Acta Universitatis Sapientiae

Informatica

Volume 6, Number 2, 2014

Sapientia Hungarian University of Transylvania
Scientia Publishing House

Contents

Acta Univ. Sapientiae, Informatica, **6,** 2 (2014) 163–177

A new model for the linear 1-dimensional online clustering problem

András LONDON
University of Szeged
Institute of Informatics
email: london@inf.u-szeged.hu

József NÉMETH
University of Szeged
Institute of Informatics
email: nemjozs@inf.u-szeged.hu

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

Áron PELYHE
University of Szeged
Institute of Informatics
email: pelyhe@inf.u-szeged.hu

Abstract. In this study, a mathematical model is presented for an online data clustering problem. Data clustering plays an important role in many applications like handling the data acknowledgment problem and data stream management in real-time locating systems. The inputs in these problems are data sequences, each containing several data elements. Each data element has an arrival time and a weight that reflects its importance. The arrival times are not known in advance, and some data elements never arrive. Hence the system should decide which moment is optimal for forwarding the collected data for processing. This requires finding a good trade-off between the amount of collected information and the waiting time, which may be regarded as a minimization problem. Here, we investigate several online algorithms and present their competitive analysis and average case studies. Experimental results, based on simulations using artificially generated data, are also presented and they confirm the efficiency of our methods.

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

1 Introduction

Online optimization is concerned with the type of problems where decisions should be made without knowing the whole input data. These class of problems are usually called online problems (in contrary, the offline problems are those problems where the whole input is available when the decision is made). In several parts of computer science, economics and operational research many problems can be solved only in an online manner. For some good discussions of this see, for instance [2, 12, 14].

In online clustering problems the goal is the classification of points into sets in an online fashion such that a given objective function, which depends on the distance between any two points in the same cluster, is minimized. Points that arrive consecutively have to be assigned to clusters at the time of arrival. Previous results on the online data clustering problem for data sequences can be found, for example, in [5, 11], with unit sized clusters and in [7, 8, 9] with variable sized clusters.

The problem that is closely related to the data clustering problem examined here first emerged during the study of a real-time locating system developed by the Fraunhofer IIS and investigated and generalized by Németh et al. in [17]. Below, we apply a different mathematical model for the problem, which is a minimizing problem in contrast to the maximizing one defined in [17].

A real-time locating system (RTLS) serves to determine positions of objects with high precision. It has various applications in transport and logistic, ambient assisted living, emergency mission support and also in sports, such as in the so-called Chip-in-the-Ball technologies. As an example, the locating system developed by the Fraunhofer IIS is a radio-based system operating with a local positioning infrastructure using two measurements (the "angle of arrival" and the "round trip time") to detect the position of objects. For a detailed description of RTLS systems see [3] and [4].

In such a system, an object (like the ball of a football game or the shin pad of a player) is equipped with a tag which periodically broadcasts radio signals to infrastructure nodes (such as receiver devices in the stadium). Besides the positioning data, a radio signal also carries the user data and an ID. The signals having the same ID belong to the same locating cycle, which we call a burst. The measured parameters from a given burst data set are distributed over the infrastructure nodes. After a measurement is made for the individual parts of the burst data set, the data elements are forwarded to the central positioning server (via a transport protocol). After receiving sufficient data (ideally the total burst data set), the positioning server can determine the

positions of the objects by going through the following algorithmic steps in the server: (1) data filtering, (2) data clustering (3) burst filtering (4) position calculation (5) position filtering.

In this study, we focus on the second step. Our aim is to forward the data as quickly as possible for burst filtering, and also minimize the number of unprocessed data elements to get more precise results in position calculations. The nature of the problem requires that we work in an online manner. The algorithm we designed also has to handle transmission errors and disturbances like provisional coverage (screening) of an infrastructure node, network packet losses and processing delays. Therefore, the data clustering algorithm cannot simply wait for all elements of a burst; a more sophisticated approach is needed! Evidently, the crucial point of the algorithm is to determine the time when the collected data items have to be sent to the server for calculating the positions. Two different objectives should be considered at the same time: (i) the positioning server should receive as much available data from the infrastructure nodes as possible, and (ii) the system is not allowed to wait too long for the incoming data, since the long delays decrease the relevance of the calculated position. The usual principle in a real-time calculation of positions is that a fairly-good position is better than a more accurate position determined too late. We will define a minimization problem that takes into account both goals. The second goal is considered directly, while the first goal appears in the objective function in an indirect way.

Here, collecting the data and calculating the positions of several tags can be done in parallel and independently of each other, hence we will assume that there is only one tag in the system and the infrastructure nodes collect the data only from this tag. We should also mention that this problem is similar to the online data acknowledgment problem (see e.g. [15, 1, 10, 16, 18]). In a computer network, from a communication aspect, data is sent by packets and in the data acknowledgment problem, given a sequence X of packet arrival times, the goal is to partition X into subsequences, where the end of a subsequence is defined by an acknowledgment. One acknowledgment may acknowledge many packets, but if an algorithm waits too long, the sender might resend the packets and this would result in congestion of the packets in the network. Here, our methods may also be applied to handle the data acknowledgment task.

Below, we present a mathematical model and describe the corresponding objective function for the problem. In Section 3, we apply the method of competitive analysis, which is often used to evaluate the quality of online algorithms (see [14] for a nice summary on competitive analysis). We will prove a result which states that there is no competitive algorithm for the problem.

The average case study is also presented for real-world situations, where arrival times of the input elements follow a certain probability distribution. For some discussions on the probabilistic analysis of algorithms see [6] and [13]. Lastly in Section 4, we present experimental results got from using different algorithms with artificially generated data which we used to verify the efficiency of our algorithms.

2 The proposed mathematical model

The input of the data clustering problem is a sequence of cycles $X=(X^1, X^2, \dots)$, where each cycle X^k is a sequence of data elements $(x_1^k, \dots, x_m^k)$ where m is the number of all possible incoming data elements (and it is equal to the number of infrastructure nodes) in a cycle. The reception time of data x_i^k is denoted by t_i^k $(i = 1, 2, \dots, m;\ k = 1, 2, \dots)$. Since there is no guarantee that all data elements will arrive in each cycle, let $t_i^k = \infty$ if x_i^k does not arrive. The bursts will be denoted by $B^1, B^2, \dots$ where $B^k = \{x_i^k \in X^k : t_i^k < \infty\}$. The difference between the reception times of the last and first data elements of the same burst is called the length of the burst.

The exact time when an algorithm decides to send the collected data in burst B^k to the server will be denoted by p_k $(k = 1, 2, \dots)$ and it is called the positioning time. Let $\hat{B}^k(t) = \{x_i^k : t_i^k < t\}$; thus, $\hat{B}^k(p_k) = \hat{B}^k$ is the subset of data elements in burst B^k which is sent to the computer for positioning.

Usually, the infrastructure nodes (wireless smart items, goniometers) have different technical properties (type, position, accuracy), hence the data collected may not all have the same importance. In our model, we assign a weight w_i to data x_i^k $(k = 1, 2, \dots)$, which denotes the importance of the data with respect to the infrastructure node that collected it. Without loss of generality,

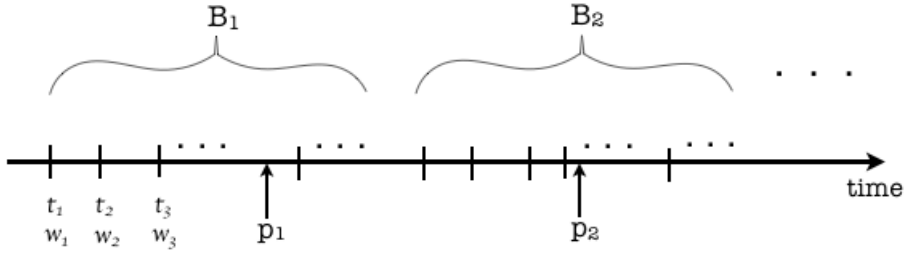


Figure 1: An example for the clustering of the input data signals

we may assume that $\sum_{i=1}^{m} w_i = 1$. Figure 1 shows an example of the input and the notations. We should add that other attributes (i.e. more information) are available in real-time locating systems like the type of the measurements (see the AoA value and RTT value in [4]), which are essential for calculating the positions, but are not needed in calculations using data clustering algorithms.

To evaluate the performance of the algorithms that we are going to devise, first we have to define an objective function which measures their efficiency. In our definition, we simultaneously take into account two objectives of the algorithm, namely (i) the waiting time for the first input data element (which is the time elapsed between the starting time of the burst and the positioning time) should not be too long, and (ii) the amount of data that could be lost (that is, $|\{x_i^k \in B^k \setminus \hat{B}^k : i = 1, 2, \dots, m\}|$ for burst B^k) should be small. In addition, the second objective is integrated over the time-dependency, meaning that a data element arriving later carries less weight in the cost function. Let $r_k = \min_i\{t_i^k : x_i^k \in B^k\}$ be the reception time of the first incoming data in burst B^k. For a given k, we will define the objective function f_k for burst B^k, which has to be minimized, as

$$f_k(t) = \lambda(t - r_k) + (1 - \lambda) \sum_{i:x_i^k \in B^k \setminus \hat{B}^k(t)} \frac{1}{1 + t_i^k} w_i, \tag{1}$$

where $\lambda \in [0, 1]$ is a constant parameter which measures the unit latency. Since the functions $f_1, f_2, \dots$ are independent (because each data element belongs to exactly one burst), by summing them up, we can get the overall cost; that is,

$$F(p_1, p_2, \dots) = \sum_k f_k(p_k), \tag{2}$$

where p_k is the positioning time of burst B^k. Since minimizing F is equivalent to minimizing all its terms, it is sufficient to consider a single term and solve the problem for it, i.e. just consider burst B. Let us assume that the reception time of the first input data element $x_1$is $t_1 = 0$ and denote the positioning time in burst B by p. What we would like to do is to minimize the function f with the form

$$f(p) = \lambda p + (1 - \lambda) \sum_{i:x_i \in B \setminus \hat{B}(p)} \frac{1}{1 + t_i} w_i. \tag{3}$$

Here, the first term is viewed as the loss of latency of the first data element to arrive (where $\lambda \in [0, 1]$ is the cost of the unit latency), while the second term is the sum of the weights of all data arriving after positioning. We shall assume that the longer we have to wait for a data, the less important it is.

3 Analysis of the model

3.1 Competitive analysis

An online algorithm for a minimization problem is said to be c-competitive, if the value of the cost function calculated by using this online algorithm is not more than c times the optimum value of the cost function (obtained by using the offline algorithm) and this holds for all possible input data streams. Formally, for an arbitrary algorithm $\mathcal{A}$ and an input sequence X the value of the cost function in the solution obtained by using $\mathcal{A}$ is $\mathcal{A}(X)$. Let $OPT(X)$ denote the optimal value of the cost function (offline optimum) for the input X. The online algorithm $\mathcal{A}$ is c-competitive if $\mathcal{A}(X) \leq c \cdot OPT(X)$.

3.1.1 Analysis without constraints

First, we will assume that there is no any restriction on the input data sequence X. We will show that there exists no constant competitive algorithm for the problem, as the following theorem states.

Theorem 1 *There is no competitive online algorithm for the online data clustering problem that uses the objective function defined by* (3)*. More precisely, for every constant* K *there exists an input sequence* X *such that the competitive ratio is larger than* K*.*

Proof. Let us consider the following input sequence. Let x_1 be the first data element of burst B, which arrives at the infrastructure node 1 at time $t_1 = 0$. If the online algorithm chooses $p > 0$ as the positioning time and if there are no more data elements, the value of the cost function expressed in terms of p is positive, while the offline optima is 0; thus the algorithm is not competitive. If the online algorithm sends the first data element of the burst for positioning immediately after it arrives (i.e. x_1 in t_1; we will call it No Waiting Time Algorithm (NWT)), then in worst case, each other element arrives at time $\delta > 0$ and hence the competitive ratio of the NWT is

$$\frac{(1-\lambda)\sum\limits_{i:x_i \in B \setminus \hat{B}(0)} \frac{1}{1+\delta} w_i}{\lambda\delta} = \frac{1}{\delta(1+\delta)} \frac{1-\lambda}{\lambda} \sum_{i:x_i \in B \setminus \hat{B}(0)} w_i \approx \frac{(1-\lambda)}{\lambda\delta^2},$$

which can be an arbitrary large constant if δ is set close to 0. □

3.2 Average case study

We have just found a negative result which states that there is no competitive algorithm, meaning that any algorithm can be easily fooled by a "malicious"' input data sequence. Although in general the input data sequences of real-time positioning systems (or of a data acknowledgment problem) are not like the worst case examples applied in the proof of Theorem 1, they can be modeled by a series of reception times that follow a certain probability distribution. Here, we will assume that the reception time t_i of $x_i \in X$ is a random variable with a given probability distribution F and we also assume that $\{t_i : x_i \in B\}$, $i = 1, \ldots, m$ are independent. Furthermore we will assume that t_i and w_i are independent of each other. Since the sum of the weights (of all data that may ideally arrive) is 1, the expected value of w_i is $1/m$ $(i = 1, 2, \ldots, m)$ by using the linearity property of the expected value.

3.2.1 Constant waiting time algorithm

Our goal is to minimize the expected cost of the online algorithm if the distribution of t_i's is given in advance. As before, let p be the positioning time in burst B. Let $K_i = w_i/(1 + t_i)$ if $t_i > p$ and $x_i \in B$, and let $K_i = 0$ otherwise. Using this notation, we get the objective function f with the form

$$f(p) = \lambda p + (1 - \lambda) \sum_{i=1}^{m} K_i. \tag{4}$$

The expected value of K_i is

$$\begin{aligned} E[K_i] &= E\Big[\frac{1}{1 + t_i} w_i\Big] \Pr(t_i > p) = \\ &= E[w_i] E\Big[\frac{1}{1 + t_i}\Big] \Pr(t_i > p) = \frac{1}{m} E\Big[\frac{1}{1 + t_i}\Big] \Pr(t_i > p), \end{aligned} \tag{5}$$

obtained by using the independency property of t_i and w_i. By using the linearity property of the expected value and (5), we get that

$$\begin{aligned} E[f(p)] &= \lambda p + (1 - \lambda) E\Big[\sum_{i=1}^{m} K_i\Big] = \\ &= \lambda p + (1 - \lambda) m E[K] = \\ &= \lambda p + (1 - \lambda) \int_{p}^{\infty} g_{\mathcal{F}}(t)\,dt \int_{0}^{\infty} \frac{1}{1 + t} g_{\mathcal{F}}(t)\,dt. \end{aligned} \tag{6}$$

Here, $g_{\mathcal{F}}$ is the probability density function of the distribution $\mathcal{F}$ and t is a random variable with distribution $\mathcal{F}$. If $\mathcal{F}$ is given, then this formula can be calculated numerically and then it can be optimized with respect to p in a minimization problem.

3.2.2 Constant waiting time algorithm for data streams from normal distribution

Usually, the reception times of a data stream of a real-time positioning system are considered to follow (or at least can be approximated by) a normal distribution, since in such systems, the probability of having large differences between the arrival times among the data elements is small in general. Now, let t be a random variable with a normal distribution ($t \sim \mathcal{N}(\mu, \sigma^2)$) having mean μ and variance σ^2. Then the expected cost of f can be calculated as

$$\begin{aligned} E[f(p), t \sim \mathcal{N}(\mu, \sigma^2)] = \\ &= \lambda p + (1-\lambda)\frac{1}{2\pi\sigma^2}\cdot \\ &\cdot \int_p^\infty \exp\Big[-\frac{(t-\mu)^2}{2\sigma^2}\Big]dt \int_0^\infty \frac{1}{1+t}\exp\Big[-\frac{(t-\mu)^2}{2\sigma^2}\Big]dt \quad = \\ &= \lambda p + c(1-\lambda)\frac{1}{2\pi\sigma}\sqrt{\frac{\pi}{2}}\mathrm{Erf}\Big[\frac{\mu-p}{\sqrt{2}\sigma}\Big], \end{aligned} \tag{7}$$

where

$$c = \int_p^\infty \frac{1}{1+t}\exp\Big[-\frac{(t-\mu)^2}{2\sigma^2}\Big]dt \in [0,1]$$

is a numerically computable constant and

$$\mathrm{Erf}[x] = \frac{2}{\sqrt{\pi}}\int_0^x \exp[-x]dx = 1 - \frac{2}{\sqrt{\pi}}\int_x^\infty \exp[-x]dx.$$

By differentiating wrt p we find that the expression (7) achieves its maximum value if the positioning time p is

$$p = \mu + \sqrt{2\log\Big[\frac{\lambda 2\pi\sigma^2}{c(1-\lambda)}\Big]}\sigma. \tag{8}$$

In the case where we know (or we can estimate) the parameters of the normal distribution (for each burst) of the arrival times, we get a constant waiting time method (CWT), where the positioning time of a burst is given by (8).

4 Experimental evaluation

We saw above that in the worst case there is no efficient algorithm for positioning. Then we showed that in the average case (considering an arbitrary probability distribution $\mathcal{F}$ of the reception times) a simple constant waiting time algorithm can achieve the best possible (minimal) expected cost, but it strongly depends on the expected value and variance of $\mathcal{F}$, which is usually not known. Below, we will define a more sophisticated algorithm that tries to learn a fairly good value of the positioning time p (which is close to the online optimum) by using the optimal values of the previous bursts. We should mention here that similar parameter learning algorithms have also been designed for the data acknowledgment problem and they are described in [16] and [18].

4.1 Variable waiting time algorithm

Now we will describe a variable waiting time algorithm for the data clustering problem. In this algorithm, each burst B^k has a starting time r_k, which is the reception time of the first data element having burst ID k. As in the description of the model, $\hat{B}^k(p) \in B^k$ is the set of those data elements in B^k that is sent to the computer for positioning. The set of data elements that have arrived before the time $\hat{t}$ in burst B^k, is denoted by $\hat{B}^k(\hat{t})$. The algorithm uses a variable t that denotes the waiting time for the data in each burst and it tries to learn the best possible value of t. Let $\mathtt{opt}(k)$ be the online optimal value of the cost function f for burst B^k calculated as follows: whenever a data in burst k arrives at time t^k, we calculate

$$f^{\text{online}}(t_i^k) = \lambda(t_i^k - r_k) + (1-\lambda)\Big[\sum_{i:x_i^k \in B^k(t^k)} \frac{1}{1+t_i^k} w_i + \frac{1}{1+t^k} \sum_{i:x_i^k \in X \setminus B^k(t^k)} w_i\Big], \tag{9}$$

for all $t_i^k \le t^k$ by considering the worst case that can happen; that is, if all other possible data elements arrive just after t^k. Then, $\mathtt{opt}(k) = \min\{f^{\text{online}}(t^k) : t_i^k \le t^k\}$, which is the online optimum calculated by using the data elements that had already arrived. Thus, $p_k = \text{argmin}\{\mathtt{opt}(k)\}$ is the optimal positioning time for the online algorithm. After, let $\hat{p}_k = p_k - r_k$. We note here that the online optimum becomes equal to the offline optima when a burst is ended. The

Algorithm 1: Variable waiting time algorithm (VWT)

Data: sequence of burst $B^1, B^2 \dots$
Result: positioning time p^k for each burst B^k $(j = 1, 2, \dots)$
Initialize $p_k = 0$ $(k = 1, 2, \dots)$;
foreach data element x with arrival time t **do**
  if $x \in B^k$ **then**
    $\mathrm{opt}(k) = \mathrm{opt}(B^k(t))$;
  end
  if $t - r_k \geq (\hat{p}_{j-1} + \dots + \hat{p}_{j-\ell})/k$ && $p_k \neq 0$ **then**
    $p_k = r_k + (\hat{p}_{k-1} + \dots + \hat{p}_{j-\ell})/\ell$;
  end
end

simple idea behind the construction of the algorithm is to use the average of the previously (and simultaneously) calculated positioning times for the actual burst that we are optimizing. Algorithm 1 shows the details of the learning process.

4.2 Empirical results

To analyze the performance of our algorithms, we generated the following input data stream. The number of bursts is 1000, the arriving times in each burst coming from a normal distribution with an expected value between 5 and 20 and a variance between 2 and 8. The input data can be found in `http://www.inf.u-szeged.hu/~london/1000burst_input.txt`. Figure 2 shows a simple example of the construction of the input. The bursts follow each other consecutively, such that the overlap between two bursts varies between 0 and 30 percent of the latter one. The average burst length is 20. In a burst, optimally the number of data elements is 40 (which is the number of infrastructure nodes), but we randomly delete each data element with probability 0.1 (in reality it may happen that data does not arrive at an infrastructure node). If a data item is deleted in a burst, the probability that it appears in the next one is 0.3. The weights of the data elements (related to the importance of the infrastructure nodes) lie between 0 and 50, assigned to each one with a uniform probability at the beginning.

We also devised three constant waiting time algorithms (CWT) to handle the data stream management task. CWT1 uses the time $r_k + 5$ for positioning (in the jth burst), which is generally less than the middle of the burst (i.e.

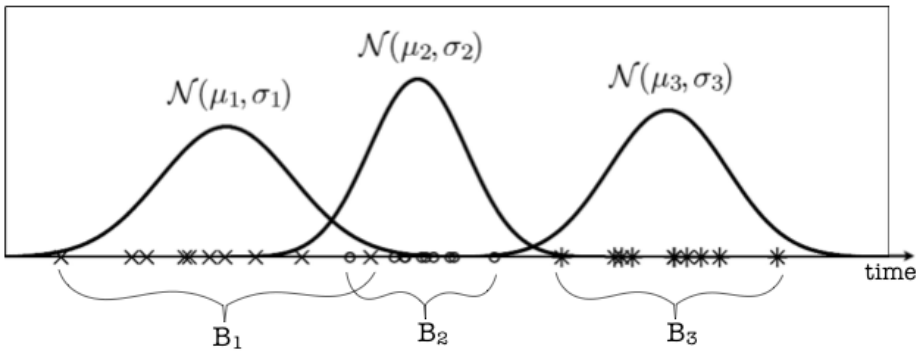


Figure 2: The structure of the input data sequence generated from a normal distribution

the starting time plus the expected value of the distribution that generates the arrival times of elements in the burst). CWT2 uses $r_k + 10$ which is close to the middle of the burst in most cases and CWT3 uses $r_k + 15$, which is generally close the the end of the burst. The learning (VWT) algorithm uses the positioning times of the last 50 bursts to calculate the waiting time for the current one. Figure 3 shows the aggregated cost of the different algorithms after a given number of cycles. As can be seen, the calculated cost using the learning algorithm approaches the offline optima after just a few cycles and remains close to it, in contrast to the constant waiting time algorithms where the aggregated costs progressively diverge farther from the offline optima.

Table 1 shows the performances values of the different algorithms, obtained by dividing the value of the total cost function of the different algorithms by the optimum value of the total cost function. The test results tell us that the learning algorithm performs well in general and remains very close to the offline optima regardless of the choice of λ (see Table 1 and Figure 4). In contrast, the efficiency of the CWT algorithms strongly depends on the choice of λ and the average length of the burst. It is not surprising that the higher the value of λ, the better the constant time algorithm will be, which chooses an earlier time for positioning, since if λ is bigger, the cost resulting from latency is also higher. This observation also holds in the reverse case. The NWT algorithm (which sends the first element that arrives in a burst for positioning), in practice, performs poorly on data streams which are like those that in real-world cases just as expected. Similar to CWT1, it only gives an acceptable result when λ is high.

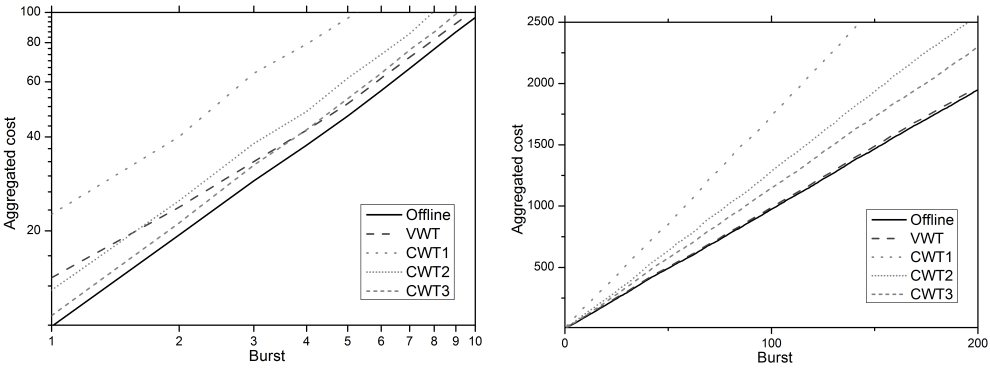


Figure 3: The aggregated cost of the different algorithms after 10 (left) and 200 (right) cycles

	$\lambda = 0.1$	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.7$	$\lambda = 0.9$
NWT	31.693	8.715	4.225	2.398	1.377
CWT1	11.482	3.346	1.787	1.232	1.333
CWT2	6.127	2.042	1.301	1.151	1.844
CWT3	3.233	1.425	1.159	1.277	2.512
VWT	1.007	1.005	1.016	1.091	1.317

Table 1: The test results

It may be concluded that the VWT algorithm is useful in general and the output values of it are close to the optimal values, even when the burst length and the parameters of the distribution of the arriving times are unknown. However, the efficiency of a constant time algorithm strongly depends on the λ and time parameters of the objective function, the average burst length and also on the distribution of the arrival times.

5 Summary

In this paper we defined an online optimization model for the data stream management problem, which arises in real-time locating systems and in a similar form in the data acknowledgment problems. We constructed different algorithms to solve the problem and analyzed them with the tool of competi-

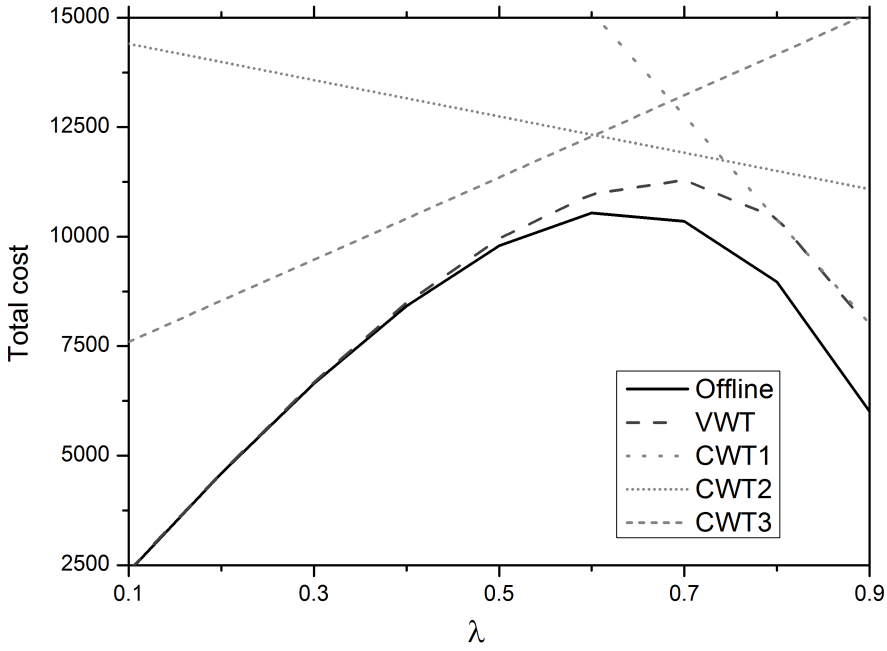


Figure 4: Total cost of the different algorithms after 1000 cycles for different λ values

tive analysis and with expected value analysis. Although we showed, by using the worst case study, that there is no competitive algorithm for this model, the average case study confirmed the validity and applicability of our model and algorithms for more realistic data streams. A more sophisticated variable waiting time algorithm that uses the average of the optimal positional times of some of the previous bursts was also created for position calculations of the subsequent bursts. We showed empirically that this learning method is useful and close to the global optimum, even when we have no a priori information about the input data stream. Some outstanding questions remain that deserve further examination. For instance if we use a more general objective function like $f(p) = \lambda g(p) + (1-\lambda)\sum_{i:t_i>p} h(t_i)w_i$ with any g and h, such that g is non-decreasing and h is non-increasing, will we get better results? The analytical study that we have already done applies for the general case, but it would be interesting to see more applications (besides the data stream management

and data acknowledgment problems) where we can apply this model with different suitable g and h functions. It is also an open question (mentioned by the authors of [17]) of how should go about solving the problem when we do not receive burst ID information. Lastly, it would be also interesting to use and analyze this objective function in the data acknowledgment problem by considering various probability distributions of the arrival times and find out whether it will lead to good performance values.

Acknowledgement

This work was supported by the European Union and the European Social Fund through project Telemedicina (Grant no.: TÁMOP-4.2.2.A-11/1/KONV-2012-0073).

András London was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TÁMOP-4.2.4.A/2-11-1-2012-0001 'National Excellence Program'.

The authors would like to thank Csanád Imreh for providing helpful comments.

References

[1] S. Albers, H. Bals, Dynamic TCP acknowledgment: Penalizing long delays, *SIAM J. Discrete Math.* **19,** 4 (2005) 938–951. ⇒165

[2] A. Borodin, R. El-Yaniv, *Online Computation and Competitive Analysis*, Cambridge University Press, 1998. ⇒164

[3] M. Brugger, T. Christ, F. Kemeth, S. Nagy, M. Schaefer, M. M.Pietrzyk, The FMCW technology-based indoor localization system, in: *Ubiquitous Positioning Indoor Navigation and Location Based Services*, 2010, pp. 1–6. ⇒164

[4] M. Brugger, F. Kemeth, Locating rate adaptation by evaluating movement specific parameters, in: *Adaptive Hardware and Systems*, 2010, pp. 127–131. ⇒164, 167

[5] T. M. Chan, H. Zarrabi-Zadeh, A randomized algorithm for online unit clustering, in: *Approximation and Online Algorithms* 2007, pp. 121–131. ⇒164

[6] E. G. Coffman, G. S. Lueker, *Probabilistic Analysis of Packing and Partitioning Algorithms*, Wiley, New York, 1991. ⇒166

[7] J. Csirik, L. Epstein, C. Imreh, A. Levin, Online clustering with variable sized clusters *Algoritmica* **65,** 2 (2013) 251–274. ⇒164

[8] G. Divéki, Online clustering on the line with square cost variable sized clusters *Acta Cybernetica* **21,** 1 (2013) 75–88. ⇒164

[9] G. Divéki, C. Imreh, Online facility location with facility movements, *CEJOR Cent. Eur. J. Oper. Res.* **19,** 2 (2011) 191–200. ⇒164

[10] D. R. Dooley, S. A. Goldman, S. D. Scott, On-line analysis of the TCP acknowledgment delay problem, *Journal of the ACM* **48,** 2 (2001) 243–273. ⇒ 165

[11] L. Epstein, R. Van Stee, On the online unit clustering problem, in: *Approximation and Online Algorithms*, 2008, pp. 193–206. ⇒164

[12] A. Fiat, G. Woeginger, *Online Algorithms: The State of the Art*, Springer, Heidelberg, 1998. ⇒164

[13] M. Hofri, *Probabilistic Analysis of Algorithms: On Computing Methodologies for Computer Algorithms Performance Evaluation*, Springer-Verlag New York, 1987. ⇒166

[14] C. Imreh, Competitive analysis, in: *Algorithms of Informatics, Vol. 1. Foundations* (ed. A. Iványi), mondAt Kiadó, Budapest, 2007, pp. 395–428. ⇒ 164, 165

[15] C. Imreh, T. Németh, On time lookahead algorithms for the online data acknowledgement problem, in: *Mathematical Foundations of Computer Science*, 2007, pp. 288–297. ⇒165

[16] C. Imreh, T. Németh, Parameter learning algorithm for the online data acknowledgment problem, *Optimization Methods and Software* **26,** 3 (2011) 397–404. ⇒ 165, 171

[17] T. Németh, S. Nagy, C. Imreh, Online data clustering algorithms in an RTLS system, *Acta Univ. Sapientiae Informatica*, **5,** 1 (2013) 5–15. ⇒164, 176

[18] T. Németh, B. Gyekiczki, C. Imreh, Parameter learning in lookahead online algorithms for data acknowledgment, *Proc. 3th IEEE Symposium on Logistics and Industrial Informatics*, 2011, pp. 195–198. ⇒165, 171

Received: September 20, 2014 • Revised: October 26, 2014

Acta Univ. Sapientiae, Informatica, **6,** 2 (2014) 178–189

Induced label graphoidal graphs

Ismail SAHUL HAMID
Department of Mathematics, The Madura College, Madurai-11
email: sahulmat@yahoo.co.in

Mayamma JOSEPH
Department of Mathematics, Christ University, Bangalore-29, India
email: mayamma.joseph@christuniversity.in

Abstract. Let G be a non-trivial, simple, finite, connected and undirected graph of order n and size m. An induced acyclic graphoidal decomposition (*IAGD*) of G is a collection ψ of non-trivial and internally disjoint induced paths in G such that each edge of G lies in exactly one path of ψ. For a labeling $f : V \to \{1, 2, 3, \ldots, n\}$, let $\uparrow G_f$ be the directed graph obtained by orienting the edges uv of G from u to v, provided $f(u) < f(v)$. If the set ψ_f of all maximal directed induced paths in $\uparrow G_f$ with directions ignored is an induced path decomposition of G, then f is called an induced graphoidal labeling of G and G is called an induced label graphoidal graph. The number $\eta_{il} = \min\{|\psi_f| : f \text{ is an induced graphoidal labeling of } G\}$ is called the induced label graphoidal decomposition number of G. In this paper we introduce and study the concept of induced graphoidal labeling as an extension of graphoidal labeling.

1 Introduction

By a graph $G = (V, E)$, we mean a non-trivial, simple, finite, connected and undirected graph. The order and size of G are denoted by n and m respectively. For terms not defined here we refer to [7].

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C38
Key words and phrases: induced acyclic graphoidal decomposition, induced label graphoidal graphs, induced label graphoidal decomposition number

A *decomposition* of a graph G is a collection ψ of its subgraphs such that every edge of G lies in exactly one member of ψ. The significance of graph decomposition problems arise from the rigor of their theoretical treatise as well as their application to a variety of fields such as coding theory, bio-informatics and various types of networks. Among the decomposition problems explored by researchers we have path decomposition introduced by Harary [8], where each element of ψ is a path and various types of path decompositions like unrestricted path cover [9], graphoidal cover [1], simple graphoidal cover [4] and so on. A detailed review of the results pertaining to graphoidal decomposition along with an array of open problem is given in [3].

Definition 1 [1] *A* graphoidal decomposition (GD) *of a graph* G *is a collection* ψ *of non-trivial paths and cycles of* G *such that*

(i) Every vertex of G *is an internal vertex of at most one member of* ψ.

(ii) Every edge of G *is in exactly one member of* ψ.

Note that by an internal vertex of a path P we mean the vertices of P other than its end vertices. A *GD* wherein no member is a cycle is called an acyclic graphoidal decomposition (*AGD*) which was introduced by Arumugam and Suresh Susheela [6]. By demanding the members of an *AGD* to be induced paths, Arumugam [2] coined another variation namely *induced acyclic graphoidal decomposition (IAGD)*. The minimum cardinality of the respective graphoidal decompositions are denoted by η, η_a and η_{ia}. The study of the parameter η_{ia} initiated by Ratan Singh and Das [10] was further extended by I. Sahul Hamid and M. Joseph [13].

Linking graphoidal decompositions and vertex labeling, Acharya and Sampathkumar [1] conceptualized the idea of graphoidal labeling as follows.

Definition 2 [1] *Let* $G = (V, E)$ *be a graph with* n *vertices and let* $f : V \to \{1, 2, \ldots, n\}$ *be a labeling of the vertices of* G. *Orient the edges* uv *from* u *to* v *provided* $f(u) < f(v)$. *Such an orientation is called a* low-to-high *orientation of* G *with respect to the given labeling* f. *By* $\uparrow G_f$ *we mean, G together with the labeling* f *with respect to which the edges of* G *are oriented from low -to- high. Let* $\pi^*(\uparrow G_f)$ *be the set of all maximal directed paths in* $\uparrow G_f$ *and* $\pi(\uparrow G_f)$ *be the set of all members of* $\pi^*(\uparrow G_f)$ *with directions ignored. We say that* f *is a* graphoidal labeling *of* G *if* $\pi(\uparrow G_f)$ *is a graphoidal decomposition of* G *and if* G *admits such a labeling* f *of its vertices, then* G *is called a* label graphoidal graph (LGG).

If G is a label graphoidal graph with a graphoidal labeling f, then the graphoidal decomposition $\pi(\uparrow G_f)$ is called the *label graphoidal decomposition of* G *with respect to the labeling* f and is denoted by ψ_f.

The following result due to Acharya and Sampathkumar [1] gives a complete characterization of label graphoidal graphs.

Recall that a vertex of a directed graph with in-degree zero is called a *source* and a vertex of out-degree zero is called a *sink*.

Theorem 3 [1] *Suppose* G *has a graphoidal labeling* f. *Then any vertex* v *of* G *with degree* > 2 *is either a sink or a source in* $\uparrow G_f$.

Theorem 4 [1] *A graph* G *is label graphoidal if and only if every odd cycle in* G *has a vertex of degree 2.*

Motivated by the observation that label graphoidal decompositions of a given graph may vary according to the nature of the graphoidal labeling, Arumugam and Sahul Hamid [5] defined the concept of label graphoidal decomposition number and obtained some fundamental results.

Definition 5 [5] *Let* G *be a label graphoidal graph. The* label graphoidal decomposition number $\eta_l(G)$ *is defined to be* $\eta_l(G) = \min\{|\psi_f| : f \text{ is a graphoidal labeling of } G\}$.

The following result obtained by Arumugam and Sahul Hamid [5] gives the value of η_l for a tree T in terms of its size and the number of vertices of degree 2.

Theorem 6 [5] *Let* T *be a tree with* b *vertices of degree 2. Then* $\eta_l(T) = m - b$.

2 Induced label graphoidal graphs

In this section we introduce the concept of induced label graphoidal graph and investigate its properties.

Definition 7 *Let* $G = (V, E)$ *be a graph of order* n *and size* m *and* $f : V \to \{1, 2, \ldots, n\}$ *be a labeling of the vertices of* G. *Let* $\uparrow G_f$ *be the* oriented graph *obtained by orienting the edges* uv *from* u *to* v, *provided* $f(u) < f(v)$ *(this orientation given to the edges of* G *is called a* low-high *orientation). If the set* ψ_f *of all maximal directed induced paths in* $\uparrow G_f$ *with directions ignored is an induced path decomposition of* G, *then* f *is said to be an induced graphoidal labeling of* G *and* G *is called an induced label graphoidal graph (ILGG).*

Example 8 *Consider the cycle* $C_n = (v_1, v_2, \ldots, v_n, v_1)$ *where* $n \geq 4$. *Define* $f : V \to \{1, 2, \ldots, n\}$ *by*

$$f(v_1) = 2,$$
$$f(v_2) = 1,$$
$$f(v_i) = i \text{ when } 3 \leq i \leq n.$$

Then $\psi_f = \{(v_2, v_1, v_n), (v_2, v_3, \ldots, v_n)\}$ *is an induced decomposition of* G, *proving that* f *is an induced graphoidal labeling and thus* G *is an induced label graphoidal graph. Certainly, triangles are not induced label graphoidal graphs and thus the cycle* C_n *is an induced label graphoidal graph only when* $n \geq 4$.

Lemma 9 *If* f *is an induced graphoidal labeling of a graph* G, *then any vertex of* G *with degree greater than two is either a sink or a source in* $\uparrow G_f$.

Proof. Suppose f is an induced graphoidal labeling of G. Assume if possible, that there exists a vertex u in G with $\deg u > 2$ such that both in-degree and out-degree of u in $\uparrow G_f$ are positive. Consider three neighbors , say x, y and z of the vertex u. Then the orientation of the edges xu, yu and zu will be one of the following.

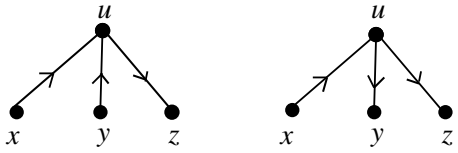


Figure 1

Concerning the first case, ψ_f will have a path P in which (x, u, z) is a section and similarly, there exists a path Q in ψ_f such that (y, u, z) is a section. Hence ψ_f is no longer a decomposition as the edge yu lies both in P and in Q. We will arrive at a similar contradiction in the later case as well. Hence every vertex of G with degree greater than two is either a sink or a source in $\uparrow G_f$. □

Theorem 10 *If* f *is an induced graphoidal labeling, then* ψ_f *is an induced acyclic graphoidal decomposition.*

Proof. Suppose f is an induced graphoidal labeling of G. By definition ψ_f is an induced path decomposition of G and moreover no member of ψ_f is a cycle, which implies that ψ_f is an induced acyclic path decomposition of G. Further

it follows from Lemma 9 that every vertex v of G having degree at least three is either a source or a sink in $\uparrow G_f$ so that the vertex v is exterior to ψ_f. Hence every vertex v in G is an internal vertex of at most one path in ψ_f. Thus ψ_f is an induced acyclic path decomposition of G such that every vertex of G is an internal vertex of at most one path in ψ_f and so ψ_f is an *IAGD* of G. □

Remark 11 *Since ψ_f is an induced acyclic graphoidal decomposition of a graph G when f is an induced graphoidal labeling, ψ_f in particular is a graphoidal decomposition of G. That is,* induced label graphoidal graphs are label graphoidal graphs*. But a label graphoidal graph need not be induced (for example consider the triangle). At the same time there are label graphoidal graphs which are induced label graphoidal as well while a labeling which served as a graphoidal labeling need not necessarily be an induced graphoidal labeling of the graph.*

For example, the labeling of cycle C_4 as in Figure 2 is a graphoidal labeling, but not an induced graphoidal labeling.

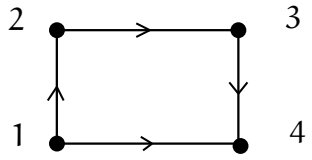


Figure 2

Theorem 12 *Induced label graphoidal graphs are triangle-free.*

Proof. If G is a graph of order n having a triangle $C = (u, v, w, u)$, then under any labeling $f : V(G) \to \{1, 2, \ldots, n\}$ of G, the orientation of the edges of C will be one of the forms given in Figure 3.

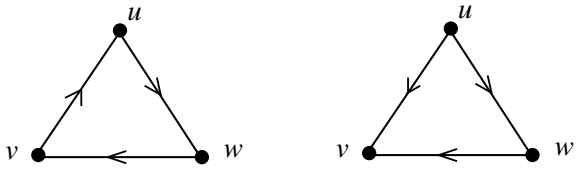


Figure 3

In the first case the path (u, w, v) is a section of a path in ψ_f and consequently ψ_f cannot be an induced decomposition. Similarly in the second case

also ψ_f contains a path having (u, w, v) as its section. Hence ψ_f will no longer be an induced graphoidal decomposition of G so that G is not induced label graphoidal. □

Corollary 13 *Complete graphs and wheels are not induced label graphoidal.*

Theorem 14 *Bipartite graphs are induced label graphoidal.*

Proof. Let G be a bipartite graph with the bipartition (X, Y) where $X = \{x_1, x_2, \ldots, x_r\}$ and $Y = \{y_1, y_2, \ldots, y_s\}$. Define $f : V(G) \to \{1, 2, \ldots, (r+s)\}$ by $f(x_i) = i$; for all $i = 1, 2, \ldots, r$ and $f(y_j) = (j + r)$; for all $j = 1, 2, \ldots, s$. Then every vertex of X is a source and that of Y is a sink in $\uparrow G_f$ so that $\psi_f = E(G)$; which of course is an induced graphoidal decomposition and thus f is an induced graphoidal labeling of G. □

Corollary 15 *Every label graphoidal graph with $\delta \geq 3$ is an induced label graphoidal graph.*

Proof. Let G be a label graphoidal graph with $\delta \geq 3$. Then Theorem 4 implies that G does not contain any odd cycle. Hence G is bipartite and so the result follows from Theorem 14. □

Theorem 16 *Induced subgraph of an induced label graphoidal graph is induced label graphoidal.*

Proof. Let G be an induced label graphoidal graph and let H be an induced subgraph of G. Let f be an induced graphoidal labeling of G. Let g be the labeling of H obtained from f by assigning the label 1 to the vertex of H which receives the minimum among the labels of the vertices of H under f and using the label 2 to the vertex receiving the next minimum on the label under f and so on. We claim that g is an induced graphoidal labeling of H. Since H is an induced subgraph of G, any member of ψ_g is either a member of ψ_f or a section of a member of ψ_f and so each member in ψ_g is an induced path in H. Hence ψ_g is an induced graphoidal decomposition of H and consequently g is an induced graphoidal labeling of H. □

Remark 17 *It follows from Theorem 16 that a graph which is not induced label graphoidal cannot be an induced subgraph of an induced label graphoidal graph. Since triangles are not induced label graphoidal graphs, any induced label graphoidal graph is triangle free which in fact is Theorem 12.*

The following theorem completely characterizes the induced label graphoidal graphs.

Theorem 18 *A graph* G *is induced label graphoidal if and only if* G *is triangle-free such that every odd cycle of* G *contains a vertex of degree 2.*

Proof. Suppose G is an induced label graphoidal graph. Remark 11 says then that G is a label graphoidal graph and so the required conditions follow from Theorem 12 and Theorem 4. Therefore, we need to verify only the converse.

Let G be a triangle-free graph such that every odd cycle in G contains a vertex of degree two. Suppose G does not contain any odd cycle. Then G is a bipartite graph and hence by Theorem 14, G is induced label graphoidal.

Now assume that G contains an odd cycle. Consider the collection $B = \{b_1, b_2, \ldots, b_k\}$ of vertices of degree two in G with minimum cardinality whose removal results in a graph with no odd cycles. Let $H = < V(G) \setminus B >$. Then H is a connected graph without any odd cycles as the vertices of degree two lying on a cycle of G cannot be cut-vertices of G. Hence H is a bipartite graph. Let (X, Y) be the bipartition of H with $X = \{x_1, x_2, \ldots, x_{n_1}\}$ and $Y = \{y_1, y_2, \ldots, y_{n_2}\}$. Now, define a labeling $f : V(G) \to \{1, 2, \ldots, n\}$ where $n = n_1 + n_2$ as follows.

$$f(x_i) = i, \text{ for all } i = 1, 2, \ldots, n_1$$
$$f(b_i) = n_1 + i \text{ for all } i = 1, 2, \ldots, k \text{ and}$$
$$f(y_i) = n_1 + k + i \text{ for all } i = 1, 2, \ldots, n_2.$$

Then one can observe that in $\uparrow G_f$ the vertices in X are sources and the vertices in Y are sinks. Also, every vertex in B is adjacent from exactly one vertex of X and adjacent to exactly one vertex of Y. Hence ψ_f consists of the edges joining a vertex of X and that of Y along with the paths of length two connecting a vertex in X and a vertex of Y having a vertex of B as an internal vertex. Now, as G is triangle-free, the initial and terminal vertices of a path of length two in ψ_f are not adjacent which implies that such a path in ψ_f is an induced path in G. Thus ψ_f is an *IAGD* of G proving that f is an induced graphoidal labeling of G. □

Corollary 19 *A label graphoidal graph* G *is induced label graphoidal if and only if* G *is triangle-free.*

Proof. Follows from Theorem 4 and Theorem 18. □

3 Induced label graphoidal decomposition number

For a graph G admitting a graphoidal labeling f, the labeling function f need not necessarily be unique; which in turn implies that G possesses more than one graphoidal decomposition. Motivated by this observation, Arumugam and Sahul Hamid [5] introduced the concept of label graphoidal decomposition number. We extend this notion further with respect to induced graphoidal decompositions.

Definition 20 *Let G be an induced label graphoidal graph. Then the* induced label graphoidal decomposition number $\eta_{il}(G)$ *is defined to be*

$$\eta_{il}(G) = \min\{|\psi_f| : f \text{ is an induced graphoidal labeling of } G\}.$$

Example 21 *(i) A graph G is given in Figure 4 with two different labelings. Observe that the two different labelings of the graph G yield two different induced acyclic graphoidal decompositions. For the first labeling f_1 we have $\psi_{f_1} = E(G)$ whereas in the second case the decomposition corresponding to the given labeling f_2 is given by $\psi_{f_2} = \{(1,2),(1,4,6),(3,6),(1,5,6)\}$. However, it can easily be verified that $\eta_{il}(G) = 4$.*

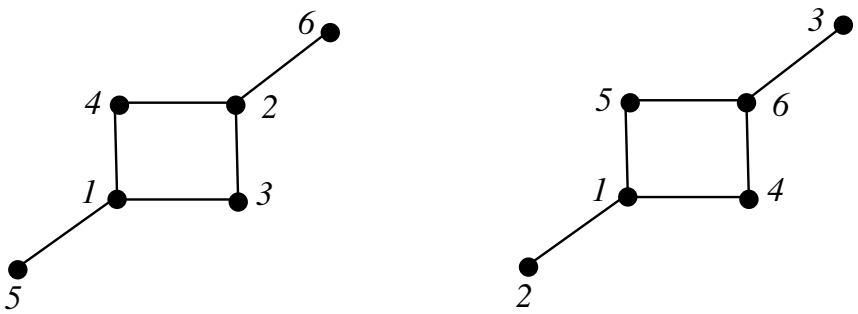


Figure 4

(ii) It follows immediately from the definition that $\eta_{il}(C_n) = 2$ for all $n \geq 4$ and $\eta_{il}(P_n) = 1$ for all $n \geq 2$.

(iii) If G is a graph with $\delta(G) \geq 3$, by Lemma 2.3 every vertex of G is either a sink or a source in $\uparrow G_f$ under any induced graphoidal labeling f of G and consequently $\eta_{il}(G) = m$.

(iv) Consider the complete bipartite graph $K_{r,s}$ *where* $2 \leq r \leq s$*, and let* (X, Y) *be the bipartition where* $X = \{x_1, x_2, \ldots, x_r\}$ *and* $Y = \{y_1, y_2, \ldots, y_s\}$*. If* $r \geq 3$*, then* $\delta(K_{r,s}) = r \geq 3$ *and so* $\eta_{il}(K_{r,s}) = rs$ *as observed above. When* $r = 2$*, define a labeling* $f : V(K_{r,s}) \to \{1, 2, \ldots, (r+s)\}$ *by*

$$f(x_1) = 1$$
$$f(x_2) = s + 2 \text{ and}$$
$$2 \leq f(y_j) \leq (s+1), \text{for each } j \text{ where } 1 \leq j \leq s.$$

Certainly, f *is an induced graphoidal labeling of* $K_{r,s}$ *with* $\psi_f = \{(x_1, y_i, x_2) : 1 \leq i \leq s\}$ *and hence* $\eta_{il}(K_{r,s}) \leq |\psi_f| = s$*. As for any induced graphoidal labeling* f *of* $K_{2,s}$*, the induced graphoidal decomposition* ψ_f *will consist of just edges and paths of length two, which implies that* $\eta_{il}(K_{2,s}) \geq s$*. Thus we have*

$$\eta_{il}(K_{r,s}) = \begin{cases} s & \text{if } r = 2 \\ rs & \text{else} \end{cases}$$

(v) As any path in a tree T *is an induced path, from Theorem 6 we have* $\eta_{il}(T) = \eta_l(T) = m - b$*, where* b *denotes the number of vertices of degree 2 in* T*.*

Remark 22 *It is possible to have two different labelings giving rise to the same minimum induced label graphoidal decomposition. For example, consider the graph* G *given in Figure 5(a) and the labeling* f_1 *and* f_2 *of* G *as given in Figures 5(b) and 5(c) respectively. Certainly* $|\psi_{f_1}| = |\psi_{f_2}| = 7 = \eta_{il}(G)$*.*

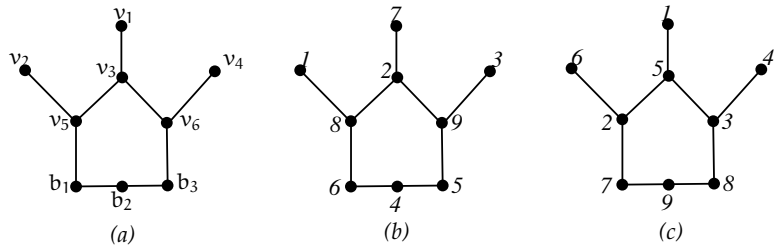


Figure 5

Let us now proceed to obtain some bounds for η_{il}. As seen earlier, by *internal vertices* of a path P, we mean the vertices of P other than its end vertices. For a graphoidal decomposition ψ of G, a vertex v is said to be *interior* to ψ if v is an internal vertex of an element of ψ and is called *exterior to* ψ otherwise.

Suppose G is an induced label graphoidal graph and let b denote number of vertices of degree 2. If f is an induced graphoidal labeling of G, we use b'_f to denote the number of vertices of degree 2 which are exterior to ψ_f. Let $b' = \min_f\{b'_f\}$ where the minimum is taken over all the induced graphoidal labeling f of G.

The following theorem which is useful to estimate the value of η_{il} for a given graph is analogous to a result for η_l given in [5].

Theorem 23 *Let G be an induced label graphoidal graph. Then $\eta_{il}(G) = m - b + b'$.*

Proof. Consider a labeling f of G with respect to which G has an induced graphoidal decomposition. Let ψ_f be the corresponding induced graphoidal decomposition. Then obviously, the interior vertices of all the elements of ψ_f are of degree 2. Therefore we have

$$\begin{aligned} m &= \sum_{P \in \psi_f} |E(P)| \\ &= \sum_{P \in \psi_f} (1 + \text{Number of vertices of degree 2 in } \psi_f) \\ &= |\psi_f| + b - b' \end{aligned}$$

so that we get $\eta_{il}(G) = m - b + b'$. □

Corollary 24 *For any induced graphoidal graph G, $\eta_{il}(G) \geq m - b$. Further, equality is obtained if and only if there exists an induced graphoidal labeling f of G such that every vertex of G with degree 2 is interior to ψ_f.*

Proof. The inequality follows from Theorem 23 as $b' \geq 0$. The rest follows from the fact that $\eta_{il} = m - b$ if and only if $b' = 0$. □

Although A graph g is 23 gives a formula to determine the value of η_{il} in terms of m and b', it is to be noted that determination of b' for a graph in general is not easy.

The following theorem gives a necessary condition for a graph G with $\eta_{il}(G) = m - b$.

Theorem 25 *Let G be an induced label graphoidal graph which is not a cycle. Then $\eta_{il}(G) = m - b$ implies that G is graph such that every cycle of G contains an even number of vertices of degree ≥ 3 of which at least one pair of vertices are non-adjacent.*

Proof. Suppose G is an induced label graphoidal graph with $\eta_{il} = m - b$. Then by virtue of Corollary 24, there exists an induced graphoidal labeling f such that every vertex of degree 2 is interior to ψ_f and of course $|\psi_f| = m - b$.

If G does not contain any cycle, then G is a tree and the result is trivial. Suppose G is not a tree and let C be a cycle in G. As any vertex of degree ≥ 3 is either a sink or a source in ψ_f and all the vertices of degree 2 are interior to it, it follows that the vertices on C with degree ≥ 3 must alternatively be sinks and sources. Hence C contains an even number of vertices of degree ≥ 3. Now, what we need to verify is that at least one pair of vertices on C with degree ≥ 3 are non-adjacent. If not, every pair of vertices on C with degree ≥ 3 are adjacent. As G is induced label graphoidal, it contains no triangles. Therefore, C contains exactly two vertices of degree ≥ 3, say u and v. Obviously, one of them, say u will be a source and the other is a sink. Then both the $u - v$ sections of C belong to ψ_f and at least one of them will not be induced which is a contradiction and hence the result. □

Lemma 26 *If* G *is an induced label graphoidal graph, there exists an induced graphoidal labeling* f *such that every vertex of degree 2 not lying on any cycle of* G *is interior to* ψ_f.

Proof. Let G be an induced label graphoidal graph. Let v_b be a vertex of degree 2 not lying on any cycle in G. If G is a tree, then the result follows from Remark 3.9 and Corollary 3.6. Suppose G is not a tree. Let w_1 and w_2 be vertices with degree ≥ 3 and nearest to v_b. Since the distance between w_1 and w_2 is > 1, it is possible to obtain a labeling f for which w_1 is a source and w_2 is a sink so that v_b is internal to ψ_f. □

Acknowledgements

This research work is supported by the Department of Science and Technology, New Delhi, India through a research project with No.SR/FTP/MS-002/2012 sanctioned to the first author.

The authors would like to thank the referee for the helpful comments and suggestions.

References

[1] B. D. Acharya, E. Sampathkumar, Graphoidal covers and graphoidal covering number of a graph, *Indian J. Pure Appl. Math.*, **18**, 10 (1987) 882–890. ⇒179, 180

[2] S. Arumugam, Path covers in graphs, *Lecture Notes of the National Workshop on Decompositions of Graphs and Product Graphs* held at Annamalai University, Tamil Nadu, January 3–7, 2006. ⇒179

[3] S. Arumugam, B. D. Acharya, E. Sampathkumar, Graphoidal Covers of a Graph: A Creative Review, *Proc. National Workshop on Graph Theory and its Applications*, Manonmaniam Sundaranar University, Tirunalveli, India, 1996, pp. 1–28. ⇒179

[4] S. Arumugam, I. Sahul Hamid, Simple graphoidal covers in graphs, *Journal of Combin. Math. Combin. Comput.*, **64** (2008) 79–95. ⇒179

[5] S. Arumugam, I. Sahul Hamid, Label graphoidal covering number of a graph, *Proc. Group Discussion on Labeling of Discrete Structures and Applications*, Mary Matha Arts College, Mananthawady, Eds. B. D. Acharya, S. Arumugam and A.Rosa, Narosa Publishing House, 2007, pp. 77–82. ⇒180, 185, 187

[6] S. Arumugam, J. Suresh Suseela, Acyclic graphoidal covers and path partitions in a graph, *Discrete Math.*, **190,** 1–3 (1998), 67–77. ⇒179

[7] G. Chartrand, L. Lesniak, *Graphs and Digraphs*, Fourth Edition, CRC Press, Boca Raton, 2004. ⇒178

[8] F. Harary, Covering and packing in graphs I, *Ann. N. Y. Acad. Sci.*, **175** (1970) 198–205. ⇒179

[9] F. Harary, A. J. Schwenk, Evolution of the path number of a graph, covering and packing in graphs II, in: *Graph Theory and Computing*, Ed. R. C. Rad, Academic Press, New York, 1972, pp. 39–45. ⇒179

[10] K. Ratan Singh, P. K. Das, *Induced acyclic graphoidal covers in a graph, World Academy of Science, Engineering and Technology*, **68** (2010) 38–44. ⇒179

[11] I. Samul Hamid, A. Anitha, On the label graphoidal covering number-I, *Trans. on Combinatorics* **1**, 4 (2012) 25–33. ⇒

[12] I. Samul Hamid, A. Anitha, On the Label graphoidal covering number-II, *Discrete Math. Algorithms Appl.* **3**, 1 (2011) 1–7. ⇒

[13] I. Samul Hamid, M. Joseph, Further results on induced acyclic graphoidal decomposition, *Discrete Math. Algorithms Appl.* **5**, 1 (2013) 1350006 (11 pages). ⇒179

Received: August 12, 2014 • Revised: September 8, 2014

Acta Univ. Sapientiae, Informatica, **6,** 2 (2014) 190–205

Remarks on the A^{**} algorithm

Tibor GREGORICS
Eötvös Loránd University, Faculty of Informatics
Budapest
email: gt@inf.elte.hu

Abstract. The A^{**} algorithm is a famous heuristic path-finding algorithm. In this paper its different definitions will be analyzed firstly. Then its memory complexity is going to be investigated. On the one hand, the well-known concept of better-information will be extended to compare the different heuristics in the A^{**} algorithm. On the other hand, a new proof will be given to show that there is no deterministic graph-search algorithm having better memory complexity than A^{**}. At last the time complexity of A^{**} will be discussed.

1 Introduction

The A^{**} algorithm is one of the graph-search algorithms that can be used to solve path-finding problems.

Path-finding problems are the tasks that can be modeled by a directed arc-weighted graph (this is the so-called representation graph). Let $R = (N, A, c)$ denote a directed arc-weighted graph where N is the set of nodes, $A \subseteq N \times N$ is the set of directed arcs and $c : A \mapsto \mathbb{R}$ is the weight function. The graphs of our interest have got only finite outgoing arcs from their nodes and there is a global positive lower limit ($\delta \in \mathbb{R}$) on the weights. These graphs are the δ-graphs.[5] Thus a path-finding problem can be represented by the triple (R, s, T) where R is a δ-graph, s denotes the start node and T denotes the set of goal nodes. The solution of this problem is a path from the start node to some goal node that can be denoted by $s \to t$ where $t \in T$.

Computing Classification System 1998: I.2.8
Mathematics Subject Classification 2010: 68T20
Key words and phrases: path-finding problem, graph-search algorithms, heuristic function, A^* algorithm, A^{**} algorithm

The cost of a path can be calculated as the summation of the cost of the arcs on this path. We will denote the smallest cost path from n to m as $n \rightarrow^* m$. This path is called as optimal path. In many path-finding problems the optimal path from the start node to some goal node is needed to be found. The value $h^*(u)$ shows the optimal cost from the node u to any goal node. The function $h^* : N \mapsto \mathbb{R}$ is called as optimal remaining cost function. The value $g^*(u)$ gives the optimal cost from the start node to the node u and the function $g^* : N \mapsto \mathbb{R}$ is named as optimal leading up cost function. We can calculate the optimal cost of the path going from the node s to any goal node via the node u in the way $f^*(u) = g^*(u) + h^*(u)$ where $f^* : N \mapsto \mathbb{R}$ is the optimal cost function. We remark that the value $f^*(s)$ denotes the cost of the optimal solution.

Graph-search algorithms try to find a path from the start node to a goal node and during their process they always record the sub-graph of the representation graph that has been discovered. This is the search graph (G). The nodes of G whose children are known are the so-called closed nodes. The other nodes of G that wait for their expansion are the open nodes. Sometimes a node may be open and closed at the same time if it has been expanded but its children (more precisely the optimal paths form the start node to its children) are not completely known. Let OPEN denote the set of the open nodes at any time. In every step the most appropriate open node will be selected and expanded, i.e. its children must be generated or regenerated. The general graph-search algorithm [5, 2, 3] evaluates the open nodes with an evaluation function $f : \text{OPEN} \mapsto \mathbb{R}$ and chooses the open node with the lowest f value for expansion.

```
procedure Graph-search
    G := (\{s\}, \emptyset) : OPEN := \{s\} : \pi(s) := nil : g(s) := 0
    while OPEN \neq \emptyset loop
        n := min_f(OPEN)
        if goal(n) then return there is a solution endif
        foreach m \in Children(n) loop
            if m \notin G or g(m) > g(n) + c(n, m) then
                \pi(m) := n : g(m) := g(n) + c(n, m)
                OPEN := OPEN \cup m
            endif
        endforeach
        G := G \cup \{(n, m) | m \in Children(n)\}
        OPEN := OPEN \cup \{(n, m) | m \in Children(n)\}
    endwhile
    return there is no solution
end
```

The cheapest paths from the start node to the nodes of G, which are found so far, must be recorded with their cost. These costs are shown by the function $g \in N \mapsto \mathbb{R}$. (It is clear that $g(u) \geq g^*(u)$ for all node u of N.) The algorithm also maintains a pointer $\pi \in N \mapsto N$ that marks the best parent of each discovered node (except the start node). The best parent of any node is the one which is along the cheapest discovered path driving from s to that node. The recorded paths form a directed spanning tree in the search graph where the root node is the start node. We will denote a recorded path driving from s to u as $s \rightarrow^\pi u$ and $c^\pi(s,u)$ will symbolize its cost.

The computation of the evaluation function of a graph-search algorithm can contain some extra knowledge about the problem. This is the so-called heuristic function $h : N \mapsto \mathbb{R}$ that estimates the remaining optimal path cost from a node to any goal node, i.e. $h(u) \approx h^*(u)$ for all nodes u of N.

The most famous heuristic graph-search algorithm is *the* A^* *algorithm*. The evaluation function of this algorithm is $f = g + h$ where the cost function g is calculated by the algorithm and the heuristic function h is derived from the problem. The A^* algorithm uses a nonnegative and admissible heuristic function. The admissibility means that the heuristic function gives a lower limit on the remaining optimal path cost from any node to a goal node, i.e. $h(u) \leq h^*(u)$ for all node u of N.

2 Definitions of the A^{**} algorithm

The A^{**} algorithm can be treated as a modification of the A^* algorithm. During the execution of A^*, it can occur that the evaluation function value of an open node n ($g(n) + h(n)$) might be smaller than the value $g(u) + h(u)$ of some node u on the recorded path from s to n. It signs that the estimation $h(n)$ is too low for the remaining path-cost hence

$$h(n) < g(u) + h(u) - g(n) = h(u) - c^\pi(u,n) \leq h^*(u) - c^\pi(u,n) \leq h^*(n).$$

The A^{**} *algorithm* has been introduced by Dechter and Pearl to correct this failure of the heuristic function.[1] Its evaluation function gives the maximum of the value $g(u) + h(u)$ for all node u on the recorded path $s \rightarrow^\pi n$, i.e.

$$f(n) = \max_{u \in \{s \rightarrow^\pi n\}} [g(u) + h(u)]$$

where h is a non-negative admissible heuristic function. Additionally, the selection breaks ties arbitrarily but in favor of goal nodes.

Corresponding to this definition, the evaluation function value of all open nodes must be always recomputed after each expansion since the recorded paths can be changed. However, the computational cost of the recalculation of the evaluation function value of all open nodes with their recorded path is too high. Russell and Norvig mention an alternative way to implement the A** algorithm. [6] They suggest that the evaluation function value of u is calculated by the following recursive formula:

$$f(u) := \max(g(u) + h(u), f(v))$$

when a better path is found to a node u after the expansion of its parent node v. Initially, $f(s) := h(s)$. The next theorem shows that the original A** algorithm and this latter alternative one work in the same way.

Theorem 1 *Both versions of the A** algorithm contain the same open nodes with the same evaluation function values in each step.*

Proof. We prove this result using induction on the number of the expansions.

Initially both versions add the start node into OPEN with the same evaluation function value ($f(s) = h(s)$). Let us suppose that the statement of the theorem is true in the i^{th} step when the open node n is selected and expanded. Let k be an arbitrary open node after this expansion. We will show that the evaluation function value of k is independent of any version of A**.

If there is no cheaper path from s to k via n, then the node k has to be in OPEN before the expansion of n and neither of the versions modify its evaluation function value.

Otherwise, we must distinguish two cases: either the node k or some of its ancestor is a child of the node n.

If k is a child of n and either k is not in G or $g(n) + c(n, k) < g(k)$ hold, both versions recalculate the value $f(k)$ after the expansion of n. According to the original version,

$$f(k) = \max_{u \in \{s \to^{\pi} k\}} [g(u) + h(u)]$$

where $s \to^{\pi} k$ is the new recorded path via the node n. According to the alternative version,

$$f(k) = \max(f(n), g(k) + h(k)).$$

But the recorded path $s \to^{\pi} n$ is not changed during the expansion of n thus the value $f(n)$ remains the same, and, by the induction hypothesis, this values

in both versions are identical, i.e.

$$f(n) = \max_{u \in \{s \to^{\pi} n\}} [g(u) + h(u)].$$

Thus the new evaluation function values of k in both versions are identical because of

$$\max_{u \in \{s \to^{\pi} k\}} [g(u) + h(u)] = \max\Big(\max_{u \in \{s \to^{\pi} n\}} [g(u) + h(u)], g(k) + h(k) \Big).$$

If k were in OPEN before the expansion of n, and one of its ancestors (let m denote it) were a child of n, and a cheaper path were discovered form s to this very ancestor via n, then the alternative version would not recalculate $f(k)$ but the original version would. Let $f^{old}(k)$ and $f^{new}(k)$ be the earlier and the new value of k maintained by the original version. By the induction hypothesis, $f(k) = f^{old}(k)$. It should be shown that $f^{new}(k) = f^{old}(k)$ because in this case $f(k) = f^{new}(k)$ is followed. Let α denotes the recorded path from s to m before the expansion of n. Let $g^{\alpha}(u)$ denote the cost of α from s to u before the expansion of n, and in this case $g(n) + c(n, m) < g^{\alpha}(m)$. It is obvious that

$$\begin{aligned} f^{old}(k) &= \max\Big(\max_{u \in \{s \to^{\alpha} m\} \setminus \{m\}} [g^{\alpha}(u) + h(u)], g^{\alpha}(m) + h(m), \\ &\qquad \max_{u \in \{m \to^{\pi} k\} \setminus \{m\}} [g(u) + h(u)] \Big), \\ f^{new}(k) &= \max\Big(\max_{u \in \{s \to^{\pi} n\}} [g(u) + h(u)], g(n) + c(n, m) + h(m), \\ &\qquad \max_{u \in \{m \to^{\pi} k\} \setminus \{m\}} [g(u) + h(u)] \Big). \end{aligned}$$

We know that $f(n) = \max_{u \in \{s \to^{\pi} n\}} [g(u) + h(u)]$ and the following inequations hold:

- $\max_{u \in \{s \to^{\alpha} m\} \setminus \{m\}} [g^{\alpha}(u) + h(u)] \leq f(n)$ because the path $s \to^{\alpha} m$ was discovered before the path $s \to^{\pi} m$,
- $g^{\alpha}(m) + h(m) \leq f(n)$ because the path $s \to^{\alpha} m$ was discovered before the path $s \to^{\pi} n$,
- $\max_{u \in \{m \to^{\pi} k\} \setminus \{m\}} [g(u) + h(u)] \leq f(n)$ because the path $m \to^{\pi} k$ was discovered before the path $s \to^{\pi} m$.

On the one hand, it follows from the above inequations that $f^{old}(k) \leq f(n)$ and since the algorithm selects n for expansion instead of k, the inequation $f(n) \leq f^{old}(k)$ must hold. Ergo, $f^{old}(k) = f(n)$. On the other hand, $f(n) \leq f^{new}(k)$ because n is on the recorded path $s \rightarrow^{\pi} k$, and $f(n) \geq f^{new}(k)$ is also true because the path $m \rightarrow^{\pi} k$ were discovered before the path $s \rightarrow^{\pi} n$. It follows that $f^{new}(k) = f(n)$. Thus $f^{new}(k) = f^{old}(k)$. □

The main consequence of this theorem is that the following lemmas and theorems that are proved on only the original version of A** are valid for both versions of A**.

Lemma 2 *The value f(m) calculated by the A** algorithm is proportional to the depth of* m.

Proof. Let $d(m)$ denote the length of the recorded path from start node to the node m. Let $d^*(m)$ denote the length of the shortest path from start node to the node m. It is obvious that $d^*(m) \leq d(m)$. By respecting the definition of A**, $f(m) \geq g(m) + h(m)$ for all open node m. We know that $h(m) \geq 0$ and the cost of the arcs are greater and equal to a positive δ. Thus we have

$$f(m) \geq g(m) + h(m) \geq g(m) > d(m) \cdot \delta \geq d^*(m) \cdot \delta.$$

□

From this lemma, it follows that the A** algorithm can find a solution if there exists a solution even if the heuristic function is non-admissible and only non-negative. This proof is analogous to the proof of the correctness of the A* algorithm. [5]

Lemma 3 *When the A** algorithm selects a node* n *for expansion, the inequation* $f(n) \leq f^*(s)$ *holds.*

Proof. If there is no solution, then $f^*(s)$ can be considered infinite. If there exists a solution, there must be a node m on the optimal solution path at the time of the selection of n so that m is in OPEN and an optimal path from s to m is recorded by algorithm, i.e. $g(u) = g^*(u)$ for all nodes u of this path. Let v denote the node of this path where

$$\max_{u \in \{s \rightarrow^{\pi} m\}} [g(u) + h(u)] = g(v) + h(v)$$

hold. The A** algorithm selects the node n instead of the node m, so $f(n) \leq f(m)$ must hold. Based on the admissible property of the heuristic function we

get

$$f(n) \leq f(m) = \max_{u \in \{s \to^{\pi} m\}} [g(u) + h(u)] = g(v) + h(v) \leq g^*(v) + h^*(v) = f^*(s).$$

□

The consequence of this lemma is that the A^{**} algorithm can find optimal solution if there exists a solution. This proof is analogous to the proof of the optimality of the A^* algorithm. [5]

At last an interesting property of the A^{**} algorithm will be proved.

Theorem 4 *If the A^{**} algorithm selects the node m after the node n for expansion, then $f(n) \leq f(m)$.*

Proof. This statement is enough to prove with two nodes that are expanded directly after each other: firstly the node n, then the node m.

If m has already been in OPEN just before n is expanded but this expansion does not find a cheaper path from s to m, then the value of $f(m)$ does not change. But $f(n) \leq f(m)$ has to hold because the algorithm selects the node n instead of the node m.

If m is not in OPEN before the expansion of n but it gets into there after n is expanded, or if m has already been in OPEN just before n is expanded and this expansion finds a cheaper path from s to m, then m must be a child of n and the recorded path π from s to m drives via n. It is obvious that

$$\max_{u \in \{s \to^{\pi} n\}} [g(u) + h(u)] \leq \max_{u \in \{s \to^{\pi} m\}} [g(u) + h(u)]$$

because n is on the path π. Thus, by respecting the definition of the evaluation function of A^{**}, $f(n) \leq f(m)$ holds. □

An important consequence of this theorem is that if A^{**} expands the same node twice, its second evaluation function value will be greater than or equal to the first one.

3 Memory complexity of the A^{**} algorithm

The memory requirement of a graph-search algorithm depends basically on the size of its search graph. This size can be mesuared by the number of the expanded nodes of this search graph. These are the so-called closed nodes. The size of the search graph is the biggest when the algorithm terminates thus the memory requirement is given with the number of the closed nodes at

termination. Because of this, we assume that the path-finding problems of our focus has got a solution, thus most of the famous graph-search algorithms – specially the A^{**} algorithm – must terminate.

Let CLOSED_S denote the set of closed nodes of the graph-search algorithm S at termination. Let X and Y be arbitrary graph-search algorithms. We can say that X is *better* than Y in a given path-finding problem if $\mathsf{CLOSED}_X \subset \mathsf{CLOSED}_Y$, and X is *not worse* than Y in a given path-finding problem if $\mathsf{CLOSED}_X \subseteq \mathsf{CLOSED}_Y$.

3.1 Comparison of different heuristics in the A^{**} algorithm

At first we will compare two A^{**} algorithms, namely A_1 and A_2 using different heuristic funtions. Let h_1 and h_2 be admissible and non-negative heuristic functions deriving from the same problem. We say – based on the work of Nils Nilsson [5] – that the A_2 algorithm using h_2 is *more informed* than the A_1 algorithm using h_1 if, for all nongoal nodes n, $h_2(n) > h_1(n)$. We would expect intuitively that the more informed algorithm would need to expand fewer nodes to find a minimal cost path. Indeed, analogously to the similar result of the A^* algorithm, we can only prove that A_2 does not expand a node that A_1 does not either.

Theorem 5 *If A_1 and A_2 are two versions of A^{**} such that A_2 is more informed than A_1, then A_2 is not worse than A_1.*

Proof. We prove this result, following to Nilsson [5], using induction on the depth of a node in the spanning tree of the search graph of A_2 at termination.

First, if A_2 expands the node having zero depth, in this case this node must be the start node, then so must A_1. (If s is a goal node, neither algorithm expand any nodes.)

Continuing the inductive argument, we assume (the induction hypothesis) that A_1 expands all the nodes expanded by A_2 having depth d or less in the A_2 search graph. We must now prove that any node n expanded by A_2 and of depth $d+1$ in the A_2 search graph is also expanded by A_1. Let us suppose the opposite of this, namely that there is a node m having depth $d+1$ expanded by A_2 but it is not expanded by A_1. (We remark that this node m may not be a goal node since it is expanded by a graph-search algorithm.)

It is trivial that m had to get into the OPEN for A_2 if A_2 expanded it. According to Lemma 3, since A_2 has expanded node m, we have $f_2(m) \leq f^*(s)$ where $f_2(m)$ is the evaluation function value of m at its expansion.

According to the induction hypothesis, since A_2 has found a path from s to m where all ancestors of m are below the level d, these ancestors are also expanded by A_1. Thus, firstly, node m must be in OPEN for A_1. Let $f_1(m)$ denote the evaluation function value of m at the termination of A_1. It is obvious that $f_1(m) \geq f^*(s)$. Secondly, the recorded path from s to m in the A_1 search graph does not cost more than the path discovered by A_2, that is, $g_1(m) \leq g_2(m)$. Thirdly, for each ancestor v of node m on the recorded path from s to m in the A_1 search graph, $f_1(v) \leq f^*(s)$ since they have been expanded by A_1. Thus, by respecting the definition of the evaluation function of A^{**}, $g_1(v) + h_1(v) \leq f^*(s)$. Because of this, $f_1(m) = g_1(m) + h_1(m)$ follows from the inequation $f_1(m) \geq f^*(s)$.

Summarizing the consequences we get that

$$f_2(m) \leq f^*(s) \leq f_1(m) = g_1(m) + h_1(m) \leq g_2(m) + h_1(m) < g_2(m) + h_2(m) \leq f_2(m),$$

but this is a contradiction. □

3.2 Comparison of the A^{**} algorithm and other admissible graph-search algorithms

In this analysis we will use the natural definition of "equally informed" allowing the algorithms compared to have access to the same heuristic information while placing no restriction on the way they use it. Accordingly, we assume that the heuristic function is a part of the parameters that specify path-finding problem-instances and correspondingly, we will represent each problem-instance by the quadruple $P = (R, s, T, h)$ where $R = (N, A, c)$ is the representation δ-graph, s is the start node, T is the set of goal nodes, and h is the heuristic function. If the heuristic function is non-negative and admissible, then these problem-instances are called *admissible problems*. Moreover we suppose that these problems have got solutions.

As the A^{**} algorithm always finds the optimal solution in an admissible problem, our aim is to compare the A^{**} algorithm to other graph-search algorithms that can also find optimal solution. These algorithms are called admissible graph-search algorithms, or shortly admissible algorithms. Famous graph-search algorithms, as the A^* algorithm, the A^{**} algorithm, the B algorithm [4], or uniform-cost search [5], belong to this algorithm class.

In order to decide if an algorithm X dominates an algorithm Y, several criteria can be used. According to one of these criteria, X dominates Y relative to a set of problems if, in every problem, X is not worse than Y. Additionally,

if Y does not dominate X, i.e. in at least one problem X is better than Y, then we say that X strictly dominates Y relative to that set of problems.

However, now we do not have to compare two algorithms but two algorithm classes. First, the A** algorithm is a non-deterministic algorithm because its OPEN set can contain several nodes with the same evaluation function value and in order to select the best one, we need some rules to break these ties. By collecting all possible tie-breaking-rules we can get a set of deterministic A** algorithms instead of the original non-deterministic one. In this sense A** can be treated as an algorithm class. Secondly, we must compare this algorithm class with all members of the admissible algorithms, i.e. the class of the admissible algorithms. Thus we must extend the concept of the dominance of algorithm to algorithm classes.

The algorithm class X is said to *dominate* the algorithm class Y relative to a set of problems if in every problem for all deterministic versions y of Y there exists a member x of X so that x is not worse than y. X *strictly dominates* Y if X dominates Y and Y does not dominate X. (This definition corresponds to the Type-1 criterion of the paper of Dechter and Pearl. [1])

Dechter and Pearl have thoroughly analyzed the memory complexity of the A* algorithms and they have shown that neither algorithm A* nor any other admissible graph-search algorithm dominate all admissible algorithms. They have proven that the A** algorithm strictly dominates the A* algorithm relative to the admissible problems. Furthermore, they have mentioned the following theorem. (See it as Theorem 10 in Dechter and Pearl's paper. [1])

Theorem 6 *There is no admissible problem where all versions of* A** *expand a node skipped by a deterministic admissible graph-search algorithm.*

Unfortunately, the proof of this theorem has got a little mistake in Dechter and Pearl's paper. In their argumentation, a new admissible problem must be constructed from an elder one but the cost of the new arc adding to the elder one may be zero. In our assumption, however, all arc-costs of the problem of interest must be greater than a positive real number. This condition guarantees that the length of the optimal solution path is finite. At first I tried to improve the original proof but I could not. A new proof is presented here.

Proof. Let us suppose indirectly that P_1 is an admissible problem where all versions of A** expand an extra node skipped by a given deterministic admissible graph-search algorithm Y.

If n denotes the extra node expanded by a version of A** but skipped by Y and C_1^* is the optimal solution cost in the problem P_1, then it is obvious that

$f(n) \leq C_1^*$ at the time of the expansion of n (see Lemma 3). In addition, there must be at least one version of A^{**} (it will be called a^{**}) so that $f(n) < C_1^*$. If this version did not exist, the evaluation function value of each extra node would be identical to C_1^*. But it must be an optimal solution path found by Y avoiding the extra nodes on order to Y could find optimal solution. The set OPEN of all versions of A^{**}always records a node u from this optimal solution path with $f(u) \leq C_1^*$. It means that a version of A^{**} can be constructed in a way that it always prefers the node u even its own extra node, but it is a contradiction.

We are going to construct a new problem P_2 from the search graph (G) maintained by a^{**} over the problem P_1 at its termination appended by a new arc (n, t) where t is a new goal node. (This graph contains all nodes of P_1 expanded by Y.) Let the cost of this new arc be $(C_1^* - f(n))/2$. This is obviously a positive value. The start node of P_2 is the same one of P_1.The heuristic function h is identical to the heuristic function of P_1 except that $h(t) = 0$. Let C_2^* denote the optimal cost from s to t.

It is easy to see that $C_2^* \leq g(n) + c(n, t) \leq f(n) + c(n, t) < C_1^*$ (where $g(n)$ is the cost of the path from s to n, found by a^{**}) so that the only optimal solution path of P_2 goes through the node n to the node t.

Now we will show that P_2 is an admissible problem. The heuristic function was admissible in the problem P_1 thus $h(u) \leq h_1^*(u)$ for all node u of G where $h_1^*(u)$ is the optimal path-cost from u to a goal node of G. Let $h_2^*(u)$ be the optimal path-cost from the node u to a goal node of $G \cup (n, t)$. Since only one new goal node was added it is conceivable that $h_2^*(u) < h_1^*(u)$ for a node u if there is a path from the node u to the node t in P_2. To discover this path, the node u must have been expanded by algorithm a^{**} during the solution of P_1 so it must be expanded during the solution of P_2. By respecting the definition of A^{**}, $f(u) \geq g(u) + h(u)$ for all open node u. If $h(u) > h_2^*(u)$, then we would have

$$f(u) \geq g(u) + h(u) > g(u) + h_2^*(u) \geq g_2^*(u) + h_2^*(u) = f_2^*(u) \geq C_2^*$$

but this contradicts that $f(u) \leq C_2^*$ (Lemma 3). Thus for all node u of G, $h(u) \leq h_2^*(u)$ as well for the new goal node t since $h(t) = 0$ by definition.

In searching P_2, algorithm Y must behave in the same way as in problem P_1, it must avoid expanding n and halting with cost C_1^*, which is higher than that found by a^{**}. This contradicts Y being an admissible algorithm and it should find the optimal solution. □

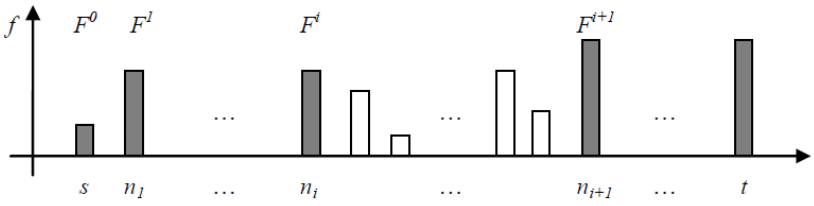


Figure 1: Execution diagram and its thershold values

4 Time complexity of the A** algorithm

The running time of a graph-search algorithm depends on the number of its iteration and on the time of one iteration. The latter factor can be estimated on the basis of only the knowledge of the problem wanted to be solved, and the first factor depends primaly on the strategy of the graph-search. Thus we will analyze just this first factor. This will be given as a function of the number of closed nodes at termination. (We suppose that the problems of our interest have got a solution thus our algorithms terminate.) Hereinafter k will denote the number of closed nodes at termination.

It is clear that the best running time is k since at least k iterations are needed to expand k nodes if every node is expanded only once. But in many problems a node may expand several times. It is known that, in the worst case, the time complexity of the A* algorithm is 2^{k-1} but there is a modification of A*, this is algorithm B; its running time is at most $\frac{1}{2}k^2$ in the worst case. [4]

An excellent tool to present the execution of a graph-search algorithm and to calculate its time complexity is the execution diagram (Figure 1). This diagram enumerates the expanded nodes with their current evaluation function values in the order of their expansions (the same node may occur several times). It is trivial that the first value is $f(s)$ and the last one is the value of the goal node found. (The goal node is selected for expansion but not expanded.) A monotone increasing subsequence F^i $(i = 1, 2, \dots)$ is constructed from the values of the diagram so that it starts with the first value $f(s)$ and then the closest non less one must always be selected. The selected values are called threshold values and the nodes belonging to these values are called threshold nodes. Let n^i denote the i^{th} threshold node where $F^i = f(n^i)$ is its threshold value. The set of nodes that are expanded between the i^{th} and the $(i+1)^{th}$ threshold nodes is called the i^{th} ditch.

Let us introduce some further notations:

- OPEN^i – the OPEN set just before the i^{th} threshold node is expanded by A^*
- $g^i(u)$ – the value $g(u)$ just before the i^{th} threshold node is expanded by A^*
- $f^{A^*}(u)$ – the evaluation function value $f(u)$ according to A^*
- $f^{A^{**}}(u)$ – the evaluation function value $f(u)$ according to A^{**}

Theorem 7 *Consider an admissible path-finding problem where the threshold values of the execution diagram of the A^* algorithm form a strictly monotone increasing number sequence. Algorithm A^{**} expands the threshold nodes of A^* in the same order and with the same threshold values if these algorithms use the same tie-breaking-rule.*

Proof. We prove this theorem using induction on the threshold nodes of the diagram of the A^* algorithm. The first threshold node is the start node and this same node is expanded by A^{**} at first. It is clear that $f^{A^*}(s) = f^{A^{**}}(s)$. Let us suppose that until the i^{th} threshold node the statement is true, that is, at the expansion of n^i both algorithms maintain the same search graphs, the same sets OPEN, the same cost function (g) values, the same parent pointers (π) and $F^i = f^{A^*}(n^i) = f^{A^{**}}(n^i)$. (This is the induction hypothesis.) We will show that between n^i and n^{i+1}, the A^* algorithm and the A^{**} algorithm expand the same nodes. It follows that the induction statement is also true for the $(i+1)^{th}$ threshold node.

Let us describe the nodes expanded by A^* in the i^{th} ditch. A node m belongs to this ditch if it gets into OPEN after the expansion of n^i and $f^{A^*}(m) < F^i$. The condition of m getting into OPEN is that there exists a path $s \to m$ via the node n^i so that it is found after the expansion of n^i and all nodes on this path between n^i and m are also in this ditch and the cost of this path is cheaper than all other path $s \to m$ discovered before. It also means that, for all nodes u on that path between n^i and m, $g(u) + h(u) < F^i$ holds. [3]

On the one hand, if the node m of the i^{th} ditch is put into OPEN by the A^{**} algorithm, then it must be expanded before the next threshold since

$$\begin{aligned} f^{A^{**}}(m) &= \max_{u \in s \to^{\pi} m} [g(u) + h(u)] = \\ &= \max\Big(\max_{u \in s \to^{\pi} n^i} [g(u) + h(u)], \max_{u \in n^i \to^{\pi} m} [g(u) + h(u)] \Big) = \\ &= \max\Big(f^{A^{**}}(n^i), \max_{u \in n^i \to^{\pi} m} [g(u) + h(u)] \Big) = \\ &= F^i < F^{i+1}. \end{aligned}$$

On the other hand, let us suppose now indirectly that A^{**} expands a node before the expansion of n^{i+1} that does not belong to the i^{th} ditch of A^*. Let us take the first such node in the order of the expansions of A^{**}. This node will be denoted by m. It is obvious that m must be put into OPEN by A^*. If $f^{A^{**}}(m) < F^{i+1}$, then the node m should be expanded by the A^* algorithm before n^{i+1}, thus m would belong to the i^{th} ditch of A^*. If $f^{A^{**}}(m) \geq F^{i+1}$, then all nodes of the i^{th} ditch would precede the expansion of m, thus n^{i+1} would be put into OPEN, too. The only chance of the expansion of m preceding the expansion of n^{i+1} is that $f^{A^{**}}(m) = F^{i+1}$. It is clear that $f^{A^{**}}(m) = g(m) + h(m)$. In this case the node m would be put into OPEN by A^* after the expansions of the i^{th} ditch and $f^{A^*}(m) = F^i$ would follow from $f^{A^{**}}(m) = g(m) + h(m)$. If the tie-breaking-rule of A^{**} selected the node m instead of n^{i+1}, then A^* would do the same, that is n^{i+1} would not be the next threshold node of A^*. This is a contradiction. □

There are two interesting side effects of this proof. First, the A^* algorithm and the A^{**} algorithm expand the same nodes between two neighboring threshold nodes. Secondly, the evaluation function values of the nodes expanded by A^{**} in the i^{th} ditch are equal to the i^{th} threshold value F^i.

We underline that the difference of execution diagrams of algorithms A^* and A^{**} under the constraints of the previous theorem is that how many times a node is expanded between two neighboring thresholds and how much its evaluation function value is. The number of the expansions of A^{**} partly depends on the order of the expansions of the nodes having the same evaluation function value. If the version of A^{**} is introduced, whose tie-breaking rule prefers the node having less costly recorded path from s, it can be prevented that the same node is expanded several times in one ditch because, after the expansion of a node, algorithm cannot find a better path to this node in this ditch. Thus the running time of this version is not worse than any version of A^*.

But what can we say when the threshold values of the execution diagram of A^* is not strictly monotone but just monotone? In this case some neighboring threshold values may be equal. In the ditches after these thresholds the A^{**} algorithm expands the nodes with the same evaluation function value. Because our tie-breaking-rule defined in the previous paragraph may prefer n^{i+1} for expansion to some node of the i^{th} ditch and this node will be expanded later or never, thus the number of the expansions of this tie-breaking-rule may be less than the number of the expansions of the best version of the A^* algorithm. On the Figure 2 our tie-breaking-rule of the A^{**} algorithm expands every node

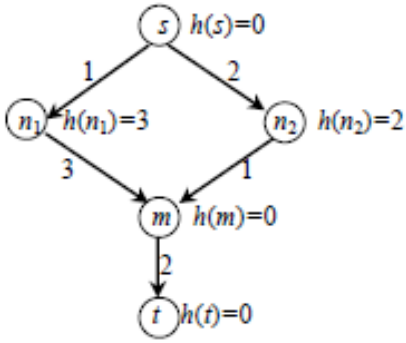


Figure 2: Example where the A^{**} algorithm is faster than algorithm B

only once but the best version of the A^* algorithm, this is algorithm B [4], expands the node m twice.

5 Summary

The A^{**} algorithm was defined by Dechter and Pearl [1] but it was introduced in a bit different form by Russell and Norviq. [6] I have shown that these versions do not differ (Theorem 1). The fact that the A^{**} algorithm is an admissible graph-search algorithm is a well-known property. But I have mentioned that, if the heuristic function applied by algorithm is not admissible but only non-negative, then A^{**} always finds a not necessarily optimal solution (if there exists a solution). I have proved that the evaluation function values of the nodes expanded by A^{**} form a monotone increasing number sequence (Theorem 4).

Then we have extended the the concept of "better-informed" derived from Nilsson [5] onto the A^{**} algorithm. It allows us to compare the memory complexity of two algorithms A^{**} using different heuristics (Theorem 5). Perhaps it is much more important to compare the memory requirement of A^{**} with other graph-search algorithm, especially the A^* algorithm using the same heuristics. All of this has been done in the excellent work of Dechter and Pearl. However there is a statement in that paper about the A^{**} algorithm whose proof has got a mistake. I have given another proof of that statement (Theorem 6).

At last I have shown (Theorem 7) that the time complexity, more precisely, the number of the expansions of a certain version of the A^{**} algorithm is not worse than the versions of A^*.

References

[1] R. Dechter, J. Pearl, Generalized best-first search strategies and optimality of A*, *Journal ACM*, **32,** 3, (1985) 505–536. ⇒192, 199, 204

[2] I. Fekete, T. Gregorics, L. Zs. Varga, Corrections to graph-search algorithms. *Proc. of the Fourth Conference of Program Designers*, ELTE, Budapest, June 1–3, 1988. ⇒191

[3] T. Gregorics, Which of graphsearch versions is the best? *Annales Univ. Sci. Budapest., Sect. Comput.* **15** (1995) 93–108. ⇒191, 202

[4] A. Martelli, On the complexity of admissible search algorithms. *Artificial Intelligence*, **8,** 1 (1977) 1–13. ⇒198, 201, 204

[5] N. J. Nilsson, *Principles of Artificial Intelligence.* Springer-Verlag, 1982. ⇒190, 191, 195, 196, 197, 198, 204

[6] S. J. Russell, P. Norvig, *Artificial Intelligence. A Modern Approach.* Prentice Hall Inc., 1995. ⇒193, 204

Received: October 7, 2014 • Revised: October 30, 2014

Acta Univ. Sapientiae, Informatica, **6,** 2 (2014) 206–209

Partitioning to three matchings of given size is NP-complete for bipartite graphs

Dömötör PÁLVÖLGYI
Eötvös Loránd University, Institute of Mathematics
email: dom@cs.elte.hu

Abstract. We show that the problem of deciding whether the edge set of a bipartite graph can be partitioned into three matchings, of size k_1, k_2 and k_3 is NP-complete, even if one of the matchings is required to be perfect. We also show that the problem of deciding whether the edge set of a simple graph contains a perfect matching and a disjoint matching of size k or not is NP-complete, already for bipartite graphs with maximum degree 3. It also follows from our construction that it is NP-complete to decide whether in a bipartite graph there is a perfect matching and a disjoint matching that covers all vertices whose degree is at least 2.

Folkman and Fulkerson [2] described bipartite graphs whose edge set can be partitioned into l_1 matchings of size k_1 and l_2 matchings of size k_2. We complement this result by showing that it is NP-complete to decide whether the edge set of a bipartite graph can be partitioned into three matchings, of size k_1, k_2 and k_3. This will follow from the NP-completeness of the following "perfect matching + matching" problem.

Input: G bipartite graph with maximum degree 3, natural number k.
Goal: Decide whether G contains an edge-disjoint perfect matching and a matching of size k.

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C30, 05C50
Key words and phrases: NP-completeness, disjoint matchings, bipartite graphs, partitioning

Proof. First we show how the hardness of the partitioning problem follows from the hardness of this problem. Notice that if G contains an edge-disjoint perfect matching P and a matching M of size k, then it also contains another matching M' which is also edge-disjoint from P, has size at least k and is such that $E(G) - P - M'$ is also a matching, as we can start alternating paths from degree two vertices of $E(G) - P - M$. Therefore G contains an edge-disjoint perfect matching and a matching of size k if and only if its edges can be partitioned into three matchings, of size n, k' and $|E(G)| - n - k'$ for some $k' \geq k$. (This was only a Cook reduction, but from our construction below it can be easily made into a Karp reduction.)

Next we show that the "perfect matching + matching" problem is NP-complete. The reduction is from MAX-2-SAT, in which the input is a conjunctive normal form such that every clause contains at most 2 literals (2CNF) and a number s and the question is whether at least s clauses are satisfiable. This problem is well known to be NP-complete [3]. Let us denote the variables of our input Ψ by $x_1, \ldots, x_n$ and the clauses by $C_1, \ldots, C_m$. From this we make a bipartite graph G of maximum degree 3 on $N = 4mn + 4m$ vertices that will contain an edge-disjoint perfect matching P and a matching M of size $k = (N - 4m + 2s)/2$ if and only if at least s clauses of the input Ψ were satisfiable.

G consists of several smaller parts, which we now describe. To every variable x_i we associate a cycle of length $4m$, denoted by X_i. The vertices of X_i are denoted (in cyclic order) by $a_1^i, b_1^i, c_1^i, d_1^i, a_2^i, b_2^i, c_2^i, d_2^i, \ldots, a_m^i, b_m^i, c_m^i, d_m^i$. To every clause C_j we associate four vertices, $u_j, v_j, u_j^{leaf}, v_j^{leaf}$ and two edges, $u_j u_j^{leaf}$ and $v_j v_j^{leaf}$.

These parts are connected as follows. If x_i is an unnegated variable of C_j, then a_j^i is connected to u_j and b_j^i is connected to v_j, while if x_i is a negated variable of C_j, then c_j^i is connected to u_j and b_j^i is connected to v_j. There are no other edges in the graph.

To see that G is bipartite, we can color all vertices $u_j, v_j^{leaf}, b_j^i, d_j^i$ with one color, and all vertices $v_j, u_j^{leaf}, a_j^i, c_j^i$ with the other.

Notice that G has exactly 2^n perfect matchings, as for each cycle X_i we can choose whether we select its edges $a_j^i b_j^i$ and $c_j^i d_j^i$ or $b_j^i c_j^i$ and $d_j^i a_{j+1}^i$ for all j (with circular indexing). The latter of these will correspond to x_i being true, the former to x_i being false.

Now, suppose that s clauses of Ψ are satisfiable. First we select an assignment of the variables satisfying s clauses and the corresponding perfect matching P, then we will show that a disjoint matching M of size s exists. For every

satisfied clause C_j, we select a variable that satisfies it, some $x_{f(j)}$. Then the edges of M will be the symmetric difference of the following three sets. For all j where $x_{f(j)}$ is unnegated in C_j, take the path $u_j a_j^{f(j)} b_j^{f(j)} v_j$. For all j where $x_{f(j)}$ is negated in C_j, take the path $u_j c_j^{f(j)} b_j^{f(j)} v_j$. Finally, take $\bigcup_i X_i \setminus P$. So expressed with a formula

$$M = \left(\bigcup_{x_{f(j)} \in C_j} u_j c_j^{f(j)} b_j^{f(j)} v_j \right) \Delta \left(\bigcup_{\bar{x}_{f(j)} \in C_j} u_j a_j^{f(j)} b_j^{f(j)} v_j \right) \Delta \left(\bigcup_i X_i \setminus P \right).$$

The fact that P corresponds to the truth assignments of the variables guarantees that $a_j^i b_j^i \notin P$ if x_i is true and $b_j^i c_j^i \notin P$ if x_i is false, so we indeed obtained a matching covering all the vertices but the leafs and further the $2m - 2s$ vertices that belong to unsatisfied clauses.

On the other hand, suppose that we have an edge-disjoint perfect matching P and matching M covering $2k$ vertices. M must obviously miss the $2m$ leafs, as it is disjoint from P. We can also suppose that M covers each vertex of each X_i, as they can be covered by a matching even after the removal of P. Now we claim that the truth assignment that corresponds to P will satisfy at least s clauses of Ψ. Indeed, if u_j is connected to a vertex from X_i, then v_j must be also connected to X_i and x_i must satisfy C_j, or M would not be a matching covering the vertices of X_i.

This finishes the proof. □

Note that if instead of MAX-2-SAT we use 3-OCC-MAX-2SAT where every variable can appear in at most 3 clauses (total of unnegated and negated occurrences), then the number of vertices is only $12n + 4m$. This problem is also known to be NP-complete, in fact even inapproximable for some small constant, see [1].

A more interesting modification proves the NP-completeness of the following problem.

Input: G bipartite graph with maximum degree 4.

Goal: Decide whether G contains two edge-disjoint matchings, P and M, such that P is perfect and M covers every vertex whose degree is at least 2.

If in our construction instead of MAX-2-SAT we use 3-SAT, then such perfect matching P and matching M, covering all but the u_j^{leaf}, v_j^{leaf} vertices, exist if and only if the original formula is satisfiable, which proves the NP-completeness. As in this reduction the maximum degree grows to 4, we leave the maximum degree 3 case open.

Acknowledgement

I would like to thank András Frank and Attila Bernáth for calling my attention to the problems, them and Csaba and Zoltán Király for several useful comments, and Marzio De Biasi for pointing me to [1].

I thank also the moral support of Institute of Mathematics of Eötvös University, Budapest and the financial support of Hungarian National Science Fund (OTKA), grant PD 104386 and the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

References

[1] P. Berman, M. Karpinski, *ICALP*, LNCS **1644** (1999) 200–209. ⇒208, 209

[2] J. H. Folkman, D. R. Fulkerson, Edge colorings in bipartite graphs. RAND Memorandum RM-5061-PR, RAND Corporation, 1966. ⇒206

[3] M. R. Garey, D. S. Johnson, L. J. Stockmeyer, Some simplified NP-complete graph problems, *Theor. Comput. Sci.* **1,** 3 (1976) 237–267. ⇒207

Received: July 11, 2014 • Revised: October 31, 2014

Acta Univ. Sapientiae, Informatica, **6,** 2 (2014) 210–229

Reconstruction of score sets

Antal IVÁNYI
Eötvös Loránd University,
Faculty of Informatics, 1117 Budapest,
Pázmány Péter sétány 1/C., Hungary
email: tony@inf.elte.hu

Abstract. The *score set* of a tournament is defined as the set of its different outdegrees. In 1978 Reid [15] published the conjecture that for any set of nonnegative integers D there exists a tournament T whose degree set is D. Reid proved the conjecture for tournaments containing $n = 1,\ 2$, and 3 vertices. In 1986 Hager [4] published a constructive proof of the conjecture for $n = 4$ and 5 vertices. In 1989 Yao [18] presented an arithmetical proof of the conjecture, but general polynomial construction algorithm is not known. In [6] we described polynomial time algorithms which reconstruct the score sets containing only elements less than 7. In [5] we improved this bound to 9.

In this paper we present and analyze new algorithms Hole-Map, Hole-Pairs, Hole-Max, Hole-Shift, Fill-All, Prefix-Deletion, and using them improve the above bound to 12, giving a constructive partial proof of Reid's conjecture.

1 Introduction

We will use the following definitions [3]. A *graph* $G(V, E)$ consists of two finite sets V and E, where the elements of V are called *vertices*, the elements of E are called *edges* and each edge has a set of one or two vertices associated to it, which are called its *endpoints* (*head and tail*). An edge is said to *join* its endpoints. A *simple graph* is a graph that has no self-loops and multi-edges.

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C20, 68R10
Key words and phrases: score set, score sequence, approximation algorithms, Reid conjecture

A *directed edge* is said to be *directed* from its *tail* and *directed* to its *head*. (The tail and head of a directed self-loop is the same vertex.)

A *directed graph* (*shortly: digraph*) is a graph whose edges are directed. If in a directed graph $(u, v) \in E$, then we say that u *dominates* v. An *oriented graph* is a digraph obtained by choosing an orientation (direction) for each edge of a simple graph. A *tournament* is a complete oriented graph. That is, it has no self-loops, and between every pair of vertices, there is exactly one edge. Beside the terms of graph theory we will use the popular terms player, score sequence, score set, point, win, loss etc.

A directed graph (so a tournament too) $F = (E, V)$ is *transitive*, if $(u, v) \in E$ and $(v, w) \in E$ imply $(u, w) \in E$.

The *order of a tournament* T is the number of vertices in T. A tournament of order n will be called an *n-tournament*.

An *(a, b, n)-tournament* is a loopless directed graph, in which every pair of distinct vertices is connected with at least a and at most $b \geq a$ edges. An (a, b, n)-tournament is *complete*, if in the matches any integer partition of c points is permitted for $a \leq c \leq b$.

The *score* (or *out-degree*) of a vertex v in a tournament T is the number of vertices that v dominates. It is denoted by $d_T^+(v)$ (shortly: $d(v)$).

The *degree sequence* (*score sequence*) of an n-tournament T is the ordered n-tuple $s_1, s_2, \ldots, s_n$, where s_i is the score of the vertex v_i, $1 \leq i \leq n$, and

$$s_1 \leq s_2 \leq \cdots \leq s_n. \tag{1}$$

An *n-regular sequence* is an increasingly ordered n-length integer sequence, that is an n-length score sequence is and n-regular sequence.

The *score set* of an n-tournament T is the ordered m-tuple $D = (d_1, d_2, \ldots, d_m)$ of the different scores of T, where

$$d_1 < d_2 < \cdots < d_m. \tag{2}$$

Figure 1 shows a $(1, 1, 4)$-tournament with score sequence $S = 0,\ 2,\ 2,\ 2$ and score set $D = \{0,\ 2\}$.

Theorem Landau [8, 10, 16] allows to test potential score sequences in linear time.

Theorem 1 (Landau [10]) *A nondecreasing sequence of nonnegative integers $S = s_1, s_2, \ldots, s_n$ is a score sequence of an n-tournament if and only if*

$$\sum_{i=1}^{k} s_i \geq \frac{k(k-1)}{2}, \quad 1 \leq k \leq n, \tag{3}$$

with equality when $k = n$.

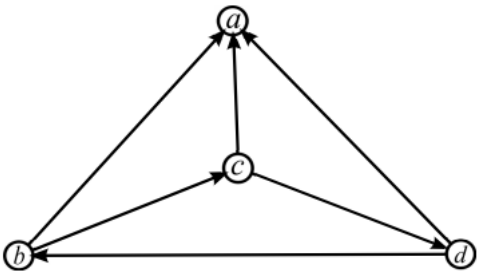


Figure 1: A tournament with score sequence 0, 2, 2, 2 and score set $\{0, 2\}$

Proof. See [10, 16]. □

To reconstruct a prescribed score set is much harder problem, then computing the score set belonging to the score sequence of a given tournament. Therefore surprising is the following conjecture published by Reid in 1978 [15]: if $m \geq 1$, $D = \{d_1, d_2, \ldots, d_m\}$ is a set of nonnegative integers, then there exists a tournament whose score set is D.

In his paper Reid described the proof of his conjecture for score sets containing 1, 2, and 3 elements, further for score sets representing an arithmetical or geometrical series. In 1986 Hager [4] published a constructive polynomial proof of the conjecture for $m = 4$ and $m = 5$.

In 2006 Pirzada and Naikoo gave a constructive proof of a special case of Theorem 3.

Theorem 2 (Pirzada and Naikoo [14]) *If $s_1, s_2, \ldots, s_m$ are nonnegative integers with $s_1 < s_2 < \cdots < s_m$, then there exists such $n \geq m$ for which there exists an n-tournament T with score set*

$$D = \left\{ d_1 = s_1,\ d_2 = \sum_{i=1}^{2} s_i,\ \ldots,\ d_m = \sum_{i=1}^{m} s_i \right\}. \tag{4}$$

Proof. See [14]. □

In 1976 Chartrand, Lesniak and Roberts [1] proved that for any finite set S of nonnegative integers there exists an oriented graph whose score set is S.

In 1989 Yao proved the conjecture of Reid.

Theorem 3 (Yao [18]) *If $m \geq 1$, $\mathcal{D} = \{d_1, d_2, \ldots, d_m\}$ is a set of nonnegative integers, then there exists a tournament T with score sequence $S = s_1, s_2, \ldots, s_n$ such, that the score set of T is D.*

Proof. See [18]. □

The proof of Yao uses arithmetical tools and only proves the existence of the corresponding tournaments, but it does not give a construction.

In 1983 Wayland [17] proposed a sufficient condition for a set D of nonnegative integers to be the score set of a bipartite tournament. This result was improved to a sufficient and necessary condition in 1983 by Petrovič [12].

In [7] we proved that the extension of Yao's theorem is not true for k-tournaments (where every pair of vertices is connected with $k \geq 2$ edges).

Now we present three lemmas allowing a useful extension of Theorem 3.

Lemma 4 *If $d_1 \geq 1$, then the score set $D = \{d_1\}$ is realizable by the unique score sequence $S = d_1^{<2d_1+1>}$.*

Proof. If $|S| = n$ and S generates D then the sum of the elements of S equals to nd_1 and also to $n(n-1)/2$ implying $n = 2d_1 + 1$. Such tournament is realizable for example so, that any player P_i gathers one points against players $P_{i+1}, \ldots, P_{i+(n-1)/2}$ and zero against the remaining players (the indices are taken mod n). □

In this lemma and later $a^{<b>}$ means a multiset, in which a is repeated b times.The form of the score sequences using this notation is called *power form.*

Lemma 5 *If the score sequence $S = s_1, s_2, \ldots, s_n$ corresponds to the score set $D = \{d_1, d_2, \ldots, d_m\}$, then $n \geq d_m + 1$.*

Proof. If the score of a vertex v is d_m, then v dominates d_m different vertices. □

Lemma 6 *If $m \geq 2$ and the score sequence $S = s_1, s_2, \ldots, s_n$ corresponds to the score set $D = \{d_1, d_2, \ldots, d_m\}$, then*

$$2d_1 + 2 \leq n \leq 2d_m, \tag{5}$$

and both bounds are sharp.

Proof. Every element of D has to appear in S. Therefore the arithmetical mean of the scores is greater, than d_1, and smaller, than d_m. From the other side n-tournaments contain $B_n = \binom{n}{2}$ edges, so the arithmetical mean of their scores is $B_n/n = (n-1)/2$, therefore

$$d_1 < \frac{n-1}{2} < d_m, \tag{6}$$

implying (5).

For example if $k \geq 0$ and $D = \{k, k+1\}$, then according to (6) $n = 2k+2$ imply the sharpness of both bounds. □

The next extension of Theorem 3 is based on Lemmas 4, 5, 6.

Theorem 7 (Iványi, Lucz, Matuszka, Gombos [6]) *If $m \geq 1$ and $D = \{d_1, d_2, \ldots, d_m\}$ is an increasingly ordered set of nonnegative integers, then*

- *there exist a tournament T, whose score sequence is S and score set is D;*
- *if $m = 1$, then $S = s_1^{<2d_1+1>}$;*
- *if $m \geq 2$, then*

$$\max(d_m + 1, 2d_1 + 2) \leq n \leq 2d_m; \tag{7}$$

- *the bounds in (7) are sharp.*

Proof. The assertion follows from the above lemmas (see [6]). □

Taking into account the remark of Beineke and Eggleton [16, page 180] we can formulate Reid's conjecture as an arithmetical statement without the terms of the graph theory. Let $D = \{d_1, d_2, \ldots, d_m\}$ be an increasingly ordered set of nonnegative integers. According to the conjecture there exist positive integer exponents $x_1, x_2, \ldots, x_m$ such that

$$S = d_1^{<x_1>}, d_2^{<x_2>}, \ldots, d_m^{<x_m>} \tag{8}$$

is the score sequence of some $(\sum_{i=1}^{m} x_i)$-tournament. Using Landau's theorem it can be easily seen that Reid's conjecture is equivalent to the following statement [13, 18].

For every set $D = \{d_1, \ldots, d_m\}$ with the property $0 \leq d_1 < d_2 < \cdots < d_m$ there exist positive integers $x_1, \ldots, x_m$, such that

$$\sum_{i=1}^{k} x_i d_i \geq \binom{\sum_{i=1}^{k} x_i}{2}, \quad \text{for } k = 1, \ldots, m-1, \tag{9}$$

and

$$\sum_{i=1}^{m} x_i d_i = \binom{\sum_{i=1}^{m} x_i}{2}. \tag{10}$$

Commenting Yao's proof Qiao Li wrote in 1989 [11]: *Yao's proof is the first proof of the conjecture, but I do not think it is the last one. I hope a shorter and simpler new proof will be coming in the near future.*

However, the constructive proof has not been discovered yet.

Our algorithms investigate only the zerofree score sets, The base of this approach is the following lemma.

Lemma 8 *Let $m \geq 2$. A sequence $S = s_1^{<e_1>}, s_2^{<e_2>}, \ldots, s_n^{<e_m>}$ is the score sequence corresponding to the score set $D = \{0, d_2, d_3, \ldots, d_m\}$ if and only if the sequence $S' = (s_2 - 1)^{<e_2>}, (s_3 - 1)^{<e_3>}, \ldots, (s_n - 1)^{<e_n>}$ is the score sequence corresponding to $D' = \{d_2 - 1, d_3 - 1, \ldots, d_m - 1\}$.*

Proof. Part *if* of the proof: If S is the score sequence corresponding to D then $s_1 = 0$ and $e_1 = 1$ that is all other players won against the player having the score $s_1 = 0$, so S' corresponds to D'.

Part *only if* of the proof: If S' corresponds to D', then we add a new score $d_1 = 0$ to D', increase the multiplicity of the other scores by 1 and get D which correspond to S. □

2 Reconstruction of score sets of tournaments

Earlier [5, 6] we proposed polynomial approximate algorithms Balancing, Shortening, and Shiftening.

By computer experiments we proved that they reconstruct all score sets with $d_m < 9$. We also described exact brute force algorithms Sequencing and Diophantine [6].

The proposed algorithms reconstruct the majority of the score sets with $d_m = 9$. , but there are three exceptional sets with $d_m = 9$. Exceptions are the sets $\{2, 4, 5, 6, 7, 8, 9\}$, $\{1, 2, 5, 6, 7, 8, 9\}$, and $\{1, 2, 4, 7, 8, 9\}$.

In this paper we present new polynomial algorithms Hole-Pairs, Hole-Max, Hole-Shift, Prefix-Shift, Fill-All and using these theorems we improve the earlier bound to $d_m < 12$. Our algorithms are based on Theorem 7. Since there are quick (quadratic) algorithms to construct n-tournaments corresponding to a given score sequence, our algorithms construct only a suitable score sequence.

If the score sequence of a tournament is S and its score set is D, then we say, that S *generates* D, or D *corresponds* to S. If D is given, then we call the corresponding score sequence *good*.

3 Hole-type algorithms

The Hole-type algorithms are based on the idea that we take the transitive score sequence $s = 0,\ 1,\ \ldots,\ d_m$ and gradually remove from it the elements corresponding to the missing elements of the investigated score set.

In a score set $D = \{d_1, d_2, \ldots, d_m\}$ there is a *k-hole* before element d_i ($1 \leq i \leq m$), if

- $d_1 = k + 1$, or
- $2 \leq i \leq m$ and $d_i - d_{i-1} = k$.

In the first case the hole is *outer*, while in the second case it is an *inner* hole. The missing elements are called *hole elements*, while the neighbors of a hole are its *lower*, resp. *upper* bound. During the maintenance of the score set D a hole is *active*, if the elements of the hole are present in s. At the beginning all holes are active. A hole is *passive*, if its elements are missing from the actual score sequence. During the reconstruction of D we gradually transform the active holes to passive. The reconstruction process is finished, when D contains only passive holes.

We prepare the work of the HOLE-type algorithms with the construction of a *hole-map*.

3.1 Algorithm HOLE-MAP

The *hole map* of a score set $D = \{d_1, \ldots, d_m\}$ is an $(m \times d_m)$-sized array $H(D) = H[1\ .\ .m, 1\ .\ .d_m]$, where the j-th column of $H(D)$ describes the j-sized active holes: $H[i, j]$ gives the beginning address of the i-th j-sized hole in D (if there exists such hole, otherwise $H[i, j]$ is undetermined).

The next algorithm HOLE-MAP generates the hole map $H(D)$, further the number of active holes $N[0](D)$ and the number of active i-holes $N[i](D)$ $(1 \leq i \leq d_m)$ in D. $N(D)$ is *the frequency vector* of the active holes.

The pseudocodes of this paper are written using the conventions proposed in the textbook [2].

Input. $m = m(D)$ $(m \geq 1)$: the number of the elements of D;
$D = \{d_1, \ldots, d_m\}$: a score set containing m elements.

Global variables. $D = \{d_1, \ldots, d_m\}$: score set;
m: the number of elements of D;
H: hole map of D.

Working variables. $i,\ j$: cycle variables.

Output: $H[1\ .\ .\ m, 1\ .\ .\ d_m]$: the hole map of D;
$N[0\ .\ .\ m] = N(D)$: the hole frequency vector of D.

HOLE-MAP(m, D)

```
01 for i = 0 to d_m                          // Line 01–03: initialization
02     N[i] = 0
03 if d_1 > 0                                // Line 03–06: is there an outer hole?
04     N[0] = 1
05     N[d_1] = 1
```

06 $H[1, d_1] = 0$
07 **for** $j = 1$ **to** $m - 1$ // Line 07–11: investigation of the inner holes
08 **if** $d_{j+1} - d_j > 1$
09 $N[d_{j+1} - d_j - 1] = N[d_{j+1} - d_j - 1] + 1$
10 $N[0] = N[0] + 1$
11 $H[N[d_{j+1} - d_j - 1], d_{j+1} - d_j - 1] = d_j + 1 + 1$
12 $H[N[0] = 0$
13 'no inner hole'
14 **return** H, N // Line 13: return of the results

If we use algorithm HOLE-MAP for the score set $D = \{2, 4, 6, 7\}$, then the input is D and $m = 4$, the size of the vector N is 7. Table 1 contains $H(D)$.

beginning/hole size	1	2	3	4	5	6	7
first hole	2	0	−−	−−	−−	−−	−−
second hole	4	−−	−−	−−	−−	−−	−−
third hole	−−	−−	−−	−−	−−	−−	−−
fourth hole	−−	−−	−−	−−	−−	−−	−−

Table 1: Hole map $H(D)$ of the score set $D = \{2, 4, 6, 7\}$

The running time of HOLE-MAP is $\Omega(d_m)$ in all cases.

3.2 Algorithm HOLE-PAIRS

Two i-sized holes form a *hole pair*. The hole which appear earlier in D is called *lower hole*, while the other one is called *upper hole*.

Our algorithms do not change the result of the matches between players in the active holes and any other player. Since for the initial transitive sequence is it true that the players are defeated by any other player having larger score, this property remains true (in such sense that there exists such tournament which realizes the given score sequence and has this property) for the players whose score is in an active hole after the application of HOLE-PAIRS and HOLE-MAX. We can call the such sequences *hole-transitive*.

The next algorithm HOLE-PAIRS forms the maximal possible number, that is $\lfloor N[i]/2 \rfloor$, of pairs of i-holes of D for $1 \leq i \leq d_m$ and removes the hole elements from the corresponding sequence.

Input. $N(D)$: the frequency vector of the active holes in D.

Global variables. $D = \{d_1, \ldots, d_m\}$: score set;
m: the number of elements of D;

H: hole map of D;
$s = 0, 1, \ldots, d_m$: the starting transitive score sequence;
Working variables. $i,\ j$: cycle variables.
Output. $t = t_0, t_1, \ldots, t_{d_m}$: the sequence produced by HOLE-PAIRS;
$M(D)$: the frequency vector of the active holes in D.

HOLE-PAIRS(N, s)

```
01 for i = 0 to d_m                                  // Line 01–03: initialization
02     M[i] = N[i]
03     t[i] = s[i]
04 for i = d_m downto 1                  // Line 04–11: processing of the hole pairs
05     while M[i] > 1
06         for j = H[M[i], i] to H[M[i], i] + i − 1
                                  // Line 06–07: maintenance of the upper hole
07             t[j] = t[H[M[i], i]] − 1
08         for j = H[M[i] − 1, i] to H[M[i] − 1, i] + i
                                  // Line 08–09: maintenance of the lower hole
09             t[j] = t[H[M[i − 1] + i, i]] + 1
10         M[i] = M[i] − 2                       // Line 10: updating of M[i]
11 return t, M                                   // Line 11: return of the results
```

If the input of HOLE-PAIRS is the hole map $H(D)$ of the score set $D = \{2, 4, 6, 7\}$, then the algorithm starts with the transitive score sequence $t = 0, 1, \ldots, 7$, and processing the 1-holes at 3 and 5 gets the shortened sequence $t = 0, 1, 2, 4^3, 6, 7$. The number of active holes in D is reduced to $M[0] = 1$.

The running time of HOLE-PAIRS is $\Theta(d_m)$ in all cases.

3.3 Algorithm HOLE-MAX

If $M[0](D) > 0$, then we have at least one active hole and can simply eliminate the largest one: if the largest hole is a c-hole, then we add the elements $d_m + 1,\ d_m + 2,\ \ldots,\ d_m + c$ to t and reduction of these elements to d_m allows us to increase the elements of the c-hole to its upper bound, as the next algorithm HOLE-MAX makes.

Input. $M(D)$: the updated frequency vector of the active holes in D;
$t = t_0, t_1, \ldots, t_{d_m}$: the reduced sequence produced by HOLE-PAIRS.
Global variables. D: score set;
m: the number of elements of D;
H: hole map of D.
Working variables. $i,\ j,\ k$: cycle variables.

Output. $u = u_0, u_1, \ldots, u_{d_m+c}$: the reduced score sequence produced by HOLE-MAX;
$O(D)$: the updated frequency vector of the active holes in D;
c: the size of the largest active hole in t.

HOLE-MAX(M, t)

```
01 for i = 0 to d_m                          // Line 01–02: initialization
02     O[i] = M[i]
03 c = d_m                                   // Line 03–14: elimination of the largest hole
04 while M[c] = 0                            // Line 04–05: seeking of the largest hole
05         c = c − 1
06 for j = 0 to d_m                          // Line 06–09: initialization u
07     u_j = t_j
08 for j = d_m + 1 to d_m + c
09     u_j = j
10 for k = 1 to c                            // Line 10–12: maintenance of the lower hole
11     u_{H[1,c]+k} = u_{H[1,c]+c}
12     u_{d_m+k} = u_{d_m}
13 O[c] = O[c] − 1                           Line 13–14: updating of number of active holes
14 O[0] = O[0] − 1
15 return u, c, O                            Line 15: return of the result
```

If $D = \{2, 4, 6, 7\}$ and the input of algorithm HOLE-MAX is the sequence $t(D)$, then HOLE-MAX eliminates the remained hole and we get the score sequence $u = 2^3, 4^3, 6, 7^3$ corresponding to D.

Running time of HOLE-MAX is $\Omega(d_m)$ in all cases.

3.4 Algorithm HOLE-SHIFT

After algorithm HOLE-PAIRS there is at most one i-hole for every i, that is $0 \le O[i] \le 1$ for every $1 \le i \le d_m$. Algorithm HOLE-MAX eliminates the largest hole by appending c scores $d_m + 1, \ldots, d_m + c$ to the input sequence t and reducing these scores to d_m.

Let the player having u_j points be T_j $(1 \le j \le d_m + c)$. The following algorithm HOLE-SHIFT uses the power form $w = w_1^{<e_1>}, \ldots, w_q^{<e_q>}$ of the input sequence u. An element w_k of w is called *point-sender* if its exponent e_k is greater than 1. If w_j and w_k are upper bounds of active holes and $j < k$, then T_k can send $e_k - 1$ points—through the player T_j—to the players in the lower active hole to increase their scores to w_k.

The *hole list* is a pair of vectors consisting of the *begin vector* $b(D) = b_1, \ldots, b_{O[0]}$ and the *length vector* $l(D) = l_1, \ldots, l_{O[0]}$, where b contains the beginning indices of the active holes in increasing order, while l contains the length of the corresponding holes. E.g. if $D = \{2, 5, 9\}$, then $b(d) = 0,\ 3,\ 6$ and $l(D) = 2,\ 2,\ 3$.

Hole-Shift uses also the score sequence $w = w_1^{<e_1>} \ldots w_n^{<e_n>}$ which is the power form of u. If $2 \leq j \leq q,\ e_j > 1$, and $w_j - w_{j-1} = 1$, then $e_j - 1$ players from the players having w_j points can give one pointto any player having smaller index. These $e_j - 1$ points are called *free points*. The following algorithm Hole-Shift extends this idea for the case $w_j - w_{j-1} > 1$.

The next algorithm Hole-Shift finds the largest holes in the investigated score sequence and tries to remove this hole by shifting of the points from the point-sender scores to the scores in the given active hole.

Input. $O(D)$: $O[0]$ is the number of active holes in D, $O[i]$ $(1 \leq i \leq d_m)$ is the number of active i-holes in D;
V: the sum of the sizes of the active holes in D;
$u = u_0, u_1, \ldots, u_{d_m+c}$: the reduced sequence produced by Hole-Max.

Global variables. D: score set;
m: the number of elements of D;
H: hole map of D;
c: the length of largest hole removed by Hole-Shift.

Working variables. $b(D) = b_1, \ldots, , b_{O[0]}$: the begin vector of the active holes of D;
$l(D) = l_1, \ldots, l_{O[0]}$: the length vector of the active holes of D;
$w(D) = w_1^{<e_1>} \ldots w_q^{<e_q>}$: power form of u;
$b(D) = b_1, \ldots, b_{O[0]}$ q: the length of the power form of u;
g: size of the actual largest active hole;
a: the index of the investigated largest active hole;
$h = O[0]$: the number of active holes;
s: difference between two consecutive scores in w;
r: number of points required by the investigated active hole to shift scores in an active hole to theirs upper bound;
$i,\ j,\ k$: cycle variables.

Output. $v = v_1, \ldots, v_{d_m+c}$: the reduced score sequence produced by Hole-Shift;
$P(D)$: the updated frequency vector of D.

Hole-Shift(O, u)

01 **for** $i = 0$ **to** d_m // Line 01–02: initialization

```
02      P[i] = O[i]
03 for i = 1 to d_m + c
04      v[i] = u[i]
06 w_1 = u_1                    // Line 06–14: computation of the power form of u
07 e_1 = 1
08 q = 1
09 for k = 2 to d_m + c
10      if u_k = u_{k-1}
11         e_q = e_q + 1
12      else q = q + 1
13           w_q = u_k
14           e_q = 1
15 if d_1 > 0 and u_1 == 0              // Line 15–24: computation of b, l, and h
16    b_1 = 0
17    l_1 = d_1
18    h = 1
19 else h = 0
20       for i = 1 to m + c - 1
21           if d_{i+1} - d_i ≥ 2
22              h = h + 1
23              b_h = d_i + 1
24              l_h = d_{i+1} - d_i - 1
25 while P[0] > 0                                  // Line 25: while is active hole
26         g = d_m                  // Line 26–31: finding of the largest active hole
27         a = 1
28         for i = 1 to h
29             if l_i > l_a
30                a = i
31                g = l_a
32         r = g(g + 1)/2                // Line 32: number of the required points
33         while r > 0
34                j = 1                  // Line 34–36: finding the starting score;
35                while w_j < b_a + l_a and e_j = 1 and w_j - w_{j-1} > r and j ≤ q
36                      j = j + 1
37                if j > q                                    // Line 37–38: defeat
38                   return                    // 'HOLE-SHIFT is not sufficient'
39                y = ⌊r/(w_j - w_{j-1})⌋
40                r = r - min(e_j - 1, y)(w_j - w_{j-1})
41                e_j = e_j - min(e_j - 1, y)
```

```
42        for i = 1 to l_a
43            w_{j-1} + 1 = w_{j-1} + l_a
44        q = q - l_a
45 return P, w, v
```

The running time of Hole-Shift is $O(md_m)$, since it is determined by the cycles in Lines 33–41, and there are at most m holes and for every hole at most d_m players can give a point.

3.5 Algorithm Fill-All

If $d_m < 10$, then our previous algorithms can reconstruct all possible score set, but if $d_m = 10$, then there are two critical sets, and if $d_m = 11$, then there are three ones. If $d_m = 10$, then the score sets $\{1, 3, 4, 5, 8, 10\}$ and $\{1, 2, 3, 5, 8, 10\}$, while if $d_m = 11$, then the score sets $\{2, 3, 4, 5, 6, 7, 8, 9, 11\}$, $\{1, 2, 3, 5, 7, 11\}$, and $\{1, 2, 3, 4, 5, 6, 7, 8, 11\}$ are unsolvable for them.

The following Fill-All reconstructs these score sets. The basic idea of Fill-All is that at first we add new elements to the starting sequences, sufficient to cover the point requirements of all holes. If we are lucky then the number the additional points is equal to the total point requirement. Otherwise we gradually increase the number of additional scores and try to hide the additional points.

Input. $Q(D)$: $Q[0]$ is the number of active holes in D, $Q[i]$ $(1 \leq i \leq d_m)$ is the number of active i-holes in D.

Global variables. D: score set;
m: the number of elements of D.

Working variables. T: the total point requirement of the holes;
a: the number of added new scores;
$p = a(a-1)2$: the number of additional points;
q: the number of free additional points;
$w(D) = w_1^{<e_1>} \dots w_q^{<e_q>}$: power form of u;
$e = e_1, \dots, e_q$: e_i $(1 \leq i \leq q)$ is the exponent od w_i;
i, j: cycle variables.

Output. $e = e_1, \dots, e_m$: the exponents of the score sequence corresponding to D.

```
Fill-All(m.D)
01 T = 0   // Line 01–06: computation of the point requirement of the holes
02 if d_1 > 0
03    T = d_1
```

```
04 for i = 2 to m
05     if d_i − d_{i−1} > 1
06         T = T + (d_i − d_{i−1})(d_i − d_{i−1} − 1)/2
07 e_1 = 1                          // Line 07–15: computation of the exponents
08 if d_1 > 0
09    e_1 = d_1
10 for j = 2 to m − 1
11     e_j = 1
12     if d_j − d_{j−1} > 1
13         e_j = d_j − d_{j−1}
14 a = min(k | k(k − 1)/2 ≥ T)  // Line 14: first number of additional scores
15 for k = a to d_m     // Line 15–23: testing of the potential score sequences
16     p = k(k − 1)/2                  // Line 16: number of additional points
17     q = p − T                  // Line 17: number of free additional points
18     for i = 1 to m − 1
19         if e_{i+1} − e_i > 1 and d_{i+1} − d_i ≤ q
20             f = min(⌊q/(d_i − d_{i−1}⌋, e_{i−1})
21             e_{i−1} = e_{i−1} − f
22             e_i = e_i + f
23             q = q − f(d_i − d_{i−1})
24             if q == 0
25                 return e
26 print 'algorithms are not sufficient'          // Line 26–27: algorithms
27 stop                                          // could not reconstruct D
```

The running time of FILL-ALL is $O(d_m^2)$ in all cases.

4 Algorithm PREFIX-DELETION

The earlier algorithms can not reconstruct the score sets $\{1, 7, 10\}$ and $\{1, 7, 11\}$.

Part of the earlier described algorithm SHORTENING [5, 6] is the deletion of the leading zero element from a score set, and decreasing of the remaining elements by 1.

Now we generalize this idea. If the score set $D = \{d_1, \ldots, d_m\}$ begins with 1, then according to the theorem of Landau a corresponding score sequence can contain one, two or three 1's. If it contains exactly three 1's, then each of the corresponding players gathered one point in their minitournament, therefore provisionally deleting them we get the smaller score set $D' = \{d_2 - 3, d_3 - 3, \ldots, d_m - 3\}$. In the concrete case, when $D = \{1, 4, 7\}$ BALANCING results the solution $s(D') = 4^6, 7^6$, resulting $s(D) = 1^3, 7^6, 10^6$.

In general case we have to investigate also the cases when the corresponding sequence contains one or two 1's.

It is a natural idea to extend the investigation to the general case $1 \leq d_1 \leq k$, when according to Landau's theorem the corresponding sequences can start with $1 \leq p \leq 2d_1 + 1$ d_1's.

Since in the case $d_m < 12$ it is sufficient to consider the case $k = 1$ and $p = 3$, we present only pseudoprogram for this special case.

In the program we call the procedure SEQUENCE-BASE which handles the results of the previous reconstruction algorithms for all score sets with $d_m < 12$. If we wish to reconstruct a score set, whose largest element is d_m, then the corresponding data base has to contain 2^{d_m-1} score sequence, but the search in it requires only $O(m)$ time.

Its input is the reduced variant of D ($D' = \{d_2 - 3, d_3 - 3, \ldots, d_m - 3\}$), and the output is the score sequence corresponding to D' ($w = w_1, \ldots, w_y$), further its length (y).

Input. $Q(D)$: the updated hole frequency vector;
$v = v_0, v_1, \ldots, v_{d_m+c}$: the reduced sequence produced by HOLE-MAX.

Global variables. D: score set;
m: the number of elements of D;
H: hole map of D-

Working variables. $D' = \{d'_1, \ldots, d'_{m-1}\} = \{d_2, \ldots, d_m\}$: the shortened score set;
$w(D) = w_1^{<e_1>} \ldots w_q^{<e_q>}$: power form of u;
i, j: cycle variables.

Output. $x = x_1, \ldots, x_{y+3}$: the score sequence corresponding to D.

PREFIX-DELETION(Q, v)

```
01 for i = 1 to d_m                          // Line 01–05: initialization
02     R[i] = Q[i]
03     w[i] = v[i]
04 for i = d_m + 1 to d_m + c
05     w[i] = v[i]
06 if m < 4 or d_1 ≠ 1 or d_2 < 3            // Line 06–07: defeat
07     'PREFIX-DELETION is not sufficient'
08 for i = 1 to m − 1
09     d'_i = d_{i+1}
10 SEQUENCE-BASE(D')                 // Line 10: call of SEQUENCE-BASE
11 x_1 = x_2 = x_3 = 1 // Line 11: reconstructed sequence begins with three 1's
12 for j = 4 to y + 3 // Line 12–13: computation of the further elements of x
```

13 $x_j = w_{j-3}$
14 **return** x // Line 14: return of the results

Running time of PREFIX-DELETION is $\Omega(d_m)$ in all cases.

5 The main program

The previous pseudocodes are procedures. These procedures are called by the following main program MAIN.

MAIN(m, D)

01 HOLE-MAP(m, D) // Line 01: construction of the hole map
02 **if** $N[0] = 0$ // Line 02–04: printing the solution
03 **print** 'no hole in the score set'
04 **stop**
05 HOLE-PAIRS(N) // Line 05: construction of the hole pairs
06 **if** $M[0] = 0$ // Line 06–08: printing the solution
07 **print** t
08 **stop**
09 HOLE-MAX(M, t) // Line 09: filling of the longest hole
10 **if** $O[0] = 0$ // Line 10–12: printing the solution
11 **print** u
12 **stop**
13 HOLE-SHIFT(P, u) // Line 13: shifting of the holes
14 **if** $P[0] = 0$ // Line 14–16: printing the solution
15 **print** v
16 **stop**
17 PREFIX-DELETION(Q, v) // Line 17: shifting of the holes
18 **if** $Q[0] = 0$ // Line 18–20: printing the solution
19 **print** w
20 **stop**
21 FILL-ALL(N, w) // Line 21: filling of the holes
22 **if** $q = 0$ // Line 22–24: printing the solution
23 **print** e
24 **stop**
25 PREFIX-DELETION(Q, v) // Line 25: deletion a prefix
26 **if** $Q[0] = 0$ // Line 26–28: printing the solution
27 **print** w
28 **stop**
29 **print** 'the algorithms are not sufficient' // Line 29–30: defeat
30 **stop**

In worst case the running time of MAIN is $O(d_m^2)$.

6 Simulation results

Algorithm BALANCING reconstructs all score sets with $d_m < 6$, but can not reconstruct the score sets $\{1, 3, 6\}$ and $\{1, 2, 3, 5, 6\}$ [6]. For these critical score sets algorithm SHORTENING gives the solutions $1^3, 3, 6^5$ and $1^2, 2, 3^2, 5, 6$. These algorithms can not reconstruct the score sets $1, 2, 3, 5, 7$ and $1, 2, 3, 4, 6, 7$. In the first case algorithm SHIFTENING results a corresponding score sequence $1^2, 2, 3^2, 5, 7^3$, while in the second case it gives $1, 2, 3, 4^2, 6^4, 7$ [5].

n	D	s	Algorithms	DIOPHANTINE
1	$\{2, 4, 5, 6, 7, 8, 9, 10\}$	$3, 2, 1, 1, 1, 1, 2, 2$	M + S	$4, 1, 1, 1, 1, 1, 2, 1$
2	$\{2, 3, 5, 7, 9, 10\}$	$3, 1, 2, 3, 2, 2$	P + M + S	$3, 2, 2, 2, 1, 1$
3	$\{1, 7, 10\}$	$3, 6, 6$	X	same
4	$\{1, 4, 5, 7, 9, 10\}$	$2, 3, 1, 3, 2, 2$	P + M + S	$3, 1, 3, 2, 1, 1$
5	$\{1, 3, 6, 8, 9, 10\}$	$2, 2, 4, 1, 2, 2$	P + M + S	$1, 5, 2, 1, 1, 1$
6	$\{1, 3, 6, 7, 8, 10\}$	$2, 2, 3, 2, 1, 3$	P + M + S	$3, 1, 4, 1, 1, 1$
7	$\{1, 3, 4, 7, 9, 10\}$	$2, 2, 1, 4, 2, 2$	P + M + S	$3, 1, 4, 1, 1, 1$
8	$\{1, 3, 4, 5, 8, 10\}$	$2, 2, 1, 1, 3, 5$	F	$2, 2, 2, 1, 3, 1$
9	$\{1, 2, 5, 7, 9, 10\}$	$2, 1, 3, 3, 2, 2$	P + M + S	$2, 1, 5, 1, 1, 1$
10	$\{1, 2, 5, 6, 7, 8, 10\}$	$2, 1, 3, 1, 1, 2, 3$	P + M	$2, 2, 2, 1, 1, 2, 1$
11	$\{1, 2, 4, 8, 9, 10\}$	$2, 2, 1, 4, 1, 4$	P + M	$1, 3, 2, 4, 1, 1$
12	$\{1, 2, 4, 6, 9, 10\}$	$2, 1, 2, 2, 3, 4$	P + M	$2, 1, 2, 4, 1, 1$
13	$\{1, 2, 3, 7, 8, 10\}$	$2, 1, 1, 4, 2, 4$	P + M	$1, 1, 4, 2, 2, 1$
14	$\{1, 2, 3, 5, 8, 10\}$	$2, 1, 1, 2, 3, 5$	F	$1, 2, 2, 2, 3, 1$
15	$\{1, 2, 3, 5, 8, 9, 10\}$	$2, 1, 2, 1, 3, 1, 3$	P + M	$1, 2, 1, 4, 1, 1, 1$
16	$\{1, 2, 3, 5, 7, 10\}$	$2, 1, 1, 3, 1, 6$	A	$2, 1, 1, 2, 4, 1$
17	$\{1, 2, 3, 5, 6, 9, 10\}$	$2, 1, 2, 1, 1, 3, 3$	P + M	$2, 1, 1, 1, 4, 1, 1$
18	$\{1, 2, 3, 4, 6, 7, 10\}$	$2, 1, 1, 2, 1, 1, 5$	P + M	$2, 1, 1, 1, 1, 4, 1$

Table 2: Reconstruction results of the critical score sets ending with $d_m = 10$

BALANCING, SHORTENING, and SHIFTENING reconstruct the majority of score sets with $d_m < 9$. Exceptions are the sets $\{1, 2, 3, 5, 7, 8\}$ and $\{1, 2, 3, 4, 6, 7, 8\}$. In the first case HOLE-MAX gives a corresponding score sequence $1^2, 2, 3^2, 5^2, 7, 8^2$, while in the second case algorithm HOLE-PAIRS presents the solution $1^2, 2, 3^2, 4, 6, 7, 8$.

If $d_m = 9$, then for the first three algorithms the critical sets are $\{2, 4, 5, 6, 7, 8, 9\}$, $\{1, 2, 5, 6, 7, 8\ 9\}$ and $\{1, 2, 4, 7, 8, 9\}$. HOLE-MAX solves these problems:

corresponding sequences are in the first case $2^3, 4^2, 5, 6, 7, 8, 9^2$, in the second case $1^2, 2, 5^2, 6, 7, 8, 9^2$, and in the third case $1^2, 2^2, 4, 7^3, 8, 9^3$.

n	D	s	Algorithms	DIOPHANTINE
1	$\{3,5,6,7,8,9,10,11\}$	$4,2,1,1,1,1,2,3$	M + S	$5,1,1,2,1,1,1,1$
2	$\{3,4,5,6,8,9,10,11\}$	$4,1,1,1,2,1,2,3$	M + S	$5,1,1,1,1,1,2,1$
3	$\{3,4,5,6,7,10,11\}$	$4,1,1,1,1,6,1$	M + S	$4,2,1,1,2,1,1$
4	$\{2,4,5,6,7,8,9,10,11\}$	$3,2,1,1,1,1,1,2,2$	M + S	$4,1,1,1,1,1,1,2,1$
5	$\{2,3,5,9,10,11\}$	$3,1,2,5,3,1$	M + S	$1,3,5,1,1,1$
6	$\{2,3,5,7,9,10,11\}$	$3,1,2,2,2,1,4$	P + M + S	$3,2,2,2,1,1,1$
7	$\{2,3,4,8,9,11\}$	$3,1,1,4,1,6$	M + S	$3,1,3,2,2,1$
8	$\{2,3,4,5,6,8,9,10,11\}$	$3,1,1,1,1,2,1,2,2$	M + S	$3,1,2,1,1,1,1,1,1$
9	$\{2,3,4,5,6,7,8,10,11\}$	$3,1,1,1,1,1,1,3,2$	M + S	$3,1,1,1,2,1,1,1,1$
10	$\{2,3,4,5,6,7,8,9,11\}$	$3,1,1,1,1,1,1,2,3$	F	$3,1,1,1,1,2,1,1,1$
11	$\{1,7,11\}$	$3,2,14$	X	$3,1,3,2,1,1,1$
12	$\{1,4,5,7,9,10,11\}$	$2,3,1,3,1,2,2$	P + M + S	$3,1,3,2,1,1,1$
13	$\{1,3,5,6,7,10,11\}$	$2,2,2,1,1,3,4$	P + M + S	$3,1,3,2,1,1,1$
14	$\{1,3,4,5,8,10,11\}$	$2,2,1,1,4,2,2$	P + M + S	$2,2,2,1,3,1,1$
15	$\{1,2,6,8,9,11\}$	$2,1,4,3,1,4$	P + M + S	$2,2,4,2,1,1$
16	$\{1,2,5,9,10,11\}$	$2,1,3,4,1,5$	P + M + S	$2,2,3,4,1,1$
17	$\{1,2,4,8,10,11\}$	$2,1,2,5,2,3$	P + M + S	$1,2,4,3,1,1$
18	$\{1,2,4,8,9,11\}$	$2,1,2,5,1,4$	P + M + S	$1,2,4,2,2,1$
19	$\{1,2,4,7,9,10,11\}$	$2,1,2,4,1,2,2$	P + M + S	$1,3,2,3,1,1,1$
20	$\{1,2,4,6,9,10,11\}$	$2,1,3,1,3,2,2$	P + M + S	$2,1,2,4,1,1,1$
21	$\{1,2,4,5,7,10,11\}$	$2,1,2,1,2,3,4$	P + M + S	$2,1,2,1,4,1,1$
22	$\{1,2,3,7,8,9,10,11\}$	$2,1,1,4,1,1,2,3$	M + S	$2,1,2,3,1,2,1,1$
23	$\{1,2,3,6,10,11\}$	$2,1,1,3,7,1$	M + S	$1,1,2,6,1,1$
24	$\{1,2,3,5,8,10,11\}$	$2,1,1,2,4,2,2$	P + M + S	$1,2,1,4,1,1,1,1$
25	$\{1,2,3,5,7,11\}$	$2,1,1,2,2,7$	F	$1,2,1,1,6,1$
26	$\{1,2,3,5,7,10,11\}$	$2,1,1,2,2,3,4$	P + M + S	$2,1,1,2,2,3,4$
27	$\{1,2,3,5,7,8,11\}$	$2,1,1,2,2,1,6$	P + M + S	$2,1,1,2,2,3,1$
28	$\{1,2,3,5,6,9,11\}$	$2,1,1,2,1,3,5$	P + M + S	$2,1,1,3,1,3,1$
29	$\{1,2,3,5,6,8,11\}$	$2,1,1,2,1,2,6$	P + M + S	$2,1,1,2,1,4,1$
30	$\{1,2,3,4,9,10,11\}$	$2,1,1,1,5,2,4$	M + S	$1,1,2,3,4,1,1$
31	$\{1,2,3,4,5,9,10,11\}$	$2,1,1,1,1,4,2,3$	P + M	$2,1,1,1,2,4,1,1$
32	$\{1,2,3,4,5,6,7,8,11\}$	$2,1,1,1,1,1,1,3,4$	F	$1,1,2,1,1,1,1,3,1$

Table 3: Reconstruction results of the critical score sets ending with $d_m = 11$

If $d_m = 10$, then there are 18 sequences which are not reconstructable if we use only the algorithms BALANCING, SHORTENING, and SHIFTENING. Table 2

contains a possible reconstruction of these sequences. The used algorithms are P = Hole-Pairs, M = Hole-Max, S = Hole-Shift, X = Prefix-Shift, and F = Fit-All. The table contains also a shortest solution found by the brute force algorithm Diophantine described in [6]. Diophantine uses the algorithm described by Knuth [9, page 392].

If $d_m = 11$, then there are 72 sequences which are not reconstructable if we use only the algorithms Balancing, Shortening, and Shiftening. The majority of these sets can be reconstructed adding only P = Hole-Pairs and M = Hole-Max to the three basic algorithms. Table 3 is similar to the previous Table 2, but it contains only those examples (32 sets), whose reconstruction requires at least one of the algorithm Hole-Shift, Prefix-Shift, and Fit-All.

It is interesting to analyze the length of the critical sets. According to Theorem 7 for the length n of the score sequences corresponding to a score set $D = \{d_1, d_2, \ldots, d_m\}$ hold the bounds $\max(d_m + 1, 2d_1 + 2) \leq n \leq 2d_m$, and these bounds are sharp.

According to the lower bound in the case $d_m = 10$ we get $n \geq 11$. The data represented in Table 2 show, that Dophantine in 17 cases reaches this minimal length (the exception is $D = \{2, 3, 5, 7, 9, 10\}$) the approximate algorithms generate longer solutions in all cases.

If $d_m = 11$, then in the case of the critical sets the lower bound is $n \geq 12$. In the majority of cases Diophantine founds solutions of length 12, while the approximate algorithms never find a solution of length 12. The exact algorithm Diophantine in 29 cases found shorter sequence than the polynomial algorithms.

7 Summary

Checking all relevant score sets by polynomial time approximate algorithms we proved Theorem 3 for score sets whose maximal element is less than 12. Our proof is constructive, since we generated score sequences corresponding to the investigated score sets.

Acknowledgement

The author thanks the unknown referee for the proposed useful corrections, further PhD student of Eötvös Loránd University János Elek for making the computer experiments with algorithm Diophantine

References

[1] G. Chartrand, L. Lesniak, J. Roberts, Degree sets for digraphs, *Periodica Math. Hung.*, **7** (1976) 77–85. ⇒212

[2] T. H. Cormen, Ch. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms* (third edition), The MIT Press/McGraw Hill, Cambridge/New York, 2009. ⇒216

[3] J. L. Gross, J. Yellen, P. Zhang, *Handbook of Graph Theory*, CRC Press, Boca Raton, FL, 2013. ⇒210

[4] M. Hager. On score sets for tournaments, *Discrete Math.*, **58** (1986) 25–34. ⇒ 210, 212

[5] A. Iványi, J. Elek, Degree sets of tournaments, *Studia Univ. Babeş-Bolyai, Informatica*, **59** (2014) 150–164. ⇒210, 215, 223, 226

[6] A. Iványi, L. Lucz, T. Matuszka, G. Gombos, Score sets in multitournaments, I. Mathematical results, *Annales Univ. Sci. Budapest., Rolando Eötvös Nom., Sectio Comp.*, **40** (2013) 307–320. ⇒210, 214, 215, 223, 226, 228

[7] A. Iványi, B. M. Phong, On the unicity of the score sets of multitournaments, in: *Fifth Conference on Mathematics and Computer Science* (Debrecen, June 9–12, 2004), University of Debrecen, 2006, 10 pages. ⇒213

[8] A. Iványi, S. Pirzada, Comparison based ranking, in: ed. A. Iványi, *Algorithms of Informatics*, Vol. 3, mondAt, Vác, 2013, 1209–1258. ⇒211

[9] D. E. Knuth, *The Art of Computer Programming*, Volume 4A. Addison Wesley, Upper Saddle River, NJ, 2011. ⇒228

[10] H. H. Landau, On dominance relations and the structure of animal societies. III., *Bull. Math. Biophysics*, **15** (1953) 143–148. ⇒211, 212

[11] Q. Li, Some results and problems in graph theory, *New York Academy of Science*, **576** (1989) 336–343. ⇒214

[12] V. Petrović, On bipartite score sets, *Zbornik Radova Prirodno-matematičkog Fakulteta Ser. Mat., Universitat u Novom Sadu*, **13** (1983) 297–303. ⇒213

[13] S. Pirzada, A. Iványi, M. A. Khan, Score sets and kings, in ed. A. Iványi, *Algorithms of Informatics*, mondAt, Vác, 2013, 1337–1389. ⇒214

[14] S. Pirzada, T. A. Naikoo, On score sets in tournaments, *Vietnam J. Math.*, **34** (2006) 157–161. ⇒212

[15] K. B. Reid, Score sets for tournaments, *Congressus Numer.*, **21** (1978) 607–618. ⇒210, 212

[16] K. B. Reid, Tournaments: Scores, kings, generalizations and special topics, *Congressus Numer.*, **115** (1996) 171–211. ⇒211, 212, 214

[17] K. Wayland, Bipartite score sets, *Canadian Math. Bull.*, **26** (1983) 273–279. ⇒ 213

[18] T. X. Yao, On Reid conjecture of score sets for tournaments. *Chinese Science Bull.*, **34** (1989) 804–808. ⇒210, 212, 213, 214

Received: June 28, 2014 • Revised: September 29, 2014

Acta Univ. Sapientiae, Informatica, **6,** 2 (2014) 230–251

Statistical complexity and generalized number system

Ágnes FÜLÖP
Loránd Eötvös University, Budapest
Faculty of Informatics
email: fulop@caesar.elte.hu

Abstract. We apply the concept of statistical complexity to understand the dynamical behaviour of the time series by the probability distribution. This quantity allows to distinguish between the random, regular motion and the structural complexity in finite systems. We determined the numerical approximation of the statistical complexity of the Lozi attractor and the generalized number system.

1 Introduction

In this article we discuss the statistical complexity [25], which provides a description of a finite measured sequence to specify more complicated dynamical structures. It was extended on wide range of sciences [23, 1, 15].

The idea of complexity was introduced in different forms. We mention some of them: algorithmic complexity (Kolmogorov) [22], amount of information about the past required to predict the future (Crutchfield, Young) [7], complexity of finite sequence (Lempel, Ziv) [24].

There are more questions in the real word, where the statistical complexity is applied. We referred some example as more realistic gas of particles [4, 5], the effective method in the hydrological systems [12], the statistical features

Computing Classification System 1998: J.2
Mathematics Subject Classification 2010: 68U20
Key words and phrases: statistical complexity, Shannon entropy, number theory, Lozi map, chaos, strange attractor

of the behaviour for DNA [36], the earthquake magnitude time series [26], chaotic motion in Logistic map [13], biological application [32].

The notion of statistical complexity is defined by the concept of the information theory i.e. the entropy and the disequilibrium. The Shannon entropy specifies the gain of the information storage in the disordered system and the disequilibrium characterizes the amount of distance from the equiprobability distribution.

While the entropy allows to describe the direction of flow and the Lyapunov exponent characterizes the chaotic orbits, we can not specify the whole strange attractor in the finite dimensional space. The statistical complexity enables to determine the inner structure of the dynamical system and the location of the strange attractor is shown in the parameter space. It is compared with the complexity of generalized number system, which contains periodic paths.

The numerical simulation plays important role in the chaotic motion, because there are numerous problems, which can not be solved analytically.

We calculated the spectrum of statistical complexity of the two dimensional piecewise Lozi map, which is not differentiable and contains chaotic region. It is compared with the statistical complexity of the finite approximation of the set B_γ on the lattice.

The structure of the article is the following:

Section 2 contains the introduction of the statistical complexity to consider different measurement. The chaotic motion is described in Section 3. The definition of the generalized number system and the fundamental set are investigated in Section 4. The numerical results are displayed in Section 5.

2 Complexity

In this section we introduce the statistical complexity following the effective entropy by P. Grassberger [16] and the main concept by R. López-Ruiz, H. L. Manchini, X. Calbet (1995) [25, 28, 2]. This definition was extended to the generalized statistical complexity measures by M. T. Martin et al. (2006) [29] for different types of entropy and disequilibrium.

We investigate the notation of a measured sequence [14]. Let us denote $y_1, \dots, y_n$ the time series, where y_i means the measurement of the quantity y at time $t_i = t_0 + i\Delta t$, and the time interval $\Delta t > 0 \in \mathbb{R}$.

The $\underline{x}^{(n)}$ denotes the trajectory of length n in $\mathbb{R}^d$, which is a time sequent of the measurement. The kth point of the path of length n is denoted by $x_k^{(n)} (k = 1, \dots, n)$. We will study the series of $x_1^{(n)}, x_2^{(n)}, \dots, x_n^{(n)}$ as a time

sequent. The set K contains the points of some orbits $x_k^{(n)}$ $(k = 1, \ldots, n)$.

We will apply the notation of symbolic dynamics, because the idea of complexity is more general concept than this application.

There are M different values of measurements. Each path $\underline{x}^{(n)}$ for a finite n corresponds to symbolic sequence $O^{(n)} = (o_1, o_2, \ldots, o_n)$, where the symbol o_k $(k = 1, \ldots, n)$ is chosen from the set $\{1, \ldots, M\}$. Let us consider a time series of length N', where $N' >> n$. Then a given sequent $O^{(n)}$ appears with probability $P(O^{(n)})$ along this long series of length N'. The unit of the time interval Δt equals to a constant in this description.

2.1 Statistical complexity

The statistical complexity is based on the probabilistic description of a finite time series, which provides a statistical approximation of the time sequent. We introduce a measure of statistical complexity, which depends on the finite discrete probability distribution.

We define N-system. Let us assume that there are N different symbol sequences of length n $\{O_1^{(n)}, \ldots O_N^{(n)}\}$, which correspond to the set of discrete probability distribution $P \equiv \{p_1, \ldots p_N\}$, where $p_i := P(O_i^{(n)})$ $(\sum_{i=1}^{N} p_i = 1)$ and $p_i > 0$ for all i.

The first we have to consider the entropy i.e. some measure of the amount of information stored $\mathcal{H}$ and the disequilibrium D, which corresponds to the distance from the appropriate probability distribution to the equilibrium.

2.1.1 Measure of entropy and disequilibrium

The information measure was introduced as a quantity, which depends on a probability distribution $P = \{p_j, j = 1, \ldots, N\}$. In the information theory the entropy was investigated as a unique function, which corresponds to the measure of the uncertainty. The statistical complexity is defined by the Shannon entropy [33], therefore we will investigate this form in N-system:

$$\mathcal{H} = -\sum_{i=1}^{N} p_i \log p_i. \tag{1}$$

This quantity $\mathcal{H} \sim 0$, if the symbol sequence $O_c^{(n)}$ would be almost probable $(p_c \sim 1)$ and other $O_i^{(n)}$ would be very improbable $(p_c \sim 0)$. $\mathcal{H}_{max}$ notes the maximal value of $\mathcal{H}$, which reaches the uniform probability distribution $p_e = \{1/N, 1/N, \ldots, 1/N\}$ i.e. the equiprobability symbol sequence

$O_e^{(n)}$ characterizes the maximum of information for the N systems. The normalized quantity $\overline{\mathcal{H}}$ is the following $\overline{\mathcal{H}} = \mathcal{H}/\mathcal{H}_{max}$, then $0 \leq \overline{\mathcal{H}} \leq 1$, where $\mathcal{H}_{max} = \log N$.

If the system is out of equilibrium, the entropy $\mathcal{H}$ can be expanded around this maximum $\mathcal{H}_{max}$:

$$\mathcal{H}(p_1, p_2, \ldots, p_N) = \log N - \frac{N}{2} \sum_{i=1}^{N} \left(p_i - \frac{1}{N} \right)^2 + \cdots = \mathcal{H}_{max} - \frac{N}{2} D + \cdots ,$$

where the quantity $D = \sum_i (p_i - 1/N)^2$ denotes the disequilibrium.

Let us multiply this expansion by $\mathcal{H}$ in the following:

$$\mathcal{H}^2 = \mathcal{H} \cdot \mathcal{H}_{max} - \frac{N}{2} \mathcal{H} \cdot D + g(N, p_i),$$

where $g(N, p_i)$ contains the entropy multiplied by the rest of Taylor expansion terms, which presents the form $\frac{1}{N} \sum_i (N p_i - 1)^m$ with $m > 2$. If we rename $C = \mathcal{H} \cdot D$:

$$C = \frac{2}{N} \cdot \mathcal{H} \cdot (\mathcal{H}_{max} - \mathcal{H}) + 2g/N.$$

This expression shows the connection among the entropy, disequilibrium and complexity.

Let us define the function of disequilibrium D on the probability distribution $\{p_j : j = 1 \ldots, N\}$ in N-system.

The Euclidean measure have been used i.e. the quadratic distances from the probability distribution of each symbol sequences $P(O^{(n)})$ to the equiprobability $P(O_e^{(n)})$:

$$D = \sum_{i=1}^{N} (p_i - p_e)^2, \quad \text{where} \quad p_e = \frac{1}{N}. \tag{2}$$

The maximum disequilibrium is reached for dominant symbol series $O_c^{(n)}$ with $p_c \sim 1$ and $D_c \to 1$ for N is increasing, while the disequilibrium vanishes. i.e. $D \sim 0$ for $p_i \sim 1/N$. For any other probability distribution D will have value between these two extrema. The normalized disequilibrium is $\overline{D} = D \cdot D_e$, where D_e equals to $\frac{N}{N-1}$.

2.1.2 Measure of statistical complexity

The family of the complexity measure contains the product of disorder $\mathcal{H}$ and disequilibrium D for different type of the time series. This is interplay

between the information stored in the system and its disequilibrium. We will define the measure of statistical complexity C [25] in the following expression in N-system:

$$C = \mathcal{H} \cdot D = -\left(\sum_{i=1}^{N} p_i \log p_i\right)\left(\sum_{i=1}^{N}\left(p_i - \frac{1}{N}\right)^2\right). \tag{3}$$

This quantity is larger or equals to zero, i.e. $C \geq 0$. The normalized value of C is $\overline{C} = \overline{\mathcal{H}} \cdot \overline{D} = (\mathcal{H}/\log N)(D \cdot (N/(N-1)))$.

The definition of statistical complexity measure can be divided into three categories: (i) it is growing with increasing entropy, (ii) it is a convex function and equals to minimum at the $\mathcal{H} = 0$ total order and $\mathcal{H} = 1$ total disperse state and a maximum at transition level, where the probability distribution is p_e, (iii) it is decreasing with increasing entropy [29]. We will study the second case in this article.

Because the statistical complexity was defined in a finite system, therefore it depends on the scale. At each scale of observation a new set of accessible symbol sequence $O^{(n)}$ appears with its corresponding probability distribution $P(O^{(n)})$ therefore the complexity changes.

The complexity C is finite and limiting but it is not necessary a unique function of $\mathcal{H}$, there exists a range of value between a minimal value C_{min} and a maximal value C_{max}. Thus, evaluating the complexity provides more important additional information regarding the peculiarities of a probability distribution.

Two basic incidents are distinguished in the relationship between the entropy $\mathcal{H}$ and complexity C. On the one hand the time sequence can be found in any of its accessible symbol sequence $O^{(n)}$ with the same probability. All of them contribute in equal measure to the information stored. On the other hand, minimal information is enough to describe the system considering some symmetries properties and distance.

It should be noticed that different measures for complexity employed for different probability distribution. Tsallis suggested a generalisation of the Shannon-Boltzmann-Gibbs entropic measure [34] and A. Rényi introduced a definition of entropy for discrete probability distribution in 1950s [31].

The disequilibrium can be extended for various probability distribution. Jensen-Kullback divergence was investigated for relative entropies [29] and Wootters statistical distance was applied for two probability distributions [37], which can be used in the quantum mechanic.

In Section 5 we study the statistical complexity on the system out of equilibrium by numerical approximation.

3 Chaotic motion

In this section we introduce the Lyapunov exponent [8], which characterises the chaotic behaviour of dynamical systems. This quantity is expressed by probability density along the ergodic trajectory [6].

Let us introduce a map $f : \mathbb{R} \to \mathbb{R}$, where $x_{t+1} = f(x_t)$, which ($t = 0, 1, 2, 3 \dots$) corresponds to the trajectory $x_0, x_1, x_2 \dots$ at the time series $t = 0, 1, 2 \dots$. This map contains stable or unstable fixpoints x^* and it is differentiable near to the fixpoints x^*. We can expanded the map f around the fixpoint x^* upto linear expression:

$$\frac{|x_{t+1} - x^*|}{|x_t - x^*|} \approx \left| \frac{\partial f(x)}{\partial x} \right|_{x^*}.$$

The solution of this equation is written by the next form $|x_t - x| \approx \left| \frac{\partial f(x)}{\partial x} \right|_{x^*}^t = ce^{\lambda t}$, where $c \in \mathbb{R}$ is a constant value. If $\lambda < 0$, then the fixpoint x^* becomes stable. If $\lambda > 0$, then the fixpoint x^* turns into unstable, and in the case of $\lambda = 0$ the fixpoint x^* is marginal stable.

The $f(x)$ map was extended upto first order, therefore we can not determinate the whole trapping region of the fixpoint x^*.

As a consequence we defined the Lyapunov exponent:

$$\overline{\lambda} = \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n} \ln \left| \frac{\partial f(x)}{\partial x} \right|_{x_t},$$

along the trajectories, where these orbits converge to the fixpoint x_*.

The Lyapunov exponent $\overline{\lambda}$ depends on the initial condition of the trajectories, therefore we denote $x_n(x_0) \equiv f^n(x_0)$ and it can be used as $\overline{\lambda} = \lim_{n\to\infty} \frac{1}{n} \ln \left| \frac{\partial f^n(x_0)}{\partial x_0} \right|$.

If $\overline{\lambda} > 0$, then the motion is chaotic. The Lyapunov exponent is sensitive to the initial condition. The d distance of points of the trajectories increases exponentially, where the x_0 and $x_0 + \varepsilon$ points ($\varepsilon > 0$, $\varepsilon \in \mathbb{R}$) were near to each other around the unstable fixpoint x^* at the first time step ($t = 0$).

Then the form of Lyapunov exponent is the following:

$$\overline{\lambda} \approx \lim_{n\to\infty} \frac{1}{n} \ln \left| \frac{f^n(x_0 + \varepsilon) - f^n(x_0)}{\varepsilon} \right|,$$

where $\varepsilon \to 0$. This expression can be written $|f^n(x_0+\varepsilon) - f^n(x_0)| \approx |\varepsilon| e^{n\overline{\lambda}}$.

The ergodicity plays important role in the chaotic motion.

We consider the ergodic paths, where $h(x_t)$ means an absolute continuous integrable function ($h : \mathbb{R} \to \mathbb{R}$) along the trajectory. At almost all initial conditions x_0 the average of the function $h(x_t)$ is introduced as following:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n} h(x_t) = \int_0^1 h(x)P(x)dx \equiv \int_0^1 h(x)d\mu(x),$$

where $\mu(x)$ means some invariant measure, $\frac{\mu(x)}{dx} = P(x)$ is a probability density and the exact form is given by the map $f(x)$. This expression is measure invariant therefore:

$$\int_0^1 h(x)d\mu(x) = \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n} h(x_t) = \lim_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n} h(f(x_t)) = \int_0^1 h(f(x))d\mu(x).$$

The Lyapunov exponent is defined by the $P(x)$ probability density along the ergodic orbit:

$$\overline{\lambda} = \int_0^1 P(x) \ln |f'(x)| dx.$$

In the chaotic motion the strange attractor S plays similar role in the case of aperiodic motion as the attractor in the periodic motion. Let us choose that orbits, which are characterized by probability distribution $P(x)$. The set F contains the initial points of these trajectories and the set L is defined following $L = \{x|P(x) > 0\}$. Then we take the union of the set L with its closure. If the Lyapunov exponent is larger then zero on this set, than we gain the strange attractor S and the trapping region corresponds to the closure of the set F. The motion comes on this set S after finite iteration and the points of trajectory follow one to the other randomly.

We study the time series of the measurement of length n $\underline{x}^{(n)}$ and the error of initial point $x_0^{(n)}$ is $\varepsilon > 0$. Let us suppose $\overline{\lambda} > 0$, then the $\varepsilon' \approx |\varepsilon| e^{k\overline{\lambda}}$ at the kth element of sequence $x_k^{(n)}$ i.e. the error of time series is increasing as k becomes larger, we can not predict the value $x_{k+1}^{(n)}$ more exactly than $|\varepsilon'|e^{\overline{\lambda}}$. Then the value of the ergodic time series becomes unpredictable. Therefore we apply the statistical complexity to determine the location of the strange attractor in the parameter space.

On a computer the study of the dynamical system appears in a finite m dimensional space, then we have an m dimensional signal $z(t)$. In a physical

experiment a single scalar variable $u(t)$ is monitored for a system, which has an infinite dimensional phase space $\mathcal{M}''$.

In this case we restrict our attention to the dynamics on a finite dimension attractor A in the space $\mathcal{M}''$. Otherwise we generate several different scalar values $z_i(t)$ $i = 1 \ldots, \mathcal{N}$ from the original $u(t)$.

The only way to obtain several measurements from a single one is to use time delays. We choose different delays $\mathcal{T}_1 = 0, \mathcal{T}_2, \ldots, \mathcal{T}_{\mathcal{N}}$ and it can be written $z_k(t) = u(t + \mathcal{T}_k)$. We can generate an $\mathcal{N}$ dimensional signal in this manner.

The successive time derivative of the signal is formed: $z_{k+1}(t) = d^k z_1(t)/dt^k$, but the numerical differentiations produce high level of error. As usually we should measure several experiment signal produce more exactly values.

The reconstruction of the dynamical process provides an $\mathcal{N}$ dimensional image πA of an attractor A, which has finite Haussdorff dimension and it is embedded in an infinite dimensional space $\mathcal{M}''$. The projection will look different according to the choice of variables. Taken proved that theorem (1981): If we use enough variables, typically about twice the Haussdorff dimension, we shall generally get a good projection [8]. This method produces the attractor, but the realisation is difficult.

In the next section we introduce the generalized number system and the set B_γ, which is a fractal structure [11], but there are periodic motions on it, therefore it is not chaotic.

4 Generalized number system

I. Kátai investigated the concept of generalized number systems [18] in the 1970s. This idea is developed on different algebraic structures expansively as real quadratic fields [10], imaginary quadratic fields [19].

We introduce the basic definition in this section according to the literature [20, 11].

Let $\mathbb{Z}_k$ be a ring of integer vectorial in $\mathbb{R}_k$ $(k \geq 1)$. A $k \times k$ type matrix with integer elements is noted by M, where $\mathcal{L} = M\mathbb{Z}_k$. Then $\mathcal{L}$ is a subgroup in $\mathbb{Z}_k$, $O(\mathbb{Z}_k/\mathcal{L}) = t$ the order, where $t = |\det M|$. We introduce the digit set $\mathcal{A}$ in the following. Let $\mathcal{A} = \{a_0 = 0, a_1, \ldots, a_{t-1}\}$ mean a complete set of the representation of the residue classes mod M for $(t \geq 2)$. We define the number system $(\mathcal{A}, M)$, if each $n \in \mathbb{Z}_k$ can be written by uniqueness expansion form:

$$n = a_0 + Ma_1 + \cdots + M^{h-1}a_{h-1}, \qquad a_j \in \mathcal{A} \quad \text{for} \quad h > 0. \tag{4}$$

We define the function $J : \mathbb{Z}_k \to \mathbb{Z}_k$, where the ring of integer vectorial is

mapped to onto itself. There exists a unique $a_0 \in \mathcal{A}$ and $n_1 \in \mathbb{Z}_k$ such that $n = a_0 + Mn_1$ for every $n \in \mathbb{Z}_k$, i.e. let $J(n) = n_1$ be.

The set H plays fundamental role in the number system $(\mathcal{A}, M)$. Let us define H in the following:

$$H = \left\{ z | z = \sum_{i=1}^{\infty} M^{-i} a_i, \quad a_i \in \mathcal{A} \right\}. \tag{5}$$

The set H is compact. If $(\mathcal{A}, M)$ is a number system, then

$$\cup_{n \in \mathbb{Z}_k} (H + n) = \mathbb{R}. \tag{6}$$

We say that $(\mathcal{A}, M)$ is just touching covering system (JTCS), if every $n_1, n_2 \in \mathbb{Z}_k$, $n_1 \neq n_2$:

$$\lambda(H + n_1 \cap H + n_2) = 0, \tag{7}$$

where λ is the Lebesques measure. Let $\mathcal{B} = \mathcal{A} - \mathcal{A} = \{a_u - a_v | a_u, a_v \in \mathcal{A}\}$ hold. We introduce a set S following. That element $\gamma \in \mathbb{Z}_k$ is contained in the set S, which fulfils $\gamma \neq 0$ and satisfies the following expression:

$$H \cap H + \gamma \neq \emptyset. \tag{8}$$

This set is assigned by B_γ and

$$B = \cup_\gamma B_\gamma. \tag{9}$$

More detailed, if $z \in B_\gamma$, then z can be extended by this form $z = \sum_{i=1}^{\infty} M^{-i} a_i = \gamma + \sum_{i=1}^{\infty} M^{-i} a'_i$, where $a_i, a'_i \in \mathcal{A}$ and $\gamma = \sum_{i=1}^{\infty} M^{-i} e_i$, where $e_i = a_i - a'_i \in \mathcal{B}$.

We will determinate the statistical complexity of the finite approximation set B_γ on the ring of quadratic integers at a given algebraic number fields in Section 4.

In the next section we investigate a walk along the finite transition graph, which is analogous to dynamical system.

4.1 Transition graph of number system

Let us produce a finite directed labeled graph $G(S)$ according to the article [35], where the function $Q : S \to S$ ($S \subseteq \mathbb{Z}_k \setminus \{0\}$) means a walk P along the transition graph.

The elements of the set S correspond to vertices of the graph $G(S)$ and the edges can be defined as follows:

There exists a directed edge from γ_k to γ_{k+1} and it is labeled by $\delta \in \mathcal{B}$, if $Q(\gamma_k) = \gamma_{k+1}$, i.e. $\gamma_{k+1} = \gamma_k M - \delta$, $k \in \mathbb{N}$.

Let us construct the graph $G(S)$:

The elements γ_i of the set S can be computed by the following way:

∘ Initial condition: If the first element γ_1 satisfies $Q(\gamma_1) = 0$, then this element $\gamma_1 = 0$.

∘ If $\gamma_2 \in S$ and there exists en edge, which goes from γ_1 to γ_2, than $\gamma_1 \in S$.

∘ Each element of the set S fulfils the next condition. That values of outgoing and ingoing degree of the vertices γ_k are larger than zero i.e. $\deg^+(\gamma_k) > 0, \deg^-(\gamma_k) > 0$.

∘ The set S contains periodic points: $Q^r(\gamma_1) = \gamma_1$.

The loop of the algorithm:

∘ From every $\gamma_1 \in S$ an edge fits to γ_2, if $\gamma_2 = \gamma_1 M - \delta$ for $\delta \in \mathcal{B}$.

∘ If $\deg^+(\gamma_1) = 0$, than erase the vertices γ_1 and all edges, which directed to γ_1.

Repeat these finite steps, until we delete all those nodes from which no edge goes out i.e. $\deg^-(\gamma_k) = 0$ or ends, remove all coinciding edges as well.

We obtain the directed transition graph $G(S)$.

Let $P := \gamma_1 \xrightarrow{\delta_1} \gamma_2 \xrightarrow{\delta_2} \gamma_3 \ldots, \gamma_{r-1} \xrightarrow{\delta_{r-1}} \gamma_r$ be a walk of length r on the graph $G(S)$, it is labeled by $(\delta_1, \delta_2, \ldots, \delta_{r-1})$, i.e. for finite orbit of length r $Q^{(r)}(\gamma_1) = \gamma_r$.

Because $z \in B_\gamma$ was defined by expression (8):

$$z = \sum_{i=1}^{\infty} M^{-i} f_i, \quad f_i \in \mathcal{A}. \tag{10}$$

Every infinitely long walk P is assigned by the series of labels: $\delta_1, \delta_2, \ldots, \delta_{r-1}, \ldots$, where $\delta_i = f_i - f_i'$ with appropriate $f_i, f_i' \in \mathcal{A}$. Therefore $z \in B_\gamma$ can be labeled by the sequence $f_1, f_2 \ldots$.

5 Numerical results

We introduce an appropriate measure of the time series and the statistical complexity on lattice.

5.1 Probability measure and euclidean distance on lattice

Let us consider a lattice C' in the $\mathbb{R}^d$ with linear size ε ($\varepsilon > 0,\ \varepsilon \in \mathbb{R}$), where C'_j assigns the elementary box of lattice C' in the following way:

$$C' = \cup_j C'_j, \quad \text{and} \quad C'_j \cap C'_i = \emptyset, \quad \text{where} \quad j, i \in \{0, \ldots, N''^d - 1\}, \tag{11}$$

where the set of C'_j is a partition of $[0, N''\varepsilon[^d \subset \mathbb{R}^d$. Let K be a compact set, which contains the measured value y:

$$T_j = K \cap C'_j \neq \emptyset, \quad \text{where} \quad j = 1, \ldots, M' \quad \text{and} \quad K \cap C'_j = \emptyset \quad \text{for any other} \quad C'_j.$$

$$T = \cup_{j=1}^{M'} T_j, \quad \text{where} \quad T_j \cap T_i = \emptyset, \quad i \neq j.$$

The compact set $K \subset \mathbb{R}^d$ consists of every points $x_k^{(n)}$ ($k = 1, \ldots, n$) of some orbit of length n. Each path corresponds to the series of the indices j for which $x_k^{(n)} \in T_j$ and $j \in \{1, \ldots, M'\}$.

The lattice size ε and the unit of the time interval Δt equal to a constant.

We define the distance on this lattice, where a box of the linear size ε is taken as the unit length. We introduce constant values, that is a, which means the minimal distance between two points $a = \min\{|x-y| : x \neq y, x, y \in K\}$ and L, which is the diameter of the set K i.e. $L = \max\{|x - y| : x, y \in K\}$.

Let us introduce I', which contains all series of the indices $\underline{n}' = (n'_1, n'_2 \ldots, n'_n)$, where $n'_1, n'_2, \ldots, n'_n \in \{1, \ldots, M'\}$, and $M' \in \mathbb{N}$. The elements of the set $T^{(n)}_{\underline{n}'}$ correspond to the path of length n. The set $T^{(n)}_{\underline{n}'}$ is the following:

$$T^{(n)}_{\underline{n}'} = \left\{ (x_1^{(n)}, x_2^{(n)} \ldots, x_n^{(n)}) \middle| x_1^{(n)} \in T_{n'_1}, x_2^{(n)} \in T_{n'_2}, \ldots, x_n^{(n)} \in T_{n'_n} \right\}.$$

Analogously to the article [11] we define a measure by the map $\mu(T^{(n)}_{\underline{n}'})$ on the lattice:

$$\mu(T^{(n)}_{\underline{n}'}) = \frac{|T^{(n)}_{\underline{n}'}|}{|T^{(n)}|}, \quad \text{where} \quad T^{(n)} = \cup_{\underline{n}' \in I'} T^{(n)}_{\underline{n}'}. \tag{12}$$

We note $T^{(n)}_{\underline{m}'} \cap T^{(n)}_{\underline{n}'} = \emptyset$, if $\underline{n}' \neq \underline{m}'$, $\underline{n}', \underline{m}' \in I'$ and $1 < |I'| \leq M^n$ and

$$\sum_{\underline{n}' \in I'} \frac{|T^{(n)}_{\underline{n}'}|}{|T^{(n)}|} = 1. \tag{13}$$

The measure of Shannon entropy is defined for finite time series on lattice:

$$\tilde{\mathcal{H}} = -\sum_{\underline{n}'\in I'} \mu(T^{(n)}_{\underline{n}'}) \ln \mu(T^{(n)}_{\underline{n}'}). \tag{14}$$

We investigate the measure of disequilibrium on grid:

$$\tilde{D} = \sum_{\underline{n}'\in I'} \left(\mu(T^{(n)}_{\underline{n}'}) - \frac{1}{N}\right)^2. \tag{15}$$

Let us introduce the measure of complexity is the following on the lattice:

$$\tilde{C} = \tilde{\mathcal{H}}\tilde{D}. \tag{16}$$

In the next section we determine the statistical complexity $\tilde{C}$ for the Lozi map on some range of parameter a, b and for the finite approximation of the set B_γ.

5.2 Approximation of the statistical complexity

In this section we consider the indicator role of the statistical complexity i.e. it enables to point out the nonlinearity on the time series $x_1^{(n)}, \ldots, x_n^{(n)}$. The Lozi map possesses chaotic behaviour in some ranges of parameter and initial condition, but it is not everywhere differentiable, therefore we perform numerical approximation. These properties were discussed by bifurcations [3] and the Lypunov exponents [9], but the statistical complexity can be determined more easier than the other quantities.

We present the statistical complexity on the finite approximation of the set B_γ, which is obtained for a generalized number system in quadratic integer.

5.2.1 Lozi map

R. Lozi introduced a two dimensional a piecewise linear map [27], which assigns the plane into itself $f : (\mathbb{R}\times\mathbb{R}) \to (\mathbb{R}\times\mathbb{R})$, it is a homomorphism on a metric space:

$$f(x, y) = (1 + y - a|x|, bx).$$

A numerical simulation is plotted on the Figure 1.

M. Misiurewicz [30] proved that Lozi map has a strange attractor on some set of values a, b, which arises from the intersection of the images of the trapping region. He supposed six conditions:

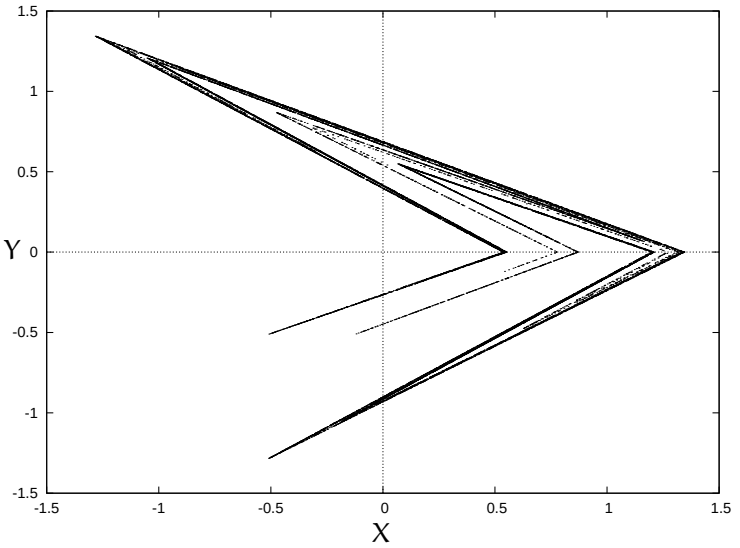


Figure 1: Lozi map ($a = 1.7, b = 0.5, N' = 10^4$ iteration)

1. $0 < b < 1, a > 0$ 2. $a > b + 1$ 3. $2a + b < 4$
4. $a > \frac{1}{2}\sqrt{3b^2 + 4 + \sqrt{(3b^2+4)^2 - 32b}}$ 5. $b < \frac{a^2-1}{2a-1}$ 6. $a\sqrt{2} > b+2$.

His theorem was proven by the geometrical verification:

That set, which satisfies these assumptions (1-6) is open and non-empty.

If the parameters fulfil the first and $a + b > 1$ conditions, the map f has two hyperbolic fixpoints: $F1 = (1/(1 + a - b), b/(1 + a - b))$ and $F2 = (1/(1 - a - b), b/(1 - a - b))$,

The stable and unstable manifold of these points W^u_{F1}, W^s_{F1}, W^u_{F2} and W^s_{F2} located on the plane (X, Y) according to the eigenvectors of the map f. The nonempty set, which satisfies the conditions 1–3 corresponds to the trapping region. Because the strange attractor equivalents to closure of unstable manifolds, he justified that subset of trapping regions, which fulfils the first and 3-5 criterion, suits to the strange attractor.

Let us consider the statistical complexity of Lozi map by numerical approximation. Particularly we study the the strange attractor on the plane of the parameter space (a, b), where $a, b \in \mathbb{R}$ and the relationship between $\tilde{\mathcal{H}}$ and $\tilde{C}$.

We applied the generalized partition of the Lozi map on the two dimensional lattice (X, Y). The kth element $x^{(n)}_k$ of time sequence of length n equals to 1, if $y > 0$, otherwise $x^{(n)}_k = 0$.

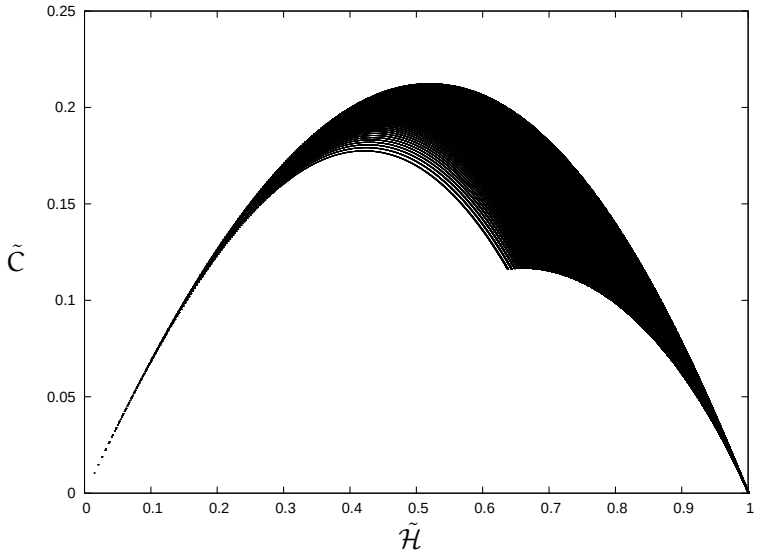


Figure 2: The points of the relationship $\tilde{\mathcal{H}} \times \tilde{C}$ for the ideal gas with uniform probability distribution ($N = 3$)

The time series of length N' ($N' >> n$) is generated by the iteration of the Lozi map and we created an appropriate N-system according to the probability distribution $\{p_1, \ldots, p_N\}$ of orbits of length n $\underline{x}^{(n)}$. Then there are N different paths of length n $\{\underline{x}_1^{(n)}, \underline{x}_2^{(n)}, \ldots, \underline{x}_N^{(n)}\}$, which corresponds to the set of the probability distribution $\{p_1, \ldots, p_N\}$, where $p_l := P(\underline{x}_l^{(n)})$ ($l = 1, \ldots, N$). It is applied to calculate the entropy $\tilde{\mathcal{H}}$, disequilibrium $\tilde{D}$ and the statistical complexity $\tilde{C}$ by appropriate measure on lattice.

Before discussing for the statistical complexity of the Lozi map we should mention that system, where the complexity does not have any intricately structure and $\tilde{C}_{max} \neq \tilde{C}_{min}$, i.e. all possible value of the discrete probability distribution appears, then the points show uniformly dispersion on the plane $\tilde{\mathcal{H}} \times \tilde{C}$ between $\tilde{C}_{min}$ and $\tilde{C}_{max}$ (Figure 2).

In contract the structure of statistical complexity for the Lozi map is characterized by the intricate dynamics on the plane $\tilde{\mathcal{H}} \times \tilde{C}$ in the range $a \in (0 : 2.5)$ and $b \in (0 : 1)$ (Figure 3).

It suggests that the structure of relationship between the entropy $\tilde{\mathcal{H}}$ and complexity $\tilde{C}$ becomes more entanglement for chaotic dynamics.

We distinguish 3 different regions of the spectrum following:

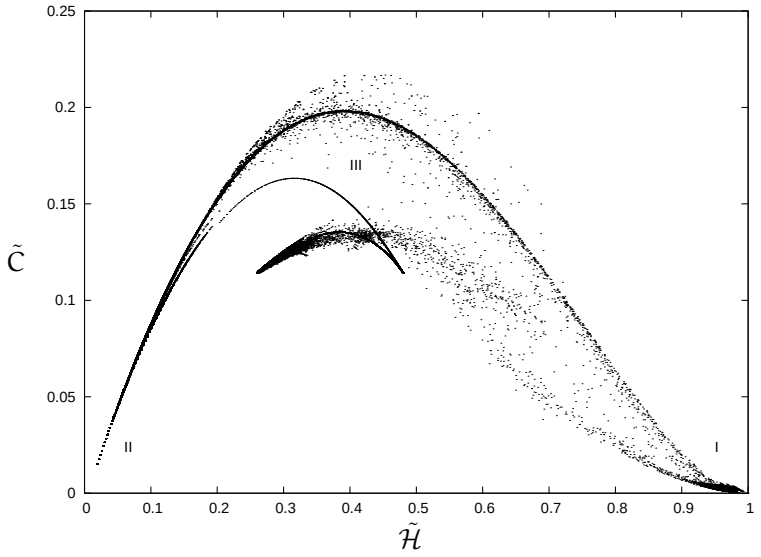


Figure 3: The $\tilde{C}(\tilde{\mathcal{H}})$ spectrum of the Lozi map ($n = 10$, $N' = 1024$)

I. The complexity $\tilde{C} \sim 0$, the entropy $\tilde{\mathcal{H}} \sim 1$: This subset corresponds to strange attractor. It is characterized by maximal entropy storage near to the equiprobable distribution $\tilde{D} \sim 0$.

II. We distinguish that region, where the $\tilde{C} \sim 0, \tilde{\mathcal{H}} \sim 0, D \sim 1$, in this case the value of entropy decreases to zero for the large amount of order.

III. Between the two extreme states (I), (II) the complexity satisfies the maximal value, which corresponds to the transition states.

The statistical complexity $\tilde{C}$ is plotted on the plane of parameter space (a, b) (Figures 4). Suitably for the case I. that region, where the value of the complexity steeply decreases to $\tilde{C} \sim 0$, at the same time the entropy increases to $\tilde{\mathcal{H}} \sim 1$ and the disequilibrium becomes to $\tilde{D} \sim 0$, corresponds to strange attractor i.e. this set is characterized by maximal entropy storage near to the equiprobable distribution.

The result of M. Misiurewitz can be compare with numerical simulation of Lozi map. These are equivalent to each other inside error.

Let us choose $\Delta\tilde{C} \sim \pm 0.01$ then $\Delta a, \Delta b \sim 0.03$ holds i.e. the triangle, which is bounded by the lines to suit the conditions 1, 3, 6. Then the measured value $\tilde{C} \sim 0, \tilde{\mathcal{H}} \sim 1$ corresponds to the theoretical consideration within a given accuracy. This region equivalents to strange attractor (Figure 4.).

We study the finite N-system on lattice, therefore we need to discuss the effect of the scaling properties. If we increase the value of N, then the complexity-

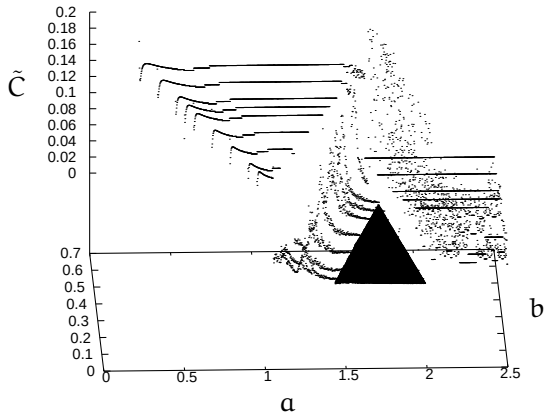


Figure 4: The statistical complexity $\tilde{C}$ of the Lozi map on the parameter plane (a, b)

curve peak goes to smaller entropy values in the numerical simulations. It means that the biggest complexity can be reached for less entropy with larger discrete probability distribution $\{p_1, \ldots, p_N\}$.

5.2.2 Finite approximation of B_γ

We introduced the idea of the generalized number system for $\mathbb{Z}_k$ in the Section 4. It can be extended to the ring of the integer $\mathbb{Z}[\theta]$ in $\mathbb{Q}[\theta]$, where θ is an algebraic integer and the element of the set forms $f(\theta) = \nu_0 + \nu_1\theta + \cdots + \nu_{n-1}\theta^{n-1}$, $\nu_j \in \mathbb{Z}$. The equivalency between the $\mathbb{Z}_k$ and $\mathbb{Z}[\theta]$ was proven [18]. The digit set is denoted by $\mathcal{A} = \{a_0, a_0, \ldots, a_{t-1}\}$ $(\subset \mathbb{Z}[\theta])$.

The map $J : \mathbb{Z}[\theta] \to \mathbb{Z}[\theta]$ is introduced by $J(\alpha) = \alpha_1$, where there exists a unique $b \in \mathcal{A}$ in $(\mathcal{A}, \theta)$ and a unique $\alpha_1 \in \mathbb{Z}[\theta]$, where $\alpha = b + \theta\alpha_1$. It can be extended for α by this expression $\alpha_l = J^{(l)}(\alpha)$.

Kátai I. and Szabó J proved that $(\theta, \mathcal{A})$ is a canonical number system if and only if $\Re\theta < 0$ and $\Im\theta = \pm 1$, where θ is a Gaussian complex integer and $\mathcal{A} = \{0, 1, \ldots, N(\theta) - 1\}$ $(N(\theta) = \theta\bar{\theta})$ [21].

According to the expression of the fundamental set H (5), it holds in this extension field $\mathbb{Z}[\theta]$. Let $\rho = 1/\theta$, where $\rho \in \mathbb{C}$, $0 < |\rho| < 1$ and $\mathcal{A} = \{0, 1\}$.

Then the analogue set H:

$$H = \left\{ z | z = \sum_{i=1}^{\infty} \rho^i f_i, \quad f_i \in \mathcal{A} \right\}. \tag{17}$$

Because $B_\gamma = H \cap H + \gamma$,

$$B_\gamma = \left\{ z \middle| z \in H, z - \gamma \in H \right\}. \tag{18}$$

Therefore all expansions of γ appear as

$$\gamma = \rho^1 e_1 + \rho^2 e_2 \ldots, \tag{19}$$

where $e_1, e_2 \cdots \in \mathcal{B} = \mathcal{A} - \mathcal{A}$. Because $e_i = f_i - f'_i$ holds, where $f_i, f'_i \in \mathcal{A}$, $(i = 1 \ldots)$, we can determine all of possible values of the digit f_i, which follows from expressions (17), (18):

$$z = \rho^1 f_1 + \rho^2 f_2 \ldots. \tag{20}$$

The elements of the set B_γ contains z over infinite sums, we will approximate them with finite sums. This set is denoted by $\widetilde{B}_\gamma$, which contains these elements for some fixed k and γ, it is written as

$$x = \sum_{i=1}^{k} \rho^i f_i, \quad f_i \in \mathcal{A}. \tag{21}$$

The Kolmogorov entropy and the fractal dimension of the finite approximation of the set B_γ were published [14] [11].

In the next section we will study the statistical complexity for the set $\widetilde{B}_\gamma$, whose every element corresponds to a subset of B_γ.

5.2.3 Statistical complexity of the set $\tilde{B}_\gamma$

In this section we present the numerical results, which is obtained for a generalised number system in quadratic integers.

In the article [17] it was proven that $\theta \in \mathbb{C}$ is a root of the polynomial of second-degree $f(x)$, whose coefficients are $a_2 = 1, a_1 = 0, \pm 1, \pm 2, a_0 = 2$. The smallest ring is $\Delta = \{1, \Theta\}$. Then

$$\cup_{\gamma \in \Delta} (H + \gamma) = \mathbb{C} \tag{22}$$

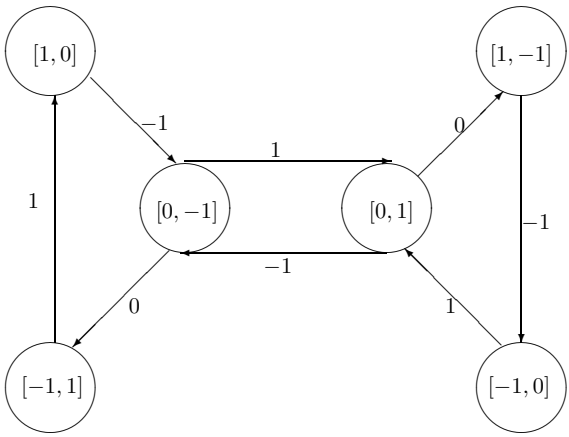


Figure 5: Transition graph for $\theta = -1 - i$ and $\mathcal{B} = \{-1, 0, 1\}$

$$\lambda((H + \gamma_1) \cap (H + \gamma_2)) = 0, \quad \gamma_1 \neq \gamma_2, \ \ \gamma_1, \gamma_2 \in \Delta \tag{23}$$

where $\gamma = \sum_{\nu=0}^{l} a_\nu \Theta^\nu$ and $a_\nu \in \mathcal{A}$. Then $(\Theta, \mathcal{A})$ is a canonical number system in a quadratic field extensions for all these Θ values.

We construct the transition graph $G(S)$ (Figure 5) according to Section 4.1. The base of the number system is chosen as $\theta = -1 - i$ and the digit set $\mathcal{A} = \{0, 1\}$. The edge is labeled by an element of the set $\mathcal{B} = \{-1, 0, 1\}$.

The steps of graph construction are the following:

◦ Every $\gamma \in \mathbb{Z}[\Theta]$ which satisfies the condition $|\gamma| \leq \sqrt{2} + 1$, we determinate $\eta = \gamma\Theta - \delta$ for $\delta \in \mathcal{B}$. A directed edge fits from γ to η, if $|\eta| \leq \sqrt{2} + 1$ holds.

◦ That γ vertices is deleted, which has no edge from γ and remove all edges which are directed to γ.

The process results the graph $G(S)$.

Let us consider the process of the graph walking P.

◦ First step in the initial condition we choose one vertex along the graph $G(S)$ randomly.

◦ The basic concept of the graph walking P is the following. It contains all of possible edges at least ones $(\delta_1, \delta_2, \ldots, \delta_k)$, i.e. this step produces the minimal length orbit.

◦ Let us take into consideration, that the outgoing degree of vertices q can be larger than 1 and the graph walking need to contain all edges, therefore the same node can appear more times resulting the perfect series of all directed ones.

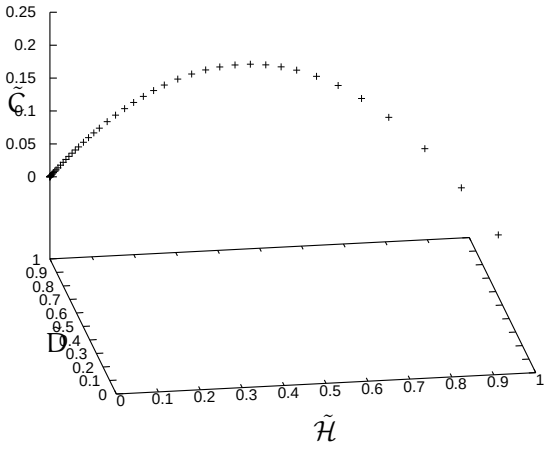


Figure 6: The spectrum $\tilde{C}(\tilde{\mathcal{H}}, \tilde{D})$ of the set $\tilde{B}_\gamma$ ($N' = 8064, n = 50$)

◦ We need to consider each of sequences $f_1, \ldots, f_r$, ($f_i \in \mathcal{A}$) to the series of labels of edges.

The N-system is determined by finite walk on the transition graph $G(S)$. The probability distribution $\{p_1, p_2, \ldots, p_N\}$ is created according to the path of length n $\underline{x}^{(n)}$ along the walk of length N'.

We determined the Shannon entropy $\tilde{\mathcal{H}}$, the disequilibrium $\tilde{D}$ and the statistical complexity $\tilde{C}$ on lattice, which is plotted on the Figure 6.

The points of the curve in three dimensional space $\tilde{\mathcal{H}} \times \tilde{D} \times \tilde{C}$ does not contain any intrinsic structure i.e. any structural complexity and it is a unique convex function $\tilde{C}(\tilde{D})$ respectively $\tilde{C}(\tilde{\mathcal{H}})$. The curve corresponds to the maximal limit of the statistical complexity $\tilde{C}_{max}$ of the curve in Figure 3. on the range $\tilde{\mathcal{H}} \in [0, 1]$ and $\tilde{D} \in [0, 1]$.

The transition graph, which was introduced analogues to dynamical process on the set $\tilde{B}_\gamma$ points to regular motion.

6 Summary

The statistical complexity is an indicator, which can be used to reveal the dynamical behaviour of the finite system. We calculated the complexity $\tilde{C}$ of

Lozi map on the plane of parameter (a, b), this map has chaotic range, where the strange attractors can be unique specified by complexity $\tilde{\mathsf{C}}$, entropy $\tilde{\mathcal{H}}$ and disequilibrium $\tilde{\mathsf{D}}$ on lattice. Here the value of entropy becomes $\tilde{\mathsf{H}} \sim 1$, the quantity of disequilibrium is $\tilde{\mathsf{D}} \sim 0$, therefore the value of $\tilde{\mathsf{C}}$ changes to 0 sharply on the parameter space. Contrary to the statistical complexity $\tilde{\mathsf{C}}$ of the set $\tilde{\mathsf{B}}_\gamma$ in the generalized number system corresponds to unique function of the entropy $\tilde{\mathcal{H}}$ and disequilibrium $\tilde{\mathsf{D}}$ and it fits to the upper limit of the spectrum $\tilde{\mathsf{C}}(\mathcal{H})$ for the Lozi map. Because the chaotic motion on the strange attractor contains aperiodic orbits in contrast to the dynamical behaviour of the set B_γ includes periodic paths, the statistical complexity is able to indicate the different properties of time series and it is localised in the finite dimensional space.

References

[1] C. Adami, N. T. Cerf, Physical complexity of symbolic sequences, *Physica D* **137,** 1–2 (2000) 62–69. ⇒230

[2] C. Anteneodo, A. R. Plastino, Some features of the López-Ruiz-Manchini-Calbet (LMC) statistical measure of complexity, *Physics Letters A* **223,** 5 (1996) 348–354, ⇒231

[3] V. Botella-Soler, J. M. Castelo, J. A. Oteo, J. Ros, Bifurcations in the Lozi map, *Journ. Phys. A: Math. Theor.* **44,** 30 (2011) 305101–305115. ⇒241

[4] X. Calbet, R. Lopez-Ruiz, Extremism complexity distribution of a monodimensional ideal gas out of equilibrium, *Physica A* **382,** 2 (2007) 523–530. ⇒230

[5] X. Calbet, R. Lopez-Ruiz, Extremum complexity in the monodimensional ideal gas: The piecewise uniform density distribution approximation, *Physica A* **388,** 20 (2009) 4364–4378. ⇒230

[6] P. Collet, J. P. Eckmann, *Iterated maps in the interval as dynamical systems*, Birkhäuser, 1980. ⇒235

[7] J. P. Crutchfield, K. Young, Inferring statistical complexity, *Phys. Rev. Lett.* **63,** 2 (1989) 105–108. ⇒230

[8] J. P. Eckmann, D. Ruelle, Ergodic theory of chaos and strange attractors, *Rev. of Modern Physics* **57,** 3 (1985) 617–656. ⇒235, 237

[9] Z. Elhadj, J. C. Sprott, A Unified Piecewise Smooth Chaotic Mapping that Contains the Hénon and the Lozi Systems, *Annual Review of Chaos Theory, Bifurcations and Dynamical Systems* **1** (2012) 50–60. ⇒241

[10] G. Farkas, Number systems in real quadratic fields, *Annales Univ. Sci. Rolando Eötvös Budapest. Sect. Comput.* **18** (1999) 47–49. ⇒237

[11] G. Farkas, A. Fülöp, The sandbox method in quadratic fields, *Annales Univ. Sci. Rolando Eötvös Budapest. Sect. Comput.* **28** (2008) 235–248. ⇒237, 240, 246

[12] G. Feng, S. Song, P. Li, A statistical measure of complexity in hydrological systems, *Hydr. Eng. Chin.* **11** (1998) 14. ⇒230
[13] G. L. Ferri, F. Pennini, A. Plastino, LMC-complexity and various chaotic regime, *Physics Letters A* **373,** 26 (2009) 2210–2214. ⇒231
[14] Á. Fülöp, Estimation of the Kolmogorov entropy in the generalized number system, *Annales Univ. Sci. Budapest Sect. Comp.* **40** (2013) 245–256. ⇒231, 246
[15] C. M. Gonzalez, H. A Larrondo, O. A. Rosso, Statistical complexity measure of pseudorandom bit generators, *Physica A* **354** (2005) 281–300. ⇒230
[16] P. Grassberger, Toward a Quantitative Theory of Self-Generated Complexity, *Int. Journ. Theor. Phys.* **25,** 9 (1986) 907–938. ⇒231
[17] K. H. Indlekofer, I. Kátai, P. Racskó, Some remarks in generalized number systems, *Acta Sci. Math.* **57** (1993) 543–553. ⇒246
[18] I. Kátai, Generalized number systems and fractal geometry, Univ. Janus Pannonius Pécs, 1995. ⇒237, 245
[19] I. Kátai, Number systems in imaginary quadratic fields, *Annales Univ. Sci. Roland Eötvös Budapest, Sect. Comput.* **14** (1994) 91–103. ⇒237
[20] I. Kátai, Generalized number systems in Euclidean spaces, *Math. and Computer Modelling* **38,** 7–9 (2003) 883–892. ⇒237
[21] I. Kátai, J. Szabó, Canonical number systems for complex integers, *Acta Sci. Math.* **37,** 3–4 (1975) 255–260. ⇒245
[22] A. N. Kolmogorov, Entropy per unit time as a metric invariant of automorphism, *Doklady of Russian Academy of Sciences*, **124** (1959) 754–755. ⇒230
[23] P. T. Landsberg, J. S. Shiner, Disorder and complexity in an ideal non-equilibrium Fermi gas, *Phys. Lett. A* **245,** 3–4 (1998) 228–232. ⇒230
[24] A. Lempel, J. Ziv, On the complexity of finite sequences, *IEEE Trans. Inform. Theory* **22**, 1 (1976) 75–81. ⇒230
[25] R. López-Ruiz, H. L. Mancini, X. Calbet, A statistical measure of complexity, *Phys. Letters A* **209,** 5–6 (1995) 321–326. ⇒230, 231, 234
[26] M. Lovallo, V. Lapenna, L. Telesca, Transition matrix analysis of earthquake magnitude sequences, *Chaos, Soliton and Fractals* **24,** 1 (2005) 33–43. ⇒231
[27] R. Lozi, Un attracteur étrange du type attracteur de Hénon, *Journal de Physique* **39** (1978) C5–9. ⇒241
[28] M. T. Martin, A. Plastino, O. A. Rosso, Statistical complexity and disequilibrium, *Physics Letters A* **311,** 2-3 (2003) 126–132. ⇒231

[29] M. T. Martin, A. Plastino, O.A. Rosso, Generalized statistical complexity measures: Geometrical and analytical properties, *Physica A* **369,** 2 (2006) 439–462. ⇒231, 234

[30] M. Misiurewicz, Strange attractors for the Lozi mappings, *Annals of the New York Academy of Sciences* **357** (1980) 348–358. ⇒241

[31] A. Rényi, *Probability Theory*, North-Holland, Amsterdam, 1970. ⇒234

[32] P. T. Saunders, M. W. Ho, On the increase in complexity in Evolution II. The relativity of complexity and the principle of minimum increase, *Journ. Theor. Biol.* **90,** 4 (1981) 515–530. ⇒231

[33] C. E. Shannon, The Mathematical Theory of Communication, *Bell System Technical Journal* **27** (1948) 379–423, 623–656. ⇒232

[34] C. Tsallis, Possible generalization of Boltzmann-Gibbs statistics, *J. Stat. Phys.* **52,** 1–2 (1988) 479–487. ⇒234

[35] J. M. Thuswaldner, Fractal and number systems in real quadratic number fields, *Acta Math. Hungary* **90,** 3 (2001) 253–269. ⇒238

[36] Z. Yu, G. Chen Rescaled range and transition matrix analysis of DNA sequences, *Comm. Theor. Phys.* **33,** 4 (2000) 673–678. ⇒231

[37] W. K. Wooters, Statistical distance and Hilbert space, *Phys. Rev. D* **23,** 2 (1981) 357–362. ⇒234

Received: August 28, 2014 • Revised: October 22, 2014

Acta Univ. Sapientiae, Informatica, **6,** 2 (2014) 252–286

Recognition of split-graphic sequences

Bilal A. CHAT
University of Kashmir
Hazratbal Srinagar-190006, India
email: bilalchat99@gmail.com

Shariefudddin PIRZADA
University of Kashmir
Hazratbal Srinagar-190006, India
email:
pirzadasd@kashmiruniversity.ac.in

Antal IVÁNYI
Eötvös Loránd University
Faculty of Informatics, H-1011
Budapest, Pázmány s. 1/A, Hungary
email: tony@inf.elte.hu

Abstract. Using different definitions of split graphs we propose quick algorithms for the recognition and extremal reconstruction of split sequences among integer, regular, and graphic sequences.

1 Basic definitions

In this paper a, b, l, m, n, p and q denote nonnegative integers with $b \geq a$ and $l + m \geq 1$. We follow the terminology of *Handbook of Graph Theory* [28] written by Gross, Yellen and Zhang.

An (a, b, n)-***graph*** is a loopless graph in which different vertices are connected at least by a and at most by b edges [43, 44]. A (b, b, l)-***graph*** is denoted by K_l^b and is called a b-***clique*** or b-***complete graph***. Clearly, $K_l^1 = K_l$, where K_l is the ***complete graph*** on l vertices. Its complement, $\overline{K}_l$ is called ***independent graph*** on l vertices.

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C30, 05C50
Key words and phrases: psplit graph, jsplit graph, bsplit graph, graphic sequence, linear time algorithm, (a, b, n)-graph, potentially split sequence

The ***join*** [12, 28, 66] of two graphs G and H is denoted by $G + H$. It has the following vertex set and edge set:

$$V(G + H) = V(G) \cup V(H)$$

and

$$E(G + H) = E(G) \cup E(H) \cup \{uv \mid u \in V(G) \text{ and } v \in V(H)\}.$$

A nonincreasing integer sequence $\sigma = (s_1, \ldots, s_n)$ with $s_1 \leq b(n-1)$ and $s_n \geq a(n-1)$ is said (a, b, n)-***regular*** [43, 44]. A $(0, b, n)$-regular sequence shortly is said b-***regular*** [17]. An integer sequence σ is said (a, b, n)-***graphic***, if it is the degree sequence of an (a, b, n)-graph G [43, 44], and such a graph G is referred to as a ***realization*** of σ. An integer sequence is called ***even,*** if the sum of its elements is even.

In this paper we denote the integer sequences by σ and the degree sequences by δ.

In 1965 Fulkerson, Hoffman and McAndrew [24] proposed the following definition of (γ, δ)-***multigraphs with capacity bounds***. Let $n \geq 1$, $\delta = (d_1, \ldots, d_n)$ and $\gamma = (c_{11}, \ldots, c_{1n}, c_{21} \ldots, c_{2n}, \ldots, c_{n-1,n}, c_{n,n})$ sequences of nonnegative integers with $c_{ii} = 0$ and $c_{ij} = c_{ji}$ for $1 \leq i < j \leq n$. Fulkerson and his coathors call δ ***degree vector***, while γ is the ***capacity vector***. Let G_γ denote the graph in which there is an edge between the vertex with degree d_i and vertex with degree d_j, if $c_{ij} = 1$. The capacity vector γ has the ***odd-cycle condition*** if the graph G_γ has the property that any two of its odd length (simple) cycles either have a common vertices or there exists a pair of vertices, one vertex from each cycle, which are connected with an edge.

With other words, the distance between two odd length cycles is at most 1. In particular, if G_γ is bipartite (has no odd length cycle) or G_γ is complete (all c_{ij} equals to 1) then γ obviously satisfies the odd-cycle condition.

An (a, b, n)-regular sequence is said ***potentially*** K_l^b-***graphic,*** if it has a realization G containing K_l^b as a subgraph. If $b = 1$, then we write simply K_l instead of K_l^1.

An (a, b, n)-regular sequence $\sigma = (s_1, \ldots, s_n)$ is said ***potentially*** A_l^b-***graphic,*** if it has a realization G containing J_l^b (definition see later) on vertices having degrees $s_1, \ldots, s_{l+m}$.

An (a, b, n)-regular sequence $\sigma = (s_1, \ldots, s_n)$ is said ***potentially*** $A_{l,m}^b$-***graphic,*** if it has a realization G containing $J_{l,m}^b$ (definition see later) on vertices having degrees $s_1, \ldots, s_{l+m}$.

A $(0, b, n)$-regular sequence $\sigma = (s_1, \ldots, s_n)$ is said ***potentially*** $J_{l,m}^b$-***graphic*** if it has a realization G containing $J_{l,m}^b$ (definition see later) on vertices having

degrees $s_1, \ldots, s_{l+m}$. If $b = 1$, then we write simply A_l insted A^1_b,, $A^1_{l,m}$ instead of $A^b_{l,m}$, $J_{l,m}$ instead of $J^1_{l,m}$, and $J_{l,m}$ instead of $J^1_{l,m}$.

Let $n \geq 2$ and $\sigma = (s_1, \ldots, s_n)$ be a nonnegative integer sequence, and k be any integer $1 \leq k \leq n$. Let $\sigma' = (s'_1, \ldots, s'_n)$ be the sequence obtained from s by setting $s_k = 0$ and $s'_i = s_i - 1$ for the s_k largest elements of s other than s_k. Let H_k be the graph obtained on the vertex set $V = \{v_1, \ldots, v_n\}$ by joining v_k to the s_k vertices corresponding to the s_k elements used to obtain s'. This operation of getting s' and H_k is called ***laying off*** s_k, s' is called ***residual sequence***, and H_k is called the ***subgraph obtained by laying off*** s_k [51].

Now we formulate several definitions of split graphs.

The classical and most distributed definition of split graphs was introduced by Földes and Hammer in 1977 [21, 22, 26, 28].

Definition 1 (Földes, Hammer [21, 22]) *An* (l, m)-***partitioned split graph*** (*shortly: psplit graph*) *is one whose vertex set can be partitioned into two disjoint subsets spanning a clique* K_l *and an independent graph* $\overline{K}_m$. *It is denoted by* $S_{l,m}$.

It is worth to mention that one of l and m can be zero, that is if $l \geq 1$, then $S_{l,0}$ is also a psplit graph, and if $m \geq 1$, then $S_{0,m}$ is also a psplit graph, and the independent graph $\overline{K}_{0,m} = S_{0,m}$ are also psplit graphs. For the number of edges $|E(S_{l,m})|$ of an $S_{l,m}$ hold the inequalities $l(l-1)/2 \leq |E(S_{l,m})| \leq l(l-1)/2 + l \cdot m$ and between these bounds every integer value is realizable.

Consider the following example (Figure 1). Let $G = (V, E)$, where $V = \{v_1, \ldots, v_5\}$ and $E = (v_1v_2, v_1v_3, v_1v_4, v_2v_3, v_2v_4, v_3v_4\}$, that is G contains six edges. Then G is $S_{3,2}$ and also $S_{4,1}$ due to the following two partitions of V: $\{v_1, v_2, v_3, v_4\}$ plus $\{v_4, v_5\}$ (Figure 1a)) containing all six edges and also is $S_{3,2}$ due to the partition $\{v_1, v_2, v_3\}$ plus $\{v_4, v_5\}$ (Figure 1b)) containing only three edges.

In 1996 Brandstädt introduced the following definition of (l, m)-multipartitioned split graphs. Let $G = (V, E)$ with $|V| = n$. $V_1, \ldots, V_k$ is a ***partition*** of V, if and only if for all $u, v \in \{1, \ldots, k\}$ with $i < j$ $V_i \cap V_j = \emptyset$ and $\bigcup_{i=1}^{k} V_i = V$. A partition $C_1, \ldots, C_m, I_1, \ldots, I_m$, with cliques C_i, $i \in \{1, \ldots, l\}$ and independent sets I_j, $j \in \{1, \ldots, m\}$ is an (l, m)-partition of V.

Definition 2 (Brandstädt [7, 8, 10]) *A graph* $G = (V, E)$ *is called an* (l, m)-***split graph***, *if its vertex set has an* (l, m)*-partition.*

In 1998 Gyárfás generalized (l, m)-psplit graphs to (l, m)-bsplit graphs.

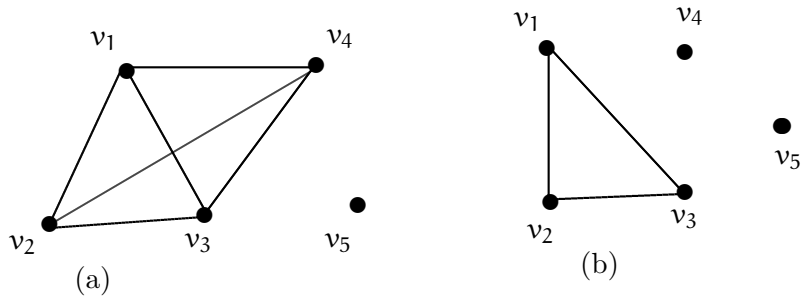


Figure 1: Partition of a psplit graph is not unique.

Definition 3 (Gyárfás [30]) *A graph* G *is called* (l, m)-***bounded split graph*** (*shortly: bsplit graph*) *if its vertex set can be partitioned into* A *and* B *so that the order of the largest clique graph in* A *is* l *and the order of the largest complete subgraph in* B *is* m.

In 2001 Hell, Klein, Protti, and Tito [41] defined (k, l)-graphs so, that their vertex set can be partitioned into k cliques and l independent sets.

In 2005 Bradstädt, Hammer, Le and Lozin studied bisplit graphs, defined as follows.

Definition 4 (Brandstädt, Hammer, Le, Lozin [9]) *A graph* G *is called* (l, m)-***bisplit graph*** (*shortly: bisplit graph*) *if its vertex set can be partitioned into a complete bipartite graph and an independent graph. It is denoted by* $B_{l,m}$.

For the number $|E(B_{l,m})|$ of edges of a bisplit graph $B_{l,m}$ hold the inequalities $l^2 \leq |E(B_{l,m})| \leq l^2 + 2lm$.

In 2007 Le and Ritter [55] defined probe split graphs (modifying the definition of interval split graphs [61]).

Definition 5 (Le, Ritter [55]) *A* $G(V, E)$ *graph is* ***probe split graphs****, if* V *can be partitoned into two parts* N *(nonprobes) and* P *(probes) where* N *is an independent set and there exists* $E' \subset N \times N$ *such that* $G' = (V, E \cup E')$ *is a psplit graph.*

In 2009 [6] Boros, Gurvich and Zverovich [6] proposed the definition of ***almost CIS-graphs***.

The following simple definition appeared in 2011 and later in the papers of different authors as Chat, Pirzada, and Yin [67, 86, 87].

Definition 6 (Yin [86, 87]) *An* (l, m)-***join split graph*** (*shortly: jsplit graph*) *is the join of* K_l *and* $\overline{K}_m$. *It is denoted by* $J_{l,m}$.

It is worth to remark, that jsplit graphs are special cases of psplit graphs: if G is a jsplit graph, then any vertex of K_l is connected with any vertex of $\overline{K}_m$, while in the corresponding psplit graph even all such edges can be absent.

Consider the following example Let $H = (V, E)$, where $V = \{v_1, \ldots, v_5\}$ and $E = (v_1v_2, v_1v_3, v_1v_4, v_1v_5, v_2v_3, v_2v_4, v_2v_5, v_3v_4, v_3v_5, v_4v_5\}$, that is H is a clique on 5 vertices and so it contains ten edges. Then H is $J_{5,0}$ and also $J_{4,1}$ due to the following partition of V: $\{v_1, v_2, v_3.v_4, v_5\}$ plus $\emptyset$ and is $J_{4,1}$ for example due to the partition $\{v_1, v_2, v_3, v_4\}$ plus $\{v_5\}$. We can remark that these partitions at the same time give psplit graphs with the same size parameters.

Let K_l^b and K_m^b be b-cliques, and let $\overline{K}_m^b$ be the complement of K_m^b, that is an empty graph on m vertices. We propose the following generalization of psplit-graphs.

Definition 7 *A* (b, l, m)-***partitioned split graph*** (*shortly:* b-*psplit graph*) *is one whose vertex set can be partitioned into two disjoint subsets spanning a* b-*clique* K_l^b *and an empty graph* $\overline{K}_m^b$. *It is denoted by* $S_{l,m}^b$.

Clearly, $S_{l,m}^1 = S_{l,m}$.

In 2011 Yin extended the definition of the jsplit-graphs to b-jsplit graphs.

Definition 8 (Yin [87]) *A* (b, l, m)-***join-split graph*** (*shortly:* b-*jsplit graph*) *is the join of* K_l^b *and* $\overline{K}_m^b$. *It is denoted by* $J_{l,m}^b$.

Clearly, $J_{l,m}^1 = J_{l,m}$.

Figure 2 shows $J_{3,2}$ (part a) and $J_{3,2}^2$ (part b).

The structure of the paper is as follows. After the basic definition (Section 1) in Section 2, 3 and 4 the most important mathematical background results connected with graphical, potentially graphical and potentially split graphical sequences are reviewed, then in Sections 5, 6 the new mathematical results are presented.

We review the known algorithms in Sections 7 and 8, while the now proposed algorithms are presented in Section 9.

The main results of the paper are that using different definitions of split graphs [4, 5, 7, 8, 9, 10, 21, 22, 23, 26, 28, 30, 67, 82, 86, 87, 88] we propose quick algorithms for the recognition and extremal reconstruction of split sequences among integer, regular [17, 45] and graphic [43, 45, 48] sequences.

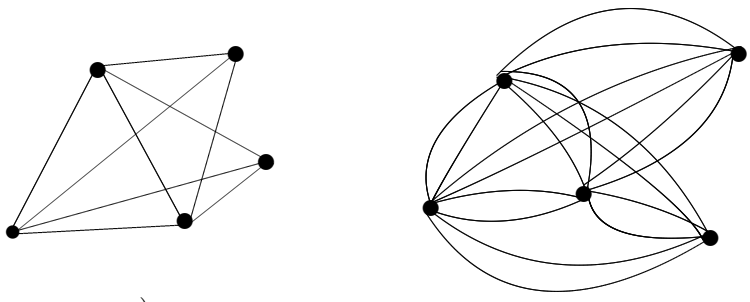


Figure 2: Jsplit graphs $J_{3,2}$ (a)) and $J^2_{3,2}$(b)).

2 Known results on graphic sequences

In 1955 Havel, in 1962 Hakimi proposed the following necessary and sufficient condition for n-regular sequences to be graphic.

Theorem 9 (Havel [32], Havel [36]) *Let $n \geq 2$. An n-regular sequence $\sigma = (s_1, \ldots, s_n)$ is graphical if and only if the sequence $\sigma' = (s_2-1, s_3-1, \ldots, s_{s_1} - 1, s_{s_1+1} - 1, s_{s_1+2}, \ldots, s_{n-1}, s_n)$ is graphical.*

Proof. See Hakimi[32], Havel [36]. □

The recursive algorithm implementing this theorem requires in worst case $\Theta(n^2)$ time. It is worth to remark, that this algorithm not only tests the sequences, but if they are graphic, the algorithm constructs a realization of the tested sequence.

In 1973 Kleitman and Wang improved the Havel-Hakimi theorem: according to their following theorem it is sufficient to test *any* nonzero element of the input sequence. The central element of their proof is the laying off the tested sequence.

Theorem 10 (Kleitman, Wang [51]) *Let $n \geq 2$. A nonnegative integer sequence σ is graphic if and only if the residual sequence obtained by laying off any nonzero element of s is graphic.*

Proof. See Kleitman [51]. □

In 1974 Chungphaisan [13] extended the definition of laying off and residual sequence to b-***laying off*** and b-***residual sequence*** as follows. Let $\sigma =$

$(s_1, \ldots, s_n)$ be an n-regular sequence and $1 \leq k \leq n$. Define $\sigma_k^{'} = (s_1', \ldots, s_{n-1}')$ to be the nonincreasing rearrangement of the sequence obtained from $(s_1, \ldots, s_{k-1}, s_{k+1}, \ldots, s_n)$ reducing by 1 the remaining largest term that has not already been reduced b times, and repeating the procedure s_k times. $s_k^{'}$ is called the *b-residual sequence* obtained from σ by ***b-laying off*** s_k.

Using the b-laying off operation Chungphaisan proved the following generalization of Kleitman-Wang theorem.

Theorem 11 (Chungphaisan [13]) *Let $n \geq 2$. A nonnegative integer sequence σ is b-graphic if and only if the b-residual sequence obtained by b-laying off any nonzero element of σ is graphic.*

Proof. See Chungphaisan [13]. □

In 1960 Erdős and Gallai gave the following necessary and sufficient condition.

Theorem 12 (Erdős, Gallai [17]) *Let $n \geq 1$. An n-regular even sequence $\sigma = (s_1, \ldots, s_n)$ is graphical if and only if*

$$\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min(d_i, k)$$

is satisfied for each integer k, $1 \leq k \leq n$.

Proof. See Erdős, Gallai [17]. □

Later several new proofs of this theorem were published, among others by to Gasharov in 1997, [25], by Tripathi and Tiagy in 2008 [77], by Tripathi, Venugopalan and West in 2010 [78].

In 1974 Chungphaisan extended Erdős-Gallai theorems to $(0, b, n)$-graphs.

Theorem 13 (Chungphaisan [13]). *Let $\sigma = (s_1, \ldots, s_n)$ be an (a, b, n)-regular even sequence. Then σ is $(0, b, n)$-graphic if and only if for each positive integer $t \leq n$,*

$$\sum_{i=1}^{t} s_i \leq bt(t-1) + \sum_{i=t+1}^{n} \min(bt, s_i).$$

Proof. See Chungphaisan [13]. □

We remark then if we use the theorems of Erdős-Gallai [17], Havel-Hakimi [32, 36], Kleitman-Wang [51] or Chungphaisan [13] to decide whether an integer sequence is graphic, the decision requires quadratic time. In 2012 Iványi

proposed an algorithm for $(0, b, n)$ graphs, then in 2012 [45] for (a, b, n)-graphs allowing the decision in worst case in $O(n)$ time.

In the worst case the algorithm based on Theorem 13 requires quadratic time, but the following assertion allows us to test the sequences in linear time. Since this is an important result, we repeat its proof.

Theorem 14 (Iványi [45]) *If* $n \geq 1$*, then the* $\sigma = (s_1, \ldots, s_n)$ $(0, b, n)$*-regular sequence is* $(0, b, n)$*-graphic if and only if*

$$\sum_{i=1}^{n} s_i \quad \textit{is even}$$

and

$$H_i > bi(y_i - 1) + H_n - H_y \quad (i = 1, \ldots, n-1), \tag{1}$$

where

$$y_i = \max(i, w_i) \quad (i = 1, \ldots, n-1). \tag{2}$$

Proof. This proof is an improved version of the proof of linearity of EGL in [48] and was published in 2012 [45]

We exploit that s is monotone and determine the capacity estimations $c_k = \min(jb, s_k)$ appearing in (1) in constant time. The base of the quick computation is again the sequence of the weight points $w(\sigma) = (w_1, \ldots, w_{n-1})$ containing the *weight points* belonging to the elements of σ, and the sequence $y(\sigma) = (y_1, \ldots, y_n)$ containing the cutting points of the elements of s. For given s_i the ***weight point*** w_i is the largest k $(1 \leq k \leq n)$ having the property $s_k \geq i$. . The ***cutting point*** y_i belonging to s_i is the maximum of i and w_i, see (2).

During the testing of the elements of s there are two cases:

a) if $i > w_i$, then the maximal contribution $C_i = \sum_{k=i+1}^{n} \min(i, s_k)$ of the actual tail of s is at most $H_n - H_i$, since the maximal contribution $c_k = \min(i, s_k)$ of the element s_k is only s_k, and so

$$C_i = \sum_{k=i+1}^{n} c_k = H_n - H_i,$$

implying the requirement

$$H_i \leq bi(i-1) + H_n - H_i; \tag{3}$$

b) if $i \leq w_i$, then the maximal contribution C_i of the actual tail of s consists of contributions of two types: $c_{i+1}, \ldots, c_{w_i}$ are equal to bi, while $c_j = s_j$ for $j = w_i + 1, \ldots, n$, therefore we have

$$C_i = bi(w_i - i) + H_n - H_{w_i}, \tag{4}$$

implying the requirement

$$H_i = bi(i-1) + bi(w_i - i) + H_n - H_{w_i}. \tag{5}$$

Transforming (5) we get

$$H_i = bi(w_i - 1) + H_n - H_{w_i}. \tag{6}$$

Considering the definition of y_i given in (2), further (4) and (5) we get the required (1). □

In 1981 Rao [69] analyzed the conditions for graphic sequences to be P-graphic.

In 2009 Hell Hell and Kirkpatrick [40] extended the concept of graphic sequences defining quasi-graphic sequences and proposing a linear time algorithm for their certifying. A state of art of certifying algorithms was published in 2011 [59] by McConell, Mehlhorn, Näher and Schweitzer.

The following assertion is the base of the quick testing of integer sequences whether they are (a, b, n)-graphic or not. In 2012 Iványi proved that theorem of Chungphaisan has the following consequence allowing the quick test of potential (a, b, n)-graphic sequences..

Corollary 15 (Iványi [45]) *If* $n \geq 2$, *then an* (a, b, n)*-regular sequence* $\sigma = (s_1, \ldots, s_n)$ *is* (a, b, n)*-graphic if and only if the sequence* $s' = (s_1 - a, s_2 - a, \ldots, s_n - a)$ *is* $(0, b - a, n)$*-regular.*

Proof. See [45] □

This corollary allows the testing of (a, b, n)-regular sequences in worst case in $O(n)$ time using algorithm Erdős-Gallai-Linear [48] or algorithm Havel-Hakimi-Chungphaisan [45].

The following sources contain results on the enumeration of graphic and b-graphic sequences [35, 45, 46, 47, 48, 49, 74].

3 Known results on A-graphic sequences

The following two results due to J. H. Yin are generalizations from 1-graphs to b-graphs of two well-known results given by A. R. Rao [50, 68, 70]

Theorem 16 (Yin [84]). *Let* $n \geq l+1$ *and* $\sigma = (s_1, \ldots, s_n)$ *be a* b*-graphic sequence with* $s_{l+1} \geq bl$*. Then* σ *is potentially* A^b_{l+1}*-graphic if and only if* s'_{l+1} *is* b*-graphic.*

Proof. See [84]. □

Theorem 17 (Yin [84]) *Let* $n \geq l+1$ *and* $\sigma = (s_1, \ldots, s_n)$ *be a* b*-graphic sequence with* $s_{l+1} \geq 2bl-1$*, then* s *is potentially* A^b_{l+1}*-graphic.*

Proof. See [84]. □

In 1978 Hakimi and Schmeichel [33] studied potentially and forcibly P-graphic sequences.

In 2009 Yin generalized a result Gould, Jacobson and Lehel [27].

Theorem 18 (Yin [85]) *If* $\delta = (d_1, \ldots, d_n)$ *is is a* b*-graphic sequence with a realization* G *containing a* b*-graph* H *as a subgraph, then there exists a realization* G' *of* δ *so that the vertices of* H *have the largest degrees of* δ*.*

Proof. See [85]. □

In 2009 Yin wrote [84] that the following assertion is a special case of Theorem 18.

Corollary 19 *A* b*-graphic sequence is potentially* K^b_l*-graphic, if and only if it is potentially* A^b_l*-graphic.*

Proof. We prove a bit stronger assertion.

It is trivial, that if an integer sequence is A^b_l-graphic, then it is b-graphic.

The sufficiency can be proved following the proof of Lemma 2.1. in [85]. □

We remark, that Theorem 18 contains only a sufficient condition.

Let $l, m,\ r$ and n be positive integers, $n \geq l+m$, and let $\sigma = (s_1,\ \ldots,\ s_n)$ be an n-regular sequence with $s_l \geq l+m-1$ and $s_{l+m} \geq l$. We construct the sequences $\sigma_1,\ \ldots,\ \sigma_l$ as follows. At first we construct the sequence

$$\sigma_1 = (s_1 - 1, \ldots, s_l - 1, s_{m+1}, s_{l+1}, \ldots, s^1_{l+m+1}, \ldots, s^1_n)$$

from σ by deleting s_1, reducing the first s_1 remaining elements of σ by one, and then reordering the last $n-l-m$ elements to be nonincreasing. For $2 \leq i \leq l$, we recursive construct

$$\sigma_i = (s_{i+1} - i, \ldots, s_l - i, s_{l+1} - i, \ldots, s_{l+1} - i, s^{i-1}_{l+m+1}, \ldots, s^i_n)$$

from σ_{i-1} by deleting $s_i - i + 1$, reducing the first $s_i - i + 1$ remaining elements of σ_{i-1} by one, and then reordering the last $n - l - m$ elements to be nonincreasingIn 2012 Yin proved the two following theorems.

In 2012 Yin proved the following two theorems.

Theorem 20 (Yin [87]) *σ is potentially A_b-graphic if and only if σ_b is graphic.*

Proof. See [87]. □

Theorem 21 (Yin [87]) *Let $n \geq l + m$ and let $\sigma = (s_1, \ldots, s_n)$ be a nonincreasing graphic sequence. If $s_{l+m} \geq l + m - 2$, then σ is potentially $A_{l,m}$-graphic.*

Proof. See [87]. □

Using the algorithm Erdős-Gallai-Linear [48] or algorithm Havel-Hakimi-Linear [45] we can decide in worst case in $O(n)$ time whether π is graphic.

The following theorem allows to decrease the expected running time of Havel-Hakimi-Split.

Theorem 22 (Yin [87]). *Let $n \geq l + m$ and let $\sigma = (s_1, \ldots, s_n)$ be an n-regular sequence. If $s_{r+s} \geq 2l + m - 2$, then σ is A_{l+m}-graphic.*

Proof. See [87]. □

In the same paper Yin [87] published a Havel-Hakimi type algorithm which constructs the corresponding $J_{l,m}$-graph.

Let $A_n = (a_1, \ldots, a_n)$ be an n-regular sequence, and $B_n = (b_1, \ldots, b_n)$ a sequence of nonnegative integers with $a_i \leq b_i$ and $a_i + b_i \geq a_{i+1} + b_{i+1}$ for $i = 1, \ldots, n - 1$. $(A_n; B_n)$ is said to be *potentially* K_{m+1}*-graphic* (resp. A_{m+1}-graphic) if there exists a graph G with vertices $v_1, \ldots, v_n$ such that $a_i \leq v_i(G) \leq b_i$ for $i = 1, \ldots, n$ and G contains K_{m+1} as a subgraph. In 2013 Yin [88] characterized $(A_n; B_n)$ so, that it is potentially A_{m+1}-graphic and potentially K_{m+1}-graphic.

In 2014 Yin [89] characterized the sequences having a realization containing an arbitrary subgraph.

In 2014 Pirzada and Chat proved the following assertion.

Theorem 23 (Pirzada, Chat [67]) *If G_1 is a realization of $\sigma_1 = (s_1^1, \ldots, s_l^1)$, containing K_l as a subgraph and G_2 is a realization of $\sigma_2 = (s_1^2, \ldots, s_m^2)$ containing K_m as a subgraph, then the degree sequence $\sigma = (s_1, \ldots, s_{l+m})$ of the join of G_1 and G_2 is K_{l+m}-graphic.*

Proof. See [67] □

4 Known results on split sequences

The ***girth*** $g(G)$ of a graph G containing at last one cycle is the length of its shortest cycle. The girth of an acyclic graph is infinite. A graph G is called ***chordal***, if it does not contain an induced subgraph with finite girth greater then 3.

In 1977 Földes and Hammer gave the following characterization of psplit graphs.

Theorem 24 (Földes, Hammer [22]) *For any graph* G *the following three conditions are equivalent:*

(i) G *and* $\overline{G}$ *both are chordal;*

(ii) $V(G)$ *can be partitioned into a complete and an empty set;*

(iii) G *does not contain an induced subgraph isomorphic to* $2K_2$, C_4 *or* C_5.

In 1993 Blázsik, Hujter, Pluhár and Tuza [5] characterized the *pseudo split graphs* defined as graphs with no induced C_4 and $2K_2$ (see also [2]). In 1998 Maffray and Preissmann [58] proved the following assertion.

Theorem 25 (Maffray, Preissmann [58]) G *is a pseudo split graph with a nonincreasing degree sequence* $\delta = (d_1, \ldots, d_n)$, *then* G *is a pseudo split graph, if* G *is a split graph or*

$$\sum_{i=1}^{q} d_i = q(q+4) + \sum_{i=m+1}^{n} d_i \tag{7}$$

and

$$d_{q+1} = d_{q+2} = d_{q+3} = d_{q+4} = d_{q+5} = q+2, \tag{8}$$

where $q = \max(i \mid d_i \geq q+4)$.

Proof. See [58]. □

The following theorem allows to design a linear time algorithm recognising the psplit graphs in linear time.

Theorem 26 (Golumbic [26]; Hammer, Simeone [34]; Tyshkevich [79]; Tyshkevich, Melnikow, Kotov [81], Wikipedia [82]) *Let the nonincreasing degree sequence of a graph* G *be* $\delta = (d_1, \ldots, d_n)$, *and let* m *be the largest value of* i

such that $d_i \geq i-1$. *Then* G *is a psplit graph if and only if*

$$\sum_{i=1}^{m} d_i = m(m-1) + \sum_{i=m+1}^{n} d_i. \tag{9}$$

If this is the case, then the m *vertices with the largest degrees form a maximum clique in* G*, and the remaining vertices constitute and independent set.*

Proof. See [34]. □

An extremal problem for 1-graphic sequences to be potentially K_l^1-graphic was considered by Erdős, Jacobson and Lehel [19], and solved by Gould et al. [27] and Li et al. [56, 57]. Recently, Yin [85] generalized this extremal problem and the Erdős-Jacobson-Lehel conjecture from 1-graphs to b-graphs.

Different split graphs are closely connected with the problems of colorings of graphs, since the clique number gives a lower bound of coloring number. E.g. Erdős and Gyárfás [18], Gyárfás and Lehel [29] deal with coloring of psplit graphs. Yin and Li [90] gave sufficient conditions for graphic sequences to have a realization with prescribed clique size.

There are many publications on the maximal clique algorithms. Recently Zavalnij [91] analysed parallel algorithms for the the solution of the maximal clique problem. This problem was earlier studied e.g. in [60, 63, 64, 75, 72, 73, 76].

In 2000 observed a bijection between nonisomorphic psplit graphs and minimal covers of a set by its subsets. Using the formula proved by Clarke [15] for the number of minimal covers, Royle [71] proved the following assertion, giving the number $p(n)$ of the nonisomorphic psplit graphs on n vertices. This result was published also by Tyshkevich and Chernak in 1990.

Theorem 27 (Royle [71], Tyshkevich, Chernak [80]) *If* $n \geq 1$*, then*

$$p(n) = \sum_{k=1}^{n} t(n-k, k), \tag{10}$$

where

$$t(n,k) = \frac{1}{n!k!} \sum_{\alpha \in P_n, \beta \in P_k} \binom{n}{\alpha} \binom{\beta}{k} \prod_i \left(\left(\prod_j 2^{(\alpha_i, \beta_j)} \right) \right), \tag{11}$$

$$\binom{n}{\alpha} = \frac{n!}{\prod_i \mu_i! i^{\mu_i}}, \tag{12}$$

where μ_i *is the number of* i*'s in the partition* α, (u, v) *denotes the greatest common divisor of* u *and* v, P_n *is the set of all partitions of* n.

Figure 3 contains the values of $p(n)$ for $n = 1, \ldots, 20$. This data are taken from *Encyclopedia of Interger Sequences* [39] containing the values of $p(n)$ for $n = 1, \ldots, 40$ vertices.

n	$p(n)$	n	$p(n)$
1	1	11	64 956
2	2	12	501 696
3	4	13	5 067 146
4	9	14	67 997 750
5	21	15	1 224 275 498
6	56	16	29 733 449 510
7	164	17	976 520 265 678
8	557	18	43 425 320 764 422
9	2 223	19	2 616 632 636 247 976
10	10 766	20	213 796 933 371 366 930

Figure 3: The number $p(n)$ of nonisomorphic psplit graphs for $n = 1, \ldots, 20$ vertices.

In 1995 Nikolopoulos proposed a constant-time parallel algorithm for the recognition of psplit graphs.

Theorem 28 (Nikolopoulos [62]) *Let* $G(V, E)$ *be a graph with* $|V| = n$ *and* $|E| = m$. *Then algorithm* Split-Recognition *decides—using* $O(nm)$ *processors— in* $O(1)$ *time whether* G *is a psplit graph.*

Proof. See [62]. □

Several papers deals with the hamiltonicity of split graphs. E.g. in 1980 Burkard and Hammer [11] gave a necessary condition of the hamiltonicity of psplit graphs.

In 1988 Peemüller analyzed the condition of Burkard and Hammer and proved new necessary conditions for hamiltonian psplit graphs.

Theorem 29 (Peemüller [65]) *Let* $G = (C, I, E_1, E_2)$ *be a psplit graph with* $|C| < |I|$. *If* G *is hamiltonian, then*

$$2|X'| - m(X', Y') + f(X', Y') \leq m(Y', \overline{Y}) - fY', \overline{Y}), \tag{13}$$

for all $X' \subset X$, $X' \neq \emptyset$, , *while* m, f *and* N *is defined in* [65].

Proof. See [65]. □

In 1998 Woeginger proved the following property of the taughness [14] of psplit graphs solving a problem posed by Kratsch, Lehel and Müller in 1996 [53].

Theorem 30 (Woginger [83]) *The toughness of psplit graphs can be computed in polynomial time.*

Proof. See [83]. □

It is worth to remark that in 1990 Burkard, Hakimi and Schmeichel [3] that recognising of the toughness of a graph is NP-hard.

In 1999 Brandstädt, Le and Spinrad [10], in 2012 Almeida, Mello and Morgana [1] studied the classification problem of split graphs.

In 2006 Kratsch, McConnell, Mehlhorn, and Spinrad [52] reviewed certifying algorithms for recognizing interval graphs and permutation graphs

In 2008 Ibarra [42] studied fully dynamical algorithms of maintenance of psplit graphs.

In 2009 Heggernes and Mancini [38] analysed the minimal completion of psplit graphs.

In 2012 LaMar [54] defined directed psplit graphs and derived conditions for integer sequences to be degree sequences of directed psplit graphs.

In 2014 Habib and Mamcarz [31] investigated split decompositions of graphs.

5 New results for A-graphic sequences

In the next result, we use the Havel-Hakimi procedure to test whether a b-graphic sequence δ is potentially $A^b_{l,m}$-graphic.

Theorem 31 *Let $b \geq 1$ and $n \geq 1$. A b-graphic sequence $\sigma = (s_1, \ldots, s_n)$ is potentially $A^b_{l,m}$-graphic if and only if σ_l is b-graphic.*

Proof. Assume that σ is potentially $A^b_{l,m}$-graphic. Then σ has a realization G with the vertex set $V(G) = \{v_1,\ ldots, v_n\}$ such that $d_G(v_i) = s_i$ for $(1 \leq i \leq n)$ and G contains $J^b_{l,m}$ on the vertices $v_1, \ldots, v_{l+m}$, where $l + m \leq n$, so that $V^b(K_l) = \{v_1, \ldots, v_l\}$ and $V(\overline{K}^b_m) = \{v_{l+1}, \ldots, v_{l+m}\}$. We will show that by applying a sequence of b-exchanges to G in order that there is one such realization G' such that $G' \setminus v_1$ has degree sequence σ_1. If not, we may choose such a realization H of b-graphic sequence σ such that the number of vertices adjacent to v_1 in $\{v_{l+m+1}, \ldots, v_{s_1+1}\}$ is maximum. Let $v_i \in \{v_{l+m+1}, \ldots, v_{s_1+1}\}$

and assume that there is no edge between v_1 and v_i and let $v_j \in \{v_{s_1+2}, \ldots, v_n\}$ and there are b edges between v_1v_j. We may assume that $s_i > s_j$, since the order of i and j can be interchanged if $s_i < s_j$. Hence there is a vertex $v_t, t \neq i, j$ such that there are b edges between v_i and v_t and no edge between v_j and v_t. Clearly $G = \big(H \setminus \{v_1^b v_j, v_i^b v_t\}\big) \bigcup \{v_1^b v_i, v_j^b v_t\}$—where $v_i^b v_j$ means that there are b edges between v_i and v_j—is a realization of σ such that $d_G(v_i) = s_i$ for $1 \leq i \leq n$, G contains $S^b_{l,m}$ on $v_1, \ldots, v_{l+m}$ with $V^b(K_l) = \{v_1, \ldots, v_l\}$ and $V(\overline{K}^b_m) = \{v_{l+1}, \ldots, v_{l+m}\}$ and G has the number of vertices adjacent to v_1 in $\{v_{l+m+1}, \ldots, v_{s_1+1}\}$ larger than that of H. This contradicts the choice of H. Repeating this procedure, we can see that σ_i is potentially A^b_{l-i}-graphic successively for $i = 2, \ldots, l$. In particular, σ_l is b-graphic.

Conversely suppose that σ_l is b-graphic and is realized by a graph G_l with a vertex set $V(G_l) = \{v_{l+1}, \ldots, v_n\}$ such that $d_{G_l}(v_i) = s_i$ for $l + 1 \leq i \leq n$. For $i = l, \ldots, 1$ form G_{i-1} from G_i by adding a new vertex v_i that is adjacent to each of $v_{i+1}, \ldots, v_{l+m}$ with b-edges and also to the vertices of G_i with degrees $s^{i-1}_{l+m+1} - b, \ldots, s^{i-1}_{d_i+1} - b$. Then for each i, G_i has degrees given by π_i and G_i contains $J^b_{l-i,m}$ on $l + m - i$ vertices $v_{i+1}, \ldots, v_{l+m}$ whose degrees are $s_{i+1} - ib, \ldots, s_{l+m} - ib$ so that $V(K^b_{l-i}) = \{v_{i+1}, \ldots, v_l\}$ and $V(\overline{K}^b_m) = \{v_{l+1}, \ldots, v_{l+m}\}$. In particular, G_0 has degrees given by σ and contains $S^b_{l,m}$ on $l + m$ vertices $v_1, \ldots, v_{l+m}$ whose degrees are $s_1, \ldots, s_{l+m}$ so that $V(K^b_l) = \{v_1, \ldots, v_l\}$ and $V(\overline{K}^b_m) = \{v_{l+1}, \ldots, v_{l+m}\}$. Hence the result follows. □

Now we prove a sufficient condition for a b-graphic sequence to be potentially A^b_l-graphic.

Theorem 32 *Let $n \geq l + m$ and let $\sigma = (s_1, \ldots, s_n)$ be a b-graphic sequence. If $s_{l+m} \geq 2bl + bm - 2$, then σ_i is potentially $\overline{A}^b_{l,m}$-graphic.*

Proof. Let $n \geq l + m$ and let $\sigma = (s_1, \ldots, s_n)$ be a nonincreasing b-graphic sequence with $s_{l+m} \geq 2bl + m - 2$. By Theorem 17, σ is potentially K^b_l-graphic and hence by Lemma 33, A^b_l-graphic. Therefore, we may assume that G is a realization of σ with a vertex set $V(G) = v_1, \ldots, v_n$ such that $d_G(v_i) = s_i$, $(1 \leq i \leq n)$ and G contains K^b_l on $v_1, \ldots, v_l$, that is, $V(K^b_l) = \{v_1, \ldots, v_l\}$ and $M = e_G(\{v_1, \ldots, v_l\}, \{v_{l+1}, \ldots, v_{l+m}\})$ (that is, the number of edges between $\{v_1, \ldots, v_l\}$ and $\{v_{l+1}, \ldots, v_{l+m}\}$) is maximum. If $M = blm$, then G contains $\overline{S}^b_{l,m}$ on $v_1, \ldots, v_{l+m}$ with $V(\overline{K}^r_m) = \{v_{l+1}, \ldots, v_{l+m}\}$. In other-words, $sigma$ is potentially $\overline{A}^b_{l,m}$-graphic. Assume that $M < blm$. Then there exists a $v_k \in \{v_1, \ldots, v_l\}$ and $v_m \in \{v_{l+1}, \ldots, v_{l+m}\}, (i \neq j)$ such that $e_G(v_k, v_m) < b$. Let

$$A = N_{G \setminus \{v_l, \ldots, v_{l+m}\}}(v_k) \setminus N_{G \setminus \{v_1, \ldots, v_l\}}(v_m),$$

$$B = N_{G\setminus\{v_1,\dots,v_{l+m}\}}(v_k) \cap N_{G\setminus\{v_1,\dots,v_l\}}(v_m).$$

Then $e_G(x,y) = b$ for $x \in N_{G\setminus\{v_1,\dots,v_l\}}(v_m)$ and $y \in N_{G\setminus\{v_1,\dots,v_{l+m}\}}(v_k)$. Otherwise, if $e_G(x,y) < b$, then $G' = (G \setminus \{vy, v_m x\}) \cup \{v_k v_m, xy\}$ is a realization of π and contains $\overline{J}^b_{l,m}$ on $v_1,\dots,v_{l+m}$ with $V(K^b_l) = \{v_1,\dots,v_l\}$ and $(\overline{K}^b_m) = \{v_{l+1},\dots,v_{l+m}\}$ such that

$$e_{G'}(\{v_1,\dots,v_l\},\{v_{l+1},\dots,v_{l+m}\}) > M,$$

which contradicts the choice of G. Thus B is b-complete. We consider the following two cases.

Case 1. Let $A = \emptyset$. Then $2bl + bm - 2 \le d_k = d_G(v_k) < bl + bm - 1 + b|B|$, and so $|B| \ge bl$. Since each vertex in $N_{G\setminus\{v_1,\dots,v_l\}}(v_m)$ is adjacent to each vertex in B by b edges and $|N_{G\setminus\{v_1,\dots,v_l\}}(v_m)| \ge 2bl + bm - 2 = bl + bm - 1$. It can be easily seen that the b induced subgraph of $N_{G\setminus\{v_1,\dots,v_l\}}(v_m) \cup \{v_m\}$ in G contains $\overline{J}^b_{l,m}$ as a subgraph. Thus π is potentially $\overline{A}^b_{l,m}$- graphic.

Case 2. Let $A \ne \emptyset$. Let $a \in A$. If there are $x, y \in N_{G\setminus\{v_1,\dots,v_l\}}(v_m)$ such that $e_G(x,y) < b$ then $G' = (G_{\setminus\{v_m x, v_m y, v_k a\}}) \cup \{v_k v_m, a v_m, xy\}$ is a realization of σ and contains $\overline{J}^b_{l,m}$ on $v_1,\dots,v_{l+m}$ with $V(K^b_l) = \{v_1,\dots,v_l\}$ and $(\overline{K}^r_m) = \{v_{l+1},\dots,v_{l+m}\}$ such that $e_{G'}(\{v_1,\dots,v_l\},\{v_{l+1},\dots,v_{l+m}\}) > M$ which contradicts the choice of G. Thus $N_{G\setminus\{v_1,\dots,v_l\}}(v_m)$ is b-complete. Since

$$|N_{G\setminus\{v_1,\dots,v_l\}}(v_m)| \ge bl + bm - 1 \text{ and } e_G(v_m, z) = b$$

for any $z \in N_{G\setminus\{v_1,\cdots,v_l\}}(v_m)$, it is easy to see that the induced b-subgraph of $N_{G\setminus\{v_1,\dots,v_l\}}(v_m) \cup \{v_m\}$ in G is b-complete, and so contains $\overline{J}^b_{l,m}$ as a b-subgraph. Thus σ is potentially $\overline{A}^b_{l,m}$-graphic. □

6 New results for split sequences

Let $n \ge l + m$ and let $\sigma = (s_1,\dots,s_n)$ be a nonincreasing sequence of nonnegative integers with $s_l \ge b(l+m) - 1$ and $s_{l+m} \ge bl$. We define sequences $\sigma_1,\dots,\sigma_l$ as follows. We first construct the sequence

$$\sigma_1 = (s_2 - b,\dots,s_l - b, s_{l+1} - b,\dots,s_{l+m} - b, s^1_{l+m+1},\dots,s^1_n)$$

from σ by reducing 1 the largest term that has not already been reduced b times, and then reordering the last $n - l - m$ terms to be nonincreasing. For $2 \le i \le b$, we construct

$$\sigma_i = (s_{i+1} - ib,\dots,s_l - ib, s_{l+1} - br,\dots,s_{l+m} - ib, s^i_{l+m+1},\dots,s^i_n)$$

from

$$\begin{aligned}\sigma_{i-1} =&(s_i-(i-1)b,\dots,s_l-(i-1)b,s_{l+1}-(i-1)b,\dots,\\ &s_{l+m}-(i-1)b,s_{l+m+1}^{i-1},\dots,s_n^{i-1})\end{aligned}$$

by deleting $s_i-(i-1)b$, reducing the first $s_i-(i-1)b$ remaining terms of s_{i-1} by one that has not already been reduced b times, and then reordering the last $n-l-m$ terms to be nonincreasing.

We start with the following lemma.

Lemma 33 *A nonincreasing integer sequence* $\sigma=(s_1,\dots,s_n)$ *is potentially* $A_{l,m}^b$*-graphic if and only if it is potentially* $J_{l,m}^b$*-graphic.*

Proof. We only need to prove that if $\sigma=(s_1,\dots,s_n)$ is potentially $J_{l,m}^b$-graphic, then it is potentially $A_{l,m}^b$-graphic. We may choose a realization G of σ with vertex set $V(G)=\{v_1,\dots,v_n\}$ such that $d_G(v_i)=s_i$ for $1\le i\le n$, the induced b-subgraph $G[\{v_1,\dots,v_{l+m}\}]$ of $\{v_1,\dots,v_{l+m}\}$ in G contains $J_{l,m}^b$ as its b-subgraph and $|V(K_l^b)\cap\{v_1,\dots,v_l\}|$ is maximum. Denote $H=G[\{v_1,\dots,v_{l+m}\}]$. If $|V(K_l^b)\cap\{v_1,\dots,d_l\}|=l$, that is, $V(K_l^b)=\{v_1,\dots,v_l\}$, then σ is potentially $A_{l,m}^b$-graphic. Assume that $|V(K_l^b)\cap\{v_1,\dots,v_l\}|<l$. Then there exists $v_i\in\{v_1,\dots,v_l\}\setminus V(K_l^b)$ and a $v_j\in V(K_l^b)\setminus\{v_1,\dots,v_l\}$. Let $A=N_H(v_j)\setminus(\{v_i\}\cup N_H(v_i))$ and $B=N_G(v_i)\setminus(\{v_j\}\cup N_G(v_j))$. Since $d_G(v_i)\ge d_G(v_j)$. We have $|B|\ge|A|$. Let us choose any subset $C\subseteq B$ such that $|C|=|A|$. Now form a new realization G' of s by a sequence of b-exchanges the b-edges of the star centralized at v_j with end vertices in A with the non b-edges of the star centralized at v_j with end vertices in C, and by a sequence of b-exchange the b-edges of the star centralized at v_i with end vertices in C with the non b-edges of the star centralized at v_i with end vertices in A. It is easy to see that G' contains $J_{l,m}^b$ on $\{v_1,\dots v_{l+m}\}$ so that $|V(K_l^b)\cap\{v_1,\dots,v_l\}|$ is larger than that of G, which contradicts to the choice of G. □

In the next result, we use the result of Fulkerson et al. [24] and prove a necessary and sufficient condition for a b-graphic sequence s to be potentially $J_{l,m}^b$-graphic.

Theorem 34 *Let* $n\ge l+m$ *and* $\sigma=(s_1,\dots,s_n)$ *be a nonincreasing even sequence of nonnegative integers, where* $s_l\ge b(l+m-1)$ *and* $s_{l+m}\ge lb$.

Then σ *is potentially* $J^b_{l,m}$*-graphic if and only if*

$$\sum_{i=1}^{p}(s_i - b(l+m-1)) + \sum_{i=b+1}^{l+p'}(s_i - bl) + \sum_{i=l+m+1}^{l+m+q} s_i \leq$$
$$r(p+p'+q)(p+p'+q-1) - rp(p-1) - 2bpp'$$
$$+ \sum_{i=p+1}^{r} \min\{bq, s_i - b(l+m-1)\}$$
$$+ \sum_{i=l+p'+1}^{l+m} \min\{b(p'+q), s_i - bl\} + \sum_{i=l+m+q+1}^{n} \min\{b(p+p'+q), s_i\}$$

for any $1 \leq l \leq n$, $1 \leq m \leq n$, *for any* p, p' *with* $0 \leq p \leq l$, $0 \leq p' \leq m$ *and* $0 \leq q \leq n-l-m$.

Proof. To prove the necessity, by Lemma 33, let G be a graph with vertex set $V(G) = \{v_1, \ldots, v_n\}$ such that $d_G(v_i) = s_i$ for $1 \leq i \leq n$ and G contains $J^b_{l,m}$ on $v_1, \ldots, v_{l+m}$ with $V(K^b_l) = \{v_1, \ldots, v_l\}$ and $V(\overline{K}^b_m) = \{v_{l+1}, \ldots, v_{l+m}\}$. The removal of the b edges induced by $\{v_1, \ldots, v_l\}$ and the b-edges between $\{v_1, \ldots, v_l\}$ and $\{v_{l+1}, \ldots, v_{l+m}\}$ results in a graph G' in which all degrees in $\{v_1, \ldots, v_l\}$ are reduced by $b(l+m-1)$ and all degrees in $\{v_{l+1}, \ldots, v_{l+m}\}$ are reduced by lb. For $0 \leq p \leq l, 0 \leq p' \leq m$ and $0 \leq q \leq n-l-m$, denote $P = \{v_i | 1 \leq i \leq p\}, P' = \{v_i | l+1 \ leqi \leq l+p'\}, R = \{v_i | p+1 \leq i \leq l\}, R' = \{v_i | l+p'+1 \leq i \leq l+m\}, Q = \{v_i | l+m+1 \leq i \leq q+l+m\}$ and $S = \{v_i | q+l+m+1 \leq i \leq n\}$. The degree sum in the b-subgraph induced by $P \cup P' \cup Q$ is at most $b(p+p'+q)(p+p'+q-1) - bp(p-1) - 2bpp'$. Therefore,

$$m = \sum_{i=1}^{p}(s_i - b(l+m-1)) + \sum_{i=r+1}^{b+p'}(s_i - bl) + \sum_{i=l+m+1}^{l+m+q} s_i$$
$$- b(p+p'+q)(p+p'+q-1) - bp(p-1) - 2bpp'$$

is the minimum number of edges of G' with exactly one end vertex in $P \cup P' \cup Q$. On the other hand, the maximum number of edges of G' with exactly one end

vertex in $R \cup R' \cup S$ is

$$M = \sum_{i=p+1}^{b} \min\{bq, s_i - b(l+m-1)\} + \sum_{i=b+p'+1}^{l+m} \min\{b(p'+q), s_i - bl\}$$

$$+ \sum_{i=l+m+q+1}^{n} \min\{b(p+p'+q), s_i\}$$

We observe that in the graph G', $m \le M$ is true. This proves the necessity.

To prove the sufficiency, we shall use the following well-known result of Fulkerson et al. [24]. Let H be a b-graph on the vertex set $V(H) = \{v_1, \dots, v_n\}$. There exists a b-subgraph $G \subseteq H$ such that every vertex v_i has degree s_i, if and only if

$$\sum_{i=1}^{n} s_i \text{ is even}, \tag{14}$$

and for every $A, B \subseteq V(H)$ such that $A \cap B = s$, we have

$$\sum_{v_i \in A} s_i \le \sum_{v_i \in A, v_j \in V(H) \setminus B} e_H(v_i, v_j) + \sum_{v_i \in B} s_i. \tag{15}$$

We now continue to proceed with the proof of sufficiency. Let $n \ge l + m$ and $\sigma = (s_1, \dots, s_n)$ be a nonicreasing sequence of nonnegative integers, where $s_l \ge l + m - 1, s_{l+m} \ge l$ and $\sum_{i=1}^{n} s_i$ is even. Let $s' = (s'_1, \dots, s'_n)$, where $s'_i = s_i - l - m + 1$ for $1 \le i \le l$, $s'_i = s_i - b$ for $l + 1 \le i \le l + m$ and $s'_i = s_i$ for $l + m + 1 \le i \le n$. Let H be the graph obtained from K_n^b with vertex set $V(K_n^b) = \{v_1, \dots, v_n\}$ by deleting all edges of the complete b-subgraph induced by $\{v_1, \dots, v_l\}$ and all edges between $\{v_1, \dots, v_l\}$ and $\{v_{l+1}, \dots, v_{l+m}\}$. It is easy to see that s is potentially $A_{l,m}^b$-graphic if and only if H has a subgraph G with the degree sequence s' such that every vertex v_i has degree s'_i. Observe that between two disjoint odd cycles of H there is an edge. Therefore, H satisfies the odd-cycle condition and we apply (14) and (15).

Let $K = \{v_1, \dots, v_l\}, K' = \{v_{l+1}, \dots, v_{l+m}\}$ and $A, B \subseteq V(H)$ such that $A \cap B = s$. Let $A_1 = A \cap K, A'_1 = A \cap K', A_2 = A \setminus (K \cup K'), B_1 = B \cap K, B'_1 = B \cap K$, $B_2 = B \setminus (K \cup K')$ and set $p = |A_1|$, $p' = |A'_1|$, $q = |A_2|, b_1 = |B_1|, b'_1 = |B'_1|, b_2 = |B_2|$. For convenience, we denote

$$L(p, p', q) = \sum_{i=1}^{p} (s_i - b(l+m-1)) + \sum_{i=r+1}^{r+p'} (s_i - bl) + \sum_{i=r+s+1}^{r+s+q} s_i, \tag{16}$$

$$R(p,p',q) = b(p+p'+q)(p+p'+q-1) - bp(p-1) - 2bpp' + \sum_{i=p+1}^{b} \min\{bq, s_i - b(l+m-1)\} + \sum_{i=r+p'+1}^{l+m} \min\{b(p'+q), s_i - bl\} + \sum_{i=l+m+q+1}^{n} \min\{b(p+p'+q), s_i\},$$

$$L'(A,B) = \sum_{v_i \in A} s'_i = \sum_{v_i \in A_1} \{s_i - b(l+m-1)\} + \sum_{v_i \in A'_1} \{s_i - bl\} + \sum_{v_i \in A_2} s_i,$$

$$R'(A,B) = \sum_{v_i \in A, v_j \in V(H)\setminus B} e_H(v_i,v_j) + \sum_{v_i \in B} s'_i$$

$$= \sum_{v_i \in A, v_j \in V(H)\setminus B} e_H(v_i,v_j) + \sum_{v_i \in B_1} (s_i - b(l+m-1)) + \sum_{v_i \in B'_1} (s_i - lb) + \sum_{v_i \in B_2} s_i.$$

Clearly, $L'(A,B) \le L(p,p',q)$. Further $\sum_{v_i \in A, v_j \in V(H)\setminus B} e_H(v_i,v_j)$ is the number of counting the edges of H between A and $V(H)\setminus(A\cup B)$ and double counting the edges induced by A. Thus we get

$$\sum_{v_i \in A, v_j \in V(H)\setminus B} e_H(v_i,v_j)$$
$$= r(p+p'+q)(p+p'+q-1) - bp(p-1) - 2bpp' + qb(l-p-b_1)$$
$$+ b(p'+q)(m-p'-b'_1) + b(p+p'+q)(n-l-m-q-b_2)$$
$$= b(p+p'+q)(p+p'+q-1) - bp(p-1) - 2bpp' + \sum_{i=p+1}^{l-b_1} q$$
$$+ \sum_{i=l+p'+1}^{l+m-b'_1} (p'+q) + \sum_{i=l+m+q+1}^{n-b_2} (p+p'+q).$$

Therefore,

$$\begin{aligned}
R'(A,B) = & \sum_{v_i\in A, v_j\in V(H)\setminus B} e_H(v_i,v_j) + \sum_{v_i\in B_1}(s_i - b(l+m-1)) \\
& + \sum_{v_i\in B'_1}(s_i - lb) + \sum_{v_i\in B_2} s_i \\
& \geq b(p+p'+q)(p+p'+q-1) - bp(p-1) - 2bpp' + \sum_{i=p+1}^{l-b_1} q \\
& + \sum_{i=l+p'+1}^{l+m-b'_1}(p'+q) + \sum_{i=l+m+q+1+1}^{n-b_2}(p+p'+q) \\
& + \sum_{i=l-b_1+1}^{l}(s_i - b(l+m-1)) + \sum_{i=l+m-b'_1+1}^{l+m}(s_i - br) + \sum_{i=n-b_2+1}^{n} s_i \\
& \geq r(p+p'+q)(p+p'+q-1) - bp(p-1) - 2bpp' \\
& + \sum_{i=p+1}^{l}\min\{bq, s_i - b(l+m-1)\} + \sum_{i=l+p'+1}^{l+m}\min\{b(p'+q), s_i - bl\} \\
& + \sum_{i=l+m+q+1}^{n}\min\{r(p+p'+q), s_i\} \\
& = R(p,p',q).
\end{aligned}$$

It follows from $L(p,p',q) \leq R(p,p',q)$ that $L'(A,B) \leq R'(A,B)$. By (14) and (15) H is a b-subgraph G with the degree sequence s' such that every vertex v_i has degree s'_i. Hence s is potentially $A^b_{l,m}$-graphic. Thus the sufficiency is proved. □

It is easy to enumerate the $J_{l,m}$-split graphs on n vertices.

Theorem 35 *If* $l \geq 0$, $m \geq 1$, $l+m \geq 1$, *and* $b \geq 1$, *then*

1. *there are* $\lambda(b,l,m) = \binom{l+m}{l}$ *labeled* $J^b_{l,m}$ *and they are isomorphic;*
2. *there are* $\beta(b,l,m) = l+m$ *nonisomorphic* $J^b_{l,m}$.

Proof.

1. Since $J_{l,m}$ has $l+m$ vertices, therefore there are $\binom{l+m}{l}$ ways to choose the vertices of K_l. If we consider two different labeled $J_{l,m}$ jsplit graphs, then

the vertices of the clique parts correspond to each other, and the independent vertices of these graphs also correspond to each other, therefore these graphs are isomorphic.

2. Formally $J^b_{0,l+m}$, $J^b_{1,l+m-1}, \ldots, J^b_{0,l+m}$ are $l+m+1$ different possibilities, but the last two split graphs are isomorphic, therefore $\beta(l,m) = l+m$.

□

We can remark, that if $m \geq 1$, then $J_{l,m}$ is also a $J_{l+1,m-1}$ split graph.

7 Known algorithms for graphic sequences

In this section at first we present the classical Havel-Hakimi (HH) algorithm, then its testing version (HHL), which even in the worts case in $O(n)$ time decides whether an integer sequence is $(0,1,n)$-graphic. Then we describe algorithm Havel-Hakimi-pqlm-Split which in $O(n)$ time decides the similar problem for potentially $J^{(p,q)}_{l,m}$-graphic sequences, further a Havel-Hakimi type algorithm for recognition of (a,b,n).

7.1 Havel-Hakimi algorithm (HH)

If $n = 1$, then there exists one $(0,1,n)$-graphic sequence: (0). If $n \geq 2$, then Havel-Hakimi theorem (Theorem 9) gives a necessary and sufficient condition.

Input. n: the length of the sequence s $(n \geq 2)$;
$\sigma = (s_1, \ldots, s_n)$: the investigated n-regular sequence.

Output. L: logical variable ($L = 0$ signalizes, that σ is not graphic, while $L = 1$ means, that σ is $(0,1,n)$-graphic).

Working variable. i: cycle variables.

```
Havel-Hakimi(n, σ)
01 L = 0                                   // line 01–07: test of the elements of s
02 for i = 1 to n − 1
03     if s_{s_i+i} == 0                   // lines 03–04: s is not graphic
04         return L
05     for j = i + 1 to s_i + i
06         s_j = s_j − 1
07     sort (s_{i+1}, ..., s_n) in decreasing order
08 L = 1                                   // lines 08–09: s is graphic
09 return L
```

7.2 HAVEL-HAKIMI-TESTING-LINEAR algorithm (HHTL)

The original Havel-Hakimi algorithm in worst case requires quadratic time to test the $(0,1,n)$-regular sequences. Using the concepts weight point, reserve and cutting point we reduced the worst running time to $O(n)$.

The definition of the *weight point* w_i belonging to s_i was introduced in [48] in connection with ERDŐS-GALLAI-LINEAR and it is as follows. w_i is the largest k $(1 \le k \le n)$ having the property $s_k \ge i$. But if $s_1 < i$, then $w_i = 0$. EGL exploits the property w_i ensuring that if $i \le w_i$, then the key expression $\min j, s_k$ in the Erdős-Gallai theorem equals to i, otherwise equals to s_k.

Here we extend the definition to be applicable also in the proof of the linearity of CHUNGPHAISAN-ERDŐS-GALLAI. Now let w_i the largest k $(1 \le k \le n)$ having the property $s_k \ge bi$. But if $s_1 < bi$, then let $w_i = 0$. In the case $b = 1$ the new definition results the old one.

In HHL the weight point w_i determines the increment of the tail capacity when we switch to the investigation of the next element of σ.

The *remainder* r_i belonging to s_i is defined as the unused part of the actual tail capacity and can be computed by the formulas

$$r_i = w_1 - 1 - s_1$$

and

$$r_i = w_i - r_{i-1} - s_i \quad \text{for } 1 \le i \le n-1.$$

The *cutting point* y_i belonging to s_i is $\max(i, w_i)$.

The programs of this paper are written using the pseudocode conventions descibed in [16].

Input. n: number of vertices $(n \ge 1)$;
$\sigma = (s_1, \ldots, s_n)$: the investigated n-graphic sequence.

Output. L: logical variable.

Work variables. i: cycle variable;
$r = (r_1, \ldots, r_n)$: r_i the reserve belonging to s_i;
$w = (w_1, \ldots, w_n)$: w_i the weight point belonging to s_i;
$H = (H_1, \ldots, H_n)$: H_i is the sum of the first i elements of s.

HAVEL-HAKIMI-TESTING-LINEAR(n, s)

```
01 L = 0                                   // lines 01: set the probable value
02 if s_1 == 0      // lines 02–04: test of the sequence consisting of only zeros
03    L = 1
04    return L
```

```
05 if s_{s_1+1} == 0                    // lines 05–06: test of s_{s_1} in constant time
06    return L
07 H_1 = s_1                            // line 07: initialization of H
08 for i = 2 to n                       // lines 08–09: further H_i's
09      H_i = H_{i-1} + s_i
10 if H_n is odd                        // lines 10–11: test of the parity
11    return L
12 w_1 = n   // lines 12–15: computation of the first weight point and reserve
13 while s_{w_1} < 1
14          w_1 = w_1 - 1
15 r_1 = w_1 - 1 - s_1                  // lines 15–24: testing of σ
16 s_{n+1} = 0
17 for i = 2 to n - 1
18      if s_i ≤ i or s_{i+1} = 0
19         L = 1
20         return L
21      w_i = w_{i-1}
22      while s_{w_i} < i and w_i > 0
23            w_i = w_i - 1
24      if s_i > w_i - 1 + r_{i-1}      // line 24: Is σ graphic?
25         return L                     // line 25: σ is not graphic
26      r_i = w_i + r_{i-1} - s_i
27 L = 1                                // lines 27–28: σ is graphic
28 return L
```

Theorem 36 *The running time of* Havel-Hakimi-testing-Linear *is in best case* $\Theta(1)$*, and in worst case is* $\Theta(n)$**.**

Proof. If the condition in line 2 holds, then the running time is $\Theta(1)$. If not, then we reduce the actual w at most n times and the remaining operations require $O(1)$ operations for all reductions. □

7.3 Erdős-Gallai-Chungphaisan-Linear algorithm (EGChL)

The following algorithm tests the potential degree sequences of $(0, b, n)$-graphs. It is based on Theorem 13.

Input. n: number of vertices ($n \geq 1$);
$\sigma = (s_1, \ldots, s_n)$: a $(0, b, n)$-regular sequence;
b: the maximal permitted number of arcs between two vertices.

Output. 1 or 0: 1, if s is $(0, b, n)$-graphic and 0 otherwise.
Work variable. i: cycle variable;
$r = (r_1, \ldots, r_n)$: r_i is the reserve belonging to s_i;
$w = (w_1, \ldots, w_n)$: w_i is the weightpoint belonging to s_i.

ERDŐS-GALLAI-CHUNGPHAISAN-LINEAR(n, σ, b)

```
01 H_1 = s_1                                   // line 01: initialization of H_1
02 for i = 2 to n − 1          // line 02–03: computation of the elements of H
03     H_i = H_{i−1} + s_i
04 if H_n is odd                               // line 04–05: test of the parity
05   return 0
06 w = n                      // lines 06: initialization of the first weight point
07 for i = 1 to n − 1                          // lines 07–12: test of σ
08     while s_w < ib and w > 0
09           w = w − 1
10     y = max(i, w)
11     if H_i > bi(y − 1) + H_n − H_y
12       return 0
13 return 1                                    // line 13: acceptance of σ
```

Theorem 37 *The running time of* ERDŐS-GALLAI-CHUNGPHAISAN-LINEAR *is* $\Theta(n)$ *in all cases.*

Proof. Lines 01–05 require $\Theta(n)$ time. Since the value of w is strictly decreasing, lines 06–13 require $O(n)$ time, therefore the running time is $\Theta(n)$ in all cases. □

Let us consider two examples. Let $b = 3$ and $\sigma' = (13, 10, 5, 5, 4, 1)$. $H_6 = 38$ is even. If $i = 1$, then $w_i = y = 5$ and the condition in line 11 is not satisfied $(13 \leq 3 \cdot 1 \cdot (5 - 1))$. If $i = 2$, then $w_i = y = 2$ and the condition in line 11 holds $(23 > 3 \cdot 2 \cdot (2 - 1)) + 5 + 5 + 4 + 1$, therefore σ is *not* $(0, 3, 6)$-*graphic.*

Let b remain 3, but change σ to $\sigma' = (13, 10, 5, 5, 4, 3)$. The first difference comparing with the previous example comes when $i = 2$. Now $23 \leq 3 \cdot 2 \cdot (2 - 1)) + 5 + 5 + 4 + 3$, and the condition in line 11 holds for $i = 3$, 4 and 5 too, therefore σ' is $(0, 3, 6)$-graphic.

Using Corollary 15 it is easy to test an (a, b, n)-regular sequence σ whether it is (a, b, n)-graphic. We use EGCHL with input sequence $\sigma' = (s_1 - a(n - 1), \ldots, s_n - a(n - 1))$.

8 Known algorithms for split sequences

In this section we describe the linear time algorithm proposed for the recognition and reconstruction of potentially psplit and jsplit sequences.

8.1 Hammer-Simeone-PSplit algorithm (HSPS)

The following algorithms was proposed in 1981 by Hammer and Simeone [34]. Ist base is Theorem 26.

Let G be a graph with degree sequence $d = (d_1, \ldots, d_n)$.

Input. n: number of elements of δ;
$\delta = (d_1, \ldots, d_n)$: a graphic sequence.

Output. 1 or 0: 1, if d is potentially psplit sequence.

Work variable. i, k: cycle variables;
Σ_1, Σ_2: actual sums of the degrees.

```
Hammer-Simeone-Linear(n, δ)
01 k = 0                                  // line 01–02: initialization of k and S
02 Σ1 = 0
03 while d_{k+1} ≥ k − 1 and k < n         // line 03–07: computation of m
04         m = k + 1
05         Σ1 = Σ1 + d_k
06         k = k + 1
07 Σ2 = m(m − 1)
08 for i = m + 1 to n                     // lines 08–09: computation of Σ2
09      Σ2 = Σ2 + d_i
10 if Σ1 ≠ Σ2                            // lines 10–11: G is not psplit graph
11    return 0
12 return G i'is psplit, maximal clique size is m // line 12: G is psplit graph
```

Theorem 38 *Let G a graph with degree sequence δ. Algorithm* Hammer-Simeone-Linear *decides, if G is a psplit graph and computes the maximal clique size in $\Theta(n)$ time.*

Proof. Lines 01–02 require $O(1)$ time, lines 03–09 $\Theta(n)$ time and lines 10–12 $O(1)$ time. □

8.2 Further linear algorithms for psplit sequences

In 1980 Golumbic [26], in 2003 Feder et al. [20], in 2007 Heggernes and Kratsch [37] proposed linear time algorithm for the recognition of psplit graphs.

8.3 Havel-Hakimi-Testing-JSplit algorithm (HHJST)

In 2012 Yin [87] described HHJST, a Havel-Hakimi type linear algorithm for the recognition of potentially jsplit sequences.

9 New algorithms

In this section we present two simple algorithms, which decide whether a sequence of nonnegative integers is A_l^b-graphic or $J_{l,m}^b$-graphic, and if the answer is yes, then they compute the maximal suitable l too.

These algorithms require in worst case only $O(n)$ time even for (a, b, n)-regular input, and are quicker for (a, b, n)-graphic input., since then the sorting can be omitted.

We remark, that earlier only for pseudo-split graphs was published a linear time testing algoritm [58].

9.1 Algorithm Ab-l-Max

For given sequence $\sigma = (s_1, \ldots, s_n)$ of nonnegative integers and given nonnegative integer b algorithm A-b-l-Max computes the maximal l for which the sequence s is A_l^b-graphic.

Input. $n \geq 1$: the length of the sequence s;
$\sigma = (s_1, \ldots, s_n)$: a sequence of nonnegative integers;
b: the maximal permitted number of arcs between two different vertices.

Output. l: the maximal value for which d is A_l^b-graphic.

Work variable. i: cycle variable.

A-b-l-Max(n, σ, b)
01 Counting-Sort(n, σ) // line 01: sorting of σ
02 $l = 1$ // line 02: initialization of l
03 **while** $s_{l+1} \leq bl$ and $l < n$ // line 03–04: computation of l
04 $l = l + 1$
05 **return** l 'is the maximal value' // line 05: return of the maximal l

Theorem 39 *Let* b, l *and* n *be positive integers. Algorithm* A-b-l-Max *computes the maximal* l *for which* $\sigma = (s_1, \ldots, s_n)$ *is* A_l^b*-graphic in* $\Theta(n)$ *time.*

Proof. Let G be a b-graph and $s' = (s'_1, \ldots, s'_n)$ be the nonincreasingly sorted sequence consisting from the elements of s. K_l^b contains l vertices whose degrees are equal to $b(l-1)$. Therefore to find the maximal size K_l^b which is a subgraph

of G it is sufficient to find the maximal j satisfying $s'_j \geq b(j-1)$. In lines 01–04 A-b-l-Max computes this maximal l.

Lines 01 of A-b-l-Max requires $\Theta(n)$ time, lines 02 and 05 $O(1)$ time, and lines 03–04 require $O(n)$ time, so the best and worst running times of this algorithm are both $\Theta(n)$. □

As an example consider the sequence $\sigma = (6\ 6\ 1\ 6)$ and $b = 2$. Then $\sigma' = (6\ 6\ 6\ 1)$ and Ab-l-Max returns with $l = 3$. Indeed G contains K_3 as a subgraph but it does not contain K_4 as a subgraph. Since the sum of the elements of s is odd, according to theorem of Erdős and Gallai [17] s is not graphic, that is an A_l^b-graphic sequences are not always graphic.

9.2 Algorithm J-b-l-Max

For given sequence $\sigma = (s_1, \ldots, s_n)$ of nonnegative integers and given nonnegative integer b algorithm S-b-l-Max computes the maximal l for which the sequence σ is $S^b_{l,n-l}$-graphic.

If K_l^b and K_m^b are vertex disjoint and G is the join of K_l^b and $\overline{K}_m$, then in G the degrees of the vertices of K_l are equal to $b(l-1+m)$, while the degrees of the vertices of K_m are equal to bm. This observation is the base of the following algorithm S-b-l-Max.

Input. n: the length of the degree sequence s;
$\sigma = (s_1, \ldots, s_n)$: a sequence of nonnegative integers;
b: the maximal permitted number of edges between two different vertices.

Output. l: the maximal value for which σ is $S^b_{l,n-l}$-graphic or the message 'σ is not $(b, l, n-l)$-graphic'.

Work variable. i: cycle variable.

J-b-l-Max(n, σ, b)

```
01 if s_1/b is not integer                    // line 01–02: constant time test
02    return 'σ is not (b, l, n − l)-graphic'   // line 02: σ is
                                                  not (b, l, n − l)-graphic
03 Counting-Sort(n, σ)                        // line 03: sorting of σ
04 l = 1                                      // line 04: initialization of l
05 while s_{l+1} == s_1 and l < n             // line 05–06: computation of l
06         l = l + 1
07 if s_{l+1} ≠ bl                  // line 07–08: σ is not (b, l, n − l)-graphic
08    's is not (b, l, n − l)-graphic'
09 return l 'is the maximal value'
```

Theorem 40 *Algorithm* J-b-l-MAX *computes the maximal* l *for which* $\sigma = (s_1, \ldots, s_n)$ *is* $S^b_(l, n-l)$*-graphic in* $\Theta(n)$ *time.*

Proof. Let b, l, m and n be positive integers. Let G be a b-graph and $\sigma = (s'_1, \ldots, s'_n)$ be the nonincreasingly sorted sequence consisting from the elements of s'.

The next part of the proof is similar to the corresponding part of Theorem 39.

Line 01 of J-b-l-MAX requires $\Theta(n)$ time and lines 02-05 require $O(n)$ time, so the best running time is $\Theta(1)$ and the worst running time is $\Theta(n)$. □

We remark that if the input of J-b-l-MAX is sorted, then we can omit lines 03 and 04, and using logarithmic search we can reduce the worst case running time to $\Theta(\log n)$.

Acknowledgement

The authors thank the useful remarks of the unknown referee and papers of Professor Jian-Hua Yin (Hainan University).

References

[1] S. M. de Almeida, C. P. de Mello, A. Morgana, Classification problem for split graphs, *J. Brazilian Comp. Soc.*, **18,** (2) (2012) 95–101. ⇒266

[2] M. D. Barrus, Hereditary unigraphs and Erdős-Gallai equalities, *Discrete Math.*, **313,** (21) (2013) 2469–2481. ⇒263

[3] D. Bauer, S. L. Hakimi, E. Schmeichel, Recognising of tough graphs is NP-hard, *Discrete Appl. Math.*, **28** (1995) 191–195. ⇒266

[4] C. Benzaken, P. L. Hammer, D. De Werra, Split graphs of Dilworth number 2, *Discrete Math.*, **55,** (2) (1985) 123–127. ⇒256

[5] Z. Blázsik, M. Hujter, A. Pluhár, Zs. Tuza, Graphs with no induced C_4 and $2K_2$, *Discrete Math.*, **115** (1993) 51–55. ⇒256, 263

[6] E. Boros, V. A. Gurvich, I. Zverovich, On split and almost CIS-graphs, *Australas. J. Combin.,* **43** (2009) 163–180. ⇒255

[7] A. Brandstädt, Partitions of graphs into one or two stable sets and cliques, *Discrete Math.*, **152** (1996) 47–54. ⇒254, 256

[8] A. Brandstädt, Corrigendum, *Discrete Math.*, **186** (1998) 295–295. ⇒254, 256

[9] A. Brandstädt, P. L. Hammer, V. B. Le, V. V. Lozin, Bisplit graphs, *Discrete Mathematics* **299** (2005) 11–32. ⇒255, 256

[10] A. Brandstädt, V. B. Le, J. P. Spinrad, *Graph Classes: A Survey,* SIAM Monographs on Discrete Mathematics and Applications, SIAM, Philadelphia, PL, 1999. ⇒254, 256, 266

[11] R. E. Burkard, P. L. Hammer, A note on Hamiltonian split graphs, *J. Comb. Graph Theory, Series B,* **28** (1980) 245–248. ⇒265
[12] G. Chartrand, L. Lesniak, P. Zhang, *Graphs and Digraphs*, CRC Press, Boca Raton, FL, 2011. ⇒253
[13] V. Chungphaisan, Conditions for a sequences to be r-graphic, *Discrete Math.*, **7** (1974) 31–39. ⇒257, 258
[14] V. Chvátal, The toughness of graphs, *Discrete Math.*, **5** (1973) 215–228. ⇒266
[15] R. J. Clarke, Covering a set by subsets, *Discrete Math.*, **181** (1990) 147–152. ⇒264
[16] T. H. Cormen, Ch. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms* Third edition, The MIT Press/McGraw Hill, Cambridge/New York, 2009. ⇒275
[17] P. Erdős, T. Gallai, Gráfok előírt fokú pontokkal (Graphs with given degrees of vertices), *Mat. Lapok*, **11** (1960), 264–274. ⇒253, 256, 258, 280
[18] P. Erdős, A. Gyárfás, Split and balanced colorings of complete graphs, *Discrete Mathematics*, **200,** (1–3) (1999), 79–86. ⇒264
[19] P. Erdős, M. S. Jacobson, J. Lehel, Graphs realizing the same degree sequences and their respective clique numbers, in: Y. Alavi (Ed.), Graph Theory, Combinatorics and Applications, vol. 1, John Wiley and Sons, New York, 1991, 439–449. ⇒264
[20] T. Feder, P. Hell, S. Klein, R. Motwani, List partitions, *SIAM J. Discrete Math*, **16** (2003) 449–478. ⇒278
[21] S. Földes, P. Hammer, Split graphs having Dilworth number two, *Canad. J. Math.* **29,** (3) (1977) 666–672. ⇒254, 256
[22] S. Földes, P. Hammer, Split graphs, in (ed. E. Hoffman et al.) *Proc. 8th South-Eastern Conf. Combinatorics, Graph Theory Comp. Congressus Num.*, **XIX** (1977) 311–315. ⇒254, 256, 263
[23] S. Földes, P. Hammer, The Dilworth number of a graph, *Annals Discrete Math.*, **2** (1978) 211–219. ⇒256
[24] D. R. Fulkerson, A. J. Hoffman, M. H. McAndrew, Some properties of graphs with multiple edges, *Canad. J. Math.*, **17** (1965) 166–177. ⇒253, 269, 271
[25] V. Gasharov, The Erdős-Gallai criterion and symmetric functions, *Europ. J. Combinatorics,* **18** (1997) 287–294. ⇒258
[26] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, 1980. ⇒254, 256, 263, 278
[27] R. J. Gould, M. S. Jacobson, J. Lehel, Potentially G-graphical degree sequences, in: Y. Alavi (Ed.), *Combinatorics, Graph Theory and Algorithms*, vol. 1, New Issues Press, Kalamazoo, Michigan, 1999, 451–460. ⇒261, 264
[28] J. L. Gross, J. Yellen, P. Zhang. *Handbook of Graph Theory*, CRC Press, Boca Raton, FL, 2013. ⇒252, 253, 254, 256
[29] A. Gyárfás, J. Lehel, On-line and first fit colorings of graphs. *J. Graph Theory*, **12,** (2) (1988) 217–227. ⇒264
[30] A. Gyárfás, Generalized split graphs and Ramsey numbers, *J. Comb. Theory, Series A* **81,** (2) (1998) 255–261. ⇒255, 256

[31] M. Habib, A. Mamcarz, Colored modular and split decompositions of graphs with applications to trigraphs, in (ed. D. Kratsch and I. Todinca) *Graph-Theoretic Concepts in Computer Science* (40th International Workshop, WG 2014, Nouan-le-Fuzelier, France, June 25-27, 2014). Series: Lecture Notes in Computer Science, **8747,** Springer Verlag, Berlin, 2014, 263–274. ⇒266

[32] S. L. Hakimi, On the realizability of a set of integers as degrees of the vertices of a simple graph. *J. SIAM Appl. Math.*, **10** (1962), 496–506. ⇒257, 258

[33] S. L. Hakimi, E. F. Schmeichel, Graphs and their degree sequences: A survey, in: *Theory and Applications of Graphs,* Lecture Notes in Math. **642**, Springer-Verlag, Berlin 1978, 225–235. ⇒261

[34] P. L. Hammer, B. Simeone, The splittance of a graph, *Combinatorica,* **1,** (3) (1981) 275–284. ⇒263, 264, 278

[35] P. Hanion, Enumeration of graphs by degree sequence, *J. Graph Theory*, **3** (1979) 295–299. ⇒260

[36] V. Havel, A remark on the existence of finite graphs (Czech), *Časopis Pěst. Mat.*, **80** (1955), 477–480. ⇒257, 258

[37] P. Heggerness, D. Kratsch, Linear-time certifying algorithms for recognizing split graphs and related graph classes, *Nordic J. Comp.,* **14** (2007) 87–108. ⇒278

[38] P. Heggernes, F. Mancini, Minimal split completions. *Discrete Appl. Math.*, **157,** (12) (2009) 2659–2669. ⇒266

[39] A. P. Heinz, Total number of split graphs (chordal + chordal complement) on $\mathfrak{n}$ vertices. In: (ed. N. J. A. Sloane): *The On-Line Encyclopedia of the Integer Sequences*. 2014. ⇒265

[40] P. Hell, D. Kirkpatrick, Linear-time certifying algorithms for near-graphical sequences, *Discrete Math.*, **309,** (18) (2009) 5703–5713. ⇒260

[41] P. Hell, S. Klein, F. Protti, L. Tito, On generalized split graphs, *Electronic Notes Disc. Math.,* **7** (2001) 98–101. ⇒255

[42] L. Ibarra, Fully dynamic algorithms for chordal graphs and split graphs, *ACM Trans. Algorithms*, **4(4),** (2008), Art. 40, 20 pages. ⇒266

[43] A. Iványi, Reconstruction of complete interval tournaments, *Acta Univ. Sapientiae, Inform.*, **1,** (1) (2009) 71–88. ⇒252, 253, 256

[44] A. Iványi, Reconstruction of complete interval tournaments II, *Acta Univ. Sapientiae, Math.*, **2,** (1) (2010) 47–71. ⇒252, 253

[45] A. Iványi, Degree sequences of multigraphs, *Annales Univ. Sci. Budapest., Rolando Eötvös Nom., Sectio Comp.*, **37** (2012) 195–214. ⇒ 256, 259, 260, 262

[46] A. Iványi, Z. Kása, Parallel enumeration of graphical sequences (in Hungarian), *Alk. Mat. Lapok*, **31,** (2014) 1–58. ⇒260

[47] A. Iványi, L. Lucz, Degree sequences of multigraphs (in Hungarian), *Alk. Mat. Lapok*, **29,** (2012) 1–54. ⇒260

[48] A. Iványi, L. Lucz, T. F. Móri, P. Sótér, On the Erdős-Gallai and Havel-Hakimi algorithms. *Acta Univ. Sapientiae, Inform.* **3,** 2 (2011) 230–268. ⇒ 256, 259, 260, 262, 275

[49] A. Iványi, S. Pirzada. Comparison based ranking, in: ed. A. Iványi, *Algorithms of Informatics*, Vol. 3, mondAt, Vác, 2013, 1209–1258. ⇒260

[50] A. E. Kézdy, J. Lehel, Degree sequences of graphs with prescribed clique size. In: Y. Alavi et. al. (eds.) *Combinatorics, Graph Theory, and Algorithms*, vol. 2., Michigan, New Issues Press, Kalamazoo, 1999, 535–544. ⇒260

[51] D. J. Kleitman, D. L. Wang, Algorithm for constructing graphs and digraphs with given valences and factors, *Discrete Math.*, **6** (1973) 79–88. ⇒254, 257, 258

[52] D. Kratsch, R. M. McConnell, K. Mehlhorn, J. P. Spinrad, Certifying algorithms for recognizing interval graphs and permutation graphs, *SIAM J. Comput.*, **36,** (2) (2006) 326–353. ⇒266

[53] D. Kratsch, J. Lehel, H. Müller, Toughness, hamiltonicity and split graphs, *Discrete Math.*, **150** (1996) 231–245. ⇒266

[54] M. D. Lamar, Split digraphs, *Discrete Math.,* **312,** (7) (2012) 1314–1325. ⇒266

[55] V. B. Le, H. N. de Ridder, Probe split graphs, *Discrete Math. Theor. Comput. Sci.*, **9(1),** (2007) 207–238. ⇒255

[56] J. S. Li, Z. X. Song, An extremal problem on the potentially p_k-graphic sequence, *Discrete Math.*, **212** (2000) 223–231. ⇒264

[57] J. S. Li, Z. X. Song, R. Luo, The Erdős-Jacobson-Lehel conjecture on potentially p_k-graphic sequences is true, *Sci. China Ser. A*, **41** (1998) 510–520. ⇒264

[58] F. Maffray, M. Preissmann, Linear recognition of pseudo-split graphs, *Discrete Appl. Math.* **52** (1994) 307–312. ⇒263, 279

[59] R. M. McConnell, K. Mehlhorn, S. Näher, P. Schweitzer, Certifying algorithms, *Computer Science Review*, **5,** (2) (2011) 119–161. ⇒260

[60] C. McCreesh, Multi-threaded maximum clique, level 4 Project, School of Computing Science, of Glasgow University, 2013, 38 pages. ⇒264

[61] F. R. McMorris, C. Wang, P. Zhang, On probe interval graphs, *Discrete Appl. Math.*, **88(1–3),** (1998) 315–324. ⇒255

[62] S. D. Nikolopoulos, Constant-time parallel recognition of split graphs, *Inf. Proc. Letters*, **54,** (1) (1995) 1–8. ⇒265

[63] P. M. Pardalos, J. Rappe, M. G. S. Resende, An exact parallel algorithm for the maximum clique problem, *High Performance Algorithms and Software in Nonlinear Optimization, Applied Optimization*, **24** (1998) 279–300. ⇒264

[64] P. M. Pardalos, J. Xue, The maximum clique problem, *J. Global Opt.* (1994) 301–328. ⇒264

[65] J. Peemüller, necessary conditions for hamiltonian split graphs, *Discrete Math.*, **54** (1985) 45–57. ⇒265, 266

[66] S. Pirzada, *An Introduction to Graph Theory*, Universities Press, Orient Blackswan, India, 2012. ⇒253

[67] S. Pirzada, B. A. Chat, Potentially graphic sequences of split graphs, *Kragujevac J. Math.*, **38,** (1) (2014) 73–81. ⇒255, 256, 262, 263

[68] A. R. Rao, The clique number of a graph with given degree sequence, in: A. R. Rao (Ed.) *Proc. Symp. on Graph Theory*, MacMillan and Co. Limited, India, ISI Lecture Notes Series, **4** (1979) 251–267. ⇒260

[69] A. R. Rao, A survey of the theory of potentially P-graphic and forcibly P-graphic degree sequences, *Comb. Graph Theory,* Proc. Symp. (Calcutta 1980), Lect. Notes Math. **885** (1981) 417–440. ⇒260

[70] A. R. Rao, An Erdős-Gallai type result on the clique number of a realization of a degree sequence (unpublished). ⇒260

[71] G. F. Royle, Counting set covers and split graphs, *J. Integer Sequences,* **3** (2000), Article 00.2.6, 5 pages. ⇒264

[72] M. C. Schmidt, N. F. Samatova, K. Thomas, B.-H. Park, A scalable, parallel algorithm for maximal clique enumeration, *J. Parallel Dist. Comp.*, **69,** (4) (2009) 417–428. ⇒264

[73] P. S. Segundo, D. Rodríguez-Losada, A. Jiménez, An exact bit-parallel algorithm for the maximum clique problem, *Computers & Operations Res.*, **38,** (2) (2011) 571–581. ⇒264

[74] N. J. A. Sloane, The number of degree-vectors for simple graphs. In (ed. N. J. A. Sloane): *The On-Line Encyclopedia of the Integer Sequences.* 2014, http://oeis.org/A004251 . ⇒260

[75] S. Szabó, Parallel algorithms for finding cliques in a graph, *J. Physics: Conf. Series*, **268,** (1) (2011) 012030. ⇒264

[76] E. Tomita, A. Tanaka, H. Takahashi, The worst-case time complexity for generating all maximal cliques and computational experiments, Theoretical Computer Science, **363,** (1) (2006) 28–42. ⇒264

[77] A. Tripathi, H. Tyagi, A simple criterion on degree sequences of graphs. *Discrete Appl. Math.,* **156,** 18 (2008) 3513–3517. ⇒258

[78] A. Tripathi, S. Venugopalan, D. B. West, A short constructive proof of the Erdős–Gallai characterization of graphic lists, *Discrete Math.*, **310,** (4) (2010) 843–844. ⇒258

[79] R. I. Tyshkevich, The canonical decomposition of a graph, *Doklady Akademii Nauk SSSR* (in Russian), **24** (1980) 677–679. ⇒263

[80] R. I. Tyshkevich, A. A. Chernyak, Yet another method of enumerating unmarked combinatorial objects, *Mathematical Notes,* **48,** (6) (1990) 1239–1245 and (in Russian) *Mat. Zametki,* **48,** (6) (1990) 98–105. ⇒264

[81] R. I. Tyshkevich, O. I. Melnikow, V. M. Kotov, On graphs and degree sequences: the canonical decomposition (in Russian), *Kibernetica,* **6** (1981) 5–8. ⇒263

[82] Wikipedia, Split graph. http://en.wikipedia.org/wiki/Split_graph, 2014. ⇒256, 263

[83] G. J. Woeginger, The taughness of split graphs, *Discrete Math.*, **190** (1998) 195–197. ⇒266

[84] J.-H. Yin, Conditions for r-graphic sequences to be potentially $\mathsf{K}_{m+1}^{(r)}$-graphic, *Discrete Math.*, **309** (2009) 6271–6276. ⇒261

[85] J.-H. Yin, A generalization of a conjecture due to Erdős, Jacobson and Lehel, *Discrete Math.*, **309** (2009) 2579–2583. ⇒261, 264

[86] J.-H. Yin, A Rao-type characterization for a sequence to have a realization containing a split graph, *Discrete Math.*, **311** (2011) 2485–2489. ⇒255, 256

[87] J.-H. Yin, A Havel-Hakimi type procedure and a sufficient condition for a sequence to be potentially $S_{r,s}$-graphic, *Czechoslovak Math. J.*, **62,** (3) (2012) 863–867. ⇒255, 256, 262, 279

[88] J.-H. Yin, An extension of A. R. Rao's characterization of potentially K_{m+1}–graphic sequences, *Discrete Appl. Math.*, **161,** (7–8) (2013) 1118–1127. ⇒256, 262

[89] J.-H. Yin, A Rao-type characterization for a sequence to have a realization containing an arbitrary subgraph H, *Acta Math. Sin.*, **30,** (3) (2014) 389–394. ⇒262

[90] J.-H. Yin, Y.-S. Li, Two sufficient conditions for a graphic sequence to have a realization with prescribed clique size, *Discrete Math.*, **209** (2005) 218–227. ⇒264

[91] B. Zavalnij, Three versions of clique search parallelization, *J. Comp. Sci. Inf. Technology*, **2(2),** (2014) 9–20. ⇒264

Received: July 20, 2014 • Revised: November 16, 2014

Acta Univ. Sapientiae, Informatica, **6,** 2 (2014) 287–310

Finding sequential patterns with TF-IDF metrics in health-care databases

Zsolt T. KARDKOVÁCS
U1 Research
2 INFOPARK Gábor Dénes
Budapest, Hungary
email: kardkovacs@u1research.org

Gábor KOVÁCS
U1 Research
2 INFOPARK Gábor Dénes
Budapest, Hungary
email: kovacsg@u1research.org

Abstract. Finding frequent sequential patterns has been defined as finding ordered list of items that occur more times in a database than a user defined threshold. For big and dense databases that contain really long sequences and large itemset such as medical case histories, algorithm proposed on this idea of counting the occurrences output enourmous number of highly redundant frequent sequences, and are therefore simply impractical. Therefore, there is a need for algorithm that perform frequent pattern search and prefiltering simultaneously. In this paper, we propose an algorithm that reinterprets the term support on text mining basis. Experiments show that our method not only eliminates redundancy among the output sequences, but it scales much better with huge input data sizes. We apply our algorithm for mining medical databases: what diagnoses are likely to lead to a certain future health condition.

1 Introduction

The main goal of any data mining process is to find novel, interesting patterns. In the last two decades, data gathering has been accelerated, which brought the attention of data scientists from pattern or association oriented

Computing Classification System 1998: H.2.8. H.3.3. I.1.2. I.5.3. J.3.
Mathematics Subject Classification 2010: 68P10, 68T30, 68W05
Key words and phrases: sequence mining, frequent sequential pattern, TF-IDF, health care database

knowledge discovery to transition analysis or sequential data mining, which is a generalization of the former task. Sequential data mining has become an important task in many real-life applications, e.g. real-time recommendation systems, health care service optimization, fraud detection.

In the known data mining publications, absolute and relative frequencies (called support) have been used as the sole criterion in selecting patterns, while significance or interestingness have been addressed by different solutions. Published solutions suggest to calculate frequent and interesting patterns independently first, and to combine solutions into a resulting set later. This approach implicitly states that frequency is more important than significance, or more accurately: significance is taken into account if and only if an interesting pattern is frequent enough.

Note that, the two properties strongly correlate:

- if minimum frequency (threshold for support) is set too high, then interesting patterns might be omitted
- if minimum frequency is set too low, then the most of the found patterns are trivial, particular cases of others, or they are uninteresting otherwise
- there is no known golden rule how to set frequencies properly.

So independent processing guarantees information loss. Moreover, consider the following situation: there are known transactions (baskets with purchased goods) in a supermarket. Well-known pattern mining algorithms reveals frequent patterns about beers, cheese, and dumpers, however, the real profit is made from high-end products like branded whiskeys or top smart phones. Since there are a relatively few number of most profitable customers who purchase expensive goods, data mining algorithms do not discover the most significant, profitable customer needs, because they focus on frequent, mass products only.

From another point of view, if we consider that the baskets are textual documents and items are words, then well-known pattern mining algorithms would identify the most frequent words in all documents, and later on they would deal with those too frequent in all documents. As it is expected it would find tons of uninteresting patterns extremely slowly, while significant ones are completely missed. Since there are very efficient document indexing methods like *TF-IDF* [12] to handle this problem properly, our approach aims at adapting fundamental ideas from information retrieval techniques to boost sequence mining algorithms' performance.

Significance, interestingness, or relevance are vague notions. The most common definition is as follows:

Definition 1 (Absolute significance) *A data mining pattern* P *is said to be absolutely significant if* P *is previously unknown, and there exists a null model* $\mathcal{Z}$ *for which* $\Pr(\mathsf{P})$ *is statistically relevant.*

In this paper, we argue on that significance cannot be separated from the item or itemset we are analyzing at the time, i.e. there is or at least there shall be a fix point, a point of view to capture this notion properly.

Definition 2 (Relative significance) *A data mining pattern* P *is said to be relatively significant regarding an item* I *if* $\mathsf{I} \in \mathsf{P}$*,* P *is previously unknown, and there exists a null model* $\mathcal{Z}$ *for which* $\Pr(\mathsf{P}|\mathsf{I})$ *is statistically relevant.*

While our approach is more restrictive, it is easy to prove that absolute significance can be addressed by relative ones. In this paper, we introduce a novel algorithm called REVIEW (RElevance from the items' point of VIEW), a point of view oriented sequence mining algorithm, which finds all frequent and significant patterns in polynomial time. In addition, we also prove that REVIEW finds the most likely anti-patterns in a hand, i.e. those items, which tend to mutually exclude each other in itemsets. This property is beyond the capabilities of state-of-the-art algorithms.

The paper is organized as follows. In Section 2, we review the most important sequence mining algorithms and show an example of their scalability issues. We also overview the alternative definitions of importance that exist in the literature. Section 3 gives the elementary definitions of sequence mining: items, itemsets, sequences, sequence databases, and gives illustrative examples for the definitions. The algorithm we propose for finding frequent and important patterns is defined in Section 4. Empirical comparison is given in Section 5, our algorithm algorithm is tested on a real-life health care database against PrefixSpan and SPADE, the two fastest algorithms in the literature. Finally in Section 6 a brief summary is given.

2 Related work

In this section, we give an overview of the most important frequent sequential pattern mining algorithms: GSP, PrefixSpan, SPADE and SPAM. In the literature several other algorithms exist as well, however, those can be considered as the variants, extensions of the ones presented here. We also discuss the performance problems that arises when the size of the input database grows. In the literature, there are a lot of alternative definitions for importance, we review these definitions.

2.1 GSP

The GSP algorithm proposed by Srikant and Agrawal in [13] is built on the pattern of the a priori algorithm. First, it scans the database and counts the support of each item, detects all single item frequent sequences. Then in each subsequent pass a candidate generation and candidate counting takes place. Candidate generation uses the frequent sequential patterns of the previous pass: if removing the first element of a frequent sequence and removing the last element of another frequent sequence are the same, then the two sequences are joined and a new sequence with one more item is created. The candidate counting scans for each new sequence in the database counting the occurrences, and the ones with support greater than the user defined minimum support are retained as frequent sequences of the pass. The candidate generation and candidate counting are repeated until no frequent sequences are found.

2.2 PrefixSpan

PrefixSpan proposed by Pei et al. [8] is also based on the frequent pattern growth principle like GSP, however, it does not perform the search on the entire database for each candidate sequence, but on smaller projected databases. The sequence database is partitioned based on the itemsets of each frequent sequence of previous passes such that all sequences that support the frequent sequence are within the partition and the sequences not supporting are not. If several frequent sequences share the same itemset, then those use the same database partition. The hypothesis is that the support of a frequent sequence that is one item longer can be calculated on that partition as outside of that partition it is not supported. New candidate sequences are generated only locally by combining sequences that use the same partition. This method is a significant speed improvement over GSP as database partitions are smaller and because of the shared itemsets candidate counting does not need to be performed for each frequent sequence, but only for shared prefixes.

2.3 SPADE

SPADE (Sequential PAttern Discovery using Equivalence Classes) proposed by Zaki [15] aims to reduce the number of database scans and minimize computational costs. During the database scans frequent sequences of length one and two are searched for and their support is counted. The algorithm maintains an id-list for each item where each element of the id-list is a pointer to a sequence id and an itemset the item occurs in. Candidate sequences with one

more item are generated with temporal joins or intersections on the id-lists of frequent sequences of maximum length, the support is calculated in the memory, and the new sequence is frequent if the cardinality of the resulting id-list is greater than the minimum support value. Frequent sequences are clustered into smaller sub-lattices based on common prefixes that enables independent processing.

2.4 SPAM

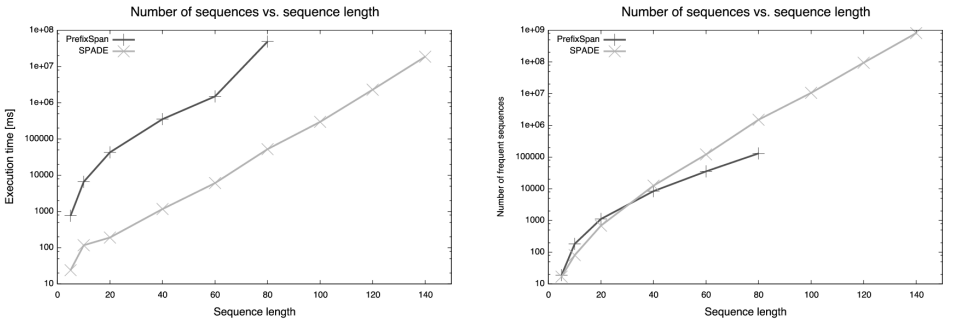
SPAM (Sequential PAttern Mining) proposed by Ayres et al. [3] assumes that the entire sequential database can completely fit in the memory and no sequences are longer than 64. The hypothesis is that frequent sequences can be found in the lexicographic tree with a simple depth-first search. Each sequence is represented with a vertical bitmap, if an item appears in a sequence then the corresponding element of the bitmap is set to one. Itemsets are generated with a bitwise and operation on the vectors of the items. Candidate sequences are generated with depth-first search from bitvectors of previous sequences and the vector of a next item in the lexicographic tree such that a bitwise and operation is performed on the two vectors, the candidate is frequent if it has more ones in its bitvector than the minimum support. The algorithm is fast, but very limited with regard to the input database.

2.5 Performance issues

In [7], Gouda and Hassaan argue that typical sequential pattern mining algorithms tend to lose their efficiency when applied to a dense database. Their experiments confirm that the execution time increases exponentially as the number of frequent sequences increases even when the execution times in their experiments remain in the order of a few hundred seconds.

We conducted similar experiments on a subset of a medical database covering 23856 out of 455514 that is about 5% of the patient data. The average length of sequences related to a patient is 307 in that sample, the average sequence size is 10.88 itemsets. The relative or absolute minimum support thresholds were set so that the number of occurrences of a frequent sequence were at least 30.

Figure 1 shows how PrefixSpan and SPADE that are the sequential pattern mining algorithms considered to be the fastest in the literature scale as we consider longer sequences from our sample set. In the experiments, we used the reference implementations available in the SPMF library [4]. The horizontal



(a) Sequence length vs. execution time

(b) Sequence length vs. frequent sequences

Figure 1: The maximum length in a sequence database vs. the execution time and the number of frequent sequences for the two fastest sequential pattern mining algorithms, PrefixSpan and SPADE

axes show the maximum length of sequences used in the mining process. The sequence lengths considered in the experiment were 5, 10, 20, 40, 60 and 80. Note that these values are still far away from 300 that is the average length in our dataset. In Figure 1a, the vertical axis is the execution time of the algorithm in milliseconds in logarithmic scale. In Figure 1b the vertical axis is the number of frequent sequences detected in logarithmic scale by the algorithms that are proportional to their memory and disc space usages.

The results not surprisingly show that the algorithms fail to produce usable results. The algorithms do not scale with the average length. The time, memory and disk space requirements are exponentially proportional to the length of the input sequences.

2.6 Significance

A general idea to find *interesting* patterns is widely discussed in the literature. A very detailed description on different aspects of significance is found in [10, 6]. According to Geng et al. [6] definition interestingness can be broken down into the following categories:

- Conciseness. A pattern is concise if it contains relatively few attribute-value pairs, while a set of patterns is concise if it contains relatively few patterns.

- Generality/Coverage. A pattern is general if it covers a relatively large subset of a dataset.
- Reliability. A pattern is reliable if the relationship described by the pattern occurs in a high percentage of applicable cases.
- Peculiarity. A pattern is peculiar if it is far away from other discovered patterns according to some distance measure.
- Diversity. A pattern is diverse if its elements differ significantly from each other, while a set of pattern is diverse if the patterns in the set differ significantly from each other.
- Novelty. A pattern is novel to a person if he did not know it before and are not able to infer it from other known patterns.
- Surprisingness. A pattern is surprising (or unexpected) if it contradicts a person's existing knowledge or expectations.
- Utility. A pattern is of utility if its use by a person contributes to reaching a goal.
- Actionability or Applicability. A pattern is actionable (or applicable) in some domain if it enables decision making about the future actions in this domain.

Some of these notions correlate, some are subjective, some are objective, and some depends on the semantics, but they share a common feature: all of them use some kind of statistical relevance metrics to describe a particular meaning of interestingness. That is why we used the notion significance in Definition 1 as a union of these possible meanings where $\Pr(\mathsf{P})$ corresponds to the proper relevance metrics depending on the use case. Note that, $\Pr(\mathsf{P})$ is a statistical function but how to calculate is undetermined in general; it can be adapted to the problem specific needs. Later on this paper, we use the term significance measure for $\Pr(\mathsf{P})$. Table 1 summarizes the most common significance measure in sequential pattern mining [6].

3 Preliminaries

In this section, we give preliminary definitions necessary for the formalization of the problem statement: the definition of the frequent sequential pattern discovery and the definition of relevant pattern discovery.

Throughout this paper, we use the following conventions:

- sets and elements of sets are denoted by capital letters and lower case letters, respectively,

Measure	**Formula**
Support	$\Pr(AB)$
Lift/Interest	$\frac{\Pr(B\|A)}{\Pr(B)}$ or $\frac{\Pr(AB)}{\Pr(A)\Pr(B)}$
Interestingness Weighted Dependency	$\left(\left(\frac{\Pr(AB)}{\Pr(A)\Pr(B)}\right)^k - 1\right)\Pr(AB)^m$
Added value	$\Pr(B\|A) - \Pr(B)$
Relative risk	$\frac{\Pr(B\|A)}{\Pr(B\|\neg A)}$
Mutual information	$\sum_i \sum_j \Pr(A_i B_j) \log_2 \frac{\frac{\Pr(A_i B_j)}{\Pr(A_i)\Pr(B_j)}}{-\sum_i \Pr(A_i)\log_2 \Pr(A_i)}$
Certainty factor	$\frac{\Pr(B\|A) - \Pr(B)}{1 - \Pr(B)}$
Conviction	$\frac{\Pr(A)\Pr(\neg B)}{\Pr(A\neg B)}$
Odds ratio	$\frac{\Pr(AB)\Pr(\neg A\neg B)}{\Pr(A\neg B)\Pr(\neg AB)}$
Yule's Q	$\frac{\Pr(AB)\Pr(\neg A\neg B) - \Pr(A\neg B)\Pr(\neg AB)}{\Pr(AB)\Pr(\neg A\neg B) + \Pr(A\neg B)\Pr(\neg AB)}$
Cosine	$\frac{\Pr(AB)}{\sqrt{\Pr(A)\Pr(B)}}$

Table 1: Some probability based objective measures for data mining[6]

- itemset and items are taken from the beginning of the latin alphabet,
- $|X|$ denotes the size of X where X is a set of attributes or itemsets,
- we use R, S symbols for database relations defined as subsets of a Cartesian products, and $r, s \ldots$ for tuples, records or elements of a relation. If $r \in R$ and $R \subseteq A \times B$ then $r(a, b)$ is short form to say $a \in A$, $b \in B$, and $r[A] = a$, $r[B] = b$ where $r[X]$ stands for the attribute values of r on attribute set X (projection in relational database theory),
- identifiers are denoted by letters near I,
- $\top$ and $\bot$ stand for logical values `true` and `false`, respectively,
- T, t are used for time related sets and variables, respectively,

patient 1			patient 2			patient 3			patient 4		
I	T	A	I	T	A	I	T	A	I	T	A
1	234	a	2	57	f	3	186	h	4	33	a
1	234	b	2	63	g	3	186	i	4	93	k
1	234	c	2	74	g	3	186	a			
1	234	d	2	78	e	3	186	j			
1	237	e	2	78	g	3	186	e			
						3	199	a			
						3	199	e			

Table 2: A small anonymized piece of the database

- we also introduce the symbol $\mathcal{D}_R(X)$, which denotes the domain of an attribute set X in a relation R, i.e. $\mathcal{D}_R(X) = \{r[X] | r \in R\}$.

Let the input database be defined as follows, the definition is analogous to the one in [15].

Definition 3 (Sequence database) *Let* $A = \{a_1, a_2, \ldots, a_n\}$ *be a finite set of items, where* $n \geq 1$, T *is a non-empty set of timestamps, and* I *is a non-empty set of unique identifiers. Let a relation* R *be defined over* $I \times T \times A$*, i.e.* $R \subseteq I \times T \times A$*, then* R *is a sequence database. For simplicity and better understanding, we use the notion* $R(ITA)$ *to express* R *is determined by sets* I*,* T*, and* A*, i.e.* $R \subseteq I \times T \times A$ *in that order. If* $|I| = 1$ *in a sequence database* $R(ITA)$*, then* R *is called a sequence database.*

Example Table 2 shows a small set of records from the anonymized health care database we use in this paper. The columns of the table reflect relation R. The twelve columns of the table are organized into four groups of three columns. Each column group represents a patient. The first column in a group is a patient identifier, the real identifier is replaced with an integer number. The second column in a group is a timestamp, the real date is replaced with an integer number. The third column in a group contains the items, the real treatment codes are transformed to letters of the alphabet.

Definition 4 (Ordering of items) *Let a binary relationship* $\leq$: $R \times R \rightarrow \{\top, \bot\}$ *be defined on sequence databases such that the ordering of elements of* R *is determined by the natural ordering over* T*.* $\leq$ *is an ordering, i.e. it is transitive, antisymmetric, reflexive, and total. For simplicity, we also use* $\leq$

on relations such that if $S_1, S_2 \subseteq R$ *and* $S_1, S_2 \neq \emptyset$*, then* $S_1 \leq S_2$ *if and only if* $\forall s_1 \in S_1 \forall s_2 \in S_2 : s_1 \leq s_2$.

In other words, ordering of items in sequence databases are based on time related attributes. If time representation in sequence database does allow a clear distinction between when two events have happened, then we assume they are simultaneous events.

Definition 5 (Sequence) *Let* $R(ITA)$ *be a sequence database, and* $\mathcal{S} = < S_1, S_2, \dots S_n >$ *be defined as an ordered set of relations where* $\forall i : 1 \leq i \leq n \implies S_i \subseteq R$ *such that*

$$S_i, S_j \in \mathcal{S} : 1 \leq i < j \leq n \implies S_i \leq S_j, \neg S_j \leq S_i.$$

We say $\mathcal{S}$ *is a sequence if and only if*

$$\begin{aligned} \forall r, s : \; & r \in S_i, s \in S_j \implies r[I] = s[I] \\ \forall r, s : \; & \Big(r \in S_i, s \in S_j \implies r[T] = s[T] \Big) \iff i = j \end{aligned}$$

for all $S_i, S_j \in \mathcal{S}$*. Since the identifiers are the same in the sequence, and there are itemsets that share the same timestamps, we use the representation* $\mathcal{S} =< A_{t_k}, \dots, A_{t_l} >$ *for better readability where* $t_i \in T$*, where* $A_{t_i} \subseteq A$ *is a set of elements indexed by their shared timestamps. We also introduce the following notions:*

- $|\mathcal{S}| = n$ *denotes the length (number of relations) of the sequence,*
- $U(\mathcal{S})$ *stands for the shared identifier in* $\mathcal{S}$*,*
- *and* $\tau(S_i)$ *(or* $\tau(A_i)$*) for the shared timestamp in* S_i *where* $1 \leq i \leq n$.

This means that in sequence $\mathcal{S}$ all elements share the same identifier, and within an A_{t_i} itemset the elements are unordered as they are considered to have occurred simultaneously. If an itemset A_{t_i} precedes A_{t_j} in $\mathcal{S}$ ($A_i \leq A_j$), then all items in A_{t_i} precede any item in A_{t_j}. We also introduce operators on sequences to deal with more complex problems.

Definition 6 (Operators on sequences) *Let* $\mathcal{S}_1 =< A_{t_k}, \dots, A_{t_l} >$ *and* $\mathcal{S}_2 =< A_{t_m}, \dots, A_{t_n} >$ *be two sequences defined on* $R(ITA)$*. We say*

- $\mathcal{S}_1$ *is a proper subsequence of* $\mathcal{S}_2$ *denoted by* $\mathcal{S}_1 \sqsubseteq \mathcal{S}_2$ *if and only if* $U(\mathcal{S}_1) = U(\mathcal{S}_2)$ *and* $\forall A_{t1} \exists A_{t2} : A_{t1} \in \mathcal{S}_1, A_{t2} \in \mathcal{S}_2 \implies A_{t1} \subseteq A_{t2}, \tau(A_{t1}) = \tau(A_{t2})$,

- $\mathcal{S}_1$ *is a subsequence of* $\mathcal{S}_2$ *denoted by* $\mathcal{S}_1 \preceq \mathcal{S}_2$ *if and only if for all* a_1, a_2, *and* $t1, t2 \in T$ *there exist* $t3, t4 \in T$ *such that*

$$a_1 \in A_{t1}, a_2 \in A_{t2}, t1 \leq t2 \implies a_1 \in A_{t3}, a_2 \in A_{t4}, t3 \leq t4$$

  where $A_{t1}, A_{t2} \in \mathcal{S}_1$, *and* $A_{t3}, A_{t4} \in \mathcal{S}_2$,

- *the union of sequences for which* $U(\mathcal{S}_1) = U(\mathcal{S}_2)$ *denoted by* $\mathcal{S}_1 \cup \mathcal{S}_2$ *is defined as an* $\leq$*-ordering preserving merge of these sets such that if* $A_{t1} \in \mathcal{S}_1$, $A_{t2} \in \mathcal{S}_2$ *and* $\tau(A_{t1}) = \tau(A_{t2})$ *then the resulting* $A_t = A_{t1} \cup A_{t2}$,

- *the intersection of sequences for which* $U(\mathcal{S}_1) = U(\mathcal{S}_2)$ *denoted by* $\mathcal{S}_1 \cap \mathcal{S}_2$ *is defined as the largest possible subsequence* $\mathcal{S}$ *in number of items for which* $\mathcal{S} \sqsubseteq \mathcal{S}_1$ *and* $\mathcal{S} \sqsubseteq \mathcal{S}_2$,

- *the difference of the sequences for which* $U(\mathcal{S}_1) = U(\mathcal{S}_2)$ *denoted by* $\mathcal{S}_1 \backslash \mathcal{S}_2$ *is defined as the largest possible subsequence* $\mathcal{S}$ *in number of items for which* $\mathcal{S} \sqsubseteq \mathcal{S}_1$ *and there is no* $\mathcal{S}'$ *such that* $\mathcal{S}' \sqsubseteq \mathcal{S}$ *and* $\mathcal{S}' \sqsubseteq \mathcal{S}_2$ *if and only if* $\mathcal{S}_1$ *is not a subsequence of* $\mathcal{S}_2$. *The difference does not exist otherwise.*

- $\mathcal{S}_2$ *is the prefix cut of* $\mathcal{S}_1$ *by an item* $a \in A$ *denoted by* $\varphi(\mathcal{S}_1, a)$ *if and only if* $\mathcal{S}_2 \sqsubseteq \mathcal{S}_1$ *and if there exists* $A_t \in \mathcal{S}_1$ *such that* A_t *is a set for which* $a \in A_t$ *then* $\max_{A_i \in \mathcal{S}_2}(\tau(A_i)) \leq \tau(A_t)$. *In this paper, we call maximum cut of* $\mathcal{S}_1$ *by* $a \in A$ *(*$\Phi(\mathcal{S}_1, a)$*) the union of all possible prefix cuts of* $\mathcal{S}_1$ *by* $a \in A$.

Notice that, there can be many different sequences with the same identifier according to Definition 5, and there is no sequence with the length of 0. It is easy to prove that the maximal number of closed sequences in a sequence database $R(ITA)$ equals to $\mathcal{D}_R(I)$, hence every closed sequence has a natural identifier: the elements of set I in our database.

Definition 7 (Closed sequences) *Let a sequence* $\mathcal{S} =< S_1, S_2, \ldots, S_n >$ *be defined on a sequence database* $R(ITA)$. *We say* $\mathcal{S}$ *is a closed sequence and it is denoted by* $\overline{\mathcal{S}}$ *if and only if*

$$\forall r \exists S_i \; r \in R, S_i \in \mathcal{S}, r[I] = U(\mathcal{S}) \implies r \in S_i$$

for some $1 \leq i \leq n$. *The largest possible set of closed sequences in* $R(ITA)$ *is called macseq and it is denoted with* Σ.

Example 8 *Table 3 shows the records of Table 2 transformed into the form used for representing sequence databases in the literature. The first column is the sequence identifier, which comes from the patient identifying* I *attribute of Table 2. The second column contains the sequences, where each sequence is a comma separated list of itemsets shown in braces. The ordering of the itemsets is determined by attribute* T*. If the* T *value is identical for two* A *items, then those appear in the same itemset.*

I	Σ
1	$<(a,b,c,d),(e)>$
2	$<(f),(g),(g),(e,g)>$
3	$<(h,i,a),(j,e)>$
4	$<(a),(k)>$

Table 3: Relation R transformed to a sequence database

Sequence $<(a),(e)>$ *is a subsequence of both the sequence identified by* $I = 1$ *and* $I = 3$*. In both sequences itemset* (a) *is a subset of the first itemset, and* (e) *is a subset of the second itemset.*

The following terms are the foundations for capturing the proper concept of frequent sequences:

Definition 9 (Support of sequences) *We introduce the following support metrics for a sequence* $\mathcal{S} =< S_1, S_2, \ldots, S_n >$ *defined on* $R(ITA)$*:*

- *the support of* $\mathcal{S}$ *denoted by* $\mathrm{supp} : \mathcal{S} \to [0,1]$*:*

$$\mathrm{supp}(\mathcal{S}) = \frac{||\{\mathcal{S}_i \mid \mathcal{S}_i \in \Sigma, \mathcal{S} \preceq \mathcal{S}_i\}||}{||\Sigma||},$$

  where $||S||$ *stands for the number of elements in set* S*,*

- *the conditional support of* $\mathcal{S}$ *assuming there is a* $\mathcal{S}_i \in \Sigma$*, which has an element containing* $a \in A$ *is*

$$\mathrm{supp}(\mathcal{S}|a) = \frac{||\{\mathcal{S}_i \mid \mathcal{S}_i \in \Sigma, \mathcal{S} \preceq \Phi(\mathcal{S}_i, a)\}||}{||\{\mathcal{S}_i \mid \mathcal{S}_i \in \Sigma, \exists A_t \in \mathcal{S}_i, a \in A_t\}||}.$$

  If there is no sequence in Σ *that contains* a*, then let* $\mathrm{supp}(\mathcal{S}|a) = 0$*,*

- *the conditional unsupport of* $\mathcal{S}$ *assuming there is a* $\mathcal{S}_i \in \Sigma$ *that contains an item* $a \in A$ *is*

$$\text{supp}(\mathcal{S}|\neg a) = \frac{||\{\mathcal{S}_i \mid \mathcal{S}_i \in \Sigma, \mathcal{S} \preceq \mathcal{S}_i, \forall A_t \in \mathcal{S}_i : \ a \notin A_t\}||}{||\{\mathcal{S}_i \mid \mathcal{S}_i \in \Sigma, \forall A_t \in \mathcal{S}_i, a \notin A_t\}||}.$$

If each sequence in Σ *contains* a*, then let* $\text{supp}(\mathcal{S}|\neg a) = 0$.

Example 10 *Table 4 show the support of sequences with length of one based on Table 3. Items* a *and* e *occur in three different sequences. Though* g *has three occurrences as well, those are limited to a single sequence.*

Σ	$\text{supp}(\mathcal{S})$
a	3
b	1
c	1
d	1
e	3
f	1
g	1
h	1
i	1
j	1
k	1

Table 4: Support of items

Theorem 11 *Let* $\mathcal{S}_1 =< A_{t_1}, \ldots, A_{t_n} >$*, and* $\mathcal{S}_2 =< A_{t_1}, \ldots, A_{t_{n-1}} >$ *be two sequences defined on* $R(ITA)$*, then*

$$\forall a \in A_{t_n} : \ \text{supp}(\mathcal{S}_1) \leq \text{supp}(\mathcal{S}_2|a).$$

Proof. According to Definition 9, $\text{supp}(\mathcal{S})$ equals to number closed sequences that contain $\mathcal{S}$ as a pattern divided by the number of all closed sets. Firstly, let assume that A_{t_n} consists of a single item a. In that case, the numerator of $\text{supp}(\mathcal{S}_1)$ and $\text{supp}(\mathcal{S}_2|a)$ is the same since $\mathcal{S}_1$ equals to $\mathcal{S}_2$ followed by an a. The denominator is different: for $\text{supp}(\mathcal{S}_2|a)$ it is the number of closed sequences containing a in one of its elements, which must be less or equal than the number of all closed sequences. As a consequence, $\text{supp}(\mathcal{S}_1) \leq \text{supp}(\mathcal{S}_2|a)$.

If A_{t_n} contains more than one element, then $\text{supp}(\mathcal{S}_1)$ is determined by the least frequent item in A_{t_n}. That is, the numerator of $\text{supp}(\mathcal{S}_2|a)$ must be greater or equal to one of $\text{supp}(\mathcal{S}_1)$ also leads to $\text{supp}(\mathcal{S}_1) \leq \text{supp}(\mathcal{S}_2|a)$. □

4 Relevance base candidate selection

The problem of frequent sequential pattern discovery has been defined in [1]: *Given a set of sequences, where each sequence consists of a list of elements and each element consists of a set of items, and given a user-specified min_support threshold, sequential pattern mining is to find all frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is no less than min_support.* This definition originally targeted the mining of database transactions, however this very same definition can be applied to a much wider range of problems. In our case, the set of sequences are identified by attribute I in the elements of relation R. The elements of sequences are ordered by attribute T, and hence form a list. As it is possible for elements of R to have the same $\mathsf{t} \in \mathsf{T}$ value, the elements of that list are not individual items, but a set of items.

Frequent sequence construction is based on the hypothesis that all subsequences of a frequent sequence are frequent sequences themselves, formally if $\mu \leq \text{supp}(\mathcal{S})$, then $\forall \mathcal{S}' \preceq \mathcal{S} \implies \mu \leq \text{supp}(\mathcal{S}')$ where $\mu \in [0, 1]$ stands for min_support. This assumption makes it possible to build a lattice of subsequences. The lattice can be constructed bottom-up with increasing length on the pattern of the a priori algorithm or by combining frequent subsequences already detected based on their prefixes.

4.1 Problem statement

Average Apriori like algorithms are quasi linear whenever the size of frequent itemsets are small; polynomial in the sum of the size of the input (transactions) plus output (frequent patterns) [11]. That is, if every shopper buys every item, the algorithm must output each subset of A items. The basic characteristics do not change for sequence mining using distributed Apriori-like algorithms [2], however, the size of the input is multiplied by the length of sequences. In other words, sequence mining algorithms are exponential for long sequences or large inputs.

As Figure 1 indicates for the case of sequential databases that contain very long sequences, an improvement is necessary in the candidate generation for sequential pattern mining algorithms. Counting the number of occurrences may be simply infeasible for mining some real life datasets, like in health care database.

Moreover, interesting patterns are not necessarily frequent ones. If some items are frequent enough among sequences by themselves independently from

others, then frequent pattern mining algorithms always include those in almost all elements of the output, which increases the size of the output, and decreases significantly the interestingness of such patterns. As a consequence, the problem is how to improve the computational performance with or by increasing the interestingness of patterns, or to be more precise: how to improve the overall performance by pruning non-relevant or independent consumptions.

Consider a database of medical diagnoses; anamneses can easily be constructed by building sequences joining diagnoses/treatment by patient identifiers. Almost all frequent sequences contain diagnoses frequent in the population such as flu or hypertension that are not necessarily relevant in general for the course of the main case. Such items should not be considered when building frequent sequences and provide a basis for prefiltering. We call the prefiltered set of frequent sequences frequent-and-relevant sequences.

In this section, we propose a new method for calculating importance metrics, which combine relevance and support of a sequence when generating candidate sequences. The basic idea is that candidate generation and prefiltering of the sequences should take place at the same time to reduce the search space and hence the computational complexity.

The idea for prefiltering in our support calculation method comes from text mining where the TF-IDF [12] metric has been successfully used to connect different documents based on their contents. The sequence metric is similar to the SIF-IDF metric defined in [9] for protecting sensitive data in databases.

Importance metric of a pattern is derived from two values associated with two sequences generated by the pattern. For a pattern we maintain two sets of key-value pairs: one in which support values are calculated on closed sequences of identifiers that appear in the pattern, and another one set of those that do not. The former one is called *frequent set*, the latter is called *inverse set*. The normalized rate of relative occurrences of an item in the frequent and inverse sets is a suitable parameter for that item for filtering when generating a new candidate sequence.

In the next section, we give the definitions necessary for formalizing this idea.

4.2 Definitions

Definition 12 (Frequent set of a pattern) *Let $\mathcal{S}$ be a pattern over a relation $\mathsf{R}(\mathsf{ITA})$, and F be a function which maps sequences to a set of a set of item-number pairs such that*

$$\mathsf{F}(\mathcal{S}) = \{(\mathfrak{a}, \mathrm{supp}(\mathcal{S}|\mathfrak{a}))|\ \mathfrak{a} \in \mathsf{A}\}.$$

Definition 13 (Inverse set of a pattern) *Let $\mathcal{S}$ be a pattern over a relation $\mathsf{R}(\mathsf{ITA})$, and $\overline{\mathsf{F}}$ be a function which maps sequences to a set of a set of item-number pairs such that*

$$\overline{\mathsf{F}}(\mathcal{S}) = \{(\mathfrak{a}, \mathrm{supp}(\mathcal{S}|\neg\mathfrak{a})) \mid \mathfrak{a} \in \mathsf{A}\}.$$

$\mathsf{F}(\mathcal{S})$ and $\overline{\mathsf{F}}(\mathcal{S})$ contain information on each item, e.g. $(\mathfrak{a}, \mathfrak{n}) \in \mathsf{F}(\mathcal{S})$, and $(\mathfrak{a}, \mathfrak{m}) \in \overline{\mathsf{F}}(\mathcal{S})$. Notice that, a correlation between values $\mathfrak{n}$, $\mathfrak{m}$ might indicate relevance. If $\mathfrak{m} \simeq 0$ and $\mathfrak{n} \geq \mu$ then $\mathcal{S} \to \mathfrak{a}$ (i.e. $\mathcal{S}$ is followed by $\mathfrak{a}$) show high correlation, which means that the presence of $\mathcal{S}$ as a series of events highly suggests $\mathfrak{a}$ to be happening. If $\mathfrak{m} \geq \mu$ and $\mathfrak{n} \simeq 0$, then $\mathcal{S}$ and $\mathfrak{a}$ show high inverse correlation, i.e. the pattern of $\mathcal{S}$ almost always inhibits the event $\mathfrak{a}$ to happen.

Let $\mathsf{F}(\mathcal{S})[\mathfrak{a}] = \mathfrak{n}$ be a shorthand for the fact that $(\mathfrak{a}, \mathfrak{n}) \in \mathsf{F}(\mathcal{S})$. Let $\mathsf{E}(\mathsf{F}(\mathcal{S}))$, $\mathrm{Var}(\mathsf{F}(\mathcal{S}))$, and $\mathrm{Sum}(\mathsf{F}(\mathcal{S}))$ be the mean, deviation, and the sum of $\mathsf{F}(\mathcal{S})[\mathfrak{a}]$ values, respectively, for all $\mathfrak{a} \in \mathsf{A}$.

Definition 14 (Relevance measure) *Let* Imp *be defined as an importance measure on a sequence $\mathcal{S}$, and $\mathfrak{a} \in \mathsf{A}$ item of a relation $\mathsf{R}(\mathsf{ITA})$ such that*

$$\mathrm{Imp}(\mathcal{S}, \mathfrak{a}) = \begin{cases} 0 & \text{if } \mathrm{Sum}(\mathsf{F}(\mathcal{S}))\mathrm{Sum}(\overline{\mathsf{F}}(\mathcal{S})) = 0 \\ \dfrac{\left|\dfrac{\mathsf{F}(\mathcal{S})[\mathfrak{a}]}{\mathrm{Sum}(\mathsf{F}(\mathcal{S}))} - \dfrac{\overline{\mathsf{F}}(\mathcal{S})[\mathfrak{a}]}{\mathrm{Sum}(\overline{\mathsf{F}}(\mathcal{S}))}\right|}{\max\left(\dfrac{\mathsf{F}(\mathcal{S})[\mathfrak{a}]}{\mathrm{Sum}(\mathsf{F}(\mathcal{S}))}; \dfrac{\overline{\mathsf{F}}(\mathcal{S})[\mathfrak{a}]}{\mathrm{Sum}(\overline{\mathsf{F}}(\mathcal{S})]}\right)} & \text{otherwise} \end{cases},$$

where $|\mathfrak{n}|$ stands for the absolute value of a number $\mathfrak{n}$. We say $\mathcal{S}$ is a relevant antecedent of $\mathfrak{a}$ if $\mathrm{Imp}(\mathcal{S}, \mathfrak{a})$ *is greater or equal to a certain threshold.*

Relevance measure indicates that there is a connection between the frequency and rareness of an item, that is, if an item occurs in every sequence or that item occurs in no sequences, then relevance is equally 0 according to Definition 14. Nevertheless, if there is an item $\mathfrak{a}$ which always appears before or together with an item $\mathfrak{b}$ then relevance is 1 because $\mathsf{F}(\mathcal{S}, \mathfrak{b}) = 1$ and $\overline{\mathsf{F}}(\mathcal{S}, \mathfrak{b}) = 0$, where $\mathcal{S}$ consists of a single itemset that has a single value $\mathfrak{a}$ ($\mathcal{S} =< \{\mathfrak{a}\} >$ for short). By symmetry, if $\mathfrak{a}$ never occurs together or before $\mathfrak{b}$ in any sequences then relevance is still 1 indicating some kind of rejection or inhibition. For example, those who buy lactose free products will not buy milk or cheese. Also notice that, relevance both measures frequencies and infrequences, which leads to a re-formulation how important an item $\mathfrak{a}$ regarding

a preliminary series of events $\mathcal{S}$. That is, it is a potential relative significance measure (see Definition 2).

Definition 15 (Importance measure) *Let* Ind *be defined as a measure on* $\mathcal{S}$ *sequences, and* $\mathtt{a} \in \mathsf{A}$ *items of a relation* $\mathsf{R}(\mathsf{ITA})$ *such that*

$$\mathrm{Ind}(\mathcal{S}, \mathtt{a}) = \frac{\mathrm{Imp}(\mathcal{S}, \mathtt{a}) - \mathsf{E}_{\mathtt{a} \in \mathsf{A}}(\mathrm{Imp}(\mathcal{S}, \mathtt{a}))}{\mathrm{Var}_{\mathtt{a} \in \mathsf{A}}(\mathrm{Imp}(\mathcal{S}, \mathtt{a}))}$$

We say $\mathcal{S}$ *is an import antecedent of* $\mathtt{a}$ *if* $|\mathrm{Ind}(\mathcal{S}, \mathtt{a})|$ *is greater or equal to a certain threshold.*

Importance is a normalized value of relevance to measure how much $\mathcal{S} \rightarrow \mathtt{a}$ is unusual. In most of the cases, mean value of relevance shall be about 0, i.e. occurrences of items are independent in general. Statistically, if absolute value of the $\mathrm{Ind}(\mathcal{S}, \mathtt{a}) \geq 3$ (the triple of the variance), then it is an outlier value that is usually a strong indicator for a deep connection between variables.

We propose Algorithm 1 for identifying important sequences in the sequence database R. The inputs of the algorithm are the database R itself, a μ minimum support threshold, and a ν importance threshold. The output is the set of frequent-and-relevant (important) sequences Σ_f. In the body of the algorithm, a loop variable k, the frequent set $\mathsf{F}(\mathcal{S})$, the inverse set $\overline{\mathsf{F}}(\mathcal{S})$, and the set of new sequences Σ_c are used locally.

The algorithm works as follows. In the initialization phase (line 1), we add items as sequences of length 1 to the Σ_f set, if their support is over the minimum threshold μ. The main loop iterates over the sequences of maximum length. First, it removes all elements from the Σ_f new important sequence set (line 8). It computes the frequent $\mathsf{F}(\mathcal{S})$ and the inverse $\overline{\mathsf{F}}(\mathcal{S})$ sets for the current $\mathcal{S}$ sequence (line 10). If there are candidate postfix items, then we iterate over it, and filter the sequences with the formula of Definition 15. As threshold, we utilize ν an importance threshold input parameter (line 11). If the importance is over that threshold, then that item c is appended to the end of $\mathcal{S}$ (line 12), and the new sequence is added to the important set of sequences (line 13). The main loop is repeated until the Σ_c set generated is not an empty set. If no further candidate sequences can be generated, the algorithm returns the Σ_f set (line 17), otherwise the elements if Σ_c are added to Σ_f (line 18). If all sequences are processed, the k maximum length loop variable is increased (line 20), and the main loop is restarted.

Lemma 16 *Algorithm 1 identifies all frequent patterns, which have a support greater or equal to a min_support according to Definition 9, but the important ones are returned.*

input : $\mathsf{R}(\mathsf{ITA})$ database, μ minimum support threshold, ν importance threshold
output: Σ_f set of important sequences
data : k cycle variable, Σ_c set of new sequences, $\mathsf{F}(\mathcal{S})$ frequent next item set, $\overline{\mathsf{F}}(\mathcal{S})$

```
1  /* Initialization                                          */
2  k := 1;
3  Σ_f = {S|a ∈ A, S =< {a}_{-∞} >, supp(S) ≥ μ };
4  /* Main loop                                               */
5  while true   :
6  do
7      foreach S ∈ Σ_f where len(S) = k do
8          Σ_c := ∅;
9          foreach a ∈ A do
10             Compute the sets F(S, a) and F̄(S, a);
11             if |Ind(R, S, c)| ≥ ν then
12                 S' := concat(S, c);
13                 Σ_c := Σ_c ∪ {S'};
14             end
15         end
16         if Σ_c = ∅ then return Σ_f;
17         ;
18         Σ_f := Σ_f ∪ Σ_c;
19     end
20     k := k + 1;
21 end
```

Algorithm 1: Importance based frequent sequential pattern generation

According to Theorem 11, conditional support $\mathrm{supp}(\mathcal{S}|\mathsf{a})$ is greater or equal to the $\mathrm{supp}(\mathcal{S} \to \mathsf{a})$. It means, that by generating $\mathsf{F}(\mathcal{S})$ Algorithm 1 finds all candidates for which support is greater or equal to a certain threshold. However, if either a or $\mathcal{S}$ is independent, or too frequent in general, it entails $\overline{\mathsf{F}}(\mathcal{S}, \mathsf{a}) \simeq \mathsf{F}(\mathcal{S}, \mathsf{a})$ and as such is omitted from the output. As a consequence, Algorithm 1 is a one-step method to find frequent *and* relevant patterns.

Lemma 17 (Infrequent important candidate) *Algorithm 1 can identify important sequences with regard to the* ν *importance threshold that are not frequent regarding a* μ *threshold.*

Definition 15 is independent from the minimum support threshold μ, so it is possible to construct an example, where the statement of Lemma 17 holds. If $\Sigma = \{<\{a\},\{b\},\{c\}>,<\{c\},\{d\}>\}$, and $\mu = 60\%$, then subsequence $<\{a\},\{b\}>$ can not be frequent as it occurs only in the first closed sequence. However, it is important because $|\mathrm{Ind}(<\{a\},b>)| \geq \nu$ for an appropriate ν because b is always preceded by a.

Algorithm 1 builds important sequences on the pattern of GSP. The number of database scans is two times the number of important sequences identified: the computation of sets $F(\mathcal{S})$ and $\overline{F}(\mathcal{S})$ requires a scan each. With regard to candidate generation and data structure efficiency, there is a lot of room for improvements.

Lemma 18 (Candidate generation) *All subsequences of important sequences generated by Algorithm 1 are important sequences.*

Lemma 18 gives a property similar to that exists in case of sequential pattern generation algorithms, and this way patterns can not only be grown, but joined as well.

5 Empirical analysis

5.1 Application to medical data

In this section, we present the experiments we conducted on real-life clinical data. The clinical database was anonymized [5] before use.

We defined one sequence for each unique patient identifier, i.e. the I set comprises patient identifiers. Treatments and diagnoses have unique medical codes that define the A itemset. Treatment and diagnosis timestamps are aggregated on daily level, that is, two treatments that happened on the same day are considered to be simultaneous and have the same $t \in T$ element associated with them.

The properties of the data set are shown in Table 5. The total number of patient records is about 67 million, which is the size of the relation in the context of this paper. The patient cases, which is equal to the number of natural identifiers, is in the order of 10^5. The average number of examinations of a patient is in the order of 10^2, this value is the average number of items in

a sequence. The average number of days when examinations are performed or diagnoses are given on a patient is around 6, this value is the average number of itemsets in the maximum sequences. The number of treatments, i.e. the number of items is around 10^4.

Number of records in the database	66870306
Number of natural identifiers	455514
Average length of maximum sequences	146.8
Average size of maximum sequences	6.18
Number of items	9291

Table 5: Properties of the data set

5.2 Experimental results

The experiments we conducted on an Oracle Sun Server X3-2 with 256GB RAM and 32 cores of 4 Intel Xeon E5-2660 CPUs. The mining processes were allowed to use up to 48GB of RAM and 200GB of disk space.

As the experiments shown in Section 2.5 use the API of [4], where the algorithms are implemented in Java, we used the Java implementation of our method. We experimented with the implementation of PrefixSpan provided by [14], however that run out of the 200GB disk space limit before finishing. The minimum support threshold was set to the absolute value of 10 occurrences in all cases. In REVIEW, we used a 3 as the importance threshold to provide output sets of similar size as the other two algorithms for short sequences. User time usage and memory usage were both measured with the UNIX command `time`.

Since the preliminary experiments with PrefixSpan and SPADE have shown that these algorithms are not able to process this amount of data within reasonable time, once again we have used a random sample and limited the maximum length of sequences to 5, 10, 20, 40, 60, 80, 100, 120 and 140. The highest value used is still below the average length of sequences in the whole database. Table 6 shows the properties of these samples.

Figure 2 compares REVIEW with PrefixSpan and SPADE over the same dataset with the same minimum support threshold settings. The figures show how the execution time requirements and the disk space usage scale as the maximum length of input sequences grows. The number shown is the average of three runs. The algorithms are deterministic, so the number of frequent sequences does not vary. We represent the output sequences in the same form

Max. length	Sequences	Length	Itemsets
5	2150	5048	3441
10	4082	18064	28255
20	6416	50781	17379
40	8979	124289	36448
60	10499	197817	53894
80	11571	271818	71685
100	12263	333065	86099
120	12789	390307	100058
140	13186	441514	112012

Table 6: Properties of the sample data sets

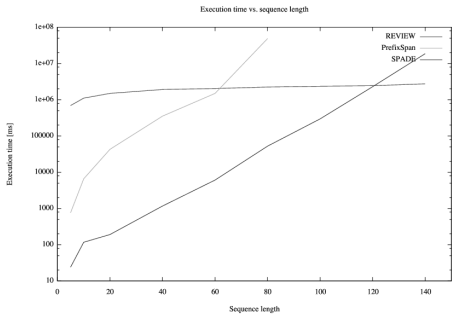
on the disk, so we consider that the number of output sequences is proportional to the disk usage.

REVIEW has been found to scale better as the length and number of input sequences grow than Preview and PrefixSpan. The chart on the left shows that though REVIEW has a high initial time requirement, however it does have a much lower gradient on the log scale than the other two. Around the sequence length of 60 REVIEW becomes quicker than PrefixSpan, and around the sequence length of 120 it surpasses SPADE in speed. Though REVIEW works over the same search space as shown in Theorem 11, it is more effective in filtering frequent sequence candidates than the other algorithms, and yet it keeps the relevant information.

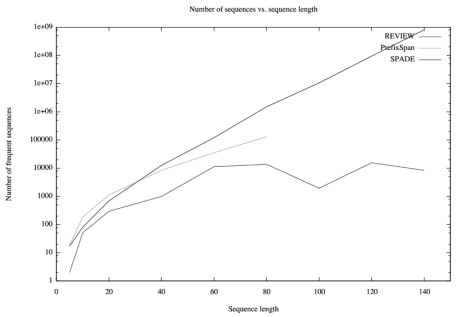
There is no correlation between the memory consumption and the efficiency of the algorithms in case of REVIEW and SPADE. The PrefixSpan implementation used up all of the available memory, while the other two remained well below the limit. In the latter cases, memory consumption seems to depend rather on the Java virtual machine, than the complexity of the algorithm.

6 Conclusions

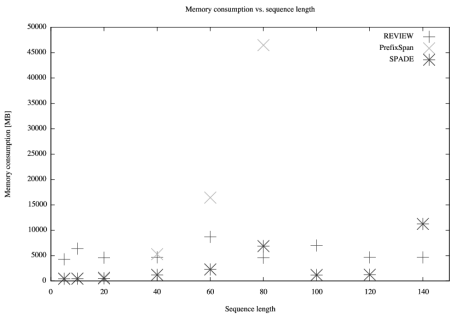
Many studies have elaborated sequential pattern mining methods to improve the overall performance because of the time complexity issues. However, a problem arises when the length of the frequent sequences increases or the number of apriori frequent items is high enough. The previously developed sequential pattern mining algorithms address the performance issue only regardless of whether a pattern with two or more items correlate somehow or



(a) Sequence length vs. execution time



(b) Sequence length vs. frequent sequences



(c) Sequence length vs. memory consumption

Figure 2: Performance of REVIEW against PrefixSpan and SPADE. The maximum length in a sequence database vs. the execution time, the number of frequent sequences and memory consumption

they co-occurrence is frequent because their apriori frequencies are independently high among customers. On the other hand, sequential pattern mining algorithms often ignore niche segments' patterns due to their relative infrequencies.

In this paper, we proposed REVIEW, a new approach how to deal with frequent closed sequences. It iteratively calculates the conditional frequencies of patterns and their possible follow-ups for those closed sequences in which a follow-up appears, and those in which it does not. If measures show statistically significant differences then pattern is extended by the follow-up item, and it is found to be important, and a new cycle with the extended sequence begins. The algorithm stops when there can be made no extensions.

We proved that this method finds all relevant and frequent sequential patterns in linear time regarding the number of closed sequences. We demonstrated by experiments that our method significantly improves performance of those known from literature on a health care database where both independent, apriori frequent items, and long sequences are both present at the same time. Moreover, REVIEW also pointed out that not so frequent diagnoses show strong correlation with others which would be missed by other methods.

Acknowledgement

Publishing of this work and the research project were funded by the European Union and co-financed by the European Social Fund under the name of ,,MEDICSPHERE – Complex, multipurpose, ICT technology driven medical, economic, and educational use of clinical data" and grant number TÁMOP-4.2.2.A-11/1/KONV-2012-0009.

References

[1] R. Agrawal, R. Srikant, Mining sequential patterns, *Proc. Eleventh International Conference on Data Engineering*, Taipei, Taiwan, 1995, pp. 3–14. ⇒300

[2] L. M. Aouad, Nhien-An Le-Khac, T. M. Kechadi, Performance study of distributed apriori-like frequent itemsets mining, *Knowledge and Information Systems*, **23**, 1 (2009) 55–72. ⇒300

[3] J. Ayres, J. Gehrke, T. Yiu, J. Flannick, Sequential pattern mining using bitmaps, *Proc. Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Canada, July 2002, pp. 429–435. ⇒291

[4] P. Fournier-Viger, SPMF – an open-source data mining library, 2014. ⇒ 291, 306

[5] T. Z. Gál, G. Kovács, Z. T. Kardkovács, Survey on privacy preserving data mining techniques in health care databases, *Acta Univ. Sapientiae, Informatica*, **6,** 1 (2014) 33–55. ⇒305

[6] L. Geng, H. J. Hamilton, Interestingness measures for data mining: A survey, *ACM Computing Surveys (CSUR)*, **38,** 3 (2006) ⇒292, 293, 294

[7] K. Gouda, M. Hassaan, Mining sequential patterns in dense databases, *International Journal of Database Management Systems (IJDMS)*, **3,** 1 (2011) 179–194. ⇒291

[8] J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation, *Proc. International Conference Management of Data (ACM-SIGMOD '00)*, Dallas, USA, May 2000, pp. 1–12. ⇒290

[9] T. P. Hong, C. W. Lin, K. T. Yang, S. L. Wang, A heuristic data-sanitization approach based on TF-IDF, *Proc. 24th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems*, *Lecture Notes in Artificial Intelligence* **6703** (2011) 156–164. ⇒301

[10] K. McGarry, A survey of interestingness measures for knowledge discovery, *The Knowledge Engineering Review*, **20,** 1 (2005) 39–61. ⇒292

[11] P. W. Purdom, D. Van Gucht , D. P. Groth, Average-case performance of the apriori algorithm, *SIAM Journal on Computing*, **33,** 5 (2004) 1223–1260. ⇒300

[12] G. Salton, E. A. Fox, H. Wu, Extended boolean information retrieval, *Communications of ACM*, **26,** 12 (1983) 1022–1036. ⇒288, 301

[13] R. Srikant, R. Agrawal, Mining sequential patterns: Generalizations and performance improvements, *Proc. 5th International Conference on Extending Database Technology: Advances in Database Technology (EDBT '96)*, *Lecture Notes in Security and Cryptology* **1057**, (1996) 3–17. ⇒290

[14] Y. Tabei, An imprementation of PrefixSpan (prefix-projected sequential pattern mining), 2008. ⇒306

[15] M. J. Zaki, SPADE: An efficient algorithm for mining frequent sequences, *Machine Learning*, **42,** 1–2 (2001) 31–60. ⇒290, 295

Received: September 11, 2014 • Revised: November 10, 2014

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