The nodes are reordered. Repeat this procedure, until the size of the nodes reaches a limit which is small enough to recognize the prime.

4.2.3 The path finding algorithm

The path finding algorithm (T) in the 'tree' then has three main stages:

(T_0) Step (D) is being applied for n_0 looking for the minimal choice of $d(n_0) = \log(n_0)^{\delta_0}$ for which D is successful with a kind of brute force strategy, using a loop in which the value of d_0 is incremented until there exists at least one descendant $n_{1,j}$, as the next step requires a non-empty list of $n_{1,j}$'s. This part is called just once at the beginning.

The next steps are repeated for $i = 1, 2, \ldots, k$, where n_k is the first probable prime that can be proven prime directly.

(T_1) Keeping the value of $\lambda = \lambda_{i,j}$ above 1.5, determine the value of $d_{i,j}$ for given $s_{i,j}$, $b_{i,j}$ for each newly found $n_{i,j}$, and store a list of (at most 100 of) the best values according to *suitability*. Sort the list of the best hundred nodes and select the best as n_{i+1}.

(T_2) Apply Step (D) again on n_{i+1}, with the parameter $d_{i+1} = \log^{\delta_{i+1}}(n_{i+1})$; if no new node is found increase the δ_{i+1} by 0.1, repeat ordering and selection until at least one new $n_{i+1,j}$ is found. Once Step (D) is successful, go back to (T_1) with the new list of nodes as input.

5 Experiments

In this section we are going to point out some typical situations that occur during the process of *ECPP* and that are handled differently by Magma-ECPP and Modified-ECPP. We describe the effects of these differences with respect to outcome and running time in one particular example.

We ran Magma-ECPP and Modified-ECPP on a probable prime with more than one thousand digits and show the output that describes the first three steps of the process. We present the number of digits and the first 10 and last 10 digits of big numbers occuring during the process. Describing the whole process or presenting whole numbers in the process for probable primes of this size would be far too space-consuming, as it usually consists of more than a hundred iterations and contains numbers in the size of the input probable prime. The output of the whole process for both implementations and the proof provided by Modified-ECPP can be found on one of the authors' homepage. Follow the links: http://www.math.ru.nl/~kiss/Modified-ECPP_Proof.pdf, http://www.math.ru.nl/~kiss/Modified-ECPP_Output.pdf and http://www.math.ru.nl/~kiss/Magma-ECPP_Output.pdf. For more big probable primes (coming from the generalized Pascal triangles), tested by Modified-ECPP refer to [6]. Note that G. Farkas and G. Kallós already have dealt with such numbers and tested them with the *ECPP*. Refer to [5] about the details.

The test was running in Magma v2.19 on a machine with 8001 Mb RAM and eight 2.5 GHz Intel Xeon processors.

Modified-ECPP is currently using different variants of trial divisions to factor the curve orders. With bound $t = 1000$ it applies normal trial division, and after that a batch trial division with bound b_i working on a list of $m_{i,j}$'s instead of factoring them one by one. The value of the penalty p is 0.8.

On an iteration we mean in the case of Magma-ECPP that we run (D) on one discriminant and in the case of Modified-ECPP that we run (D) on a set of discriminants up to $\log^{d_i}(n_i)$ (starting from a previously reached limit, or from the beginning of the discriminant list). On one step in both cases we mean a sequence of iterations, which either results in at least one new node, or runs out of discriminants and has to backtrack to another node. Going back to the same node is not considered as backtracking. In the implementation of Modified-ECPP one step also contains (T1) running on the new nodes, if there is any.

5.1 Example

The input number

```
n= 8565190451...2658848547
```

was used. This number has 1015 digits.

5.1.1 Magma-ECPP

In the first step Magma-ECPP needs 58 iterations to provide a new node and uses trial division to factor m_1.

```
% Pmax=4000000
% N_0=8565190451...683952658848547 1015 digits
% next D is 0 at 1.950000s
% next D is 7 at 10.180000s
...
% next D is 14683 at 495.940000s
% next D is 14083 at 513.280000s
% Cofactor after sieve is a probable prime
% D[[0]]=14083
%
% End of depth 0 at 513.990000 s
```

In the second step, Magma-ECPP fails to find an n_2 after running through a fixed list of discriminants twice, supposedly using trial division, Pollard's $p-1$ and Pollard's ρ in the first turn, and other, possibly harder and more time consuming, methods in the second, thus backtracking to n_0.

```
% Pmax=4000000
% N_1=8985609131...1613020571 1009 digits
% next D is 0 at 514.760000s
% next D is 7 at 522.780000s
...
% next D is 991 at 1140.930000s
% next D is 19963 at 1151.780000s
%!% No next D
%!% Forced to retry, snif...
% next D is 0 at 1163.620000s
% next D is 7 at 1184.430000s
...
% next D is 991 at 2754.460000s
% next D is 19963 at 2778.880000s
%!% No next D
```

```
%!% One redo was not enough...
% Backtracking
% End of depth -1 at 2803.580000 s
```

In the third step Magma-ECPP, as it backtracked in the previous step, uses n_0 again and starts from the first discriminant that was not used in the first step. It does not find a new n_1 just using trial division on the rest of the discriminants, so it applies stronger factorization methods running through the same set of discriminants from the beginning. As there is only one successful discriminant in the set, it finds the n_1 produced also by the first step. Therefore it gets to an endless loop finding the same n_1 and backtracking to n_0 again.

```
% Pmax=4000000
% N_0=8565190451...683952658848547 1015 digits
% next D is 19963 at 2805.250000s
% next D is 2339 at 2814.730000s
%!% No next D
%!% Forced to retry, snif...
% next D is 0 at 2825.470000s
% next D is 7 at 2847.600000s
...
% next D is 14683 at 4084.320000s
% next D is 14083 at 4114.750000s
% Cofactor after sieve is a probable prime
% D[[0]]=14083
% End of depth 0 at 4115.470000 s
```

Magma-ECPP needs 58 iterations to produce a new node in the first step and has to backtrack after the second step, and as it does not find another node where it could continue, gets stuck in an endless loop.

5.1.2 Modified-ECPP

By default, Modified-ECPP uses parameters $b_i = \log^2(n_i)$ and $s_i = \log(n_i)$, where b_i is the bound to the primes used to factor the m_i-s and s_i is the bound to primes used to factor the discriminants (refer to section 4.2.2). This configuration found a path to the leaves in 1246.99 seconds after 139 steps and it finished the proof using this path in 977.65 seconds; thus the total time was 2224.64 seconds.

Below we provide more information on the process in a configuration where we put $s_i = \log^{1.3}(n_i)$, as the first configuration does not find the node where Magma-ECPP gets stuck (because limit s_i is too low to factor 14083, a prime

in fact), and it is therefore complicated to compare the default case with the Magma-ECPP output.

In the first step, Modified-ECPP provides two $n_{1,j}$-s for $n_0 = n$ after 4 iterations. The discriminant bound $d_0 = \log^{\delta_0}(n_0)$ is increased after each iteration, moving up from $\delta_0 = 1$ to 1.3, where it finds two appropriate nodes, $n_{1,1}$ and $n_{1,2}$. It initiates bound $\delta = 1.2$ for both of them. $n_{1,2}$ is the same number that Magma-ECPP gets stuck on and as it is smaller than $n_{1,1}$, it is selected at the end of this step.

```
SLimit:  23946.87 1.3
BLimit:  5461424.06 2.0

n0 8565190451...683952658848547 1015 digits
DLimit, delta:  2336.97, 1.00
Filtering discriminants and reduction takes 7.700 seconds for
711 D-s, where 56 was succesful On average that is 0.011
The time for 56 trial divisions is 0.000 seconds, 0.000 on average
Batch trial division takes 3.270 seconds for 56 D, 0.058 on average
Time of Miller-Rabin test for 55 is 2.280 seconds, 0.041 on average
This resulted 0 new nodes
...
delta: 1.30
Filtering discriminants and reduction takes 33.380 seconds for
3930 D-s, where 50 was succesful On average that is 0.009
The time for 50 trial divisions is 0.020 seconds, 0.000 on average
Batch trial division takes 3.220 seconds for 50 D, 0.064 on average
Time of Miller-Rabin test for 46 is 3.580 seconds, 0.078 on average
This resulted 2 new nodes

lambda 1.93
delta 1.20
The total time of estimation is  3.140
lambda 1.49
delta 1.20
The total time of estimation is  3.050

1 level completed in 97.930 seconds
```

In the second step Modified-ECPP neither provides new node in one iteration, with bound $\delta_1 = 1.2$, therefore the *suitability* values had to be reevaluated and the other child is selected, as after the failure of the iteration the *suitability* of $n_{1,2}$ is increased to 1.3.

```
n1:  898560913...903551613020571 1009 digits
```

```
SLimit:  23763.64 1.3
DLimit:  10947.23 1.2
BLimit:  5397264.73590251461588576356885 2.0
Filtering discriminants and reduction takes 27.250 seconds for
3331 D-s, where 130 was succesful On average that is 0.008
The time for 130 trial divisions is 0.030 seconds, 0.000 on average
Batch trial division takes 3.410 seconds for 130 D, 0.026 on average
Time of Miller-Rabin test for 127 is 4.960 seconds, 0.039 on average
This resulted 0 new nodes

2 level completed in 36.790 seconds
```

It has more luck with $n_{1,1}$; in the third step provides four new nodes and initiates $\delta_{2,j}, 1 \leq j \leq 4$ for them.

```
n1:  1928293492...0513347699 1010 digits
SLimit:  23773.79 1.3
DLimit:  10951.55 1.2
BLimit:  5400813.29 2
Filtering discriminants and reduction takes 28.930 seconds for
3333 D-s, where 166 was succesful On average that is 0.009
The time for 166 trial divisions is 0.040 seconds, 0.000 on average
Batch trial division takes 3.480 seconds for 166 D, 0.021 on average
Time of Miller-Rabin test for 153 is 9.190 seconds, 0.060 on average
This resulted 4 new nodes

lambda 1.95
delta 1.20
The time of estimation is  3.110
...
lambda 2.06
delta 1.20
The time of estimation is  3.140

3 level completed in 51.070
```

Modified-ECPP provides two new nodes after 4 iterations in the first step. Note that it does not apply any heuristics in (T_0). Then it picks up the same node $n_{1,2}$ where Magma-ECPP got stuck and also has bad luck with this number, as it produces no new node after one iteration. Therefore it had to backtrack to $n_{1,1}$. In this case λ does not help us very much as it underestimated the necessary amount of discriminants to provide a new node, but as there is another node produced on the same level, Modified-ECPP does not

even have to go back to the previous level to provide a new node. In the third step it provides four new nodes in one iteration; λ seems to work better there. The algorithm reached the small primes in 1794.97 seconds after 146 steps. Completing the proof using this path took 1119.23 seconds, and thus the total running time was 2914.2 seconds.

Note that one iteration for Modified-ECPP takes a list of discriminants, this way the 4 iterations from the first step were running through discriminants up to 23946.

The first step took 513.99 seconds for Magma-ECPP and 97.93 seconds for Modified-ECPP. The difference probably comes from the fact that Magma-ECPP has to force the m_i-s with strong factoring methods and big factoring bounds to provide a new node while Modified-ECPP works with a much bigger set of discriminants (up to 10^{10}) and therefore it can use batch trial division and lower factoring bounds. In this case the bounds on lower levels in the tree are around 5300000. In the third step Modified-ECPP provides 4 new nodes with one iteration, where the predictions have been already applied on the value of D_1. The goal of the estimation of the initial value of δ_i using λ is to decrease the probability of backtracking and selecting the same node more often by predicting the minimal interval where one iteration will run successfully. This gives also the basis of the *suitability* value to predict on which node do we have to process the least number of discriminants to provide at least one new node. The total number of iterations in this case is 187 where 146 provides a new node. The other iterations provided the same node as the input or provided another node to backtrack. We do not count the four iterations of the first step in the number of the iterations, as there is no heuristics applied.

6 Remarks and conclusions

We described a strategy applied during the process of *ECPP*; produce more nodes in a step in the recursion, estimate the *suitability* of the new nodes and select the most suitable. With this strategy we try to avoid backtracking (or repetition on the same node) by estimating the interval for discriminants to search through for each new node and by predicting which $n_{i,j}$ to pick in order to be able to successfully continue. 78% of the iterations ran in the test terminated successfully on the nodes and intervals selected this way.

As we can see, the predictions give us no hundred percent certainty to avoid such situation; for instance, in the second step from the example $n_{1,2}$ provided no new node, but at least we have an idea, what we can expect in theory

from considering a node. Roughly speaking we could not avoid backtracking or repetition after 22% of the iterations in this case. What to do then?

First of all we are trying to avoid backtracking to the previous level because that might make our whole effort on the current level useless, and also because the size of the numbers is growing going up in the tree. There we can use the penalty value, that is added to the nodes in the previous levels' *suitability*, influencing the probability of selecting a node from previous levels. The default value of the penalty is high, 0.8. After one unsuccessful iteration we increase the value *suitability* with 0.1, and thus 0.8 would mean 8 unsuccessful iterations on the node; this is a tough condition, does not occur frequently. Of course different nodes starts from different *suitability*, thus in practice we do not always need 8 unsuccessful iterations; if none of the nodes on the current level is suitable enough, the numbers are just to get a feeling about the size of the penalty. In this example backtracking to the previous level does not occur.

The goal of producing more nodes in a step is to make the implementation more robust against backtracking to the previous level, as the nodes in the current level are likely more suitable, and to avoid running multiple iteration on the same node by switching between the nodes on the same level. In addition, if it turns out that a node is not as successful as estimated, and thus produces no new node after the first iteration on the expected interval, we become more careful and try to take small steps at a time. We increase the value $\delta_{i,j}$ of the current node $n_{i,j}$ with 0.1 after each unsuccessful iteration and reorder the list of possible nodes to see whether our selected node is still the best. This way it tries to keep backtracking fast and flexible. That is what happens in the second step in the example.

Modified-ECPP was tested with several numbers with around 1000 digits and it provided proof in each situation in 2000–3000 seconds. Magma-ECPP was running on the same numbers failed on the number from the example, and provided proof in 2000–8000 seconds in every other cases, depending on whether we backtrack or not there are bigger differences in running time. It seems that the running times of Modified-ECPP are more balanced for numbers with similar size.

The implementation works on a list of preprocessed discriminants up to 10^{10}, thus it can process the discriminants fast, without the need of factoring the discriminants or the computed m_i-s with high factoring bounds. By default Modified-ECPP runs with configuration $s_{i,j} = \log(n_{i,j})$ and $b_{i,j} = \log^2(n_{i,j})$, which for numbers with such size are around 2 300 and 5 300 000, and in our tests always terminated successfully.

As it still has to backtrack in a number of situations, there are further plans to tune the strategy to reduce this number by involving all the important parameters $s_{i,j}, b_{i,j}, d_{i,j}$ (refer to Section 4.2.2) to determine *suitability*. Also the implementation is not the final version, stronger factoring methods are going to be applied on the $m_{i,j}$-s in order to gain shorter paths and to be able to prove primality for bigger numbers.

Acknowledgements

We are greatly indebted to Prof. Wieb Bosma for his scientific consultations.

References

[1] M. Agrawal, N. Kayal, N. Saxena, PRIMES is in P, *Annals Math.* **160,** 2 (2004) 781–793. ⇒35

[2] A. O. L. Atkin, F. Morain, Elliptic curves and primality proving *Math. Comp.* **61,** 203 (1993) 29–68. ⇒36, 39, 41, 43

[3] D. Bernstein, Proving primality in essentially quartic random time, *Math. Comp.* **76,** 257 (2007) 389–403. ⇒35

[4] W. Bosma, J. Cannon, C. Playoust, The Magma algebra system. I. The user language, *J. Symbolic Comput.* **24,** 3-4 (1997) 235–265. ⇒41

[5] G. Farkas, G. Kallós, Prime numbers in generalized Pascal triangles, *Acta Tech. Jaur.* **1,** 1 (2008) 109–118. ⇒45

[6] G. Farkas, G. Kallós, G. Kiss, Large primes in generalized Pascal triangles, *Acta Univ. Sapientiae, Inform.* **3,** 2 (2011) 158–171. ⇒45

[7] J. L. Hafner, K. S. McCurley, On the Distribution of Running Times of Certain Integer Factoring Algorithms, *J. Algorithms* **10,** 4 (1989) 531–556. ⇒43

[8] A. K. Lenstra, H. W. Lenstra Jr., Algorithms in number theory in: *Handbook of Theoretical Computer Science* (vol. A) 1990, Elsevier, Amsterdam, pp. 673–715. ⇒36, 41

Received: March 10, 2013 • Revised: May 31, 2013

Acta Univ. Sapientiae, Informatica, **5,** 1 (2013) 53–62

Nonexistence of a Kruskal–Katona type theorem for double-sided shadow minimization in the Boolean cube layer

Maksim BASHOV
Lomonosov Moscow State University
email: max.bashov@gmail.com

Abstract. A double-sided shadow minimization problem in the Boolean cube layer is investigated in this paper. The problem is to minimize the size of the union of the lower and upper shadows of a k-uniform family of subsets of $[n]$. It is shown that if $3 \le k \le n-3$, there is no total order such that all its initial segments have minimal double-sided shadow.

Denote by $\binom{[n]}{k}$ the family of all subsets of the set $[n] = \{1, 2, \dots, n\}$ having the size k. Let $\mathcal{F} \subseteq \binom{[n]}{k}$. *The lower shadow* $\Delta\mathcal{F}$ is the $(k-1)$-uniform family of sets A such that there exists $B \in \mathcal{F}$, $A \subset B$. Similarly, *the upper shadow* $\nabla\mathcal{F}$ is the $(k+1)$-uniform family of sets A such that there exists $B \in \mathcal{F}$, $A \supset B$. *The double-sided shadow* $\boxtimes\mathcal{F}$ is the union of the families $\Delta\mathcal{F}$ and $\nabla\mathcal{F}$.

A family $\mathcal{F} \subseteq \binom{[n]}{k}$ is *minimal in terms of lower shadow* (*upper shadow, double-sided shadow*) if $|\Delta\mathcal{F}| \le |\Delta\mathcal{G}|$ (corr., $|\nabla\mathcal{F}| \le |\nabla\mathcal{G}|$, $|\boxtimes\mathcal{F}| \le |\boxtimes\mathcal{G}|$) for each family $\mathcal{G} \subseteq \binom{[n]}{k}$ such that $|\mathcal{G}| = |\mathcal{F}|$.

A set A *lexicographically precedes* a set B ($A <_{\text{lex}} B$) if and only if $\max((A \setminus B) \cup (B \setminus A)) \in B$.

Kruskal [8] and Katona [7] described solutions of the single-sided shadow minimization problem:

Kruskal–Katona theorem. *The initial lexicographical segments of* $\binom{[n]}{k}$ *are minimal families in terms of lower shadow.*

A simple modern proof of this theorem is given in [1]. Clements and Lindström [6] generalized this result to the products of chains. Analogues of the

Computing Classification System 1998: G.2.1
Mathematics Subject Classification 2010: 05D05, 06A07, 68R15
Key words and phrases: shadow minimization, Boolean cube, double-sided shadow, ideal weight

Kruskal–Katona theorem are also proved for a wide variety of structures, including the products of stars and their dual posets [5].

The Kruskal—Katona theorem describes minimal families in terms of a total order defined on $\binom{[n]}{k}$. In this paper we prove that it is impossible to describe the solutions of the double-sided minimization problem as initial segments of a total order.

We say that a total order defined on $\binom{[n]}{k}$ is *minimizing* if all its initial segments are minimal in terms of double-sided shadow. The following statement is proven in [2]:

Theorem 1 [2] *If $k = 3$ and $n \geqslant 8$, then there does not exist a minimizing total order on $\binom{[n]}{k}$. For every total order defined on $\binom{[n]}{k}$ there exists a number $m \leqslant 4n - 14$ such that the initial segment of the order having the length m is not minimal in terms of double-sided shadow.*

Since $\binom{[n]}{k}$ is isomorphic to $\binom{[n]}{n-k}$, a similar result holds for $k = n - 3$.

It is known [4] that the lexicographical order is minimizing when $k \geq n-2$, i. e., in this case families minimizing the lower shadow are also minimal in terms of double-sided shadow. Similarly, if $k \leq 2$, then the colexicographical order is minimizing the double-sided shadow.

Define a partial order induced by the standard shifting operator (see, e.g., [1]) on $\binom{[n]}{k}$. Let A and B be elements of $\binom{[n]}{k}$, $A = \{a_1, a_2, \ldots, a_k\}$, $B = \{b_1, b_2, \ldots, b_k\}$, and their elements are sorted in ascending order, i. e., $a_1 < a_2 < \ldots < a_k$, $b_1 < b_2 < \ldots < b_k$. The set A *precedes* the set B ($A \sqsubseteq B$) if $a_i \leq b_i$ for each $i = 1, 2, \ldots, k$. A family $\mathcal{F}$ is *an ideal* if $A \in \mathcal{F}$ and $B \sqsubseteq A$ imply $B \in \mathcal{F}$. By $\mathcal{I}(A)$ denote the minimal ideal containing A, i. e., the family $\{B \in \binom{[n]}{k} \mid B \sqsubseteq A\}$.

Theorem 2 [2] *The family $\mathcal{C}_1(n, k) = \mathcal{I}(\{2, 3, \ldots, k, n\})$ is the unique minimal in terms of double-sided shadow ideal of $\binom{[n]}{k}$ having cardinality $1+k(n-k)$.*

The following statement gives a simple additive formula for the size of an ideal's shadow.

Lemma 3 [2] *If $\mathcal{F} \subseteq \binom{[n]}{k}$ is an ideal, then $|\boxtimes\mathcal{F}| = \sum_{A \in \mathcal{F}} s(A)$, where $s(A) = s_l(A) + s_u(A)$, $s_l(A) = \min([n] \setminus A) - 1$, $s_u(A) = n - \max A$.*

The weight function s is monotone with respect to the partial order $\sqsubseteq$:

Lemma 4 [2] $A \sqsubseteq B$ *implies* $s(A) \geq s(B)$.

Let $A = \{1, 2, \dots, m, a_{m+1}, \dots, a_{k-1}, a_k\} \in \binom{[n]}{k}$, $a_{m+1} > m+1$. Denote

$$A^0 = \{2, 3, \dots, m+1, a_{m+1}, \dots, a_{k-1}, n\}.$$

Put also $\widehat{\mathcal{I}}(A^0) = \left\{B \in \binom{[n]}{k}, B^0 = A^0\right\}$. Suppose $p = \min(\{q : a_q > q+1\} \cup \{k\})$. The poset $(\widehat{\mathcal{I}}(A^0), \sqsubseteq)$ contains a least point

$$\{1, 2, \dots, p-1, a_p, \dots, a_{k-1}, a_{k-1}+1\}$$

in case $p \neq k$ and a least point $\{1, 2, \dots, k\}$ in case $p = k$. Denote this least point by A^1.

By $\mathcal{F}|_q$ we denote the family $\{A \in \mathcal{F}, s(A) = q\}$.

Lemma 5 [4] *Let* $A^0 \in \binom{[n]}{k}\Big|_0$, $\mathcal{F} = \widehat{\mathcal{I}}(A^0) \setminus \{A^1\}$ *if* $A^1 = \{1, 2, \dots, k\}$, *and* $\mathcal{F} = \widehat{\mathcal{I}}(A^0)$ *otherwise. Then* $\mathcal{F}$ *is isomorphic as a poset to the product of two chains having lengths* $s_l(A^1)+1$ *and* $s_u(A^1)+1$, *and the layers of the product correspond to the families* $\widehat{\mathcal{I}}(A^0)|_m$ *under the isomorphism.*

Lemma 6 $A^0 \sqsubseteq B^0$ *if and only if* $A^1 \sqsubseteq B^1$.

Proof. In case $A^1 = \{1, 2, \dots, k\}$ the statement follows from the facts that $A^0 = \{2, 3, \dots, k, n\}$ is the least set of the family $\binom{[n]}{k}\Big|_0$ in terms of the partial order $\sqsubseteq$, and A^1 is the least set of $\binom{[n]}{k}$.
Suppose

$$\begin{aligned} A^0 &= \{2, 3, \dots, p, a_p, \dots, a_{k-1}, n\},\ a_p > p+1, \\ B^0 &= \{2, 3, \dots, q, b_q, \dots, b_{k-1}, n\},\ b_q > q+1. \end{aligned}$$

Then

$$\begin{aligned} A^1 &= \{1, 2, \dots, p-1, a_p, \dots, a_{k-1}, a_{k-1}+1\}, \\ B^1 &= \{1, 2, \dots, q-1, b_q, \dots, b_{k-1}, b_{k-1}+1\}. \end{aligned}$$

It remains to note that the condition $A^0 \sqsubseteq B^0$ and the condition $A^1 \sqsubseteq B^1$ are equivalent to $p \geq q$, $a_j \leq b_j$ if $p+1 \leq j \leq k-1$. □

Suppose $A^0 = \{2, 3, \dots, p, a_p, \dots, a_{k-1}, n\}$, $a_p > p+1$. Note that if $A^0 \neq \{2, 3, \dots, k, n\}$, then the possible immediate successors of A^0 in terms of $\sqsubseteq$ are

$$\begin{aligned} A^{l0} &= \{2, 3, \dots, p-1, p+1, a_p, \dots, a_{k-2}, a_{k-1}, n\}, \\ A^{u0} &= \{2, 3, \dots, p-1, p, a_p, \dots, a_{k-2}, a_{k-1}+1, n\}, \\ \tilde{A}^0_j &= \{2, 3, \dots, p-1, p, a_p, \dots, a_j+1, \dots, a_{k-2}, a_{k-1}, n\}. \end{aligned}$$

Also note that $s(A^{l1}) = s(A^{u1}) = s(A^1) - 1$, and $s(\tilde{A}^1_j) = s(A^1)$.

Suppose $<_{\min}$ is a minimizing total order defined on $\binom{[n]}{k}$. The properties of the standard shifting operator [1] imply that without loss of generality we can suppose that the initial segments of $<_{\min}$ are ideals, i. e., $A \sqsubseteq B$ implies $A <_{\min} B$.

By $\mathrm{ord}(A)$ denote the number of sets preceding A in the order $<_{\min}$.

Lemma 7 *Let $<_{\min}$ is a minimizing order on $\binom{[n]}{k}$, $A <_{\min} B <_{\min} C$ and $A^0 = C^0$. Then $A^0 = B^0 = C^0$.*

Proof. Suppose $B^0 \neq A^0$.

Without loss of generality, $A = A^1$, $B = B^1$. Indeed, since $A^1 \sqsubseteq A$, $B^1 \sqsubseteq B$, it holds that $A^1 <_{\min} A$, $B^1 <_{\min} B$. If $A^1 <_{\min} B^1$, we put $A' = A^1$, $B' = B^1$, $C' = C$. If $A^1 >_{\min} B^1$, we put $A' = B^1$, $B' = A^1$, $C' = B$.

Without loss of generality, for all D such that $D <_{\min} A$, $D^0 \neq A^0$ it holds that $D^0 <_{\min} A$. If this property doesn't hold, we put $A' = D^1$, $B' = A$, $C' = D^0$ and check the property again. This reassignment doesn't break the property from the previous paragraph, and an infinite chain of reassignments is impossible since $A' <_{\min} A$ and the family $\{A' \colon A' <_{\min} A\}$ is finite.

We can assume that B is the least set of the family $\{B \colon B^0 \neq A^0, B >_{\min} A\}$ in terms of the order $<_{\min}$. Since the initial segments of $<_{\min}$ are ideals, by Lemma 6 the set B^0 is either incomparable to A^0 in the order $\sqsubseteq$, or is an immediate successor of A^0.

We can assume that C is the least set of the family $\{C \colon C^0 = A^0, C >_{\min} B\}$ in terms of the order $<_{\min}$.

Denote by $\mathcal{F}$ the initial segment of $<_{\min}$ having the length $\mathrm{ord}(B)$. Since $<_{\min}$ is a minimizing order, and $\mathcal{F} \cup \{B\}$ is its initial segment, it holds that $|\boxtimes(\mathcal{F} \cup \{B\})| \leq |\boxtimes(\mathcal{F} \cup \{C\})|$, therefore

$$s(B) \leq s(C). \tag{1}$$

Since $A \sqsubseteq C$, $A^0 = C^0$, it follows from Lemma 5 that $s(C) \leq s(A) - 1$.

Denote by $\mathcal{G}$ the initial segment of $<_{\min}$ having the length $\mathrm{ord}(A)$.

Suppose A is incomparable to B in the partial order $\sqsubseteq$. In this case the family $\mathcal{G} \cup \{B\}$ is an ideal, and by the definition of a minimizing order it holds that $|\boxtimes(\mathcal{G} \cup \{A\})| \leq |\boxtimes(\mathcal{G} \cup \{B\})|$, therefore $s(A) \leq s(B)$. This implies $s(C) < s(B)$, a contradiction with (1).

Suppose A is comparable to B in the partial order $\sqsubseteq$. Note that $A^0 \neq \{2, 3, \ldots, k, n\}$, because Theorem 2 implies that $\mathcal{C}_1(n, k) = \widehat{\mathcal{I}}(\{2, 3, \ldots, k, n\})$ is an initial segment of $<_{\min}$. Then, since A^0 is an immediate predecessor of B^0

in the order $\sqsubseteq$, $s(B) \geq s(A) - 1$. Therefore (1) implies $s(B) = s(A) - 1$, and either $B^0 = A^{l0}$, or $B^0 = A^{u0}$.

Without loss of generality, $B^0 = A^{u0}$, that is

$$\begin{aligned} A =& \{1, 2, \ldots, p, a_{p+1}, \ldots, a_{k-2}, q-1, q\}, \\ B =& \{1, 2, \ldots, p, a_{p+1}, \ldots, a_{k-2}, q, q+1\},\ a_{p+1} > p+2. \end{aligned}$$

Since $A' = \{1, 2, \ldots, p, a_{p+1}, \ldots, a_{k-2}, q-1, q+1\} \sqsubseteq B$, it holds that $A' <_{\min} B$. Note also that $q+1 \leq n$, and $n - q \geq 1$.

Consider the family $\mathcal{A} = \big\{A \cup \{r\} \setminus \{q\}, q+1 < r \leq n\big\}$. Note that for every $A'' \in \mathcal{A}$ it holds that $s(A'') < s(A) - 1 = s(B)$, and $\mathcal{A}$ is a chain in the partial order $\sqsubseteq$. Let us show that for every $A'' \in \mathcal{A}$ it holds that $A'' <_{\min} B$. Suppose A'' is the least set in $\mathcal{A}$ such that $A'' >_{\min} B$. Then

$$|\boxtimes(\mathcal{F} \cup \{A''\})| = |\boxtimes\mathcal{F}| + s(A'') < |\boxtimes\mathcal{F}| + s(B) = |\boxtimes(\mathcal{F} \cup \{B\})|,$$

a contradiction with the minimality of the family $\mathcal{F} \cup \{B\}$, which is an initial segment of $<_{\min}$.

Note that $C = A \cup \{p+1\} \setminus \{p\}$, since for every $A'' \in \widehat{\mathcal{I}}(A^0)$ distinct from the sets A, A' and $A \cup \{p+1\} \setminus \{p\}$ it holds that $s(A'') < s(B) = s(C)$. Therefore $\mathrm{ord}(B) = \mathrm{ord}(A) + s_u(A)$.

Let us show that the immediate successors of B in the order $<_{\min}$ are the sets $B \cup \{r\} \setminus \{q+1\}$, $q+1 < r \leq n$. Suppose that the immediate successor of $(B \setminus \{q+1\}) \cup \{r\}$ is $D \neq B' = B \cup \{r+1\} \setminus \{q+1\}$. The initial segments of $<_{\min}$ having lengths $\mathrm{ord}(D)$ and $\mathrm{ord}(D) + 1$ are minimal in terms of double-sided shadow, and therefore

$$s(D) \leq s(B \cup \{r+1\} \setminus \{q+1\}) < s(C). \tag{2}$$

The families

$$\mathcal{H} = \mathcal{F} \cup \big\{B\big\} \cup \big\{B \cup \{s\} \setminus \{q+1\}, q+1 < s \leq r\}$$

and $\mathcal{H} \cup \{D\}$ are ideals, therefore $D \in \widehat{\mathcal{I}}(A^0)$ implies $D = C$, and $D \in \widehat{\mathcal{I}}(B^0)$ implies $D = B'$. Hence, $D \notin \widehat{\mathcal{I}}(A^0) \cup \widehat{\mathcal{I}}(B^0)$. Since $D^1 <_{\min} A$, $D^0 \neq A^0$ implies $D^0 <_{\min} A$, it holds that $D = D^1$. If D is incomparable to A in terms of $\sqsubseteq$, then it follows from (2) that

$$|\boxtimes(\mathcal{G} \cup \{D\})| = |\boxtimes\mathcal{G}| + s(D) < |\boxtimes\mathcal{G}| + s(C) < |\boxtimes\mathcal{G}| + s(A) = |\boxtimes(\mathcal{G} \cup \{A\})|,$$

and that is a contradiction with the fact that $\mathcal{G} \cup \{A\}$ is an initial segment $<_{\min}$ and therefore is minimal in terms of double-sided shadow. If D is comparable

to A and incomparable to B, then, since $\mathcal{H}$ is an ideal, Lemma 6 implies that D^0 is an immediate successor of A^0 in the order $\sqsubseteq$, and $s(D) \geq s(A) - 1 = s(C)$, a contradiction with (2). Hence, D is comparable to B. Since D^0 is an immediate successor of B^0 in $\sqsubseteq$ and $s(D) < s(C)$, it holds that $D = \{1, 2, \ldots, p, a_{p+1}, \ldots, a_{k-2}, q+1, q+2\}$. Since $B \cup \{q+2\} \setminus \{q+1\} \sqsubseteq D$, it is true that $B \cup \{q+2\} \setminus \{q+1\} <_{\min} D$ and $r > q+2$. But then $s(B \cup \{r+1\} \setminus \{q+1\}) < s(D)$, a contradiction with (2).

Thus, $\mathcal{F}' = \mathcal{F} \cup \{B\} \cup \{B \cup \{r\} \setminus \{q+1\}, q+1 < r \leq n\}$ is an initial segment of $<_{\min}$. Denote by D an immediate successor of $B \cup \{n\} \setminus \{q+1\}$ in terms of the order $<_{\min}$. Let us show that $s(D) \geq s(C) - 1$. Indeed, if $D \in \widehat{\mathcal{I}}(A^0)$, then $D = C$, because $\mathcal{F} \cup \{D\}$ is an ideal. Similarly, $D \notin \widehat{\mathcal{I}}(B^0)$. If $D \notin \widehat{\mathcal{I}}(A^0) \cup \widehat{\mathcal{I}}(B^0)$, then $s(D) \geq s(A)$ when D is incomparable to B, and $s(D) \geq s(B) - 1 = s(C) - 1$ in case $B \sqsubseteq D$. Therefore

$$|\mathbb{X}(\mathcal{F}' \cup \{D\})| = |\mathbb{X}\mathcal{F}| + s(B) + \sum_{r=q+2}^{n} s(B \cup \{r\} \setminus \{q+1\}) + s(D) =$$

$$= |\mathbb{X}\mathcal{F}| + \sum_{r=0}^{n-q-1} (s(C) - r) + s(D).$$

Consider the family $\mathcal{F}'' = \mathcal{F} \cup \{C\} \cup \{C \cup \{r\} \setminus \{q\}, r < q \leq n\}$. Note that $|\mathcal{F}''| = |\mathcal{F}' \cup \{D\}| = |\mathcal{F}| + n - q$, and

$$|\mathbb{X}\mathcal{F}''| = |\mathbb{X}\mathcal{F}| + s(C) + \sum_{r=q}^{n} s(C \cup \{r\} \setminus \{q\}) =$$

$$= |\mathbb{X}\mathcal{F}| + \sum_{r=0}^{n-q} (s(C) - r) = |\mathbb{X}\mathcal{F}| + \sum_{r=0}^{n-q-1} (s(C) - r) + s(C) - (n - q).$$

If $n - q > 2$, it follows that $|\mathbb{X}\mathcal{F}''| < |\mathbb{X}(\mathcal{F}' \cup \{D\})|$, a contradiction with the minimizing property of $<_{\min}$. If $n - q = 1$, then there does not exist D such that $B \sqsubseteq D$, $B^0 \neq D^0$ and $s(D) = s(B) - 1$, therefore $s(D) \geq s(C)$, and again $|\mathbb{X}\mathcal{F}''| < |\mathbb{X}(\mathcal{F}' \cup \{D\})|$. □

Corollary 8 *Let $<$ be a total order defined on $\binom{[n]}{k}$ such that all its initial segments are ideals, $A < B < C$ and $A^0 = C^0 \neq B^0$. Then there exists a number m such that the initial segment of $<$ having the length m is not a minimal family in terms of double-sided shadow, and $m \leq \min\{\mathrm{ord}(B), \mathrm{ord}(A) + 2n\}$.*

Thus, the families $\widehat{\mathcal{I}}(A^0)$ are segments of any minimizing total order. The following statement describes how the sets from these families are sorted by a minimizing order.

Lemma 9 *Suppose* $<_{\min}$ *is a minimizing order,* $s(A) > 0$, $s_l(A) = p$, $s_u(A) = n - q$, *and* B *is the immediate successor of* A *in* $<_{\min}$. *If* $s_l(A^1) < s_u(A^1)$, *then*

$$B = \begin{cases} A \cup \{q+1\} \setminus \{q\}, & \text{if } q < n, \\ A \cup \{p+1, n - s_u(A^1)\} \setminus \{p, q\}, & \text{if } q = n. \end{cases} \tag{3}$$

If $s_l(A^1) > s_u(A^1)$, *then*

$$B = \begin{cases} A \cup \{p+1\} \setminus \{p\}, & \text{if } p > 0, \\ A \cup \{p+1, q+1\} \setminus \{s_l(A^1)+1, q\}, & \text{if } p = 0. \end{cases} \tag{4}$$

And if $s_l(A^1) = s_u(A^1)$, *then either for each* $A \in \widehat{\mathcal{I}}(A^0)$ *its immediate successor in* $<_{\min}$ *is defined by* (3), *or for each* A *it is defined by* (4).

Proof. This statement follows from Lemma 5, monotonicity and symmetry of the function s defined on $\widehat{\mathcal{I}}(A^0)$ and the two-dimensional case of the Clements–Lindström theorem. □

The order of families $\widehat{\mathcal{I}}(A^0)$ in the minimizing order is described by the following two statements.

Lemma 10 *Suppose* $<_{\min}$ *is a minimizing order, for each* $C^0 \sqsubseteq A^0$ *it holds that* $C^0 <_{\min} B^0$, *for each* $C^0 \sqsubseteq B^0$ *it holds that* $C^0 <_{\min} A^0$, *and* $s(A^1) < s(B^1)$. *Then* $A^1 <_{\min} B^1$.

Proof. Assume the converse. Suppose A^0 and B^0 meet the conditions of the lemma, and $A^1 >_{\min} B^1$. Denote by $\mathcal{F}$ the initial segment of $<_{\min}$ having the length $\operatorname{ord}(B^1)$. Then $\mathcal{F} \cup \{A^1\}$ is an ideal, and since $A^1 \notin \mathcal{F}$,

$$|{\boxtimes}(\mathcal{F} \cup \{A^1\})| = |{\boxtimes}\mathcal{F}| + s(A^1) < |{\boxtimes}\mathcal{F}| + s(B^1) = |{\boxtimes}(\mathcal{F} \cup \{B^1\})|,$$

a contradiction to minimality of the initial segment $\mathcal{F} \cup \{B^1\}$. □

Lemma 11 *Suppose* $<_{\min}$ *is a minimizing order, for each* $C^0 \sqsubseteq A^0$ *it holds that* $C^0 <_{\min} B^0$, *for each* $C^0 \sqsubseteq B^0$ *it holds that* $C^0 <_{\min} A^0$, *and* $s(A^1) = s(B^1)$, $\max\{s_l(A^1), s_u(A^1)\} > \max\{s_l(B^1), s_u(B^1)\}$. *Then* $A^1 <_{\min} B^1$.

Proof. Assume the converse. Suppose A^0 and B^0 meet the conditions of the lemma, and $A^1 >_{\min} B^1$. Denote by $\mathcal{F}$ the initial segment of $<_{\min}$ having the length $\operatorname{ord}(B^1)$. Without loss of generality we can assume that $s_u(A^1) =$

$n - q' \geq s_l(A^1) = p'$, $s_u(B^1) = n - q'' \geq s_l(B^1) = p''$. Then since $s_l(B^1) > s_l(A^1) \geq 0$, Lemma 9 states that the family

$$\mathcal{F}' = \mathcal{F} \cup \{B^1\} \cup \{B^1 \cup \{r\} \setminus \{q''\}, q'' < r \leq n\} \cup \{B^1 \cup \{p'' + 1\} \setminus \{p''\}\}$$

is an initial segment of $<_{min}$. Consider the family

$$\mathcal{F}'' = \mathcal{F} \cup \{A^1\} \cup \{A^1 \cup \{r\} \setminus \{q'\}, q' < r \leq n + q' + 1 - q''\}.$$

Note that $|\mathcal{F}'| = |\mathcal{F}''| = |\mathcal{F}| + n - q'' + 2$, and since $n - q'' = s_u(B^1) \geq s_l(B^1) \geq 1$, it holds that

$$|\boxtimes \mathcal{F}''| = |\boxtimes \mathcal{F}| + \sum_{r=0}^{n-q''+1} (s(A^1) - r) = |\boxtimes \mathcal{F}| + \sum_{r=0}^{n-q''+1} (s(B^1) - r) <$$
$$< |\boxtimes \mathcal{F}| + \sum_{r=0}^{n-q''} (s(B^1) - r) + s(B^1) - 1 = |\boxtimes \mathcal{F}'|,$$

a contradiction to the definition of a minimizing order. □

Since the least set of $\binom{[n]}{k} \setminus \mathcal{C}_l(n,k)$ in terms of $\sqsubseteq$, namely,

$$\{1, 2, \ldots, k-2, k+1, k+2\}$$

belongs to $\widehat{\mathcal{I}}(\{2, 3, \ldots, k-1, k+1, n\})$, the families $\mathcal{C}_l(n,k)$ and $\widehat{\mathcal{C}}_l(n,k) = \mathcal{C}_l(n,k) \cup \widehat{\mathcal{I}}(\{2, 3, \ldots, k-1, k+1, n\})$ are initial segments of $<_{min}$. The family $\binom{[n]}{k} \setminus \widehat{\mathcal{C}}_l(n,k)$ contains two minimal sets. In case $n \neq 2k$ we can infer which one is smaller in terms of $<_{min}$.

Lemma 12 *If $n > 2k$, $A \in \widehat{\mathcal{I}}(\{2, 3, \ldots, k-1, k+2, n\})$, $B \in \widehat{\mathcal{I}}(\{2, 3, \ldots, k-2, k, k+1, n\})$, and $<_{min}$ is a minimizing order, then $B <_{min} A$.*

Proof. Note that $A^0 = \{2, 3, \ldots, k-1, k+2, n\}$ and $B^0 = \{2, 3, \ldots, k-2, k, k+1, n\}$ meet the conditions of Lemma 11, and

$$\max(s_l(A^1), s_u(A^1)) = n - k - 2 < n - k - 1 = \max(s_l(B^1), s_u(B^1)).$$

□

Theorem 13 *If $4 \leqslant k < \frac{n}{2}$, then there does not exist a minimizing total order on $\binom{[n]}{k}$. For every total order defined on $\binom{[n]}{k}$ there exists a number $m \leqslant 1 + k(n-k) + (k-1)(2n-2k-3)$ such that the initial segment of the order having the length m is not minimal in terms of double-sided shadow.*

Proof. Consider the initial segment $\mathcal{L}$ of the order $<_{\min}$ having the length

$$|\mathcal{I}(\{2,3,\dots,k-1,k+2,n\})| = 1 + k(n-k) + (k-1)(2n-2k-3).$$

It follows from Lemmas 7, 9, 10, 12 that this segment is the union of the ideal $\mathcal{I}(\{2,3,\dots,k-2,k,k+1,n\})$ and the family of $n-2k$ sets belonging to $\widehat{\mathcal{I}}(\{2,3,\dots,k-3,k,k+1,k+2,n\})$. Counting the size of the double-sided shadow by Lemma 3, we get

$$|\boxtimes\mathcal{L}| = |\boxtimes\mathcal{I}(\{2,3,\dots,k-1,k+2,n\})| + (n-2k)(k-3),$$

a contradiction to the definition of a minimizing order. □

Since $\binom{[n]}{k}$ is isomorphic to $\binom{[n]}{n-k}$, the same result holds when $\frac{n}{2} < k \leq n-4$.

It is known [3] that there is a nested system of minimal sets contained in $\mathcal{C}_1(n,k) \cup \{\{1,2,\dots,k-2,k+1,k+2\}\}$, therefore it is impossible to get an estimate for m lower than $2 + k(n-k)$.

Now consider the case $n = 2k$, $k \geqslant 5$.

Theorem 14 *If $5 \leqslant k = \frac{n}{2}$, then there does not exist a minimizing total order on $\binom{[n]}{k}$. For every total order defined on $\binom{[n]}{k}$ there exists a number $m \leqslant 1+k^2+(2k-3)(k-2)+\frac{k(k-1)^2}{2}$ such that the initial segment of the order having the length m is not minimal in terms of double-sided shadow.*

Proof. Consider the family

$$\mathcal{F} = \mathcal{I}(\{3,4,\dots,k+1,n\}) \cup \mathcal{I}(\{2,3,\dots,k-2,k,k+2,n\}).$$

Since $\binom{[n]}{n/2}$ has a symmetry, without loss of generality we can assume that

$$\{2,3,\dots,k-1,k+2,n\} >_{\min} \{2,3,\dots,k-2,k,k+1,n\}.$$

Lemmas 10 and 11 determine the subsequent order of the families $\widehat{\mathcal{I}}(A^0)$ in the minimizing order, and the initial segment $\mathcal{L}$ of $<_{\min}$ having the length $|\mathcal{F}|$ contains the ideals $\mathcal{I}(\{3,4,\dots,k+1,n\})$, $\mathcal{I}(\{2,3,\dots,k-1,k+3,n\})$ and the set $\{1,2,\dots,k-2,k+4,k+5\}$. Counting the size of the shadow, we get $|\boxtimes\mathcal{L}| = |\boxtimes\mathcal{F}| + k - 4$, a contradiction to the definition of a minimizing order. □

Now we consider the cases not covered by Theorems 1, 13 and 14. There does not exist a minimizing order on $\binom{[8]}{4}$, and the maximal size m of a nested system of minimal families equals 42, since the only minimal ideals having size 41 are $\mathcal{I}(\{2,4,5,8\}) \cup \mathcal{I}(\{2,3,7,8\})$ and $\mathcal{I}(\{2,3,6,8\}) \cup \mathcal{I}(\{3,4,5,8\})$, while

the only minimal ideal having size 42 is $\mathcal{I}(\{2,4,6,8\})$. There does not exist a minimizing order on $\binom{[7]}{3}$, and $m = 15$, since the only minimal ideal having size 15 is $\mathcal{I}(\{1,6,7\})$, and it does not contain $\mathcal{C}_1(7,3)$. Finally, there is a minimizing order on $\binom{[6]}{3}$ distinct from lexicographical:

$$\{1,2,3\} <_{\min} \{1,2,4\} <_{\min} \{1,2,5\} <_{\min} \{1,2,6\} <_{\min} \{1,3,4\} <_{\min}$$
$$\{1,3,5\} <_{\min} \{1,3,6\} <_{\min} \{2,3,4\} <_{\min} \{2,3,5\} <_{\min} \{2,3,6\} <_{\min}$$
$$\{1,4,5\} <_{\min} \{1,4,6\} <_{\min} \{2,4,5\} <_{\min} \{2,4,6\} <_{\min} \{1,5,6\} <_{\min}$$
$$\{2,5,6\} <_{\min} \{3,4,5\} <_{\min} \{3,4,6\} <_{\min} \{3,5,6\} <_{\min} \{4,5,6\}.$$

Thus, a minimizing order exists on $\binom{[n]}{k}$ only in cases $k \le 2$, $k \ge n-2$, and $n = 6$, $k = 3$.

Acknowledgements

The author would like to thank his advisor Prof. Alexander A. Sapozhenko for formulation of the problem and constant attention to the work. The author also thanks Dmitry V. Chistikov for useful discussions.

References

[1] R. Ahlswede, H. Aydinian, L. H. Khachatrian, More about shifting techniques, *European J. Combin.* **24,** 5 (2003) 551–556. ⇒53, 54, 56

[2] M. A. Bashov, On minimisation of the double-sided shadow in the unit cube, *Diskr. Mat.*, **23,** 4 (2011) 115–132. ⇒54

[3] M. A. Bashov, Minimal families in terms of double-sided shadow in the Boolean cube layer, *Electron. Notes in Discrete Math.*, **38** (2011) 117–122. ⇒61

[4] M. A. Bashov, Minimal in terms of double-sided shadow subsets of Boolean cube layer distinct from circles, *Diskretn. Anal. Issled. Oper.*, **19,** 5 (2012) 3–20. ⇒54, 55

[5] S. L. Bezrukov, U. Leck, Macaulay posets, *Electron. J. Combin.* (2004), Dynamic Survey DS12, http://www.combinatorics.org/. ⇒54

[6] G. F. Clements, B. Lindström, A generalization of a combinatorial theorem of Macaulay, *J. Combin. Theory* **7** (1969) 230–238. ⇒53

[7] G. O. H. Katona, A theorem of finite sets, *Proc. Tihany Conference*, New York, 1966, pp. 187–207. ⇒53

[8] J. Kruskal, The number of simplices in a complex. In: *Mathematical Optimization Techniques*, Berkeley, Los Angeles, Univ. of California Press, 1963, 251–278. ⇒53

Received: February 5, 2013 • Revised: May 31, 2013

Acta Univ. Sapientiae, Informatica, **5,** 1 (2013) 63–76

Location-stamp for GPS coordinates

Éva ÁDÁMKÓ
University of Debrecen
Faculty of Informatics
email: adamko.eva@inf.unideb.hu

Attila PETHŐ
University of Debrecen
Faculty of Informatics
email: petho.attila@inf.unideb.hu

Abstract. Anybody can make a time information authentic easily nowadays with the help of a time-stamp by a Certification Authority. In this paper, we propose a similar service for mobile devices—which have GPS receiver–to authentication GPS coordinates. This service name is location-stamping, and we propose two protocols for this service.

1 Introduction

Nowadays the applications which based on the Global Positioning System (GPS) gain more and more place for themselves, like the locating, passenger guiding, traffic controlling, tracking or navigation system. These services became vital parts of our everyday life. A big percentage of the mobile devices have a GPS receiver too.

In 2001 Alf Zugenmaier and Matthias Kabatnik in [8] introduced a location-stamp service for mobile telephone network. Their solution authenticates cell information. The usage of the protocol mentioned in this article is impossible in the case of GPS because the Certification Authority and the Location Measurement System carry on two-way communication with each other. In our case the Location Measurement System is the GPS, where the communication is one-way, because the satellite only can send the information, but can't receive it. In this article we propose two solutions for the cryptographic authentication of GPS coordinates. The basic idea is that the data received and/or

Computing Classification System 1998: K.6.5
Mathematics Subject Classification 2010: 68P30
Key words and phrases: location authentication, GPS, location stamp

computed by the GPS device are sent to a trusted organization, which can be for example a certification authority. Of course the GPS device digitally signs the message. If this information is compatible with other information available by the organization, then it signs the GPS coordinates. The basic difference between the two solutions is that while in the first case we assume that we have access to the row data sent by the satellites and received by the GPS device, thus they can be signed; however in the second case we have access only to the computed GPS coordinates. Thus the first protocol is favorable, but the second is safe enough in most applications. This paper is the essentially revised and extended version of our publication [1].

The paper is organized as follows. In Section 2 we give a brief introduction to the basics of Global Positioning Systems (GPS). In Section 3 we compare the geodesic and cryptographic notion of authenticity. Furthermore we describe situations when cryptographic authentication of GPS data may be important. Finally in Section 4 we present two protocols, which are able to solve the basic problem of authentication of GPS coordinates.

2 Global Positioning System

"The Global Positioning System (GPS) is a space-based global navigation satellite system that provides location and time information anywhere on or near the Earth." [3] The position calculation is based on trilateration with satellites orbiting in space, and the measuring system is a receiver, which communicates with the satellites through radio-waves. This communication is one way because the satellites only can send the information, but cannot receive it. Several software exist which are able to calculate the GPS coordinates from the "raw" data which come from the satellites. For this reason, a GPS receiver calculates the actual coordinates with a special software, from the data received from the satellites through radio-waves. The Global Positioning System has three distinct segments. These segments are the space segment, the control segment and the user segment. The space segment formed from a constellation of 24 satellites. The control segment consists of the stations, which control the satellites from the Earth and last the user segment means anybody, who receives the data which come from the satellites. See more about the Global Positioning System in the following books [3, 2].

3 Authenticity

It is very important to define the differences between the geodesic and the cryptographic authenticity, to understand the aims of this paper.

3.1 Geodesic authenticity

If we measure the position of an object, we can do it accurate with the help of the Global Positioning System. But this accuracy can mean different measuring result for different people. "For example to a hiker or soldier in the desert, accurate means about 15m. To a ship in coastal waters, accurate means 5m. To a land surveyor, accurate means 1cm or less. GPS can be used to achieve all of these accuracies in all of these applications, the difference being the type of GPS receiver used and the technique employed." [4] In some cases we need extremely high accuracy. For example in the OPERA project of CERN the distance of the source of the CNGS neutrino beams at CERN and the OPERA detector at Gran Sasso, which is about 730 km was measured with a precision of 20 cm. [3] It is possible to correct the calculations with a lot of different methods. So we may declare that the GPS coordinates of an object are more accurate from geodesy viewpoint when we can calculate them with much smaller difference. In other words, it means, we can calculate them more precisely. To improve the precision of GPS coordinates the mathematical methods must be made more accurate. The devices are validated regularly, which means that their results are compared with the results of authentic devices.

3.2 Cryptographic authenticity

In this paper we deal with the cryptographic authenticity of GPS coordinates. We use a lot of data while we work with the GPS system, for example the raw data arriving from a satellite, the time information or the calculated coordinates. The task of cryptography is to prevent these data from changing during the process of the calculations. The changes can be made by for example a malicious person, a virus, a modified device or modified software sometimes it may happen by chance.

The cryptographic authenticity—as any other prevention—costs time and money. Only such data should be prevented, which are worth enough. We should ask the question: is the authenticity of these data important? The answer depends on the application. It is true that for example in passenger guiding, or in navigation not really important the above-mentioned certainty because the calculation of GPS coordinates takes so little time and it happens so often that there is no chance to change them and there is no point in changing them too. At the topic of vehicle-tracking, certainty-problem also forced a little bit, but for example, if you would like to prove with GPS coor-

dinates that the fences of your neighbor is on your place it is quite certain that you will need some kind of authentication before the official establishments to prove your truth. Until in an argument of a parcel the Land Register proceed, and make exact measurement, but it cannot do in any cases due to capacity problems.

For the foregoing case here is an example, and this will be our main example: A supervisor of the Land Registry finds a field where the ragweed had proliferated. He wants to fine the owner. To prove his truth he makes an official report:

- locates the area with a GPS device,
- signs the report digitally,
- and asks for an authentic time stamp.

After all these, he proves when the report was made and that he made it. If the aforesaid cases come on for trial, then neither the penalized driver nor the civil servant can prove that their GPS coordinates are match the place where they made it. Despite the fact that the supervisor certified the location information by his digital signature.

They cannot prove that the location information is correct because not a single people or device is fully trusted too. Electronically saved or transmitted data can be changed easily, thus without some kind of provable signature the authenticity of data is always questionable. That is why we need some kind of service, which helps us to certify our place authentically, by GPS coordinates.

In this article we suggest a solution to this problem. Our solution that is the location-stamp may be similar to the time-stamp for the above problems. The time stamp is provided by a Certification Authority. The plan with the location stamp is similar, we need an organization that is independent from the measurement and can guarantee that nobody modified the results we got. We think this location stamp service workable by a Certification Authority too. Actually, the aim is that we should make us independent from the person who makes the measurement and the device which makes it.

4 Problem formulation

As for the precision of the device, there are several options for the cryptographic authenticity:

- First we trust the person and the device in all cases. We accept the measured coordinates, and the time provided by the device.

- Second we trust the person and the device, but we do not trust the time information of the device. We accept the measured coordinates, but we do not accept the time provided by the device.

- Third we neither trust the person nor the device. We neither accept the measured coordinates nor the time provided by the device.

In this paper we only deal with this third option and we introduce two solutions for that option. First there is a higher safety solution, which is the driver-level solution, and then there is a lower safety solution that is the software-level solution. The differences between these two solutions are the following: in the first case the authentic software is built in the driver level of a mobile device. The data are signed immediately the device received them. Thus the provided security is as high as possible. On the other hand it is very hardware dependent. In the second case the authentic software is on the level of the operating system. Its security is a little bit less than in the other solution. Thus we complement this version with a trilateration for the mobile device for added security.

We use the following cryptographic primitives during the work of the protocols: digital signature, hash function and time stamp. We do not detail the working of these cryptographic primitives here, but we recommend the book [7] for the interested readers for the easier understanding.

5 Protocols

5.1 High-safety solution: Driver-level

In this solution our aim is to get the raw data before somebody could modify them. We try to build our authentic software in a very deep layer of the process, so we would like to build it in the driver level.

5.1.1 Participants and notations

GPS is the Global Positioning System. The satellites of this system provide the data from which the GPS receiver calculates the coordinates of the actual position.

MD is the Mobile Device. The device with a GPS receiver, we make the positioning and the authentication with the help of this device.

CA is the Certification Authority. It provides the authenticate time and

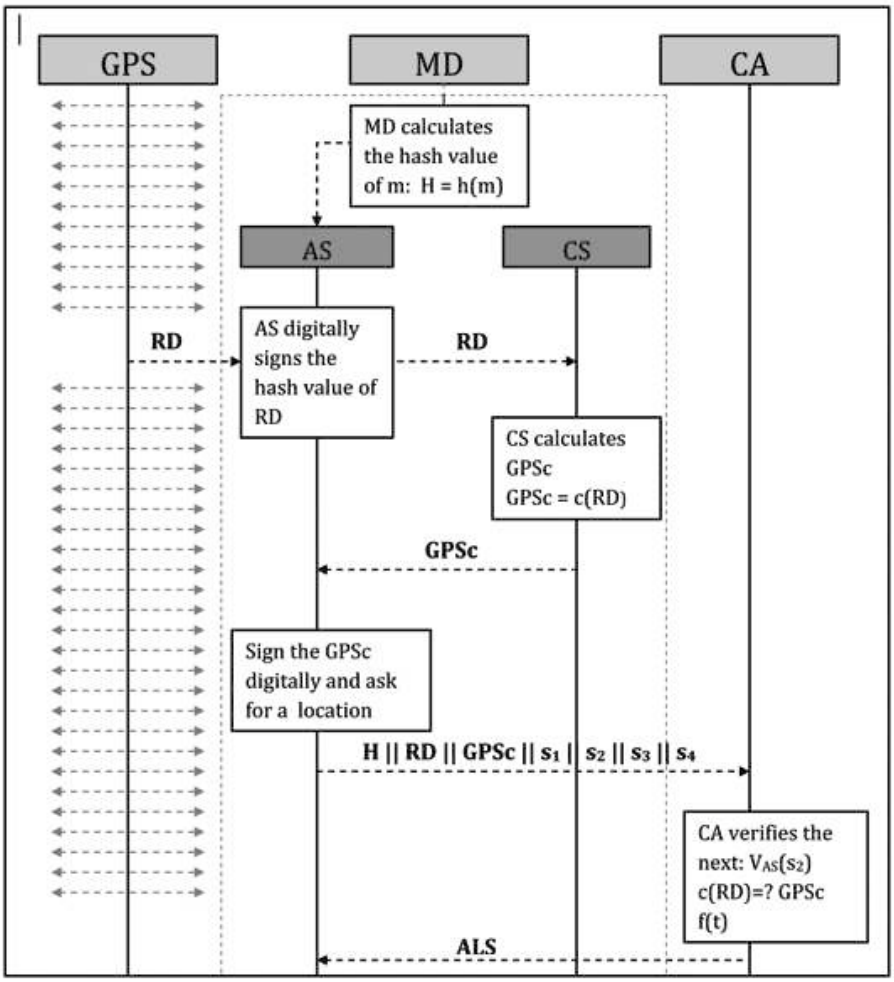


Figure 1: Driver-level protocol

location stamp, this is an organization, which is independent from the measurement and can guarantee that nobody modified the results we got.

AS is the Authentic Software. This software makes the authentication for the raw data (which come from the satellites) and for the calculated GPS coordinates.

CS is the Calculator Software. This software calculates the GPS coordinates of the actual position from the raw data come from the satellites.

M is the text or photo or some other data, that we want to authenticate with a location-stamp.

$h(\dots)$ is the hash function.

H is the hash value of the M.

$c(\dots)$ is the calculator function, which calculates the current position from the raw data that come from the satellites.

RD is the raw data come from one of the satellites of Global Positioning System.

$GPSc$ is the GPS coordinate calculated by the calculator software.

$\mathsf{S}_{\mathsf{AS}}(\dots)$ is the signature of the data in parenthesis with the private key of the authentic software.

$\mathsf{S}_{\mathsf{CA}}(\dots)$ is the signature of the data in parenthesis with the private key of the certification authority.

$\mathsf{V}_{\mathsf{AS}}(\dots)$ is the verification of the data in parenthesis with the public key of the authentic software.

$\mathsf{V}_{\mathsf{CA}}(\dots)$ is the verification of the data in parenthesis with the public key of the certification authority.

s_{i} is the ith signed data.

$TIME$ is the time information.

t is the time information of RD.

ALS is the authentic location stamp, generated by the certification authority.

$f(\dots)$ is the freshness checking function.

5.1.2 Protocol

1. MD calculates the hash value of $\mathsf{M} : \mathsf{H} = \mathsf{h}(\mathsf{M})$
2. $\mathsf{MD} \rightarrow \mathsf{AS} : \mathsf{H}\|\mathsf{M}$
3. AS digitally signs H with its private key: $\mathsf{s}_1 = \mathsf{S}_{AS}(\mathsf{H})$
4. $\mathsf{GPS} \rightarrow \mathsf{AS} : \mathsf{RD}$
5. AS digitally signs the hash value of RD with its private key: $\mathsf{s}_2 = \mathsf{S}_{\mathsf{AS}}(\mathsf{h}(\mathsf{RD}))$
6. $\mathsf{AS} \rightarrow \mathsf{CS} : \mathsf{RD}$
7. CS calculates the actual position from RD : $\mathsf{GPSc} = \mathsf{c}(\mathsf{RD})$
8. $\mathsf{AS} \leftarrow \mathsf{CS} : \mathsf{GPSC}$
9. AS digitally signs the hash value of GPSc with its private key: $\mathsf{s}_3 = \mathsf{S}_{\mathsf{AS}}(\mathsf{h}(\mathsf{GPSc}))$

10. AS concatenates H, s_1 , RD, s_2 , GPSc and s_3 and takes its hash value and then digitally signs this hash value with its private key:
$s_4 = S_{AS}(h(H||RD||GPSc||s_1||s_2||s_3))$
11. $AS \rightarrow CA : H||RD||GPSc||s_1||s_2||s_3||s_4$
12. CA verifies that the raw data were signed by AS and CA verifies that GPSc can be computed from RD, and checks the freshness of the t.
$V_{AS}(s_2)$
$c(RD) = ?GPSc$
$f(t)$

    12.1. if the answer is true for all questions, then CA makes the authentic location-stamp:
    $ALS = TIME||S_{CA}(h(H||RD||GPSc||s_1||s_2||s_3||s_4||TIME))$
    $AS \leftarrow CA : ALS$

        12.1.1. AS verifies that really the CA signed the location-stamp that it got:
        $V_{CA}(ALS)$

            12.1.1.1. if the answer is true, then AS accepts the authentic location-stamp

            12.1.1.2. if the answer is false, then AS starts a new location-stamp request with step 4.

    12.2. if the answer is false, then CA rejects to generate the authentic location-stamp
    $AS \leftarrow CA : rejection$

5.1.3 Protocol description

The protocol, described in the previous subsection, have three important participants, these are the satellites of the Global Positioning System, the mobile device and the certification authority. The mobile device generates a print of the data,—we want to stamp with an authentic location stamp—initially with an eligible hash function, this is necessary because of the digital signing. After this the authentic software, which is built in the driver of the mobile device, gets the data from the three GPS satellites, and then it digitally signs these data with its own private key presently. This signing is required in order that nobody is able to falsify during the computational process the raw data arriving from the satellites. The authentic software located in the driver of the mobile device so it can protect the data from the attack of any software

installed on the operation system of the mobile device. Once the authentic software digitally signed and stored the raw data, sends it to the calculator software. The calculator software calculates the current GPS coordinates from the present raw data and sends back the result to the authentic software. The authentic software digitally signs these data too. Now we arrived at the point that the authentic software is able to ask for an authentic time stamp from the certification authority. So the authentic software sends a request to the certification authority, this request contains the data hash value, the raw data and the calculated coordinates concatenated and digitally signed with its own private key. Then the certification authority generates a nonce value in order to ensure the freshness of the protocol and gives it back to the software. The authentic software appends the nonce to the previous request and turns it back to the certification authority, which checks that really the authentic software sent the request. If the result of the verification is right then the certification authority checks that GPSc can be computed from the raw data, if the answer is true, then it generates the location stamp, which also includes a time stamp too.

5.2 Lower-safety solution: Software-level

The protocol of the previous section is hardware dependent. This is because we signed the raw data received by the GPS device from the satellites. Our aim in the sequel is to describe a less hardware dependent authentication. Thus we cannot assume to have access to the raw data, but only calculated GPS coordinates. Hence to authenticate the data of the GPS device, the trusted organization has to have own data which it can compare with received ones. The mobile phone services have cell information, but they are usually not accurate enough to fix the location the GPS device. By our knowledge this is possible in bigger cities where the mobile network coverage is broad enough. Then the mobile phone service has independent information on the location of the GPS device, which can be compared to the data it sends to the trusted organization. This second protocol is only applicable if the above assumption holds. After this preparation we present the details of the protocol.

5.2.1 Participants and notations

Here we mention only those symbols which differ from participants or notations in the previous protocol, other symbols denote the same as above. *MPSP* is the Mobile Phone Service Provider, this provides the cell information for a mobile identifier.

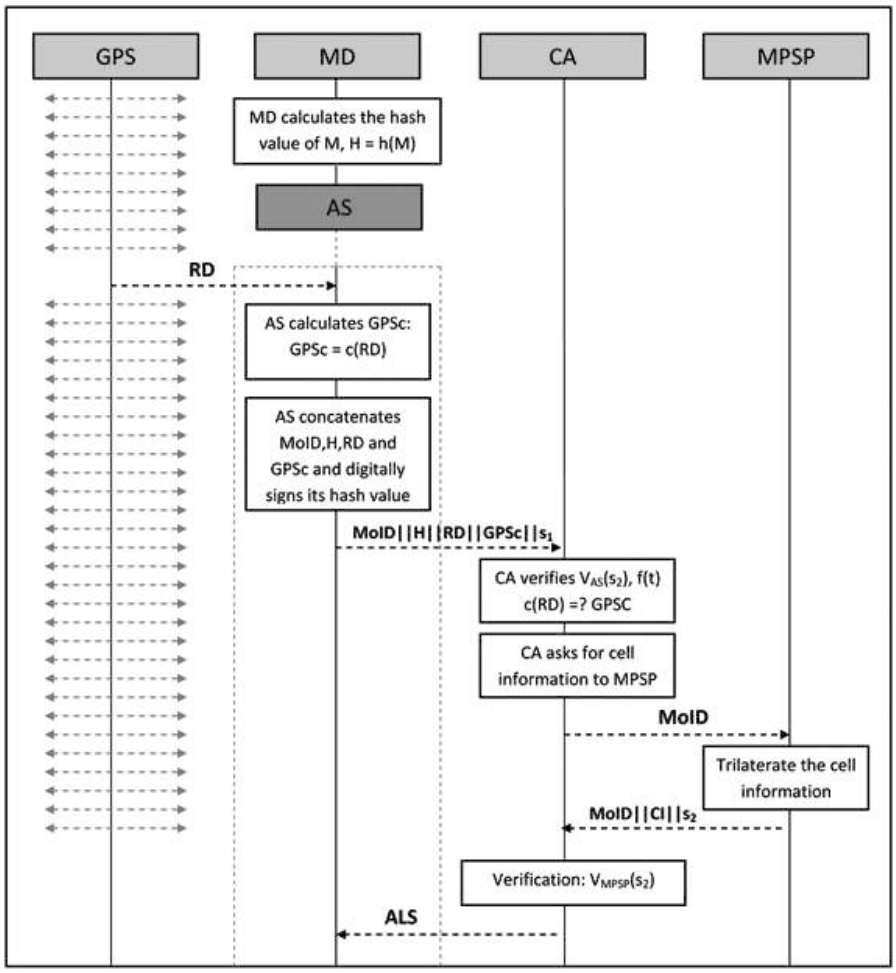


Figure 2: Sotware-level protocol

AS is the Authentic Software. This software makes the authentication for the calculated GPS coordinates.

MoID is the Mobil identifier, a number from which the MPSP can identify the current mobile device.

$t(\dots)$ is the trilateration function, trilaterates the CI from the MoID.

CI is the cell information from the MPSP.

$ck(\dots)$ is the checking function, which checks if a GPS coordinates are in

the area which is defined by the cell information.

$S_{MPSP}(\ldots)$ is the signature of the data in parenthesis with the private key of the mobile phone service provider.

$V_{MPSP}(\ldots)$ is the verification of the data in parenthesis with the public key of the mobile phone service provider.

5.2.2 Protocol

1. MD calculates the hash value of $M : H = h(M)$
2. $MD \rightarrow AS : H||M$
3. $GPS \rightarrow AS : RD$
4. AS calculates the actual position from RD : $GPSc = c(RD)$
5. AS concatenates MoID, H, RD and GPSc and digitally signs its hash value with its private key:
   $s_1 = S_{AS}(h(MoID||H||RD||GPSc))$
6. $AS \rightarrow CA : MoID||H||RD||GPSc||s_1$
7. CA verifies that the raw data were signed by AS and CA verifies that GPSc can be computed from RD and checks the freshness of the t.
   $V_{AS}(s_1)$
   $c(RD) = ?GPSc$
   $f(t)$

   7.1. if the answer is true then:
   $CA \rightarrow MPSP : MoID$ and asks for a cell information

   7.1.1. MPSP trilaterates CI from the MoID:
   $t(MoID) = CI$
   $s_2 = S_{MPSP}(MOID||CI)$
   $CA \leftarrow MPSP : MOID||CI||s_2$

   7.1.2. CA verifies that really the MPSP signed the data, which it got, and, that the GPSc matches to the CI:
   $V_{MPSP}(s_2)$
   $ck(GPSc, CI) = true$

   7.1.2.1. if the answer is true, then CA makes the authentic location-stamp:
   $ALS = TIME||SCA(h(MoID||H||RD||GPSc||s_1||s_2||CI||TIME))$
   $AS \leftarrow CA : ALS$

   7.1.2.1.1. AS verifies that really the CA signed the location-stamp that it got:
   $V_{CA}(ALS)$

7.1.2.1.1.1 if the answer is true, then AS accepts the authentic location-stamp

7.1.2.1.1.2 if the answer is false then AS starts a new location stamp request with the step 3.

7.1.2.2. if the answer it false then CA asks a new cell information with the step

7.2. if the answer is false, then CA rejects to generate the authentic location stamp
$AS \leftarrow CA : \mathsf{rejection}$

5.2.3 Protocol description

Against the protocol, described in the previous chapter, in this part the protocol has four important participants, the satellites of the Global Positioning System, the mobile device, the certification authority and the mobile phone service provider of the actual mobile device. There are some other differences between the two protocols, in the previous solution the authentic software is built in the driver level of the mobile device, in the actual case the software is installed on the operation system of the mobile device as it usually. The mobile device initially generates a print of the data with a hash function as same as the previous case. After this the authentic software, which is installed on the mobile device, gets the data from the three GPS satellites, and calculates the current GPS coordinates from the present raw data. Then it concatenates these two values, the document hash value and the mobile device identifier and digitally signs with its own private key. After this the authentic software asks for an authentic time stamp from the certification authority with the help of these digitally signed data. The certification authority generates a nonce value in order to ensure the freshness of the protocol and gives it back to the software. The authentic software appends the nonce to the previous request and turns it back to the certification authority, which checks that really the authentic software sent the request. Now the certification authority sends the identifier of the mobile device to the mobile phone service provider, which trilaterates the cell information for the device and gives it back. In this point only the verification remains behind. If the result of the verification is right, namely the calculated GPS coordinates matches to the cell information, and the private key belongs the authenticate software, then the certification authority generates the location stamp, which also includes a time stamp too.

6 Attacks

In the field of authentication of GPS information there are two main type of the possible attacks, these are jamming and spoofing.

- Jamming

In the course of jamming the attacker try to interrupt the connection between GPS satellites and GPS receivers. It is fairly an easy task, considering that the GPS satellites are orbiting in the space 20 000 km far from the Earth. This is the first reason why the broadcast signals are not too powerful and the other reason is that this communication happens over wireless connection. Therefore, the aim of the attacker is that make the satellites inaccessible for the receivers. In order to achieve the former goals the attacker produce an obstruction into the connection.

- Spoofing

In contrast with jamming, in the course of spoofing attack the connection between satellites and receivers is in good working order. Spoofing cause a much more dangerous situation. The attacker transmits a more powerful signal than the signal broadcast by the GPS satellites. From this point the receiver will think that this modified signal is the original which comes from the satellites. The fact that the receiver will receive this modified signal means that the attacker can fake the location information of the receiver, and this can mislead the user.

Compared to some other authentication method for GPS coordinates [5, 6] none of our protocols protects against jamming and only one of them protects against spoofing, but this was not the goal that we would have liked to achieve. Neither spoofing nor jamming is relevant to our case because this protocol is intended to use in the civil service. In the case that someone disrupts or terminates information flow - so the jamming occurred -, then there is simply no data that needs to be validated. There is a low-level security against spoofing in the second protocol, but this type of attack is not probable, because these data we would like to verify are not at the high classified level. So these data are not worth so much to make sense of spoofing—falsifying the GPS signals—. If you would like to verify a high classified data, then this can easily achieved by strengthen our protocol with another anti-spoofing solution, our make some changes on one of these protocols. Maybe in the future we will use some of these solutions to amplify the security of our protocol. In summary, our solution protects data from that point that they are in the mobile device.

7 Conclusion

In this article, we describe two protocols to authenticate GPS coordinates in a mobile device, and we did not give solutions to jamming or spoofing attack. So this service can only provide against for example the following type of attacks: modified software on the mobile device (which calculate false coordinates), direct adding a fake location information to the device (by hand, or by sms, or via email) or some analogue attacks. In a second article we would like to analyze the security and complexity of these protocols.

References

[1] É. Ádámkó, A. Pethő, Helyszín-bélyegzés, hitelesített GPS koordináták, in: *Az elmélet és a gyakorlat találkozása a térinformatikában, Ed.: Dr. Lóky József*, Debrecen, Hungary, 2011, pp. 381–388. ⇒64

[2] A. El-Rabbany, *Introduction to GPS: The Global Positioning System* (2nd edition), The Artech House Press, 2006. ⇒64

[3] B. Hofmann-Wellenhof, H. Lichtenegger, J. Collins, *Global Positioning System: Theory and Practice*, The Springer Press, 1993. ⇒64, 65

[4] C. Kennedy, GPS Basics, *GeoPlane Services.* (1999), http://www.geoplane.com/gpsbasics.pdf ⇒65

[5] M. G. Kuhn, An Asymmetric Security Mechanism for Navigation Signals, *Sixth Information Hiding Workshop*, Toronto, Canada, 2004, pp. 239–252. ⇒75

[6] S. Lo, D. De Lorenzo, P. Enge, D. Akos, P. Bradley, Signal Authentication, A Secure Civil GNSS for today, *InsideGNSS, Technical Article*, 2009 September-October, pp. 30–39. ⇒75

[7] A. J. Menezes, P. C.van Oorshot, S. A. Vanstone, *Handbook of applied cryptography*, The CRC Press, 1997. ⇒67

[8] A. Zugenmaier, M. Kabatnik, Location stamps for digital signatures: a new service for mobile telephone networks, *ICN'01 Proc. First International Conference on Networking, Part 2*, Colmar, France, 2001, pp. 20–30. ⇒63

Received: March 19, 2013 • Revised: June 1, 2013

Acta Univ. Sapientiae, Informatica, **5,** 1 (2013) 77–102

Test software quality issues and connections to international standards

Attila KOVÁCS
Eötvös Loránd University
Faculty of Informatics
email:
attila.kovacs@compalg.inf.elte.hu

Kristóf SZABADOS[1]
Eötvös Loránd University
Faculty of Informatics
email:
kristof.szabados@ericsson.com

Abstract. This paper examines how ISO/IEC 9126-1 and ISO/IEC 25010 quality models can be applied to software testing products in industrial environment. We present a set of code smells for test systems written in TTCN-3 and their categorization according to quality model standards. We demonstrate our measurements on industrial and ETSI projects, and provide a method for estimating their effects on product risks in current projects.

1 Introduction

In our fast changing world the usage of electrical devices belongs to the everyday life of the society. These devices contain software helping the navigation to destinations, supporting the communication with other people, driving the production, distribution and consumption of energy resources. Software drives companies, trades on the markets, takes care of people's health.

All of these systems must fulfill very strict (but different) quality restrictions. In the telecommunication area "five nines" (99.999%) availability allows only 5.26 minutes downtime per year — often including planned upgrades and maintenance. Companies producing these systems perform strategically several

Computing Classification System 1998: D.2.9
Mathematics Subject Classification 2010: 68Q60
Key words and phrases: ISO/IEC 9126, ISO/IEC 25010, TTCN-3, test system architecture, software quality
[1]Corresponding author

activities to ensure the required level of quality. In 1986 the software engineer Barry Boehm observed that the cost of detecting, reporting and correcting defects increases exponentially by the time they are found in the software development process [2]. At that time the overall magnitude of software costs was estimated to be roughly $140 billion per year, worldwide. Since then the size and complexity of software systems have been growing constantly together with the quality expectations against these systems.

Clearly, as the size and complexity of software systems grows, so does the size and complexity of their tests. Before Y2K tests were mostly designed and executed manually. Nowadays every corporation aims at automating their tests which produces large scale test architectures. In the telecom area this pressure facilitated the ETSI[2] to develop a scripting language used in conformance testing of communicating systems and a specification of test infrastructure interfaces that glue abstract test scripts with concrete communication environments. This programming language standard is called TTCN-3[3] and offers potentials for reducing test maintenance costs significantly.

Companies developing complex software systems require quality standards, models and methods to define, perform and institutionalize their quality management processes. The ISO[4] and IEC[5] published the multiple standards 9126 [11] and 25000 [16] which approach software *product* quality. Other standards like ISO/IEC 9001 or CMMI[6] [4] focus on the quality of the software *processes*. GQM[7] [25] describes *measurement* techniques used in software development, while PSP[8] [27] and TSP[9] [10] aims at the human resources and personal processes used during software development.

Regarding the software testing area applying TMMi[10] organizations can improve their test processes. ISTQB[11] offers certifications of competences in software testing. A wide range of software testing tools supports the testing processes, including test management, functional and non-functional testing, etc. Yet, the theoretical research and graphical tool support lacks behind these quality trends in testing.

[2] European Telecommunications Standards Institute
[3] Testing and Test Control Notation – 3
[4] International Organization for Standardization
[5] International Electrotechnical Commission
[6] Capability Maturity Model Integration
[7] Goal Question Metric
[8] Personal Software Process
[9] Team Software Process
[10] Test Maturity Model Integration
[11] International Software Testing Qualifications Board

The paper is organized as follows. In Section 2 we present the earlier results related to our subject. In Section 3 we give a bird's-eye view on ISO/IEC 9126 and ISO/IEC 25010 models. Section 4 contains the list of code smells we have analyzed and the categorization of them. We implemented and measured the quality of the tests via these code smells on actual industrial projects and we present the findings in Section 5. Section 6 summarizes the results of this paper.

2 Earlier results and related work

In 2007 Zeiss et al. [29] published a model for test specification derived from ISO 9126 concentrating only on internal quality attributes. This model is divided into 7 main characteristics: *test effectivity*, *reliability*, *usability*, *efficiency*, *maintainability*, *portability* and *re-usability*. Each main characteristic is divided into sub-characteristics. Most of these main and sub-characteristics are re-interpreted and re-named from ISO 9126 in order to be more appropriate in the context of testing. For example the *suitability* sub-characteristic of the *functionality* main characteristic in the ISO 9126 model is renamed to *test effectivity / test coverage*. The reason for this, as they explained, is that "in the context of test specification, the *suitability* aspect is characterized by the *test coverage*. Coverage constitutes a measure for test completeness". Another important part of this work is the definition of the *re-usability* main characteristic. *Re-usability* is an important aspect in case of test specifications. The specifications of functional tests, performance tests and load tests might be different, but the test data could be reused in all of these test suites. Another example could be functional tests: once developed, they might be used for regression testing purposes.

In the present work we take a different way of looking at the tests. Our attention is to look at tests as software products. So, instead of re-interpreting the quality standards for testing, we re-interpret the testing for software product quality. The previously mentioned article chooses TTCN-3 as a *test specification language*. In this work TTCN-3 is viewed as a *programming language*. Software products written in TTCN-3 have to be analyzed in order to fulfill quality requirements by applying quality metrics. As we see the two standpoints is not contradicting but rather complementing each other.

In the paper of Bánsághi et al. [1] one of the cornerstones was the comparison of the models ISO 9126 and ISO 25010. The article comes to the conclusion that even though the new model is broader, both models suffer

from the fact that "different parties with different views of software quality can select different definitions". They state that although both of the standards offer a good frame of reference for software product quality, neither of them offer a practically applicable method for *assessing* quality. The other cornerstone was the fact that there is a wast literature proposing numerous ways of measuring software quality metrics without providing traceable and easily applicable translation to the multi-faceted notation of quality. To bridge this gap the authors analyzed the rules of the PMD [24] and FxCop [8] static source code analyzers and classified them into the quality characteristics of the standards. PMD analyzes Java source code and looks for possible code smells, like duplicate code, unused local variables, etc. FxCop analyzes managed code assemblies in the .NET framework common language runtime. They have classified 203 PMD and 225 FxCop rules into ISO 9126 and ISO 25010 categories. They have found that most rules fall into the *maintainability* characteristic in both standards.

This work tries to extend [1] with quality rules for testing systems.

The following questions arise naturally: are the TTCN-3 language and the test systems written in it complex enough for such an analysis? Is it worth to regard the test language as a programming language? The answer is yes. It was already shown [26] that current testing systems written in TTCN-3 are not only large in size, but are also very complex. The importation graph-structure of these software systems shows scale-free properties. This was one of our main motivations why the TTCN-3 language and systems need to be studied deeper.

3 Bird's-eye view on ISO/IEC product quality standards

It is well-known that specification and evaluation of software quality attributes are key factors to ensure adequate business quality. Yet, software quality is still considered to be a trait that can not be measured [17]. This attribute is based on the fact that in general quality is a multidimensional concept. The understanding of quality depends on the entity of interest, attributes of that entity and the viewpoint of the observer. In order to achieve high software quality appropriate quality characteristics must be defined and taken into account. These characteristics should drive the significant architectural and design decisions. Software quality management should help to ensure that the required level of quality is reached.

3.1 ISO/IEC 9126

ISO/IEC 9126-1 defines quality as "the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs" [11]. This standard has later been revised and separated into 4 parts: part one defines the characteristics and viewpoints of software product quality [12]; part two provides metrics for the external quality, as the totality of characteristics of the software product from an external point of view [13]; part three assesses a software system's internal quality, as the totality of characteristics of a software product from an internal point of view [14]. It provides metrics that measure the software itself. Part four evaluates metrics by the quality in use, as the end user's view of the quality of the software product, as it is used in the specific usage context [15].

ISO/IEC 9126-1 contains a two part model. The first part is applicable for modelling the internal and external quality of a product, while the second part models the quality in use of a software product. Generally, *internal quality* is understood as something attainable on reviews by doing static code analysis. Internal quality can be used early in the development process to predict the quality of the final product. *External quality* describes properties of software as it interacts with its environment. *Quality in use* is perceived by the end user who executes the software product. These product quality views represent different stages of the development, but instead of being completely independent, they influence each other.

ISO/IEC 9126-1 defines the same frame for modelling internal and external quality (shown on Figure 1), which can be instantiated for both using different set of metrics. The model itself describes 6 characteristics: *functionality*, *reliability*, *usability*, *efficiency*, *maintainability* and *portability*. Each of these characteristics has several further sub-characteristics. The importance of the characteristics varies depending on the view point concerned.

3.2 ISO/IEC 25010

ISO/IEC 9126 has been recently replaced by ISO/IEC 25010 [16]. It revises the old standard and incorporates the same characteristics with some amendments.

- The scope of quality models includes computer systems and quality in use from system perspective.
- Several new characteristics and sub-characteristics were added. Some characteristics were escalated from being sub-characteristics (security), some were removed, some were renamed to have more appropriate names.

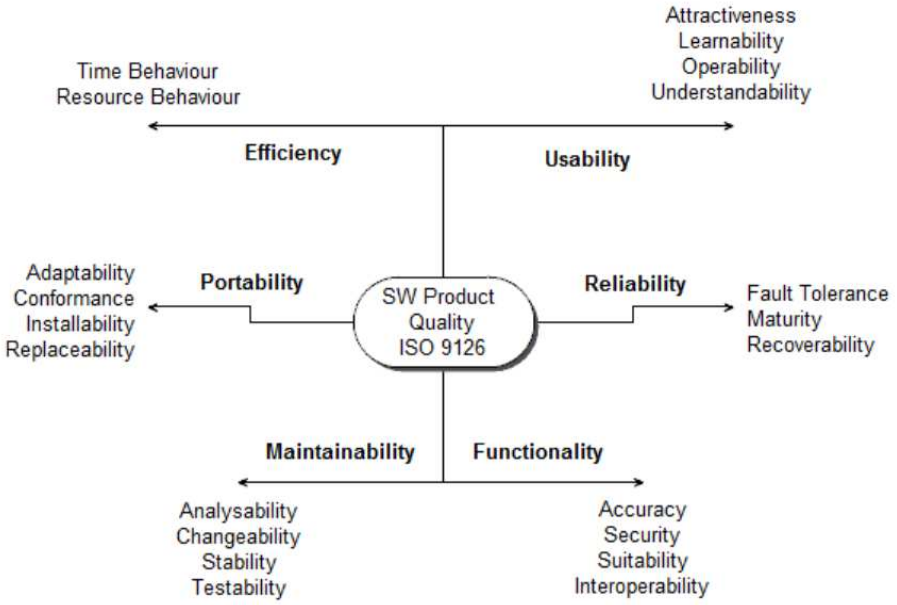


Figure 1: Quality characteristics of a software product in ISO/IEC 9126

- Internal and external quality models were combined as the product quality model.

This standard states that "this International Standard is intended to be used in conjunction with the other parts of SQuaRE series of International Standards (ISO/IEC 25000 to ISO/IEC 25099), and with ISO/IEC 14598 until superseded by the ISO/IEC 2504n series of International Standards".

ISO/IEC 25010 categorizes the product quality properties into eight characteristics: *functional suitability*, *performance efficiency*, *compatibility*, *usability*, *reliability*, *security*, *maintainability* and *portability*. With each characteristics being composed of a set of related sub-characteristics (see Figure 2).

Quality in use attributes are categorized into five characteristics: *effectiveness*, *efficiency*, *satisfaction*, *freedom from risk*, *context coverage*. The definition of quality in use compared to ISO/IEC 9126 also includes the quality of hardware, operating environment and the characteristics of users, tasks and social environment as having an effect on stakeholders.

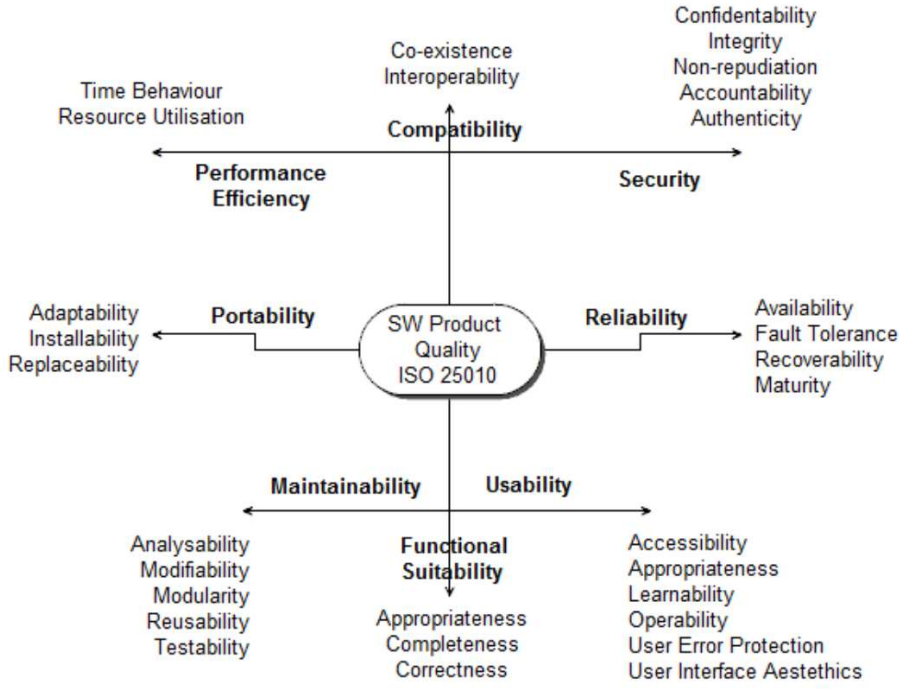


Figure 2: Software product quality according to ISO/IEC 25010

4 Code smells and categorization

As we stated earlier our starting point is different from the previous researches. They usually tried to map constructs from the testing context onto software product quality rules. Our point of view approximates from the other direction, i.e., we map software product quality constructs on software systems written in TTCN-3. We started to analyze existing systems which are actually used for testing.

Another serious difference is that we are able to consider incomplete systems: software products which are part of some larger software products (and might contain smaller software products within themselves). In this context software code itself is viewed as a product. For example a library of functions that can be used to send and receive messages on a protocol, or a framework that enables the user to do load testing (after configuring the specific messages to be sent/received). Both examples are software systems that could be used

in a standalone mode or as part of a bigger architecture. At the same time both examples share the property of being products that are used by other programmers or test designers. These software products are delivered to the customer in the form of source codes.

This point of view means that *usability* also becomes a characteristic that can be measured by a static source code analyzer. This way *usability* and *maintainability* seems to be hard to separate so we have to refine the distinction. In our understanding *maintainability* means the attributes related to maintaining the actual product, *usability* means the usage from a higher abstraction level or by an external product. In case of a TTCN-3 function, in a vague sense, this could mean that the quality attributes of the function's body most likely belong to the *maintainability* characteristic, while the quality attributes of the formal parameter list belongs to the *usability* characteristic. If a function has too many formal parameters than it may cause problems for the party trying to invoke (use) it. While the actual implementation of some feature is the internal responsibility of the product, it could be changed without creating any noticeable side effect for the external calling party.

4.1 Code smells

A *code smell* is a hint that something has gone wrong somewhere in the code. In order to find valid code smells that can be used to measure the quality of TTCN-3 testing systems we have analyzed the archives of the Quality Assurance Organization of a large telecom company and the code smells already detected by existing static analyzers in other languages.

First we have reviewed every Inspection Record (source code review documents) created from year 2006. We have also reviewed all entries in the Trouble Report (errors and problems found in released products) database. These records have proven to be very usefull for the rest of the research. All of them were code quality issues which may became show stoppers at some point in the project's life cycle.

Second, we have also reviewed the rules of PMD [24], FxCop [8], Checkstyle [3], FindBugs [6], xUnit Patterns [20], Martin Fowler's book on refactoring [7] and TRex [28] to see if there are any static analyzer rules that can be used in testing and in particular for the TTCN-3 language. Checkstyle is a highly configurable static code analyzer for Java, mostly directed to issues related to breaking naming conventions. Findbugs is an open source static code analyzer operating on Java bytecode. xUnit Patterns is an extensive collection of unit testing specific code smells. Martin Fowler's book provides an extensive list of

code smells, their description and refactoring methods for "deodorizing them". As TTCN-3 is not an object oriented language and was designed with a testing oriented mindset, most of the reviewed rules could not be applied. Another important difference is that the standard of the TTCN-3 language is changing rapidly. Visibility checking was introduced to the standard in year 2009 [5]. This change came with the *public*, *private* and *friend* words becoming keywords and no longer usable as identifiers. After the introduction of the keywords the source codes that contained them were no longer backward compilable.

TRex [22, 23] has an extensive list of code smells for TTCN-3 with the following entries: (1) Duplicated Code, (2) References, (3) Parameters, (4) Complexity, (5) Default Anomalies, (6) Test Behaviour, (7) Test Configuration, (8) Coding Standards (9) Data Flow Anomalies, (10) Miscellaneous, having altogether 38 smells. The Duplicated Code smell mentioned in this article describes 7 separate smells differentiating the dupplication of template data, local data, inline-data, component data, statement, conditional branches and Alt branches. They showed the practicability of their smells in the test systems *SIP v4.1.1*, *HiperMan v2.3.1* and *IPv6 v1.1* having altogether 61282 lines of code. Based on this work we extended and redesigned the list of code smells we found to be applicable to TTCN-3. They are as follows:

1. FIXME tags: Developer markings of severe incorrect or missing features.
2. TODO tags: Developer markings of incorrect or missing features.
3. Circular importation: The import relation of modules forms at least one loop.
4. Duplicated code: Very similar code exists in more than one location.
5. Similar functions: Several functions differing only in literal values.
6. Mergeable templates: Similar data structures, that could be merged into a single parameterized one.
7. Long statement blocks: A block of statements that has grown too large.
8. Too many parameters: A long list of formal parameters.
9. Excessively short identifiers: The name of an identifier is too short to reflect it's functions.
10. Excessively long identifier: The name of an identifier is too long.
11. Divergent naming: The identifier breaks the naming conventions.
12. "Private" group: Public definitions categorized in a group called "private".
13. Internal comments: Internal comments indicate too complicated code.
14. Missing comments: All methods should be commented.

15. Type in method name: The return type's name is redundant in the method name.
16. Module in method name: The containing module is mentioned in the method name.
17. Visibility embedded in name: Visibility rules evaluated by user.
18. Incomplete literals: Some fields of literals and constants are initialized unbound.
19. Initialize with constant: Structured value declared without initial value.
20. Dummy fields in constants: Field always overridden, should be left unbound.
21. Goto detection: Goto is considered to break structured programming rules.
22. Unnecessary imports: Module importations that are unnecessary.
23. Unused global definitions: Some global definitions are not used.
24. Unused local definitions: Some local definitions are not used.
25. Unnecessary operations: Operations never executed.
26. Unchecked module parameter: The module parameter is used before being checked.
27. Push definition to component: Functions running on a component define the same local variable.
28. Pull definition to local: A component member is only used in a few functions.
29. Unused return value: The result or error handling of the function call is missing.
30. Unused started return value: Information sent back is not reachable.
31. Infinite loops: Loops the code could not exit from.
32. Busy wait: Waiting for message in an event based system with polling.
33. Non-private private definitions: Public definitions used only internally.
34. Excessive rotation size: List rotation size should not exceed the size of the list.
35. Consequtive assignments to an entity: Assignments could be merged to a single assignment.
36. Sequential "if" statements: If possible should be changed to "if-else" conditions.
37. Size check in loop limit: The size of an unchanged list is checked in every iteration.
38. Reused loop variables: Loop variable declared and used outside the loop.
39. Unnecessary condition: The condition can be evaluated by the static analyzer.

40. Conditional complexity: Too large conditional logic blocks.
41. Explicit condition check: Explicitly check the value of a boolean condition.
42. Boolean evaluation with branching: All of the branches only set a single logical value.
43. Mergeable conditions: Consecutive conditionals do exactly the same operations.
44. If without else: In testing software all execution paths should be handled, at least logged.
45. Method with single condition: All statements of a function are in a single conditional.
46. Too many branches on a value: Switching on a value with consecutive conditionals.
47. Not written inout parameter: Reference passing used when not needed.
48. Not written out parameter: Result not calculated and passed back.
49. Not written variable: Variable declaration when constant would suffice.
50. Restrictable templates: Templates that could be more restricted based on their usage, but are not.
51. Dead code: Code fragment which is executed but not used anywhere.
52. Code commented out: Instead of removing it code was commented out.
53. Empty blocks: An empty code block.
54. Setverdict without reason: The testcase verdict is set without attached reason.
55. Variant outside Encodes: Encoding variants are specified without context.
56. Functions containing Stop: The execution is stopped inside a function, instead of the testcase.
57. Valueof used with value: The valueof function (used to convert a template to a value) is used with a value parameter.
58. Magic number: Numeric literals in the code.
59. Magic string: String literals inside the code.
60. XML tags in strings: XML encoding is simulated via string manipulation.
61. Nested block depth: The nesting of constructs exceeded a given level.
62. Indicent exposure: Too much of the module is exposed to the public.
63. Inappropriate intimacy: Dependencies on other module's implementation details. Functions using definitions only from an other module should be moved there. Members used only by a single external module should be moved there.
64. Feature envy: The function uses only an other module's attributes.

65. Divergent change: Changes touch completely different parts of a module.
66. Shotgun surgery: A change requires several changes in several modules.
67. PTC created, not started: A Parallel component is not started.
68. Isolated PTC: A parallel component is not connected to the test system.
69. Un-needed "runs on": There is no need for restricting a function to a specific component.
70. Contrieved complexity: Complex design patterns, where simpler would suffice.
71. Incorrect indentation: The code is not well indented.
72. Divergent naming of files: The names of files does not follow the naming conventions.
73. Incorrect pre-processability indication: Pre-processablity is not indicated in file extension.
74. Ordering of definitions: Definitions declared out of order.
75. Filling in values one-by-one: Structured value is filled in in several statements.
76. Private definitions published: A public function returns with a private definition creating a potential security hole.
77. Floating point equality check: Floating point numbers should not be compard directly.
78. Public/private keywords: The public/private keywords are used as identifiers.
79. Select without default branch: A select statement does not have "case else" branch.
80. Switch density: The ratio of branches are too high in the code.
81. Logic inversion: the whole conditional expression is negated.
82. Cyclometric complexity: The number of decision points in a method, plus one for the method entry.
83. NPath complexity: The number of acyclic execution paths in a method. Similar to Cyclometric complexity, but also takes into account the nesting of statements.
84. Break/continue usage: Break and continue statements are used incorrectly.
85. Unreachable code: A part of the code that can not be reached.
86. Using "*" for mandatory fields: Optionality is indicated for a mandatory field.

4.2 Categorization

The classification process was a technical review. In the review meeting each rule was discussed and decisions were made for belonging or not to some characteristic and sub-characteristic class. Each rule was categorized into the class to which it most likely belongs. Most likely means that more than 66% of the review meeting members agreed. In this way there were several rules which fell into multiple categories. For example the rule "infinite loops" belongs to *functionality/suitability* as most likely the program was not intended to operate like that, while it also belongs to the *efficiency/time behavior* since a program running in an infinite loop is most likely wasting resources. During the review we have agreed not to categorize the "FIXME / TODO tags" rule. The content and severity of this rule depends on the information the developers wished to make visible. As such each instance may belong to any of the characteristics, completely independently from any other instance. The result of the categorization review can be seen on Figure 3 and Figure 4.

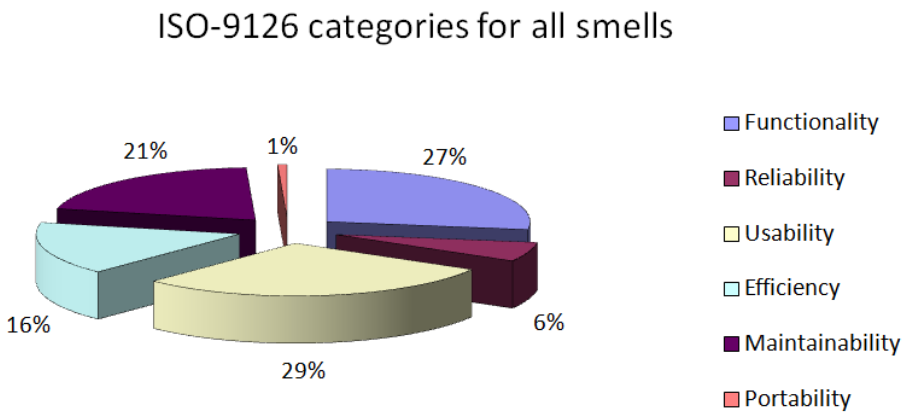


Figure 3: Code smells categorized according to ISO/IEC 9126-1

It can be observed that some of the categories of ISO/IEC 9126 kept their coverage ratio when the smells were categorized against ISO/IEC 25010, while others were re-partitioned. *Functionality*, *reliability*, *efficiency* and *portability* were almost identically mapped with respect to the categories of *functional suitability*, *reliability*, *performance efficiency* and *portability*. While the contents of the *usability* and *maintainability* categories were distributed among

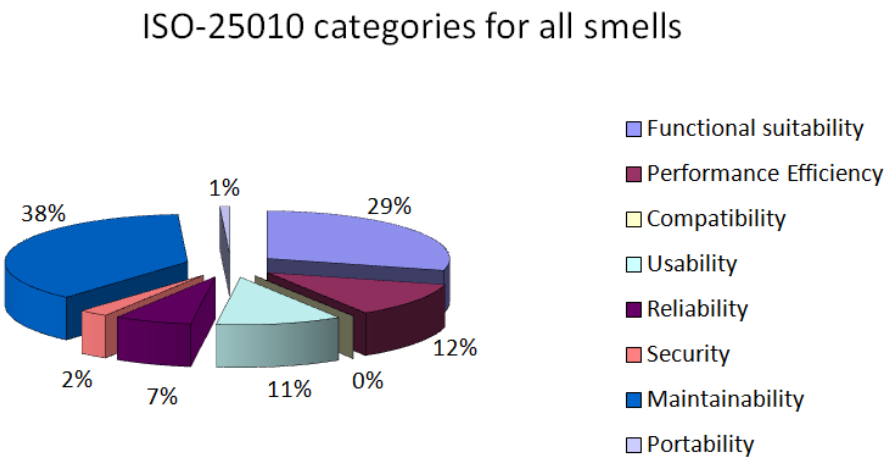


Figure 4: Code smells categorized according to ISO/IEC 25010

the *usability*, *security* and *maintainability* with much more focus placed on *maintainability*. The reason behind this comes from our understanding of these categories. For example the code smells "unused global definitions" and "unused local definitions" were categorized into *functionality* and *usability* according to ISO/IEC 9126. When evaluated against ISO/IEC 25010 we found them to be fitting only into the *functional suitability/completeness* category. This categorization is still valid, since having unused definitions (either global or local) in the source code can indicate that something might not work correctly, since some elements are unused that were planned to be used. The question of why we choose to not fit these smells into the *usability* category is somewhat less obvious. ISO/IEC 9126 defines *usability/understandability* as "attributes of software that bear on the users' effort for recognizing the logical and its applicability". The decision in the review meeting was that unused global and local definitions fall into this category, since for someone who should like to understand the relations and operations of such code, these definitions provides confusion and a large amount of extra work. ISO/IEC 25010 defines *usability* as the "degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". Although unused definitions decrease the effectiveness of using a software framework, we decided that it may have just a little obstacle to actually hinder usability. In modern IDEs the effect of these

smells is only having a few more elements in the tools "code completion" or "recommendation" lists which can quickly be overcome by the user.

Another example is the case of the code smell "conditional complexity". It is clear that this is a maintenance related problem according to both standards. Yet, we decided that this code smell only fits the *usability* category in ISO/IEC 9126, but not in ISO/IEC 25010. While "conditional complexity" provides problems for users of the functionality willing to understand its internal operations, it is not necessarily hindering them to achieve specific goals in a single specified context of use.

5 Measuring the test quality via code smells

To make sure that our code smells are useful we have analyzed 16 projects. All projects were real life projects at a large telecommunication company except 6 of them which were standard ETSI TTCN-3 test projects.

The size range of this set of projects[12] stretches from 21 174 to 794 829 with the average of 205 438 effective lines of code. The number of modules per project varies from 4 to 583 with the average of 174 modules. The average module size was 1183 lines of TTCN-3 code. Some projects contain only a few tests for very simple systems, others are large frameworks with complex tests. Some of these are organized in a single project, others form hierarchies of sub-projects. We believe that this range is not only the widest studied so far, but should also cover most usages and ways of working with TTCN-3.

It is important to mention that the measurements data were gathered in the time frame of a full year. For this reason the results do not represent the general level of quality at any given point of time.

In the time-frame of the research project we were able to implement and measure 35 code smells. Most of the measured code smells are valuable as they point out existing issues. In fact, most of them were present in the projects in a large quantity. There was only a single code smell (excessive rotation size) which was not manifesting in any analyzed projects. Other smells either manifested in almost every project, or were localized in only a few ones.

1 **FIXME tags:** Together there were 121 occurrences, 91% of them were in 3 projects.
2 **TODO tags:** There were 522 occurrences, 78% of them were in 4 projects.

[12] These were complete projects, i.e., a project were considered together with all its sub-projects

3 Circular importation: There were 70 occurrences. In one of the projects 25% of the modules belonged to a circular import cycle.

7 Long statement blocks[13]**:** There were 499 occurrences, 97% in two projects. One of them had a hit in every 600 lines in average.

8 Too many parameters[14]**:** There were 1648 occurrences, we have found hits in almost each project proportionally to their sizes. There were only two projects without this warning sign.

11 Divergent naming: There were 244480 occurrences, 79% of them in 5 projects. There was a project with extremely high hit ratio comparing to its size (more than 50%).

14 Missing comments: There were 2202 occurrences, 75% of them in three projects. The highest hits/project size ratio was in one of the smallest project.

15 Type in method name: There were 12343 occurrences, mainly proportional to the sizes of the projects, except one case: the fourth biggest project had no hits to this smell.

16 Module in method name: There were 731 occurrences, 81% of them in three projects. There were 5 projects without any hits.

17 Visibility embedded in name: There were 250 occurrences, 75% in two projects. One of them, a small sized project, had 88 hits. There were 11 projects without any occurrences.

19 Initialize with constant: We found together 61333 occurrences, 66% of them were in two projects. One of them (a middle sized project) had 42% of all hits.

21 Goto detection: We found occurrences in 6 projects, one (middle sized) project had the 98% of the hits.

22 Unused imports: There were 25597 occurrences, 90% in the two largest projects.

23 Unused global definitions: There were 26032 occurrences, 61% in three of the four largest projects. It is interesting to note that there were not any hits in the third largest project.

24 Unused local definitions: There were together 9717 occurrences, 57% in the largest project. In two (a large and a medium sized) projects there were no hits at all.

29 Unused return value: There were 5426 occurrences, 66% in two (a small and a middle sized) projects. In 9 projects there were no hits at all.

30 Unused started return value: There were 416 occurrences in two projects, basically one of them (a middle sized project) had all of the hits.

31 Infinite loops: There were 35 occurrences in 9 projects, 24 hits in two (a large and a middle sized) projects.
32 Busy wait: There was only one occurrence.
33 Non-private private definitions: There were 14390 occurrences, the second largest project has 21% of them. It must be noted that a middle sized project had 11% of all the hits.
34 Excessive rotation size: There was no occurrence at all.
37 Size check in loop limit: There were 3251 occurrences, 85% in three of the four largest project. In the third largest project there was only 1 hits.
40 Conditional complexity: There were together 3089 occurrences, 72% in three projects. One of them (a middle sized project) had 38% of the hits.
47 Not written inout parameter: There were 771 occurrences, 62% in two projects. The second largest project had 40% of all hits.
48 Not written out parameter: There were 68 occurrences, 55% of them in two large projects. In 6 projects there were no hits at all.
49 Not written variable: There were 36133 occurrences, 69% of them in the largest project.
53 Empty blocks: There were 3358 occurrences, 50% of them in one (middle sized) project. 63% of all the hits were in two projects.
54 Setverdict without reason: There were 15525 occurrences, 72% of them were in one (middle sized) project.
55 Variant outside encode: Altogether there were 290 occurrences in 2 (a large and a middle sized) projects.
56 Functions containing STOP**:** There were 1075 occurrences, 91% of them in the third largest project. In 6 projects there were no hits at all.
57 Valueof used with value: There were 49 occurrences in two projects, almost all hits were in the largest project.
58 Magic number: There were 54592 occurrences, 64% in two (the largest and a middle sized) projects.
59 Magic string: There were together 457288 occurrences, 74% were in two (the largest and a middle sized) projects.
81 Logic inversion: There were 2019 occurrences, 67% of them in a middle sized project.
85 Unreachable code: There were only 8 occurrences in four projects.

The connection of the implemented smells to the examined international standard can be seen in the Figures (5–6).

Zhang et al. [30] provided a systematic literature review on code smells

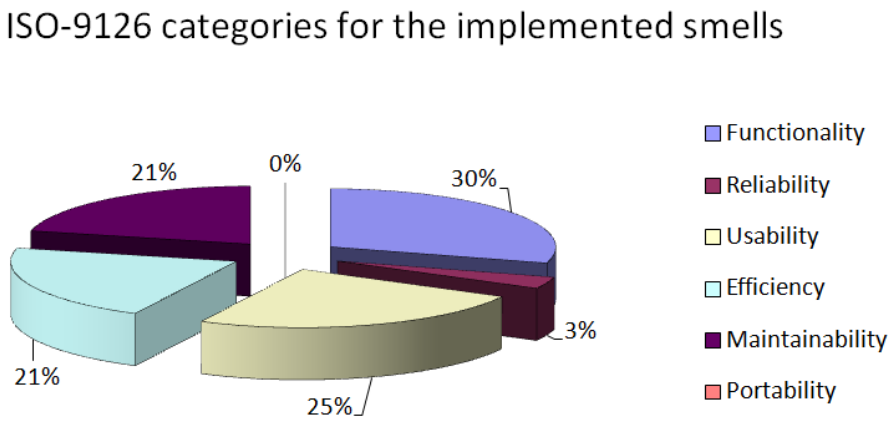


Figure 5: The implemented code smells categorized according to ISO/IEC 9126-1

and refactoring strategies based on papers published by IEEE and six leading software engineering journals from 2000 to June 2009. Nearly half of the identified papers (49%) described methods or tools to detect code smells, one-third focused on the interpretation of code smells, and 15% centered on refactoring. There were only a few study investigating the impact of code smells on maintenance or other quality attributes [21, 19, 18], but none of them were applicable to our test quality smells. We found an interesting machine learning approach for the calculation of thresholds for software metrics to evaluate quality attributes [9] but we decided to apply a pragmatic approach.

In order to have an impression about the usefulness of the previous smells we calculated the project risk factors in the usual way:

$$\text{RiskFactor}(\text{proj}) = \sum_{\text{smell}} \text{RelativeOccurrence}(\text{proj}, \text{smell}) \times \text{Impact}(\text{smell})\,.$$

For the impact estimation we used three classes:

1	– small impact,
2	– medium impact,
3	– large impact.

There were 4 smells classified into the large-impact class (with ordinal numbers from the smell enumeration): 12, 17, 18, 19; nine smells were classified into

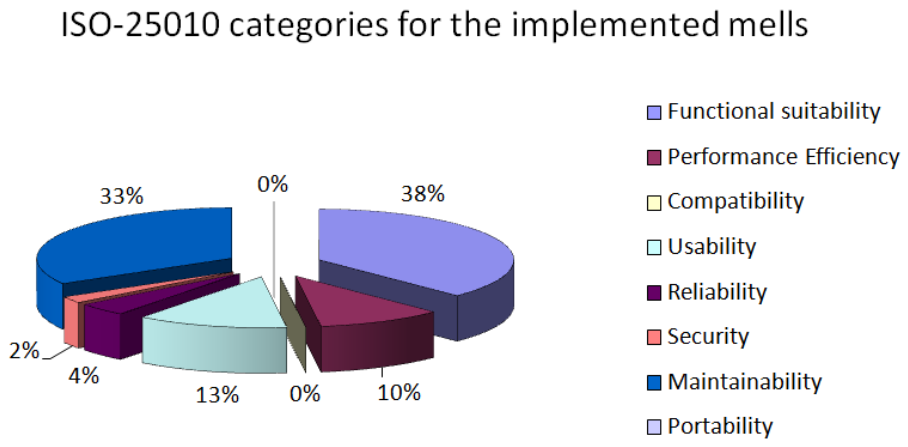


Figure 6: The implemented code smells categorized according to ISO/IEC 25010

the small impact class: $2, 3, 6, 13, 14, 20, 27, 29, 34$; all the others belonged to the medium impact category.

In order to determine the classification of the relative occurrences[15] of the smells we used *smell-baselines* on the measured data. For a smell S the smell-baseline S_b means that the smell S is acceptable to occur in every S_b effective lines of code in average. Then, we applied the following categories:

0	– no smell occurrence,
1	– rare occurrences ($S_{actual} > S_b$),
2	– occasional occurrences ($S_b \geq S_{actual} > S_b/2$),
3	– likely occurrences ($S_b/2 \geq S_{actual} > S_b/8$),
4	– frequent occurrences ($S_b/8 \geq S_{actual}$).

Here S_{actual} means the actually measured relative occurrence in a given project.

Let see an example. Based on the ETSI projects the smell-baseline for the smell *MagicNumber* is 50. In project P with size 135845 eLOC the actual (measured) value was 5657 occurrences, i.e.,

$$MagicNumber_{actual} = 135845/5657 = 24.$$

[15]Here relative occurrence means the size normalized occurrence.

Hence, this smell occurs more than twice often then the baseline, therefore

$$\text{RelativeOccurrence}(\mathsf{P},\ \textit{MagicNumber}) = 3.$$

After calculating the relative occurrences for all smells in project P we are able to determine the risk factor of this project. Then, we can compute the quality level of the project P by

very high	if	$0 < \text{RiskFactor}(\mathsf{P}) <= \mathsf{T}$,
high	if	$\mathsf{T} < \text{RiskFactor}(\mathsf{P}) \leq 2\mathsf{T}$,
medium	if	$2\mathsf{T} < \text{RiskFactor}(\mathsf{P}) \leq 3\mathsf{T}$,
low	if	$3\mathsf{T} < \text{RiskFactor}(\mathsf{P}) \leq 4\mathsf{T}$,
very low		otherwise.

The threshold T was calculated as the average value of the risk factors times $2C/5$, where the correction factor C is between 1 and 2. In our case C was 1.6.

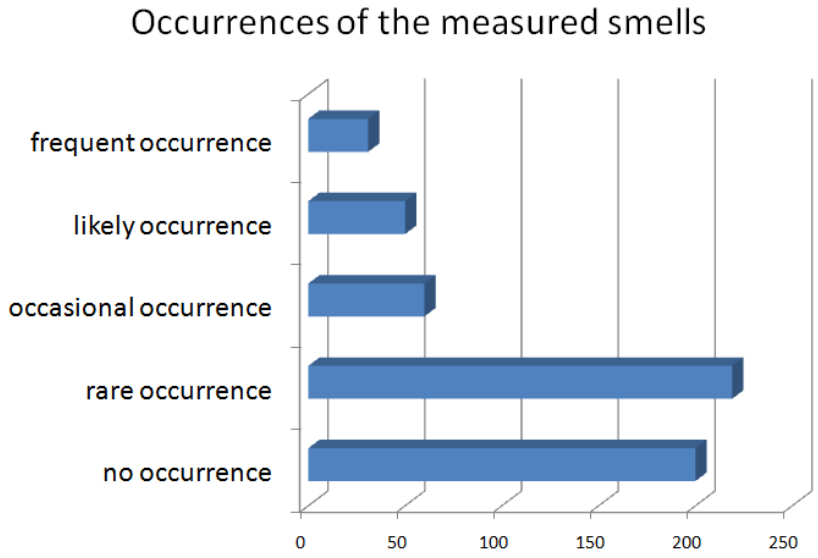


Figure 7: Relative occurrence distribution of the measured smells for all projects

The smell-baselines were determined on the basis of the 6 ETSI projects. We assumed further that the ETSI projects have good (or very good) quality, i.e., we forced them to fall into the high or very high quality category by adjusting

the correction factor C. Figure 7 shows the relative occurrence distribution of the measured projects for all smells.

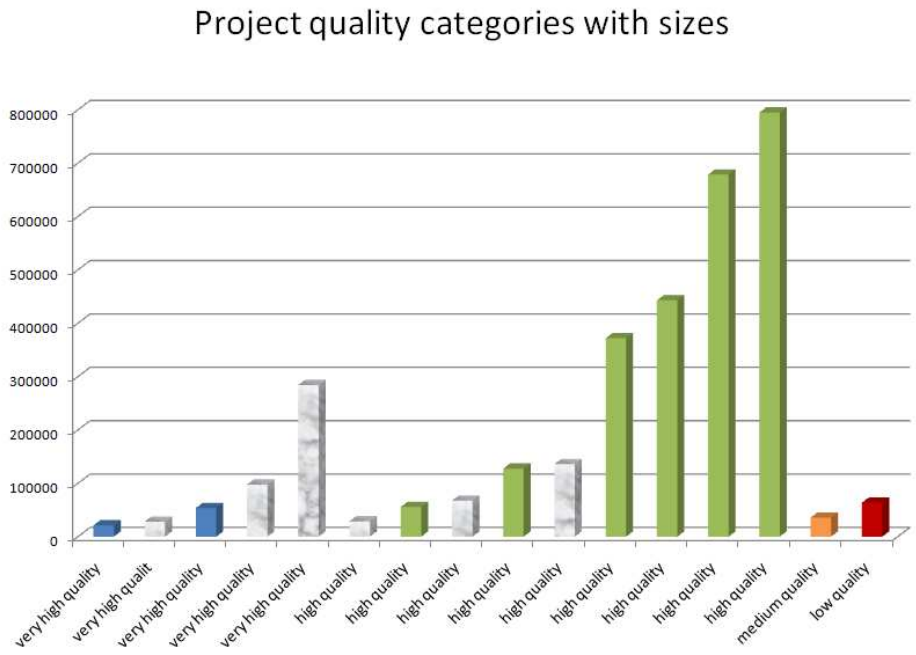


Figure 8: Project qualities with sizes

The average value of the risk factors were 67.75, hence $T = 43$. The quality categories of the projects can be seen in Figure 8. We note that we found no correlations between quality categories and project sizes, but there were no "big" project with very high quality. It must also be noted that altogether there were only one medium and one low quality project. The ETSI projects in Figure 8 are marked with marble filling.

Figure 9 and 10 show the most occurred (i.e. likely or frequently) code smell penetration for the low quality project according to the ISO 9126 and ISO 25010 models.

It would be an interesting continuation of this research to analyze the spreading of both the widely spread and the localized smells further. Local smells might be the result of misunderstanding some documentation or not sufficient knowledge transfer. It might also happen that some of the functionalities of the language are only used, by (a few) users. At the other end of the spectrum the

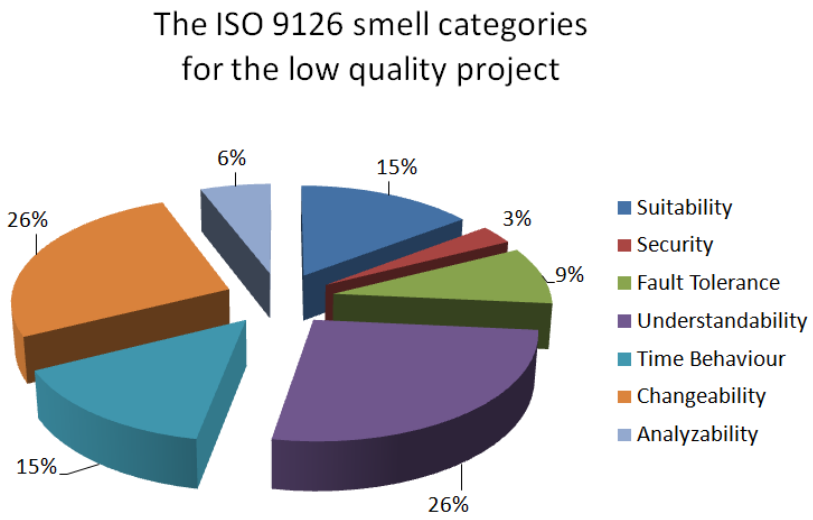


Figure 9: The most occurred code smells for the low quality project categorized according to ISO/IEC 9126-1

widely spread smells might be the results of misconceptions spreading among projects.

6 Summary

In this paper we examined the ISO/IEC 9126 and ISO/IEC 25010 software quality standards from the point of view of test software products. We analyzed how large scale test frameworks written in TTCN-3 can be adapted to these standards. We presented a list of code smells which were collected from PMD, FxCop, Checkstyle, FindBugs, TRex and code smells which were identified during reviews or trouble reports in actual industrial systems.

We conclude that the set of code smells we have shown overlaps with all main characteristics in the ISO/IEC 9126 quality standard. On the other hand they overlap all but the *compatibility* main characteristic of the ISO/IEC 25010 standard. We found that *compatibility* in the ISO/IEC 25010 meaning is not applicable in our situations. ISO/IEC 25010 defines compatibility as "degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while

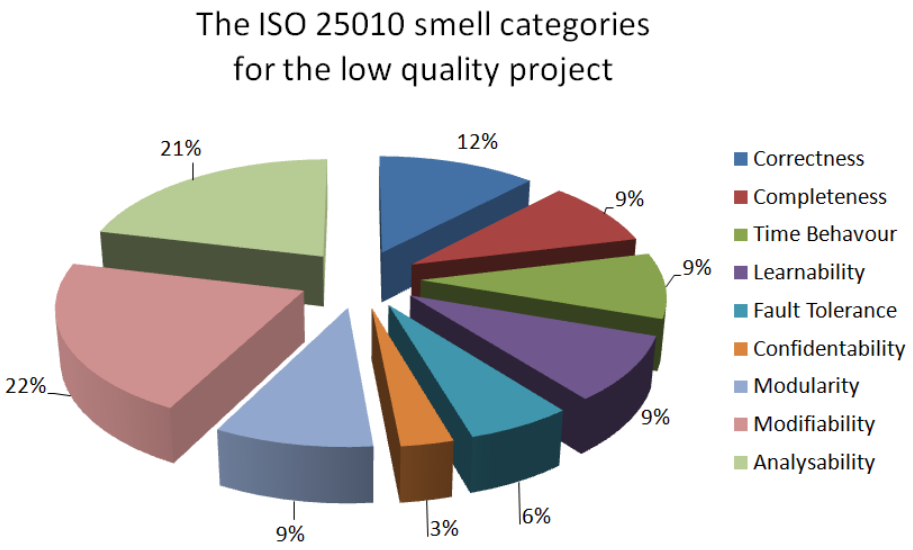


Figure 10: The most occurred code smells for the low quality project categorized according to ISO/IEC 25010

sharing the same hardware or software environment". The TTCN-3 language abstracts away all hardware and software environments standardizing the internal communications of the system.

It is interesting to see that the rules according to ISO/IEC 9126 are how well distributed. This is no longer true for ISO/IEC 25010 where *maintainability* takes up 38% of all rules while it was only 21% according to ISO/IEC 9126.

7 Further work

Our future plans involve the measurement of all of the code smells in actual projects to see how well we could *estimate* the quality of a TTCN-3 test software product. Later we would like to rise the abstraction level and include high level metrics and measure architectural quality. This way we could profile the quality of the project on several levels. We could then measure quality for designers, architects and managers. All of which are valid point of views but have different understanding and requirements. We also plan to measure runtime characteristics in order to increase our understanding of software product quality.

It is also in our plans to analyze the spreading of issues among projects. We would like to research why some of the code smells are localized to only a few projects, while others are present in almost every project. This might bring up very interesting questions related to ways of working, information distribution, or other higher level quality aspects of the projects.

8 Acknowledgments

This research would not have been possible without the help of the Quality Assurance Organization and the Software Technology (DUCI SWT) department of our industrial partner Ericsson. They were kind enough to allow us access to their databases and some of their TTCN-3 source codes and provide funding. These proved to be invaluable to find code smells that are of real life significance.

We would also like to thank Bernadett Diana Iván for her help in processing the fault and review databases. Her data mining knowledge allowed us to quickly extract and process large amount of data recorded in the span of 6 years. We would like to thank also Dániel Poroszkai for implementing many code smell metrics thoroughly.

References

[1] A. Bánsághi, B. G. Ézsiás, A. Kovács, A. Tátrai, *Source Code Scanners in Software Quality Management and Connections to International Standards*, *Annales Univ. Sci. Budapest. Sect. Comput.*, **37** (2012) 81–92. ⇒79, 80

[2] B. W. Boehm, Ph. N. Papaccio, Understanding and Controlling Software Costs, *IEEE Transactions on Software Engineering*, **14,** 10 (1988) 1462–1477. ⇒78

[3] *Checkstyle*, http://checkstyle.sourceforger.net ⇒84

[4] *CMMI institute*, http://cmmiinstitute.com/ ⇒78

[5] ETSI ES 201 873-1 v4.5.1: *Methods for Testing and Specification (MTS)*, The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language. ⇒85

[6] *FindBugs*, http://findbugs.sourceforge.net ⇒84

[7] M. Fowler, *Refactoring, Improving the Design of Existing Code*, Addison-Wesley, 1999. ⇒84

[8] *FxCop*, http://msdn.microsoft.com ⇒80, 84

[9] S. Herbold, J. Grabovszki, S. Waack, *Calculation and Optimization of Thresholds for Sets of Software Metrics. Empirical Software Engineering*, Springer, Netherlands, May 2011. ⇒94

[10] W. S. Humphrey, *The Team Software Process*, (2000) Technical Report, CMU/SEI-2000-TR-023, ESC-TR-2000-023 ⇒78

[11] ISO/IEC 9126:1991, *ISO Standard for Software Engineering – Product Quality* Revised by ISO/IEC 9126–1:2001. ⇒78, 81

[12] ISO/IEC 9126-1:2001: *ISO Standard for Software Engineering – Product Quality – Part 1: Quality Model.* ⇒81

[13] ISO/IEC TR 9126-2:2003: *ISO Standard for Software Engineering – Product Quality – Part 2: External Metrics.* ⇒81

[14] ISO/IEC TR 9126-3:2003: *ISO Standard for Software Engineering – Product Quality – Part 3: Internal Metrics.* ⇒81

[15] ISO/IEC TR 9126-4:2004: *ISO Standard for Software Engineering – Product Quality – Part 4: Quality in Use Metrics.* ⇒81

[16] ISO/IEC 25010:2011: *ISO Systems and Software Engineering - Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models.* ⇒78, 81

[17] S. H. Kan, *Metrics and Models in Software Quality Engineering*, Addison-Wesley, Boston, 2003. ⇒80

[18] F. Khomh, M. Di Penta, Y-G. Gueheneuc, An Exploratory Study of the Impact of Code Smells on Software Change-proneness, *Proc. 16th Working Conference on Reverse Engineering*, 2009, pp. 75–84. ⇒94

[19] W. Li, R. Shatnawi, An empirical study of the bad smells and class error probability in the post-release object-oriented system evolution, *Systems and Software* **80,** 7 (2007) 1120–1128. ⇒94

[20] G. Meszaros, *xUnit Test Patterns: Refactoring Test Code*, Addison-Wesley Professional, 2007. ⇒84

[21] A. Monden, D. Nakae, T. Kamiya, S. Sato, K. Matsumoto, Software quality analysis by code clones in industrial legacy software, *Symposium on Software Metrics*, 2002, pp. 87–94. ⇒94

[22] H. Neukirchen, B. Zeiss, J. Grabovszki, An approach to quality engineering of TTCN-3 test specifications, *International Journal on Software Tools for Technology Transfer (STTT)*, **10,** 4 (2008) 309–326. ⇒85

[23] H. Neukirchen, M. Bisanz, Utilising code smells to detect quality problems in TTCN-3 test suites, *Proc 19th IFIP International Conference on Testing of Communicating Systems and 7th International Workshop on Formal Approaches to Testing of Software (TestCom/FATES 2007)*, June 26-29, 2007, Tallinn, Estonia. *Lecture Notes in Computer Science (LNCS)* **4581,** 2007 pp. 228–243. ⇒85

[24] *PMD*, http://pmd.sourceforge.net ⇒80, 84

[25] R. van Solingen, E. Berghout, *The Goal/Question/Metric Method, a Practical Method for Quality Improvement of Software Development*, McGraw-Hill, 1999. ⇒78

[26] K. Szabados, Structural analysis of large TTCN-3 projects, *Proc. 21st IFIP WG 6.1 International Conference on Testing of Software and Communication Systems and 9th International FATES Workshop*, *Lecture Notes in Computer Science* **5826**:, *Testing of Software and Communication Systems*, Springer-Verlag Berlin, Heidelberg, 2009 pp. 241–246. ⇒80

[27] *The Personal Software Process (PSP) Body of Knowledge*, Version 2.0; Special Report; CMU/SEI-2009-SR-018 ⇒78

[28] *TRex*, http://www.trex.informatik.uni-goettingen.de/trac ⇒84

[29] B. Zeiss, D. Vega, I. Schiferdecker, H. Neukirchen, J. Grabovszki, *Applying the ISO 9126 Quality Model to Test Specifications – Exemplified for TTCN-3 Test Specifictions*, Software Engineering, *Lecture Notes in Informatics (LNI)* **105**, 2007, pp. 231–242. Gesellschaft für Informatik, Köllen Verlag, Bonn, ⇒79

[30] M. Zhang, T. Hall, N. Baddoo, Code Bad Smells: a review of current knowledge, *J. Softw. Maint. Evol.: Research and Practice* **23,** 3 (2011) 179–202. ⇒93

Received: April 10, 2013 • Revised: June 7, 2013

Acta Univ. Sapientiae, Informatica, **5,** 1 (2013) 103–139

The s-monotone index selection rule for criss-cross algorithms of linear complementarity problems

Zsolt CSIZMADIA
FICO
email: zsoltcsizmadia@fico.com

Tibor ILLÉS
Budapest University of Technology and Economics
Department of Differential Equations
email: illes@math.bme.hu

Adrienn NAGY
FICO, PhD student at ELTE
email: adriennnagy@fico.com

Abstract. In this paper we introduce the **s**-monotone index selection rules for the well-known criss-cross method for solving the linear complementarity problem (LCP). Most LCP solution methods require a priori information about the properties of the input matrix. One of the most general matrix properties often required for finiteness of the pivot algorithms (or polynomial complexity of interior point algorithms) is sufficiency. However, there is no known polynomial time method for checking the sufficiency of a matrix (classification of column sufficiency of a matrix is co-NP-complete).

Following the ideas of Fukuda, Namiki and Tamura, using Existentially Polynomial (EP)-type theorems, a simple extension of the criss-cross algorithm is introduced for LCPs with general matrices. Computational results obtained using the extended version of the criss-cross algorithm for bi-matrix games and for the Arrow-Debreu market equilibrium problem with different market size is presented.

Computing Classification System 1998: G.1.6.
Mathematics Subject Classification 2010: 49M35, 90C20
Key words and phrases: linear complementarity problem, sufficient matrix, criss-cross algorithm, alternative and EP theorems, bi-matrix games, Arrow-Debreu market equilibrium problems

1 Introduction

Let us consider the linear complementarity problem (LCP) in the standard form: find vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, such that

$$-M\mathbf{u} + \mathbf{v} = \mathbf{q}, \quad \mathbf{u}\,\mathbf{v} = \mathbf{0}, \quad \mathbf{u},\ \mathbf{v} \geq \mathbf{0} \qquad (P - LCP)$$

where $M \in \mathbb{R}^{n\times n}$, $\mathbf{q} \in \mathbb{R}^n$ and $\mathbf{u}\,\mathbf{v} = (u_1v_1, \ldots, u_nv_n) \in \mathbb{R}^n$.

The linear complementarity problem is one of the most studied areas of mathematical programming. A large number of practical applications and the wide range of unsolved—both theoretical and algorithmic—problems make it an attractive field of research.

There are several different pivot algorithms to solve LCP problems with different matrices. The criss-cross algorithm is one of those which were developed independently—for different optimization problems—by Chang [4], Terlaky [23] and Wang [29]. Since then, the criss-cross method has become a class of algorithms that differs in the index selection rule.

Akkeleş, Balogh and Illés [1] developed their criss-cross algorithm for LCP problems with bisymmetric matrices. They used the *LIFO* (last-in-first-out) and the *MOSV* (most-often-selected-variable) pivot rules. These index selection rules are flexible in the sense that they offer a choice of alternative pivots in certain situations, while still preserving finiteness. These index selection rules have been generalized for the linear programming problem in [9]. It is an interesting question which is the widest class of matrices for which the criss-cross algorithm with the above mentioned index selection rules can be extended, preserving finiteness.

The class of sufficient matrices was introduced by Cottle, Pang and Venkateswaran [6]. Sufficient matrices can be interpreted as generalizations of P and PSD matrices. Väliaho [27] showed that the class of sufficient matrices is the same as the class of P_* matrices, introduced in [20] for interior point methods of LCP problems. It was proved by den Hertog, Roos and Terlaky [14], that the sufficient matrices are exactly those matrices for which the criss-cross algorithm with the minimal index pivot selection rule solves the LCP problem with any right-hand side vector.

There is no known efficient algorithm to decide whether a matrix is sufficient or not, and Tseng [24] has shown that the classification of column sufficiency of a matrix is co-NP-complete. (Väliaho [26] developed a non-polynomial, inductive method to check sufficiency). Most algorithms developed for LCP problems have the practically unattractive property that they need the a priori information that the matrix is sufficient or possesses some other good properties.

Fukuda, Namiki and Tamura [12] gave the first such algorithm—based on the alternative theorem for LCP of Fukuda and Terlaky [13], a generalization of the fundamental result of Farkas [10, 11]—used in the form of Existentially Polynomial (EP) theorems—that did not require a priori information on the sufficiency of the matrix. If the algorithm cannot proceed or would begin to cycle, it provides a polynomial size certificate that the input matrix is not sufficient.

For more than a decade, it was an open question whether EP theorems for linear complementarity problems could be proved using interior point algorithms or not. The first result in this direction has been published by Illés, M. Nagy and Terlaky [16]. Their EP theorem is different from that of Fukuda, Namiki and Tamura [12]: (i) there exist a solution for the dual LCP problem, (ii) there exist a solution for the feasibility problem part of the primal LCP, (iii) the matrix is not row sufficient. The polynomial size certificates for each of these statement is provided using a polynomial time interior point algorithm for linear feasibility problems. This result however is weaker than that of Fukuda, Namiki and Tamura since statement (ii) does not guarantee solvability of the primal LCP problem.

The following paper of Illés, M. Nagy and Terlaky [17] related to EP theorems contain a generalization of the affine scaling and predictor-corrector interior point algorithms to solve LCPs with general matrices in EP-sense, namely, the generalized interior point algorithms either solve the problem with rational a coefficient matrix in polynomial time or give a polynomial size certificate that the matrix does not belong to the set of $\mathsf{P}_*(\kappa)$ matrices, with arbitrary large, but apriori fixed, rational, positive κ. This EP theorem differs both from the EP theorem published by Fukuda et al. [12] and from that of Illés et al. [17], because it contains only two statements. The reason for this is due to the fact that the embedding technique [22] of linear programming can not be generalized for LCPs with sufficient matrices. Therefore, one option is to deal with LCPs that have initial, strictly positive feasible solution. In this case, the dual LCP could not have solution, therefore the related EP theorems only need to consider two cases.

In their third paper, Illés, M. Nagy and Terlaky [18] use the embedding technique introduced by Kojima et al. [20] and the related results about the solvability of the primal LCP to solve the embedded problem with a generalized path-following interior point algorithm. This allows them to prove almost the same EP theorem as Fukuda et al. [12]. There are two differences between the Fukuda-Namiki-Tamura and the Illés-Nagy-Terlaky EP theorems: (i) the later provides a decision between the options in polynomial time (ii) while, instead

of the whole class of sufficient matrices, the algorithm works only for a given subset of sufficient matrices, i.e. for any $P_*(\kappa)$ matrix, with apriori fixed $\kappa > 0$. Property (i) is an improvement compared to the EP theorem of Fukuda et al., however (ii) states that the improvement works only for a given subset of sufficient matrices instead of the whole class.

The criss-cross method using the LIFO or MOSV index selection rules have been generalized for sufficient matrices to the form of EP-theorems in [8].

This paper introduces a new variant of the criss-cross type algorithm that combines the flexibility of the **s**-monotone index selection rules with the practicability of EP-theorems: the criss-cross algorithm using **s**-monotone index selection rules is modified in such a way that in case of an arbitrary matrix M and right-hand side $\mathbf{q}$, it either solves the LCP problem, or provides a polynomial size certificate that the matrix M is not sufficient. This property improves the value of the algorithm significantly making it applicable to a wide range of LCP problems, without requiring a priori information on the properties of the matrix. Indeed, the improved freedom of the **s**-monotone pivot position selection compared to the traditional minimal index rule gives the possibility to avoid numerically instable pivots, further extending the practical applicability.

The finiteness proofs using **s**-monotone index selection rules also proves the finiteness of these rules for the criss-cross method for the linear and convex quadratic problems in a more general context, as the Karush-Khun-Tucker conditions of the linear and quadratic problems yield a linear complementarity problem with bisymmetric matrices which are known to be sufficient (when the quadratic problem is convex) making this paper a natural extension of [9].

Throughout this paper, matrices are denoted by italic capital letters, vectors by bold, scalars by normal letters and index sets by capital calligraphic letters. Columns of a matrix are indexed as a subscript while rows are indexed by superscripts. M denotes the original problem matrix, $\mathbf{q}$ the right hand side. Without loss of generality we may assume that $\mathrm{rank}(M) = n$. The (ordered) set of all indices is $\mathcal{I} := \{1, 2, \ldots, n, \bar{1}, \bar{2}, \ldots, \bar{n}\} \cup \{q\}$ which includes the index of the right hand side, and so $|\mathcal{I}| = 2n + 1$. To denote the complementarity pairs, let $\overline{\bar{\alpha}} = \alpha$ for all $\alpha \in \mathcal{I} \setminus \{q\}$, so the complementary index pair of $\bar{\alpha}$ is α. For a given basis B, we denote the nonbasic part of M by N; the corresponding set of indices for the basis and nonbasic part are denoted by $\mathcal{I}_B$ and $\mathcal{I}_N$, respectively. The corresponding short pivot tableau for B is denoted by $T := B^{-1}N$, while the transformed right hand side is denoted by $\bar{\mathbf{q}} := B^{-1}\mathbf{q}$. Individual coefficients of the short pivot tableau will be denoted by $\bar{m}_{ij}$; please note that the definition of coefficients t_{ij} will be formally provided in Section 4.1.1. The complementary variable pairs (u_l, v_l) will usually be referred to as variable pairs. When it is

advantageous, the element-wise Hadamard product will be emphasized by "$\cdot$". A vector of an appropriate size consisting of all ones will be denoted by $\mathbf{1}$.

The structure of the paper is as follows: following the introduction, the first two sections present the theoretical background used by the algorithms: Section 2 introduces the concept of sufficient matrices and their relevant algorithmically important properties, while Section 3 summarizes the $s-\mathtt{monotone}$ index selection rules. Section 4 presents new variants of the generalized criss-cross algorithm for LCP using **s**-monotone index selection rules, while Section 5 extends it to the EP-theorem case. Section 6 present computational experiments solving small scale bimatrix games and Arrow-Debreu market equilibrium problems. The paper is closed by a summary.

2 Linear complementarity problems and sufficient matrices

For most solution methods for LCPs, the solvability of the linear complementarity problems and the efficiency of algorithms depend on the properties of matrix M. For pivot methods, one of the most general classes of interest is the *sufficient matrices*, which are generalizations of positive semidefinite matrices.

Definition 1 *[6] The matrix* $M \in \mathbb{R}^{n\times n}$ *is called* column sufficient *if no vector* $\mathbf{x} \in \mathbb{R}^n$ *exists, for which*

$$\begin{cases} x_i\,(M\mathbf{x})_i \le 0 & \text{for all indices } i \in \{1,\dots,n\} \\ x_j\,(M\mathbf{x})_j < 0 & \text{for at least one index } j \in \{1,\dots,n\} \end{cases} \tag{1}$$

and we call it row sufficient *if its transpose is column sufficient. A matrix* M *is called* sufficient *if it is both column and row sufficient at the same time.*

It can be shown that column sufficient matrices are exactly those, for which the solution set of linear complementarity problems is convex [6], in fact a polyhedron; den Hertog, Roos and Terlaky [14] proved that sufficient matrices are exactly those, for which the criss-cross algorithm with the minimal index pivot rule can solve linear complementarity problems for any right-hand side vector $\mathbf{q}$, in a finite number of iterations.

The algorithms presented in this paper will use the concept of *strictly sign reversing* and *strictly sign preserving* vectors:

Definition 2 *[12] We call a vector* $\mathbf{x} \in \mathbb{R}^{2n}$ strictly sign reversing *if*

$$\begin{aligned} &x_i x_{\bar{i}} \le 0 \quad \text{for all indices } i = 1,\dots,n \\ &x_i x_{\bar{i}} < 0 \quad \text{for at least one index } i \in \{1,\dots,n\}. \end{aligned}$$

We call a vector $\mathbf{x} \in \mathbb{R}^{2n}$ strictly sign preserving *if*

$$\begin{aligned} x_i x_{\bar{i}} &\geq 0 \quad \textit{for all indices } i = 1, \ldots, n \\ x_i x_{\bar{i}} &> 0 \quad \textit{for at least one index } i \in \{1, \ldots, n\}. \end{aligned}$$

A vector of $\mathbf{x} \in \mathbb{R}$ *related to an LCP problem is called strictly sign preserving or strictly sign reversing, if the vector* $(\mathbf{x}, M\mathbf{x}) \in \mathbb{R}^{2n}$ *is strictly sign preserving or strictly sign reversing respectively.*

To simplify the definition of the next lemma, let us introduce the subspaces

$$V := \left\{ (\mathbf{u}, \mathbf{v}) \in \mathbb{R}^{2n} \mid -M\mathbf{u} + \mathbf{v} = \mathbf{0} \right\}$$

and

$$V^{\perp} := \left\{ (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{2n} \mid \mathbf{x} + M^T\mathbf{y} = \mathbf{0} \right\},$$

where $\mathbf{u}, \mathbf{v}, \mathbf{x}$ and $\mathbf{y}$ are all vectors of length n. V and $V^{\perp}$ are orthogonal complementary subspaces [12] of $\mathbb{R}^{2n}$.

Lemma 3 *[12] A matrix* $M \in \mathbb{R}^{n \times n}$ *is sufficient if and only if no strictly sign reversing vector exists in* V *and no strictly sign preserving vector exists in* $V^{\perp}$.

A basis B of the linear system $-M\mathbf{u} + \mathbf{v} = \mathbf{q}$ is called complementary, if for each index $i \in \mathcal{I}$ exactly one of the columns corresponding to variables v_i and u_i is in the basis. A short pivot tableau [19] is called complementary, if the corresponding basis is complementary. The next lemma shows the sign structure of short complementary pivot tableaux of LCPs with sufficient matrices, which is the main property of these matrices that the algorithms presented later will rely on.

Lemma 4 *[6] Let* M *be a sufficient matrix,* B *a complementary basis and* $\bar{M} = [\bar{m}_{ij} \mid i \in J_B, j \in J_N]$ *the corresponding short pivot tableau. Then*

(a) $\bar{m}_{i\bar{i}} \geq 0$ *for all* $i \in J_B$*; furthermore*

(b) for all $i \in J_B$*, if* $\bar{m}_{i\bar{i}} = 0$ *then* $\bar{m}_{i\bar{j}} = \bar{m}_{j\bar{i}} = 0$ *or* $\bar{m}_{i\bar{j}} \cdot \bar{m}_{j\bar{i}} < 0$ *for all* $j \in J_B$, $j \neq i$.

The proof of the previous lemma is constructive, so if the given structure of the matrix is violated, we can easily obtain the certificate from tableau $\bar{M}$,

that M is not sufficient. The coding size of this certificate is bounded by a polynomial of the input length of matrix M [12, 8]

By the *permutation* of $M \in \mathbb{R}^{n\times n}$, we mean the matrix $P^T MP$, where P is a permutation matrix.

Lemma 5 *[14] Let $M \in \mathbb{R}^{n\times n}$ be a row (column) sufficient matrix. Then*

1. *any permutation of matrix M is row (column) sufficient,*
2. *the product DMD is row (column) sufficient, where $D \in \mathbb{R}^{n\times n}_{+}$ is a positive diagonal matrix,*
3. *every principal submatrix of M is row (column) sufficient.*

A sufficient matrix $\bar{M}$ is also sufficient after any number of arbitrary principal pivots. The class of sufficient matrices is closed under principal block pivot operations [26], and as a consequence their properties are preserved during the criss-cross type algorithms, as the exchange pivot operations carried out by the criss-cross algorithms are equivalent to a block pivot operation of size 2×2.

The matrix of Example 6 will be used in all examples of the paper.

Example 6 *As an example of a non-sufficient matrix, consider*

$$M = \begin{pmatrix} 1 & -1 & -1 & 0 & -1 \\ -1 & 2 & 0 & 0 & -1 \\ 1 & -2 & -1 & -2 & 0 \\ -4 & -1 & -1 & 2 & 4 \\ -1 & 0 & 2 & 0 & 1 \end{pmatrix}.$$

This matrix is neither column, nor row sufficient. In this example, the operator $\cdot$ denotes the Hadamard, element-wise product. Consider the column vector

$$\mathbf{x}^T = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

then

$$\mathbf{x} \cdot (M\mathbf{x}) = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \cdot \left[\begin{pmatrix} 1 & -1 & -1 & 0 & -1 \\ -1 & 2 & 0 & 0 & -1 \\ 1 & -2 & -1 & -2 & 0 \\ -4 & -1 & -1 & 2 & 4 \\ -1 & 0 & 2 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \right] =$$

$$= \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} -1 \\ 0 \\ -1 \\ -1 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -1 \\ 0 \\ 0 \end{pmatrix}$$

a strictly sign reversing vector. Similarly, for row vector

$$\mathbf{y} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

we have

$$(\mathbf{y}M) \cdot \mathbf{y} = \begin{pmatrix} -3 & -3 & -2 & 0 & 4 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & -2 & 0 & 0 \end{pmatrix}$$

making $\mathbf{y}$ *a proof that* M *is not sufficient according to Definition 1.*

The decision problem, whether an arbitrary linear complementarity problem has a solution or not, is in $\mathbb{NP}$, and not always in co-$\mathbb{NP}$ [5], although for the class of sufficient matrices it belongs to co-$\mathbb{NP}$, and can be stated using the dual of the LCP (1) problem:

$$(\mathbf{x}, \mathbf{y}) \in V(M, \mathbf{q})^{\perp} := \left\{ (\mathbf{x}, \mathbf{y}) \mid \begin{array}{l} \mathbf{x} + M^{T}\mathbf{y} = \mathbf{0}, \quad \mathbf{q}^{T}\mathbf{y} = -1 \\ \mathbf{xy} = \mathbf{0}, \quad \mathbf{x}, \mathbf{y} \geq \mathbf{0} \end{array} \right\} \qquad (D - LCP)$$

Using the definition of the dual, the alternative theorem of LCP problems is as follows.

Theorem 7 *[13] For a sufficient matrix* $M \in \mathbb{R}^{n \times n}$ *and a vector* $\mathbf{q} \in \mathbb{R}^{n}$*, exactly one of the following statements holds:*

(1) the (P-LCP) problem has a feasible complementary solution $(\mathbf{u}, \mathbf{v})$*,*

(2) the (D-LCP) problem has a feasible complementary solution $(\mathbf{x}, \mathbf{y})$*.*

As a consequence, it is well-characterized when the $(P-LCP)$ problem has no feasible complementary solution if the matrix M is sufficient and rational, as a polynomial size certificate can be given, namely the solution of the problem $(D - LCP)$.

The alternative theorem for LCP of Fukuda and Terlaky is an interesting generalization of Farkas-lemma [10, 11]. Due to the fact, that primal and dual LCPs are more complicated problems, to show that exactly one of the alternative statements holds it is necessary to have information on the matrix property. Therefore, Fukuda and Terlaky's result, comparing to that of Farkas' needs an extra assumption, that the matrix M is sufficient.

3 The s-monotone index selection rule

The concept of **s**-monotone index selection rules have been introduced in [9], presented here for the LCP case. We say a variable moves during a pivot, if it either leaves, or enters the basis.

Definition 8 (Possible pivot sequence) *A sequence of index pairs*

$$\mathcal{S} = \{\mathcal{S}_k = (i_k, o_k) : i_k, o_k \in \mathbb{N} \text{ for some consecutive } k \in \mathbb{N}\},$$

is called a possible pivot sequence*, if*

(i) $2n = \max\{\max_{k\in\mathbb{N}} i_k, \max_{k\in\mathbb{N}} o_k\}$,

(ii) there exists a (P-LCP) in standard form with $2n$ *variables and the* $\mathrm{rank}(M) = n$*, and*

(iii) (possibly infinite) pivot sequence, where the moving variable pairs of (P-LCP) correspond to the index pairs of $\mathcal{S}$*.*

The index pairs of a possible pivot sequence are thus only required to comply with the basic and nonbasic status. It is now easy to show that

Proposition 9 *If a possible pivot sequence is not finite then there exists a (sub)set of indices,* $\mathcal{I}^*$*, that occurs infinitely many times in* $\mathcal{S}$**.**

The concept of *pivot index preference* describes how to select pivot positions when the primary algorithm offers flexibility:

Definition 10 (Pivot index preference) *A sequence of vectors* $s_k \in \mathbb{N}^{2n}$ *is called a* pivot index preference *of an index selection rule, if in iteration* j*, in case of ambiguity according to a pivot selection rule, the index selection rule selects an index with highest value in* s_j *among the candidates.*

The concept of **s**-monotone index selection rule [9] aims to formalize a common monotonicity property of several index selection rules.

Definition 11 (s-monotone index selection rule) *Let* $n \in \mathbb{N}$ *be given. An index selection rule is called* **s**-monotone*, if*

1. *there exists a pivot index preference* $s_k \in \mathbb{N}^{2n}$*, for which*

(a) *the values in the vector* $\mathbf{s}_{j-1}$ *after iteration* j *may only change for* i_j *and* o_j*, where* i_j *and* o_j *are the indices involved in the pivot operation,*

(b) *the values may not decrease.*

2. *For any infinite* possible pivot sequence $\mathcal{S}$ *and for any iteration* j *there exists iteration* $r \geq j$ *such that*

(a) *the index with minimal value in* $\mathbf{s}_r$ *among* $\mathcal{I}^* \cap \mathcal{I}_{B_r}$ *is unique (let it be* l*), where* $\mathcal{I}_{B_r}$ *is the set of basic indices, in iteration* r*,*

(b) *in iteration* $t > r$ *when index* $l \in \mathcal{I}^*$ *occurs again in* $\mathcal{S}$ *for the first time, the indices of* $\mathcal{I}^*$ *that occurred in* $\mathcal{S}$ *strictly between* S_r *and* S_t *have a value in* $\mathbf{s}_t$ *higher than the index* l*.*

To apply the **s**-monotone index selection rules for linear complementarity problems, a small extension is necessary comparing to that one for linear programming [9].

Definition 12 *An* **s***-monotone pivot rule applied to an LCP problem is symmetric, if* $s(i) = s(\bar{i})$ *always holds for any* $i \in \mathcal{I}$*.*

The criss-cross algorithms presented in this paper will apply the **s**-monotone rules in a symmetric way. As examples for **s**-monotone rules, below is the definition of an **s** vector for the minimal index, the last-in-first-out and most often selected rules.

- *Minimal index/Bland:* The variables with the smallest index is selected. Initialize **s** consisting of constant entries

$$s_i = n - i, i \in \mathcal{I}$$

where n is the number of the problem columns in the problem.

- *Last-In-First-Out (LIFO):* The most recently moved variable is selected. Initialize **s** equal to **0**. For a pivot at (k, l) in the r^{th} iteration update **s** as

$$s_i' = \begin{cases} r & \text{if } i \in \{k, l\} \\ s_i & \text{otherwise} \end{cases}$$

- *Most-Often-Selected-Variable (MOSV):* Select the variable that has been selected the largest amount of times before.

  Initialize $\mathbf{s}$ equal to $\mathbf{0}$. For a pivot at (k, l) in the r^{th} iteration update $\mathbf{s}$ as

$$s_i' = \begin{cases} s_i + 1 & \text{if } i \in \{k, l\} \\ s_i & \text{otherwise} \end{cases}$$

4 The criss-cross method

This section presents the criss-cross algorithm for linear complementarity problems using $\mathbf{s}$-monotone index selection rules.

The criss-cross algorithms for LCP problems need a starting complementary solution, for which $\mathbf{u} = \mathbf{0}$ and $\mathbf{v} = \mathbf{q}$ is a possible selection using the assumption that $\mathrm{rank}(M) = n$. Starting from the initial complementarity solution, the criss-cross method performs a sequence of so called diagonal and exchange pivots.

If in any complementary basis during the algorithm v_j is a basic variable and the value of variable v_j is infeasible and $\bar{m}_{jj} < 0$, then the algorithm, may perform a *diagonal pivot* where variable u_j enters the basis while variable v_j leaves it.

If $\bar{m}_{jj} = 0$ and the algorithm select v_j to leave the bases, then it pivots on such an index k for which $\bar{m}_{jk} < 0$. The solution obtained after a pivot like this is not complementary any more, so to restore the complementarity of the solution the algorithm pivots on the position (k, j) as well. According to Lemma 4, in case of sufficient matrices $\bar{m}_{kj} > 0$ will hold in this situation. These two pivots together are called an *exchange pivot*.

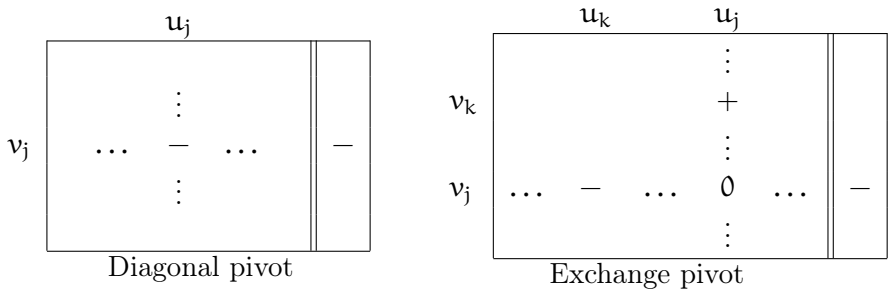


Figure 1: Diagonal and exchange pivot

During an exchange pivot, variables u_j and u_k enter the basis, while v_j and v_k leave it. We say that u_j and v_j are chosen *actively*, while u_k and v_k are chosen *passively*. Thus, the terms passively and actively refer to the order in which the indices are selected, irrespective of the variable types (i.e. the same rule applies to an exchange pivot involving different combinations like (v_j, u_k) exchanged with (u_j, v_k)). The two types of pivot operations are presented in Figure 1.

When using **s**-monotone index selection rules, the role of the **s**-vector is to maintain a history about the movements of the variables. In the LCP case, these updates are generalized as follows: in case of an exchange pivot, the vector **s** of the **s**-monotone pivot rule is updated to the passively selected pair of variables first, and only then for the actively selected pair, thus the symmetry of vector **s** (i.e. the value of any variable pair (u_i, v_i) in **s** is the same after each pivot) is also maintained. This way, from the **s** vector's point of view, an exchange pivot is considered as two pivots. This rule will play an important role in the finiteness proofs.

Note, that Lemma 5 ensures that the sufficiency of the matrix is preserved during the algorithm. Moreover, both the diagonal and the exchange pivot operations preserve complementarity as well.

The pseudo-code and flow chart of the criss-cross type algorithm using **s**-monotone index selection rules is presented in Figure 2 and Figure 3.

The sufficiency of matrix M ensures that (Lemma 4) the sign of the chosen pivot elements will be as desired in the case of exchange pivot. The algorithm terminates only if there is no solution or if it has found the solution, so it is sufficient to prove that it is *finite*. As the number of possible bases is finite, we have to show that the criss-cross type algorithm with **s**-monotone pivot rule does not *cycle*.

4.1 Almost terminal tableaux of the criss-cross method

To prove finiteness of the algorithm, we assume the contrary, i.e. that it cycles. The results of this section are a generalization of the proof presented in [8] to the **s**-monotone case.

Let us assume that an example exists for which the algorithm is not finite. The number of bases is finite, at most $\binom{2n}{n}$, so the algorithm makes an infinite number of iterations only if cycling occurs. Let us consider a minimal size cycling example. In such an example, because of minimality—and the monotonicity of **s**—every variable moves during the cycle.

Let us consider the situation described by the second criterion of **s**-monoto-

Criss-cross type algorithm with symmetric s-monotone pivot rule

Input: problem (1), where M is sufficient, $T := -M$, $\bar{q} := q$, $r := 1$. Initialize $\mathbf{s}$.
Begin
  $\mathcal{J} := \{i \in \mathcal{I} : \bar{q}_i < 0\}$.
  While $(\mathcal{J} \neq \emptyset)$ **do**
    $\mathcal{J}_{max} := \{j \in \mathcal{J} \mid s(j) \geq s(\alpha),$ for all $\alpha \in \mathcal{J}\}$, let $k \in \mathcal{J}_{max}$ arbitrary.
    If $(\bar{m}_{kk} < 0)$ **then**
      Diagonal pivot on $\bar{m}_{kk}$, update vector $\mathbf{s}$ for variable pair (u_k, v_k).
      Let $r := r + 1$.
    Else
      $\mathcal{K} := \{i \in \mathcal{I} : \bar{m}_{ki} < 0\}$.
      If $(\mathcal{K} = \emptyset)$ **then Stop:** The LCP problem has no feasible solution.
      Else
        $K_{max} = \{\beta \in \mathcal{K} \mid s(\beta) \geq s(\alpha),$ for all $\alpha \in \mathcal{K}\}$.
        Let $l \in \mathcal{K}_{max}$ arbitrary.
        Exchange pivot on $\bar{m}_{kl}$ and $\bar{m}_{lk}$.
        Update vector $\mathbf{s}$ for variable pair (u_l, v_l) as in iteration $r + 1$,
        then for variable pair (u_k, v_k) as in iteration $r + 2$.
        Let $r := r + 2$.
      Endif
    Endif
  Endwhile
  Stop: A feasible complementary solution has been computed.
End

Figure 2: The criss-cross type algorithm

nicity and with variable u_l outside the bases. Consider basis B' where variable u_l enters the bases for the first time after this situation occurred. Since u_l and v_l have the smallest value according to vector $\mathbf{s}$ and the symmetry of the rule, v_l is the only infeasible variable in basis B'. The short pivot tableaux (presented in Figures 4 and 5) for this case are as follows:

1. The algorithm chooses u_l to enter the basis.

   The diagonal element $\bar{m}_{ll} < 0$, so a diagonal pivot is possible [Figure 4, tableau (a)]: u_l enters the basis, while v_l leaves it. The vector $\mathbf{s}$ is modified symmetrically for variable pair (u_l, v_l).
2. The algorithm chooses variable u_l to enter the basis, but $\bar{m}_{ll} = 0$, so an exchange pivot is necessary [Figure 4, tableau (b)]. Variables u_l and u_j enter the basis, while variables v_l and v_j leave it. The vector $\mathbf{s}$ is modified

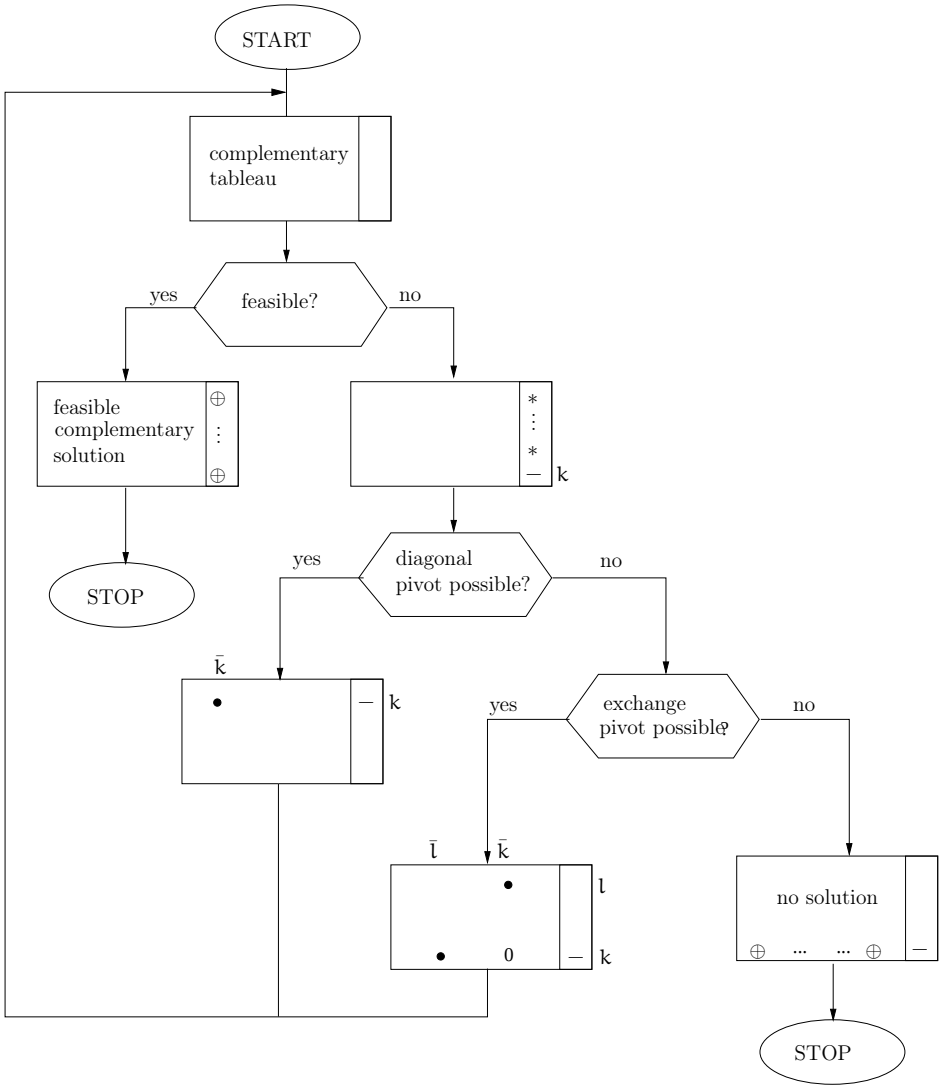


Figure 3: Flow chart of the algorithm

symmetrically (i.e. the complementary variable pairs always have the same value in $\mathbf{s}$), first for variable pair (u_j, v_j) and then for variable pair (u_l, v_l) as if in the next iteration.

The column of $\mathbf{q}$ is the same as in tableau (a). In this case, it is not important whether u_j or v_j is in the basis. We consider the case when v_j is in the basis.

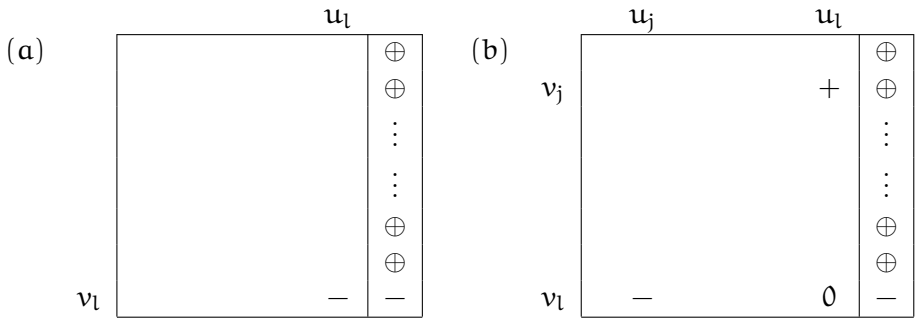


Figure 4: Variable u_l is actively selected to enter the bases

3. The algorithm chooses a variable u_j to enter the basis, but $\bar{m}_{jj} = 0$, so an exchange pivot is necessary and the algorithm chooses the variable u_l as well (passively) [Figure 5, tableau (c)].

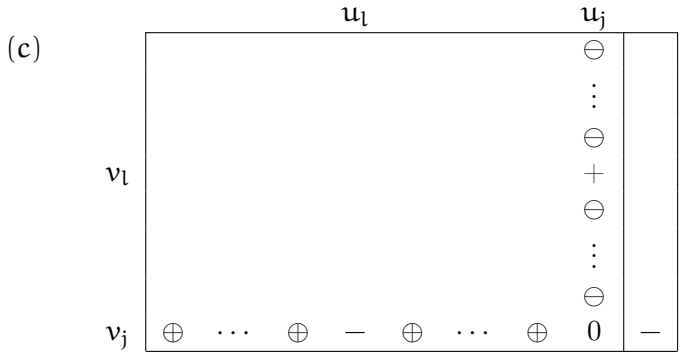


Figure 5: Variable u_l is passively selected in a a second pivot position in an exchange pivot to enter the basis

According to the situation when u_l enters the basis, the row of v_j only in the columns of u_l and $\mathbf{q}$ may contain negative elements, and because $\bar{m}_{jj} = 0$ –using Lemma 4– we also know the sign structure of the column of u_j. In this case, it is once again not important whether u_j or v_j is in the basis. We consider the case when v_j is in the basis. The vector $\mathbf{s}$ is modified symmetrically first for variable pair (u_l, v_l) and then for variable pair (u_j, v_j) as if in the next iteration.

For the purpose of the EP-theorem considered in the next section, we note that in Cases 1. and 2. only the properties of the pivot rule has been used when

filling out the sign structures, while the sufficiency of the matrix has been used in the third case for the column of u_j.

Now consider basis B'', when u_l leaves the basis for the first time after basis B'. The pivot tableau for this iteration can have three different structures (as presented in Figures 6 and 7), according to the second criterion of **s**-monotonicity.

A. According to the pivot rule, the algorithm chooses variable u_l to leave the basis, $\bar{m}_{ll} < 0$, so a diagonal pivot takes place [Figure 6, tableau (A)].

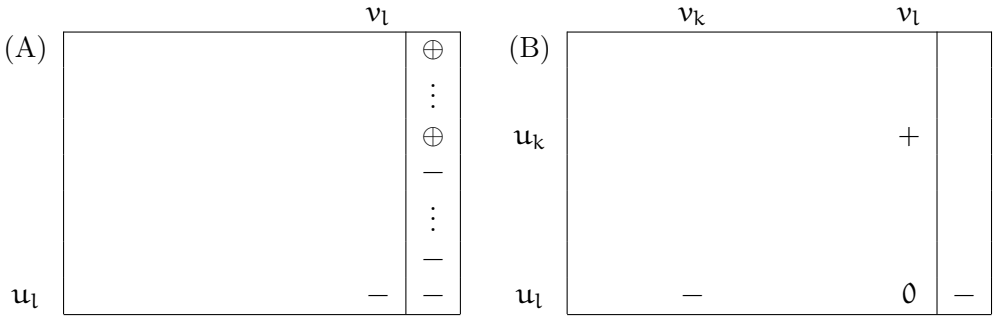


Figure 6: Variable u_l is actively selected to leave the bases

B. The pivot rule chooses variable u_l to leave the basis, but $\bar{m}_{ll} = 0$, so an exchange pivot is needed: v_k (or u_k) enters the basis, while u_k (or v_k) leaves it [Figure 6, tableau (B)].

C. The algorithm chooses variable u_k (or v_k), but $\bar{m}_{kk} = 0$, so an exchange pivot takes place and v_l enters the basis, while u_k leaves it [Figure 7, tableau (C)].

We show that none of the tableaux (a) – (c) can be followed by one of the tableaux (A)–(C) if matrix M is sufficient. For the purposes of the next section that further extends the algorithm for the EP-theorem case, it is important to make note which parts of the proofs are based on the sufficiency of the input matrix, and which on the properties of **s**-monotonicity.

4.1.1 Auxiliary lemmas

We will use the following fundamental result of pivot tableaux called the orthogonality theorem that describes an orthogonal property between different pivot

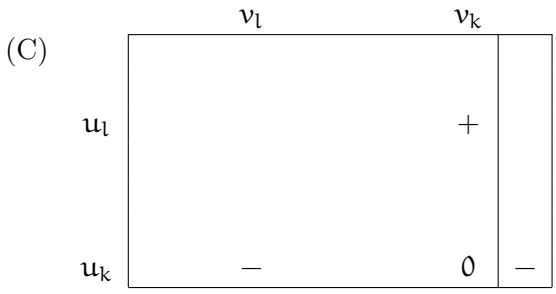


Figure 7: Variable u_l is passively selected in a second pivot position in an exchange pivot to leave the basis

tableaux of the set of vectors. Denote the row vector

$$\left(\mathbf{t}^{(i)}\right)_k = \begin{cases} t_{ik}, & \text{if } k \in \mathcal{J}_N \cup \{b\} \\ 1, & \text{if } k = i \\ 0, & \text{otherwise} \end{cases}$$

and the columns vector

$$(\mathbf{t}_j)_k = \begin{cases} t_{kj}, & \text{if } k \in \mathcal{J}_B \\ -1, & \text{if } k = j \\ 0, & \text{otherwise.} \end{cases}$$

Theorem 13 *[19] For any matrix $A \in \mathbb{R}^{m\times n}$ and with arbitrary bases B' and B'', the vectors $\mathbf{t}'^{(i)}$ and $\mathbf{t}''_j$ belonging to basic tableaux $T_{B'}$ and $T_{B''}$ respectively, are orthogonal.*

First, consider the cases that do not use the sufficiency of matrix M. We begin by showing that tableau (c) cannot be followed by tableaux (A) or (B).

Lemma 14 *Let us denote the tableau of case* (c) *by $T_{B'}$ and the tableau of* A *(or* B*) by $T_{B''}$. Consider the vectors $\mathbf{t}'^{(\bar{j})}$ and $\mathbf{t}''_q$, read from the row of the basic variable v_j in tableau $T_{B'}$, and from the column of $\mathbf{q}$ in $T_{B''}$. Then*

$$(\mathbf{t}'^{(\bar{j})})^T \mathbf{t}''_q > 0.$$

Proof. Let $\mathcal{K}'' := \{\, i \in \mathcal{I}_{B''} \mid \bar{q}''_i < 0 \,\}$. Using the second criterion of **s**-monotone pivot rules, variables referring to these indices (with the possible exception of index $\bar{j}$ and l) have not moved since B', or else their **s** value would have to be

larger than of u_l, thus the algorithm would have chosen one from $\mathcal{K}''$ instead of u_l, so $\mathcal{K}'' \subseteq \mathcal{I}_{B'} \cap \mathcal{I}_{B''}$. This indicates that $t'_{\bar{j}i} = 0$ for all indices $i \in \mathcal{K}'' \setminus \{\bar{j}, l\}$, thus

$$\sum_{i \in J'' \setminus \{\bar{j}, l\}} t'_{\bar{j}i} t''_{iq} = 0. \tag{2}$$

Since at the exchange pivot made on B', the $\mathbf{s}$ value was first updated for variable pair (u_l, v_l) and only in the next pivot for (u_j, v_j), we also know from $\mathbf{s}$-monotonicity that t''_{jq} and $t''_{\bar{j}q}$ are nonnegative (one is outside the basis, and since they have moved since B', their $\mathbf{s}$ value is bigger than of u_l).

Furthermore, it can be read from tableau (c) that $t'_{\bar{j}j} = 0$, $t'_{j\bar{j}} = 1$, $t'_{\bar{j}\bar{l}} = 0$, $t'_{j\bar{l}} < 0$ and $t'_{\bar{j}q} < 0$ so

$$t'_{j\bar{j}}\, t''_{\bar{j}q} + t'_{\bar{j}j}\, t''_{jq} + t'_{\bar{j}\bar{l}}\, t''_{\bar{l}q} + t_{\bar{j}l}\, t''_{lq} + t'_{\bar{j}q}\, t''_{qq} \geq t'_{\bar{j}\bar{l}}\, t''_{\bar{l}q} - t'_{\bar{j}q} > 0, \tag{3}$$

because $t''_{qq} = -1$ by definition, and $t''_{lq} < 0$ according to the pivot rule of the algorithm (tableaux (A) and (B)).

If $h \notin \mathcal{K}'' \cup \{j, \bar{j}, l, \bar{l}, q\}$, we know again from the tableaux that $t'_{\bar{j}h} \geq 0$ and by the definition of $\mathcal{K}''$ it holds that $t''_{hq} \geq 0$, so

$$\sum_{h \notin \mathcal{K}'' \cup \{j, \bar{j}, l, \bar{l}, q\}} t'_{\bar{j}h}\, t''_{hq} \geq 0. \tag{4}$$

The result follows as we sum up inequalities (2)–(4). □

From tableau (c) we considered the structure of the row for variable v_j, while from tableaux (A) and (B) the structure of the column of q. In none of these cases did we use the sufficiency of the matrix, and the proofs used only the combinatorial nature of the $\mathbf{s}$-monotone pivot rules. Thus tableaux (c) and (A) (or (B)) are exclusive because of the orthogonality theorem and the lemma above, regardless of the sufficiency of the matrix.

We now prove that tableaux (a) and (b) cannot be followed by tableau (C).

Lemma 15 *Let us denote tableau* (a) *(or* (b)*) by* $T_{B'}$ *and tableau* (C) *by* $T_{B''}$. *Consider the vectors* $\mathbf{t}'_q$ *and* $\mathbf{t}''^{(k)}$ *belonging to the column of* $\mathbf{q}$ *in tableau* $M_{B'}$, *and to the row of* u_k *in tableau* $M_{B''}$. *Then*

$$(\mathbf{t}''^{(k)})^T\, \mathbf{t}'_q > 0.$$

Proof. Like in the previous lemma, $\mathcal{K}''_k := \{i \in \mathcal{I}_{N''} \mid t''_{ki} < 0\} \subset \mathcal{I}_{N'}$ holds because of the second criterion of $\mathbf{s}$-monotonicity, so $t'_{iq} = 0$ for every $i \in$

$\mathcal{K}''_k \setminus \{l\}$, thus

$$\sum_{i \in \mathcal{K}''_k \setminus \{l\}} t''_{ki}\, t'_{iq} = 0. \tag{5}$$

Furthermore, for an index $h \notin \mathcal{K}''_l \cup \{\bar{j}, j, \bar{l}, l, q\}$, $t''_{kh} \geq 0$ and $t'_{hq} \geq 0$, therefore

$$\sum_{j \notin \mathcal{K}''_l \cup \{\bar{j}, j, \bar{l}, l, q\}} t''_{kj}\, t'_{jq} \geq 0. \tag{6}$$

From tableaux $T_{B'}$ and $T_{B''}$ it can be read that
$t'_{qq} = -1,\ t''_{kk} = 1,\ t''_{k\bar{k}} = t''_{kl} = t'_{lq} = 0$ and $t''_{k\bar{l}} < 0,\ t''_{kq} < 0,\ t'_{\bar{l}q} < 0,\ t'_{\bar{k}q} \geq 0$ and $t'_{kq} \geq 0$ so

$$t''_{k\bar{k}}\, t'_{\bar{k}q} + t''_{kk}\, t'_{kq} + t''_{k\bar{l}}\, t'_{\bar{l}q} + t''_{kl}\, t'_{lq} + t''_{kq}\, t'_{qq} = t'_{kq} + t''_{k\bar{l}}\, t'_{\bar{l}q} - t''_{kq} \geq t''_{k\bar{l}}\, t'_{\bar{l}q} - t''_{kq} > 0.$$

The result follows as we sum up inequalities (5) – (7).

□

We can now consider tableaux where the sufficiency of the matrix plays an important role.

In the following, we show that tableaux (a) (or (b)) cannot be followed by tableaux (A) or (B).

Lemma 16 *Let the complementary solutions* $(\mathbf{u}', \mathbf{v}')$ *and* $(\mathbf{u}'', \mathbf{v}'')$*, belonging to tableaux* (a) *(or* (b)*) and* (A) *(or* (B)*) be given. Then the Hadamard product*

$$(\mathbf{u}' - \mathbf{u}'') \cdot\ M\,(\mathbf{u}' - \mathbf{u}'') \lneqq \mathbf{0},$$

i.e. $(\mathbf{u}' - \mathbf{u}'')$ *is a strictly sign reversing vector with respect to* M.

Proof. We prove all four cases simultaneously.

$$\begin{aligned}
(\mathbf{u}' - \mathbf{u}'') \cdot\ M\,(\mathbf{u}' - \mathbf{u}'') &= (\mathbf{u}' - \mathbf{u}'') \cdot\ (\mathbf{q} + M\,\mathbf{u}' - \mathbf{q} - M\,\mathbf{u}'') \\
&= (\mathbf{u}' - \mathbf{u}'') \cdot\ (\mathbf{v}' - \mathbf{v}'') \\
&= \mathbf{u}' \cdot\ \mathbf{v}' - \mathbf{u}' \cdot\ \mathbf{v}'' - \mathbf{u}'' \cdot\ \mathbf{v}' + \mathbf{u}'' \cdot\ \mathbf{v}'' \\
&= -\mathbf{u}' \cdot\ \mathbf{v}'' - \mathbf{u}'' \cdot\ \mathbf{v}',
\end{aligned}$$

where the last equation holds because of the complementarity of the given solutions.

Let $\mathcal{K}'' := \{\, i \in \mathcal{I}_{B''} \mid \bar{q}''_i < 0 \,\}$. As before, according to **s**-monotonicity, variables indexed by $\mathcal{K}''$ have not moved since bases B', or else the algorithm

would have chosen one from them, thus for all $i \in \mathcal{K}''\setminus\{l\}$, the value of u'_i (or v''_i) and u''_i (or v'_i) is zero:

$$u'_i\, v''_i + u''_i\, v'_i = 0. \tag{7}$$

From tableau (a) (or (b)), and tableau (A) (or (B)), it can be read that $u'_l = 0, v'_l < 0$ and $u''_l < 0$, $v''_l = 0$ so,

$$u'_l\, v''_l + u''_l\, v'_l > 0. \tag{8}$$

Furthermore, for any $h \notin \mathcal{K}''$ it holds that $u'_h, v'_h, u''_h, v''_h \geq 0$, thus

$$u'_h\, v''_h + u''_h\, v'_h \geq 0. \tag{9}$$

To summarize, the vector $(\mathbf{u}' - \mathbf{u}'')$ is such that $(\mathbf{u}' - \mathbf{u}'') \cdot M\,(\mathbf{u}' - \mathbf{u}'') \lneqq \mathbf{0}$. □

Note, that the proof is constructive because the vector $\mathbf{u}' - \mathbf{u}''$ proving the lack of sufficiency of our matrix can easily be obtained from the bases B' and B''.

In the last auxiliary lemma, we consider the case when tableau (c) would be followed by tableau (C).

Lemma 17 *Let us denote tableau* (c) *by* $T_{B'}$*, and tableau* (C) *by* $T_{B''}$*. Consider the vectors* $\mathbf{t}'_j$ *and* $\mathbf{t}''^{(k)}$ *belonging to the column of* u_j *in tableau* $T_{B'}$ *and to the row of* u_k *in tableau* $T_{B''}$*. Then*

$$(\mathbf{t}''^{(k)})^T\, \mathbf{t}'_j < 0.$$

Proof. Let $\mathcal{K}''_k = \{i \in \mathcal{I}_{N''} : t''_{ki} < 0\}\setminus\{j\}$. Using the second criterion of **s**-monotone pivot rules again, the variables of the indices $\mathcal{K}''_k$ have not moved since B', so $(\mathcal{I}_{N''} \setminus \mathcal{K}''_k) \subset \mathcal{I}_{B'}$ and $\mathcal{K}''_k \subset \mathcal{I}_{N'}$, thus $t'_{ij} = 0$ if $i \in \mathcal{K}''_k$. Based on these observations, we have

$$\sum_{i \in \mathcal{K}''_k \cup \{q\}} t''_{ki}\, t'_{ij} = 0. \tag{10}$$

Furthermore, if $h \notin \mathcal{K}''_k \cup \{q, l, \bar{l}, j, \bar{j}, k, \bar{k}\}$ then $t'_{hj} \leq 0$ according to tableau (c). By the definition of $\mathcal{K}''_k$, $t''_{kh} \geq 0$, so

$$\sum_{h \notin \mathcal{K}''_k \cup \{q,l,\bar{l},j,\bar{j},k,\bar{k}\}} t''_{kh}\, t'_{hj} \leq 0, \tag{11}$$

From tableaux $T_{B'}$ and $T_{B''}$, taking the definition of vector $\mathbf{t}$ into consideration, it follows that

$$t'_{lj} = t''_{k\bar{k}} = t'_{\bar{j}\bar{j}} = t'_{qj} = t''_{kl} = 0,\ t''_{kk} = 1,\ t'_{jj} = -1 \quad \text{and } t'_{kj} \leq 0,\ t''_{k\bar{l}} < 0,\ t'_{\bar{l}j} > 0$$

so

$$t''_{kq}\, t'_{qj} + t''_{kl}\, t'_{lj} + t''_{k\bar{l}}\, t'_{\bar{l}j} + t''_{kj}\, t'_{jj} + t''_{k\bar{j}}\, t'_{\bar{j}j} + t''_{kk}\, t'_{kj} + t''_{k\bar{k}}\, t'_{\bar{k}j} < -t''_{kj}. \tag{12}$$

By the definition of the algorithm, at the exchange pivot in tableau $\mathbf{c}$, the first variable pair for which the update of the $\mathbf{s}$ vector is applied is (u_l, v_l), and only after for (u_j, v_j). Hence variable (u_j, v_j) is already considered to be moved since B', thus by the second criterion of $\mathbf{s}$-monotone pivot rules their $\mathbf{s}$ value is bigger than of index l. This is only possible if $t''_{kj} \geq 0$ either because out of bases, or because of its associated $\mathbf{s}$ value.

The result follows as we sum up inequalities (10) – (12).

□

4.2 Finiteness of the criss-cross method

In this section, we prove the finiteness of the criss-cross algorithm.

Theorem 18 *The criss-cross type algorithm with* $\mathbf{s}$*-monotone pivot rule is finite for the linear complementarity problem with sufficient matrices.*

Proof. Let us assume the contrary, that the algorithm is not finite. Because a linear complementarity problem has finitely many different bases, the algorithm can have an infinite number of iterations only if it is cycling. Then we have a cycling example. Let us choose from the cycling examples one with a minimal size. Then, every variable moves during the cycle. Taking the auxiliary lemmas into consideration, after variable u_l enters the basis after basis B', it cannot leave it again:

If it enters in case (a) or (b) and leaves the basis in case (A) or (B), Lemma 16. contradicts the sufficiency of matrix M.

If it enters in case (c) and leaves the basis in case (A) or (B), Lemma 14. contradicts the orthogonality theorem.

If it enters in case (c) and leaves the basis in case (C), Lemma 17. contradicts the orthogonality theorem.

If it enters in case (a) or (b) and leaves the basis in case (C), Lemma 15. contradicts the orthogonality theorem.

All possible cases lead to a contradiction, therefore the algorithm is finite. □

Figure 8 shows the cases in which the sufficiency of matrix T has been used in the proof of finiteness of the criss-cross type algorithm.

	(a)	(b)	(c)
(A)	*	*	
(B)	*	*	
(C)			*

Figure 8: The cases when sufficiency of the pivot matrix is used

5 EP theorems and the linear complementarity problem

This section generalizes the algorithm in the sense of EP theorems. As motivation, Example 19 demonstrates that the criss-cross algorithm may solve a LCP problem even if the matrix is not sufficient.

Example 19 *To demonstrate the criss-cross method, consider the linear complementarity problem with the matrix presented in Example 6 with the corresponding short pivot tableau where the identity matrix corresponding to* $\mathbf{v}$ *serves as an initial complementary basis.*

Solving the problem with the cross-cross method, pivoting first on diagonal elements (u_1, v_1) *and* (u_2, v_2)*:*

	u_1	u_2	u_3	u_4	u_5	
v_1	**-1**	1	1	0	1	-1
v_2	1	-2	0	0	1	-2
v_3	-1	2	1	2	0	16
v_4	4	1	1	-2	-4	6
v_5	1	0	-2	0	-1	4

Tableau 1.

	v_1	u_2	u_3	u_4	u_5	
u_1	-1	-1	-1	0	-1	1
v_2	1	**-1**	1	0	2	-3
v_3	-1	1	0	2	-1	17
v_4	4	5	5	-2	0	2
v_5	1	1	-1	0	0	3

Tableau 2.

then (u_4, v_4)*:*

	v_1	v_2	u_3	u_4	u_5	
u_1	-2	-1	-2	0	-3	4
u_2	-1	-1	-1	0	-2	3
v_3	0	1	1	2	1	14
v_4	9	5	10	**-2**	10	-13
v_5	2	1	0	0	2	0

Tableau 3.

	v_1	u_2	u_3	v_4	u_5	
u_1	-2	-1	-2	0	-3	4
u_2	-1	-1	-1	0	-2	3
v_3	9	6	11	1	11	1
u_4	-4.5	-2.5	-5	-0.5	-5	6.5
v_5	2	1	0	0	2	0

Tableau 4.

arriving at the solution $u_1 = 4, u_2 = 3, v_3 = 1, u_4 = 6.5$ *and all other variables zero. The method has found a feasible complementary solution for a problem with non-sufficient matrix.*

Example 19 compensated for an obvious matrix coefficient making the matrix non-sufficient (diagonal entry for (u_3, v_3)) with a large right hand side value. As example 20 shows, this is not necessary.

Example 20 *Consider the same* M *matrix as in Example 19, but with a different right hand side.*

	u_1	u_2	u_3	u_4	u_5	
v_1	**-1**	1	1	0	1	-1
v_2	1	-2	0	0	1	2
v_3	-1	2	1	2	0	-1
v_4	4	1	1	-2	-4	5
v_5	1	0	-2	0	-1	1

Tableau 1.

	v_1	u_2	u_3	u_4	u_5	
u_1	-1	-1	-1	0	-1	1
v_2	1	-1	1	0	2	1
v_3	-1	1	0	2	-1	0
v_4	4	5	5	-2	0	1
v_5	1	1	-1	0	0	0

Tableau 2.

Pivoting on (u_1, v_1) *arrives at a feasible complementary solution, even though the row of diagonal with the incorrect sign in respect to sufficiency stared out to be infeasible.*

An EP theorem is usually a collection of several alternative possible statements, from which one always holds, and if any statement holds, then a polynomial size (in the length of the input data) certificate for it must exist. It may

also be viewed as a general framework for making provable and practically applicable theories.

The general form of an EP (Existentially Polynomial time) theorem is as follows [3]:

$$[\forall \mathbf{x} : \mathsf{F}_1(\mathbf{x}) \text{ or } \mathsf{F}_2(\mathbf{x}) \text{ or } \ldots \text{ or } \mathsf{F}_k(\mathbf{x})]$$

where $\mathsf{F}_i(\mathbf{x})$ is a statement of the form

$$\mathsf{F}_i(\mathbf{x}) = [\exists \mathbf{y}_i \text{ for which } \|\mathbf{y}_i\| \leq \|\mathbf{x}\|^{n_i} \text{ and } \mathsf{f}_i(\mathbf{x}, \mathbf{y}_i)]\,.$$

where each n_i is a positive integer.

The extended algorithm makes use of the following two theorems.

Theorem 21 *[12] Let the matrix $M \in \mathbb{R}^{n\times n}$ be not sufficient. In this case, a certificate exists that M is not sufficient, the coding size of which is polynomially bounded by the input length of matrix M .*

Theorem 22 *[12]. For any matrix $M \in \mathbb{Q}^{n\times n}$ and vector $\mathbf{q} \in \mathbb{Q}^n$, at least one of the following statements holds:*

(1) problem (P-LCP) has a complementary, feasible solution $(\mathbf{u}, \mathbf{v})$ the encoding size of which is polynomially bounded by the input length of matrix M and vector $\mathbf{q}$.

(2) problem (D-LCP) has a complementary, feasible solution $(\mathbf{x}, \mathbf{y})$ the encoding size of which is polynomially bounded by the input length of matrix M and vector $\mathbf{q}$.

(3) matrix M is not sufficient, and there is a certificate the encoding size of which is polynomially bounded by the input length of matrix M.

Note that cases (1) and (2) are exclusive, while case (3) can hold alone or together with either case (1) or (2). It is a naturally arising condition that the entries of the matrix should be rational numbers.

We modify the extended criss-cross algorithm so that it either solves problem (P−LCP) or its dual, or proves the lack of sufficiency of the input matrix, giving a polynomial size certificate.

Lemma 4 ensures that the pivot operations can always be done if our matrix is sufficient, and if it is not, it provides the required certificate that matrix M is not sufficient.

The criss-cross type algorithm with s-monotone index selection rules in the form of EP-theorems

Input: $T = -M$, $\bar{q} = q$, $r = 1$, Initialize Q and s.
Begin
While $((\mathcal{J} := \{i \in \mathcal{I} \mid \bar{q}_i < 0\}) \neq \emptyset)$ **do**
$\mathcal{J}_{max} := \{\beta \in \mathcal{J} \mid s(\beta) \geq s(\alpha)$, for all $\alpha \in \mathcal{J}\}$.
Let $k \in \mathcal{J}_{max}$ be arbitrary.
Check $-u' \cdot v'' - u'' \cdot v''$ with the help of $Q(k)$.
If $(-u' \cdot v'' - u'' \cdot v'' \lneqq 0)$ **then**
Stop: M is not sufficient, certificate: $u' - u''$.
Endif
If $(t_{kk} < 0)$ **then**
Diagonal pivot on t_{kk}, update s.
$Q(k) = [\mathcal{J}_B, \bar{t}_q]$, $r := r + 1$.
ElseIf $(t_{kk} > 0)$
Stop: M is not sufficient, create certificate.
Else /* $t_{kk} = 0$ */
$K := \{\alpha \in I \mid \bar{t}_{k\alpha} < 0\}$
If $(K = \emptyset)$ **then**
Stop: DLCP solution.
Else
$\mathcal{K}_{max} = \{\beta \in K \mid s(\beta) \geq s(\alpha)$, for all $\alpha \in K\}$.
Let $l \in \mathcal{K}_{max}$ be arbitrary.
If ((t_k, t^k) or (t_l,t^l) sign structure is violated) **then**
Stop: M is not sufficient, create certificate.
Endif
Exchange pivot on t_{kl} and t_{lk}, update s first for (u_k, v_k), then for (u_l, v_l) as in a next iteration.
$Q(k) = [J_B, \bar{t}_q]$, $Q(l) = [\emptyset, 0]$, $r := r + 2$.
Endif
Endif
EndWhile
Stop: we have a complementary feasible solution.
End

Figure 9: The criss-cross type algorithm with **s**-monotone index selection rules in the form of EP-theorems

As Example 23 shows, in the lack of sufficiency the criss-cross method may cycle.

Example 23 *Consider the same* M *matrix as in Example 19 and Example 20 but with a another different right hand side.*

The criss-cross algorithm first pivots on (u_1, v_1)*. In the second tableau, an exchange pivot is necessary, pivoting on* (u_3, v_5) *and then* (u_5, v_3) *leading to Tableau 4.*

	u_1	u_2	u_3	u_4	u_5	
v_1	**-1**	1	1	0	1	-1
v_2	1	-2	0	0	1	2
v_3	-1	2	1	2	0	0
v_4	4	1	1	-2	-4	10
v_5	1	0	-2	0	-1	0

Tableau 1.

	v_1	u_2	u_3	u_4	u_5	
u_1	-1	-1	-1	0	-1	1
v_2	1	-1	1	0	2	1
v_3	-1	1	0	2	-1	1
v_4	4	5	5	-2	0	6
v_5	1	1	**-1**	0	0	-1

Tableau 2.

On Tableau 4, another exchange picot takes place, (v_3, u_5) *and then* (v_5, u_3) *resulting in Tableau 6.*

	v_1	u_2	v_5	u_4	u_5	
u_1	-2	-2	-1	0	-1	2
v_2	2	0	1	0	2	0
v_3	-1	1	0	2	**-1**	1
v_4	9	10	5	-2	0	1
u_3	-1	-1	-1	0	0	1

Tableau 3.

	v_1	u_2	v_5	u_4	v_3	
u_1	-1	-3	-1	-2	-1	1
v_2	0	2	1	4	2	2
u_5	1	-1	0	-2	**-1**	-1
v_4	9	10	5	-2	0	1
u_3	-1	-1	-1	0	0	1

Tableau 4.

	v_1	u_2	v_5	u_4	u_5	
u_1	-2	-2	-1	0	-1	2
v_2	2	0	1	0	2	0
v_3	-1	1	0	2	-1	1
v_4	9	10	5	-2	0	1
u_3	-1	-1	**-1**	0	0	1

Tableau 5.

	v_1	u_2	u_3	u_4	u_5	
u_1	-1	-1	-1	0	-1	1
v_2	1	-1	1	0	2	1
v_3	-1	1	0	2	-1	1
v_4	4	5	5	-2	0	6
v_5	1	1	-1	0	0	-1

Tableau 6.

Tableau 6 coincides with Tableau 2: the pivot choices have been unique as the negative right hand side values were unique: the algorithm cycles.

We now need to analyse the proofs of finiteness of the original algorithm.

Avoiding cycling: Note, that minimality of the cycling example is not necessary in the proof of finiteness of the original algorithm. Every proof remains valid for any cycling example, since those variables that do not move during a cycle do not change their basis status, thus have a zero value in the orthogonality theorem.

Consider an arbitrary cycling example. Let the index set of the variables involved in the cycling be R, and consider an iteration, when cycling has already begun. Let the basis B' such that satisfies the second criterion of **s**-monotonicity with variable x_l.

In this case the structure of tableaux $M_{B'}$ and $M_{B''}$ restricted to indices R and vector $\mathbf{q}$ is exactly like in case $(a) - (c)$ and $(A) - (C)$. Between these two tableaux, a variable whose index is not in R has not moved. Thus, in the product of the vectors analysed in Lemmas 14, 15 and 17, for the indices not in R and not q exactly one of the corresponding variables is in basis, so the contribution to the product for these indices is always zero. For the same reason, in the product of $-\mathbf{u}' \cdot \mathbf{v}'' - \mathbf{u}'' \cdot \mathbf{v}'$ in Lemma 16, the entries for the indices not in R are each zero. This proves that the proofs are valid for an arbitrary cycling example.

Handling the lack of sufficiency: Sufficiency has only been used in Lemmas 16 and 17**.** This latter one used the sign property of sufficient matrices, based on Lemma 4. Therefore, if the algorithm checks that the required sign property is fulfilled during every exchange pivot (cases (c) and (C) refer to such pivots), tableau (c) cannot be followed by tableau (C), because of the orthogonality theorem. If the required sign structure is violated, the certificate that matrix M is not sufficient is provided by the same lemma.

There remain the cases of tableaux (a)-(b) and (A)-(B)**.** Lemma 16 handled this case. The proof of the lemma is based on the product

$$-\mathbf{u}' \cdot \mathbf{v}'' - \mathbf{u}'' \cdot \mathbf{v}' \tag{13}$$

referring to such subsequent tableaux $M_{B'}$ and $M_{B''}$, where the same variable moves during both pivot operations, and in both cases this variable was chosen actively (that is, not as the second variable of an exchange pivot). Note, that we do not need the whole tableau here, the only information we use is the column of $\mathbf{q}$ (the actual complementary solution) and the set of indices in the basis. If vector (13) is strictly sign reversing, then as in the note after Lemma 16, the evidence that matrix M is not sufficient is the vector $\mathbf{u}' - \mathbf{u}''$.

Let us introduce a list $Q(p)$ $(p = 1, \ldots, n)$. Two vectors of dimension n belongs to every entry of this list. At the beginning,

$$Q(p) := \begin{bmatrix} [1,\ldots,n] \\ [0,\ldots,0] \end{bmatrix} \quad p = 1,\ldots,n.$$

When variable u_l or v_l leaves the basis during a diagonal pivot or such an exchange pivot where this variable is active (variable selected first), we modify the value of $Q(l)$ in such a way, that we write the *indices* of variables in basis to the first vector before the actual pivot operation, while we write the *values* of variables in basis before the pivot operation to the second vector:

$$Q(l) := \begin{bmatrix} [\text{ indices of variables in basis }] \\ [\text{ values of variables in basis}] \end{bmatrix}.$$

If variable u_l or v_l enters the basis passively (as the second variable of an exchange pivot), we modify the value of $Q(l)$ as:

$$Q(l) := \begin{bmatrix} [1,\ldots,n] \\ [0,\ldots,0] \end{bmatrix}.$$

An operation $Q(j) = [\{I\},\{\mathbf{h}\}]$ means that to the entry of j in the list Q, we write list I to the place of basic indices, while the values of the vector $\mathbf{h}$ in the place of $\mathbf{q}$.

Before the algorithm performs a pivot operation, it checks if the actively selected variable that enters the basis was chosen actively previously or not when it left the basis. If yes, with the help of list Q, it checks vector (13) and only after this does it modify list Q. Because the complementary pairs of variables move together during the pivot operations, it is not necessary to provide space for both of them in list Q.

Note, that because the definition of the initial values of Q and the modification of Q during a passive exchange pivot, it suffices to check the product (13) during any pivot. If tableau (a) (or (b)) is followed by tableau (A) (or (B)), the product will always be zero.

It would not be necessary to fill out list Q every time. With a slight modification of the algorithm, we would be able to save storage space as well. In the worst case, the storage space required by list Q would be the storage space required to store n^2 integer and n^2 rational numbers.

We have to investigate the case when $(P-LCP)$ has no solution. This occurs when $K = \emptyset$. The structure of the pivot tableau is shown in Figure 11. Consider the vector

$$(\mathbf{x}', \mathbf{y}') = \mathbf{t}^{(k)} \mid_{J_N \cup J_B}.$$

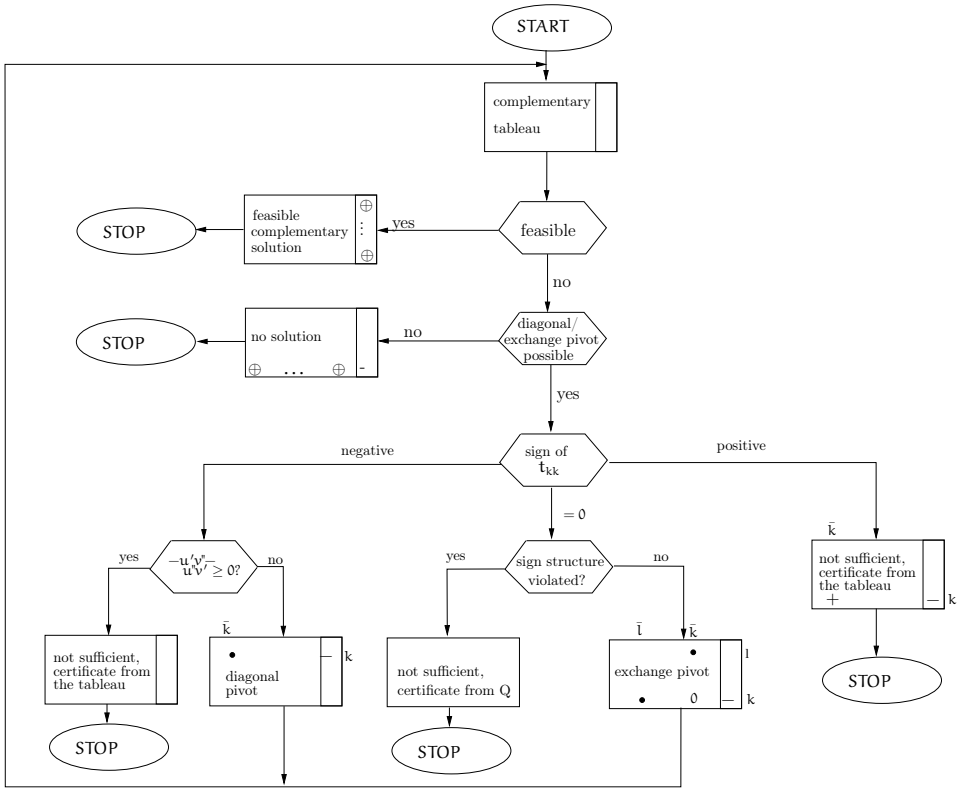


Figure 10: Flow chart of the modified criss-cross algorithm

Using the orthogonality theorem, we get that this vector is orthogonal to every row of $[-\mathsf{M}^{\mathsf{T}} \mid -\mathsf{I}]$, in other words $\mathsf{M}^{\mathsf{T}}\mathbf{x}' + \mathbf{y}' = \mathbf{0}$. Applying the orthogonality theorem to the column of the right-hand side vector $\mathbf{q}$ (in the starting basis), we have

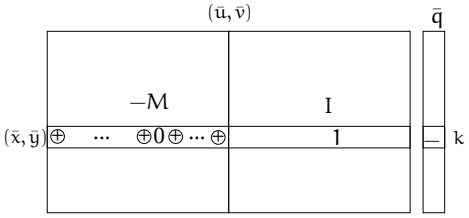
$$(\mathbf{x}', \mathbf{y}')^{\mathsf{T}}\mathbf{t}_{\mathsf{q}} \mid_{\mathsf{J}_{\mathsf{N}} \cup \mathsf{J}_{\mathsf{B}}} = (\mathbf{x}', \mathbf{y}')^{\mathsf{T}}(\mathbf{q}, \mathbf{0}) = \mathbf{x}'^{\mathsf{T}}\mathbf{q} = \mathsf{q}_{\mathsf{k}}.$$

So the vector $(\mathbf{x}, \mathbf{y}) = (\mathbf{x}', \mathbf{y}')/(-\mathsf{q}_{\mathsf{k}})$ is a solution to the problem $(\mathsf{D}-\mathsf{LCP})$, because nonnegativity and complementary follow from the structure of the pivot tableau.

Based on the discussion above, we showed that the modified criss-cross algorithm can be started from any complementarity basic solution of a general linear complementarity problem without apriori information on the matrix

property and using **s**-monotone index rule, in a finite number of iteration the algorithm stops with one of the EP theorem cases, stated in Theorem 22.

Next example shows that the modified criss-cross algorithm (see on Figure 9 and it's flowchart on Figure 10) identifies the lack of sufficiency for matrix M.



$$(x,y)_i = \begin{cases} -t_{ki}/\bar{q}_k & \text{if } i \in I_N \\ -1/\bar{q}_k & \text{if } i = k \\ 0 & \text{otherwise.} \end{cases}$$

Figure 11: The dual solution when no primal solution exists

Example 24 *As an example, consider the problem presented in Example 23 again. The extended algorithm would stop on Tableau 3: the sign of the second pivot position of the exchange pivot violates the expected sign structure.*

6 Computational experiences

In this section we provide some numerical experience using the proposed algorithm for solving general LCP problems arising from Arrow-Debreu exchange matrix problems and bimatrix games. The experiments have been carried out in Matlab, using the built in QR decomposition and update for the basis. To maximize the chance of success, in positions when the selected **s**-monotone rule (MOSV) offered flexibility of pivot selection, a random position has been selected from among the eligible choices.

The results presented here emphasize the applicability of the new variant of the criss-cross method together with the flexibility of the **s**-monotone index selection rules. For a comprehensive numerical study concentrating on the **s**-monotone index selection rules see [15].

6.1 A market equilibrium problem

Consider the exchange market equilibrium problem as described by Walras [28]. There are m traders (players) and n goods on the market, where each

good type j has a price $p_j \geq 0$. Each trader i is assumed to have an initial endowment of commodities
$\mathbf{w}_i = (w_{i1}, \ldots, w_{in}) \in \mathbb{R}^n_{\oplus}$. The traders will sell their product on the market and use their income to buy a bundle of goods $\mathbf{x}_i = (x_{i1}, \ldots, x_{in}) \in \mathbb{R}^n_{\oplus}$. A trader i has a utility function u_i, which describes his preferences for the different bundle of commodities and a budget constraint $\mathbf{p}^T\mathbf{x}_i \leq \mathbf{p}^T\mathbf{w}_i$. Finally, each trader i maximizes his individual utility function subject to his budget constraint.

Each trader optimizes his own utility function u_i with these side constraints:

$$\begin{array}{rcl} \max & u_i(\mathbf{x}_i) & \\ \mathbf{p}^T\mathbf{x}_i & \leq & \mathbf{p}^T\mathbf{w}_i \\ \mathbf{x}_i & \geq & \mathbf{0}, \end{array}$$

where the vector of prices $\mathbf{p}$ is an equilibrium for the exchange economy; if there is a bundle of goods $\mathbf{x}_i(\mathbf{p})$ (so a maximizer of the utility function u_i subject to the budget constraint) for all traders i, such that

$$\sum_{i=1}^{m} x_{ij}(\mathbf{p}) \leq \sum_{i=1}^{m} w_{ij} \quad \text{for all goods } j.$$

The exchange market equilibrium problem pursues prices where the demand $\sum_i x_{ij}(\mathbf{p})$ does not exceed the supply $\sum_i w_{ij}$ for any good j.

Arrow and Debreu [2] proved that under mild conditions, for concave utility functions, the exchange markets equilibrium exists. Using the Leontief utility function

$$u_i(\mathbf{x}_i) = \min_j \left\{ \frac{x_{ij}}{a_{ij}} : \ a_{ij} > 0 \right\},$$

where $A = (a_{ij}) \in \mathbb{R}^{n\times n}_{\oplus}$ is the Leontief coefficient matrix, Ye [25] has shown that the solution of the Arrow-Debreu competitive market equilibrium problem with Leontief's utility function is equivalent to the following linear complementarity problem:

$$A^T\mathbf{u} + \mathbf{v} = \mathbf{e}, \quad \mathbf{u} \geq \mathbf{0}, \mathbf{v} \geq \mathbf{0}, \quad \mathbf{u}\,\mathbf{v} = \mathbf{0} \ \text{ and } \ \mathbf{u} \neq \mathbf{0}$$

where the matrix A has non-negative entries.

It is easy to see that this problem will almost always have a non-sufficient matrix, and as such is a demanding problem class for the generalized criss-cross algorithm.

The computational experiences have been carried out using a 100 problems, ranging

$n \in \{10, 20, 40, 60, 80, 100, 200, 300, 400, 500\}$ taking 10 random instances for each value of n. For this problem the trivial basis corresponding to the columns of $\mathbf{v}$ is not valid, as it is a feasible solution to the problem, but $\mathbf{u} \neq \mathbf{0}$ does not hold.

To address this problem, a structural crash heuristic has been used, based on the following heuristics. Start from the empty selection $B = \emptyset$. For any set of vectors, define it's support as $\mathrm{supp}(B) = \{j : \exists i : \mathbf{a}_i \in B, a_{j,i} \neq 0\}$. In each iteration of the crash heuristics for all $i \in 1 \dots n$, if there is such an index j for which $a_{ji} \neq 0$ and $i \notin \mathrm{supp}(B)$, then add $\mathbf{a}_i$ to B, else, add the corresponding identity vector $\mathbf{e}_i$ to the B. It is easy to see, that this procedure will yield a complementary basis, and unless $A = 0$ it will contain at least one columns corresponding to $\mathbf{u}$. When the selection of the column from A was not unique, the algorithm has selected randomly. Each experiment was repeated 10 times, yielding a total of 1000 test runs.

n	Successful solves	Mean iterations	Maximum iterations
10	762	0.619	12
20	318	1.432	12
40	26	1.445	9
60	13	1.479	10
80	2	1.396	9
100	0	1.295	10
200	0	1.213	9

Table 1: Numerical experiments on random market equilibrium problems

The number of successful solves diminished very quickly with size, also the algorithm terminates after a very small iteration count, with declaring the matrix not sufficient in most cases. There has been no successful solves for $n \geq 100$ suggesting that most of the successful solves for smaller sizes strongly depend on luck associated with the crash heuristics applied.

6.2 A bimatrix game

In this problem, two companies are considering entering n markets. Entering a market has both fixed, and variable costs, and there are fixed transport charges. The problem of finding optimal profit strategies can be formulated as a bimatrix game [21]. Consider a bimatrix game defined by A and B.

Theorem 25 *The Karush-Khun-Tucker conditions of the bimatrix game can equivalently be formulated as*

$$\begin{aligned} \mathbf{x}^{\mathsf{T}}(A+B)\mathbf{y} - \alpha - \beta &\rightarrow \max \\ \mathbf{x}, \mathbf{y} &\geq \mathbf{0} \\ \mathbf{1}_m^{\mathsf{T}}\mathbf{x} &= \mathbf{1} \\ \mathbf{1}_n^{\mathsf{T}}\mathbf{y} &= \mathbf{1} \\ A\mathbf{y} &\leq \alpha\mathbf{1}_m \\ B^{\mathsf{T}}\mathbf{x} &\leq \beta\mathbf{1}_n \end{aligned}$$

where the zero valued solutions are the Nash-equilibria.

This Karush-Khun-Tucker conditions for this quadratic programming problem can be stated as

$$\begin{pmatrix} Q & P^{\mathsf{T}} \\ -P & O \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{t} \end{pmatrix} + \begin{pmatrix} \mathbf{s} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{c} \\ \mathbf{b} \end{pmatrix}$$

$$\begin{aligned} \mathbf{u}\,\mathbf{t} &= \mathbf{0} \\ \mathbf{s}\,\mathbf{v} &= \mathbf{0} \\ \mathbf{u}, \mathbf{t}, \mathbf{s}, \mathbf{v} &\geq \mathbf{0} \end{aligned}$$

where

$$Q = \begin{pmatrix} O & -A-B \\ (-A-B)^{\mathsf{T}} & O \end{pmatrix} \qquad \mathbf{u} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \in \mathbb{R}^{m+n}$$

$$P = \begin{pmatrix} B^{\mathsf{T}} & O & 0 & 0 & -1 & 1 \\ O & A & -1 & 1 & 0 & 0 \\ \mathbf{1}^{\mathsf{T}} & \mathbf{0} & 0 & 0 & 0 & 0 \\ -\mathbf{1}^{\mathsf{T}} & \mathbf{0} & 0 & 0 & 0 & 0 \\ \mathbf{0} & \mathbf{1}^{\mathsf{T}} & 0 & 0 & 0 & 0 \\ \mathbf{0} & -\mathbf{1}^{\mathsf{T}} & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \mathbf{t} = \begin{pmatrix} \alpha^+ \\ \alpha^- \\ \beta^+ \\ \beta^- \end{pmatrix} \in \mathbb{R}^4.$$

As A and B can be chosen arbitrary, it is easy to see that the resulting LCP will not necessarily be sufficient.

Experiments have been carried out for payoff matrices with $n = m = 2, 3, \ldots, 7$, with each experiment using random values for A and B and repeated 1000 times. As a trivial initial complementary bases, the identity columns corresponding to variables $(\mathbf{s}, \mathbf{v})$ have been selected.

Payoff matrix size	Local solution (obj $>$ 0.001)	Close to zero (obj $<$ 0.001)	Local solution (obj $>$ 0.01)	Close to zero (obj $<$ 0.01)
2x2	563	437	516	484
3x3	768	232	684	316
4x4	908	92	801	199
5x5	931	69	827	173
6x6	960	40	855	145
7x7	973	27	871	129

Table 2: Numerical experiments on random bimatrix games

Although the number of successful solves where the objective of the original quadratic problem is zero diminishes, the algorithm managed to find solution in a reasonable portion of the problems. Computational results are summarized on Table 2, where the optimal objective function value, due to numerical computational errors, claimed to be optimal in the interval $[0, \varepsilon)$. In case of column two of the Table 2, $\varepsilon = 0.001$, while in column three $\varepsilon = 0.01$. This shows that the computational precision may influence which problem will be declared as solved problem.

7 Summary

We have presented a variant of a generalized criss-cross type algorithm for linear complementarity problems with sufficient matrices, using **s**-monotone pivot rules. For better practical applicability, we have modified the generalized algorithm so that the a priori information on the sufficiency of the matrix is not necessary. In case of lack of sufficiency, if the algorithm cannot ensure finiteness, then it terminates and provides a polynomial size certificate that the matrix is not sufficient. We have achieved our goals using the duality theorem of linear complementarity problems [13] and with its EP theorem form [12]. With the use of flexible **s**-monotone pivot rules, the algorithm provides significant freedom in choosing the pivot position (usually during the first part of the algorithm), making it possible to avoid some numerically instable pivots.

Some supporting evidence for the applicability of the proposed algorithm has been presented by solving general linear complementarity problems arising from bimatrix games and the Arrow-Debreu market equilibrium problem, where the algorithm has proved to be applicable in (rather) small dimensions.

Acknowledgements

This research has been supported by the TÁMOP-4.2.2./B-10/1-2010-0009, Hungarian National Office of Research and Technology with the financial support of the European Union from the European Social Fund.

Tibor Illés acknowledges the research support obtained from Strathclyde University, Glasgow under the *John Anderson Research Leadership Program*.

Parts of this paper has been included in the Ph.D. thesis of Zsolt Csizmadia [7].

References

[1] A. A. Akkeleş, L. Balogh, T. Illés, New variants of the criss-cross method for linearly constrained convex quadratic programming, *Eur. J. Oper. Res.*, **157,** 1 (2004) 74–86. ⇒104

[2] K. J. Arrow, G. Debreu, Existence of an equilibrium for competitive economy, *Econometrica*, **22** (1954) 265–290. ⇒133

[3] K. Cameron, J. Edmonds, Existentially polytime theorems, in: *Polyhedral combinatorics*, (eds. W. Cook, P.D. Seymour) DIMACS Ser. Discrete Math. Theoret. Comput. Sci., American Mathematical Society, Providence, RI, 1990, pp. 83–100. ⇒126

[4] Y. Y. Chang, *Least index resolution of degeneracy in linear complementarity problems*, Technical Report 79-14, Department of Operations Research, Stanford University, Stanford, Calfornia, USA, 1979. ⇒104

[5] S. J. Chung, NP-completeness of the linear complementarity problem, *J. Optimiz. Theory Appl.*, **60,** 3 (1989) 393–399. ⇒110

[6] R. W. Cottle, J. S. Pang, V. Venkateswaran, Sufficient matrices and the linear complementarity problem, *Linear Algebra Appl.*, **114/115** (1989) 231–249. ⇒104, 107, 108

[7] Zs. Csizmadia, *New pivot based methods in linear optimization, and an application in petroleum industry*, PhD Thesis, Eötvös Loránd University of Sciences, 2007. ⇒137

[8] Zs. Csizmadia, T. Illés, New criss-cross type algorithms for linear complementarity problems with sufficient matrices, *Optim. Methods Softw.*, **21,** 2 (2006) 247–266. ⇒106, 109, 114

[9] Zs. Csizmadia, T. Illés, A. Nagy, The s-monotone index selection rules for pivot algorithms of linear programming, *European J. Oper. Res.*, **221,** 3 (2012) 491–500. ⇒104, 106, 111, 112

[10] Gy. Farkas, A Fourier-féle mechanikai elv alkalmazásai (the applications of the mechanical principle of fourier), *Mathematikai és Természettudományi Értesítő*, **12** (1894) 457–472. ⇒105, 110

[11] Gy. Farkas, Theorie der einfachen ungleichungen, *J. Reine Angew. Math.*, **124** (1901) 1–27. ⇒105, 110

[12] K. Fukuda, M. Namiki, A. Tamura, EP theorems and linear complementarity problems, *Discrete Appl. Math.*, **84,** 1-3 (1998) 107–119. ⇒105, 107, 108, 109, 126, 136

[13] K. Fukuda T. Terlaky, Linear complementarity and oriented matroids, *J. Oper. Res. Soc. Japan*, **35,** 1 (1992) 45–61. ⇒105, 110, 136

[14] D. den Hertog, C. Roos, T. Terlaky, The linear complementarity problem, sufficient matrices, and the criss-cross method, *Linear Algebra Appl.*, **187,** 1 (1993) 1–14. ⇒104, 107, 109

[15] T. Illés, A. Nagy, Computational aspects of simplex and MBU-simplex algorithms using different anti-cycling pivot rules, *Department of Operations Research, Eötvös Loránd University of Sciences, Operations Research Report*, **2012-02** (2012) submitted for publication. ⇒132

[16] T. Illés, M. Nagy, T. Terlaky, EP theorem for dual linear complementarity problems, *J. Optim. Theory Appl.*, **140,** 2 (2009) 233–238. ⇒105

[17] T. Illés, M. Nagy, T. Terlaky, Polynomial interior point algorithms for general linear complementarity problems, *Algorithmic Oper. Res.h*, **5,** 1 (2010) 1–12. ⇒105

[18] T. Illés, M. Nagy, T. Terlaky, A polynomial path-following interior point algorithm for general linear complementarity problems, *J. Global Optim.*, **47,** 3 (2010) 329–342. ⇒105

[19] E. Klafszky, T. Terlaky, The role of pivoting in proving some fundamental theorems of linear algebra, *Linear Algebra Appl.*, **151** (1991) 97–118. ⇒108, 119

[20] M. Kojima, N. Megiddo, T. Noma, A. Yoshise, A Unified Approach to Interior Point Algorithms for Linear Complementarity Problems, *Lecture Notes in Computer Science* **538**, Springer-Verlag, Berlin, 1991. ⇒104, 105

[21] C.E. Lemke, J.T. Howson, Jr., Equilibrium points of bimatrix games, *Journal Soc. Indust. Appl. Math.*, **12** (1964) 413–423. ⇒134

[22] C. Roos, T. Terlaky, J.-Ph. Vial, *Theory and Algorithms for Linear Optimization: An Interior Point Approach*, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, New York, USA, 1997. Second edition: Interior PointMethods for Linear Optimization, Springer, New York, 2006. ⇒105

[23] T. Terlaky, A convergent criss-cross method, *Optimization*, **16,** 5 (1985) 683–690. ⇒104

[24] P. Tseng, Co-NP-completeness of some matrix classification problems, *Math. Program.*, **Series A:88** (2000) 183–192. ⇒104

[25] Y. Ye, A Path to the Arrow-Debreu Competitive Market Equilibrium, *Math. Program.*, **111,** 1-2 (2008) 315–348. ⇒133

[26] H. Väliaho, Criteria for sufficient matrices, *Linear Algebra Appl.*, **233** (1996) 109–129. ⇒104, 109

[27] H. Väliaho, $\mathbf{P}_*$-matrices are just sufficient, *Linear Algebra Appl.*, **239** (1996) 103–108. ⇒104

[28] L. Walras, *Elements of Pure Economics, or the Theory of Social Wealth*, (1874) (1899, 4th ed.; 1926, rev. ed., 1954, Engl. Transl.) ⇒132

[29] Z.M. Wang, A finite conformal-elimination free algorithm over oriented matroid programming, *Chin. Ann. Math. Ser. B*, **8,** 1 (1987) 120–125. ⇒104

Received: February 19, 2013 • *Revised: April 10, 2013*

Acta Universitatis Sapientiae

The scientific journal of Sapientia Hungarian University of Transylvania publishes original papers and surveys in several areas of sciences written in English.
Information about each series can be found at
http://www.acta.sapientia.ro.

Acta Universitatis Sapientiae, Informatica

Each volume contains two issues.

Sapientia University

Scientia Publishing House

ISSN 1844-6086
http://www.acta.sapientia.ro

Information for authors

Acta Universitatis Sapientiae, Informatica publishes original papers and surveys in various fields of Computer Science. All papers are peer-reviewed.

Papers published in current and previous volumes can be found in Portable Document Format (pdf) form at the address: `http://www.acta.sapientia.ro`.

The submitted papers should not be considered for publication by other journals. The corresponding author is responsible for obtaining the permission of coauthors and of the authorities of institutes, if needed, for publication, the Editorial Board is disclaiming any responsibility.

Submission must be made by email (`acta-inf@acta.sapientia.ro`) only, using the LaTeX style and sample file at the address `http://www.acta.sapientia.ro`. Beside the LaTeX source a pdf format of the paper is necessary too.

Prepare your paper carefully, including keywords, ACM Computing Classification System codes (`http://www.acm.org/about/class/1998`) and AMS Mathematics Subject Classification codes (`http://www.ams.org/msc/`).

References should be listed alphabetically based on the Intructions for Authors given at the address `http://www.acta.sapientia.ro`.

Illustrations should be given in Encapsulated Postscript (eps) format.

One issue is offered each author free of charge. No reprints will be available.

Contact address and subscription:
Acta Universitatis Sapientiae, Informatica
RO 400112 Cluj-Napoca
Str. Matei Corvin nr. 4.
Email: acta-inf@acta.sapientia.ro

Printed by Gloria Printing House
Director: Péter Nagy

ISSN 1844-6086
http://www.acta.sapientia.ro