ITHEA

International Journal
INFORMATION THEORIES & APPLICATIONS

2010 Volume 17 Number 2

International Journal
INFORMATION THEORIES & APPLICATIONS
Volume 17 / 2010, Number 2

International Journal "INFORMATION THEORIES & APPLICATIONS" (IJ ITA)
is official publisher of the scientific papers of the members of
the ITHEA International Scientific Society

IJ ITA welcomes scientific papers connected with any information theory or its application.
IJ ITA rules for preparing the manuscripts are compulsory.
The **rules for the papers** for IJ ITA as well as the **subscription fees** are given on *www.foibg.com/ijita*.

Responsibility for papers published in IJ ITA belongs to authors.

General Sponsor of IJ ITA is the **Consortium FOI Bulgaria** (www.foibg.com).

International Journal "INFORMATION THEORIES & APPLICATIONS" Vol. 17, Number 2, 2010

Printed in Bulgaria

Edited by the **Institute of Information Theories and Applications FOI ITHEA**, Bulgaria, in collaboration with:
V.M.Glushkov Institute of Cybernetics of NAS, Ukraine,
Institute of Mathematics and Informatics, BAS, Bulgaria,
Universidad Politécnica de Madrid, Spain.

Publisher: **ITHEA**
Sofia, 1000, P.O.B. 775, Bulgaria. www.ithea.org, www.foibg.com, e-mail: info@foibg.com

ISSN 1310-0513 (printed) **ISSN 1313-0463 (online)** **ISSN 1313-0498 (CD/DVD)**

About:

In this and the next issues, we publish a collection of papers from:

– the Institute for Informatics and Automation Problems at the National Academy of Sciences of the Republic of Armenia

– the Faculty of Informatics and Applied Mathematics of Yerevan State University of the Republic of Armenia

Firstly, we will give short information about these outstanding institutions.

INSTITUTE FOR INFORMATICS AND AUTOMATION PROBLEMS

Institute for Informatics and Automation Problems (IIAP) is the leading ICT research and technology development center at the National Academy of Sciences of the Republic of Armenia. Since its establishment in 1957 IIAP is the only state supported structure for software, hardware and brainware technologies. IIAP has become the leading force in ambitious State programmes developing Information Society and Information Industry in Armenia. ICT - a proven driver for global economic activity and growth, and the convergence point of all multidisciplinary research and all State societal and economic initiatives, - requires IIAP involvement, by serving with mathematical models, electronic implementation, and ICT expertise.

Historically IIAP is linked to the first computer designed and assembled in Armenia in 1960 at the famous Yerevan Scientific Research Institute for Mathematical Machines, whose research partner IIAP became.

Scientific achievements include famous fundamental theoretical results and diverse applied ICT systems and services. First machine translation system for Armenian language was created in 1963-67, which made it possible today the development of computer support tools of Armenian language. The main characteristics of real numbers, functions, and plane curves were investigated within the framework of constructive and intuitive analysis, the estimation of the complexity of logical conclusions in classic and non-classic systems were established. The existence conditions of computable and noncomputable solutions of the general form of recursive equations were investigated. The solutions to discrete optimization problems of isoperimetry and tomography were found and applied in search engines and in pattern recognition. The inductive methods of algorithm synthesis and methods of knowledge representation for expert systems development were investigated. A symmetric cryptosystem SAFER++ was developed, and successfully participated in encryption competitions. In the area of automata theory, the recognisability and definability of languages related to homogeneouuse flow event structures were investigated.

Today the main fields of scientific and applied research include theory of algorithms; theory of automata and applications, mathematical logic; discrete mathematics and combinatorics; information theory and applied statistics; algebraic coding theory; artificial intelligence and management support systems; pattern recognition and image processing; distributed processing and data bases; scientific computations; design and testing; telecommunication and networking.

IIAP has a history of successful international collaborative R&D projects and grants within the following frameworks:

INTAS[1]

- *No. 93-1702 "Efficient symbolic computing".*
- *No. 94-3094 "Information theory and combinatorics".*
- *No. 94-469 "Mathematical and statistical research in information theory and telecommunications.*
- *No. 96-52 "Concurrent heuristics in data analysis and prediction".*
- *No. 00-397 "Data Mining Technologies And Image Processing: Theory And Applications".*
- *No. 00-626 "Data Mining Algorithm Incubator".*
- *No. 01-447 "Weak Arithmetics".*
- *No. 04-77-7173 "Data Flow Systems – algorithms and complexity".*

ISTC[2]

- *No. A-823 "Creation of High-Performance Computation Cluster and Databases in Armenia".*
- *No. A-1451 "Development of Scientific Computing Grid on the Base of Arm cluster for South Caucasus Region".*
- *No. A-1606 "Development of Armenian-Georgian Grid Infrastructure and Applications in the Fields of High Energy Physics, Astrophysics and Quantum Physics".*

EU FP4

- *GEIXS "Geological Electronic Information exchange System".*
- *AMETMAS NOE "Specific research, technological development and demonstration programme in the field of cooperation with third countries and international organizations - Scientific and technological cooperation with the countries of Central and Eastern Europe".*

[1]The independent European science foundation **INTAS** in Brussels was founded in 1993 with the objective of promoting cooperation with scientists from the New Independent States of the Soviet Union in order to preserve the scientific potential of those countries.

[2]**ISTC** (International Science and Technology Center) is an intergovernmental organization connecting scientists from countries of the Commonwealth of Independent States with their peers and research organizations in Canada, EU, Japan, Republic of Korea, Norway and the United States.

EU FP5

- *SPARTA "Security Policy Adaptation Reinforced Through Agents".*
- *ADONIS "Application Development Outsourcing to the New Independent States".*
- *TRISTAN-EAST "TRaining of IST multipliers and Awareness Nurturing in the 3rd Countries of EAST and South East Europe (NIS)".*

EU FP6

- *PORTA OPTICA "Distributed Optical Gateway to Eastern Europe".*
- *Idealist-extend "Extension of idealist34 project (the partner search and NCP support network for participants in the IST priority) in INCO Balkan and NIS countries".*

EU FP7

- *SEE GRID SCI "South East European Grid eInfrastructure for regional eScience".*
- *HP SEE "High-Performance Computing Infrastructure for South East Europe's Research Communities".*
- *BSI "Black Sea Interconnection".*
- *EGIInSPIRE "European Grid Initiative: Integrated Sustainable Pan-European Infrastructure for Researchers in Europe".*
- *IDEALIST "Your Worldwide ICT Support Network"*
- *EXTEND "EXTENDing ICT research co-operation between the European Union, Eastern Europe and the Southern Caucasus"*

IIAP has broad professional contacts in Europe (France, Germany, Hungary, Finland, Bulgaria, Netherlands, United Kingdom, Czech Republic, Poland, etc.), but also Russia, Ukraine, Georgia, Belarus and USA. List of annual publications in refereed scientific journals - more than 100. Active leadership at the helm of IIAP, always is ready to grasp new collaborative R&D opportunities, understanding their importance both for IIAP and Armenia.

IIAP designed and created the Academic Scientific Research Computer Network (ASNET-AM)[1]of Armenia and now is responsible for the developing and managing the infrastructure. IIAP is the leading institute for two State target programs funded by the Armenian Government - Creation of Armenian State Computing System and Deployment of National Grid Infrastructure. As a result the infrastructure[2] of the Armenian National Grid Initiative has been deployed.

[1]**ASNET-AM** unifies more than 50 academic, scientific, research, educational, cultural and other organizations located in the 4 cities of Armenia.

[2] The Armenian Grid infrastructure consists of seven Grid sites (about 500 cores) located in the leading research (National Academy of Sciences of the Republic of Armenia, Yerevan Physics Institute) and educational (Yerevan State University, State Engineering University of Armenia) organizations of Armenia (in Yerevan and Ashtarak cities).

IIAP acts as the EC FP7 INCO and ICT National Contact Point for Armenia.

There are many joint activities and structure inside Armenia that involve IIAP, such as

- *The United Multidisciplinary Laboratory of Information Biology of IIAP and the Institute of Molecular Biology (acting as Health NCP for Armenia),*
- *Joint investigations with National Seismic Protection Service Agency, State Hydrometeorology and Monitoring Service of Armenia, Center for Ecological Noosphere Studies of NAS RA aiming at identifying new paradigms and algorithms, mathematical analysis and computer modelling of biological, geological and eco systems.*
- Similar initiatives link IIAP and its modeling and computation power to State Department of Emergency, Ministry of Economy with projects e-Gov, and Computer for All, Broadband Armenia and others. Other cooperation links address issues such as drug design and evaluation, cosmic rays and astronomy, banking and information security, border security, languages, cultural heritage and others. IIAP is moderator between high tech and state and social projects, and regional developments both in physical infrastructural and in electronic infrastructural levels. IIAP organizes Master's degree courses of the IT department of the International-Scientific Center of NAS RA[1] and provides the postgraduate education in computer science field.

[1] ISEC (International Scientific-Educational Center of the National Academy of Sciences of Republic of Armenia) was established in 1997 on the basis of PhD studies operating at National Academy of Sciences of RA. Taking into consideration rich scientific potential of (most of the country's scientific potential is concentrated in) ISEC expanded its activity starting education in master degree programs in 2004.

FACULTY OF INFORMATICS AND APPLIED MATHEMATICS OF YEREVAN STATE UNIVERSITY

Yerevan State University was established in May, 1919 and was officially inaugurated on January 23, 1921.

The Faculty of Informatics and Applied Mathematics (IAM) of the Yerevan State University is a leading educational and scientific centre preparing fundamental research scientists as well as highly-qualified programmers.

The Faculty was founded in 1971 (initially called the Faculty of Applied Mathematics) on the base of the Chair of Computing Mathematics, which was established at the Faculty of Physics and Mathematics in 1957. A decisive role in Faculty's establishment, defining the roadmap of its scientific research and preparing human resources for the Faculty was played by such distinguished scientists as: A.A. Lyapunov, S.N. Mergelyan, Yu.I. Zhuravlev. S.N. Mergelyan was also the first Head of the Chair of Numerical Analysis - one of the three Chairs of the newly founded Faculty.

At present the Faculty of IAM consists of three chairs: the Chair of Programming and Information Technologies, the Chair of Discrete Mathematics and Theoretical Computer Science and the Chair of Numerical Analysis and Mathematical Modelling. The Staff of the Faculty includes more than 10 Doctors of Sciences and around 50 of the employees are Candidates of Science. Many of them are well-known not only in Armenia but also abroad. The Faculty prepares Bachelors specialized in "Informatics and Applied Mathematics" (both, full - and part-time). All three Chairs of the Faculty provide courses for this purpose. The Faculty provides Master degree in four different programmes. Three of them are carried out by the Faculty Chairs and one is being realized in collaboration with SYNOPSYS company. The study plan for the students of the Faculty provides a fundamental mathematical training and a wide spectrum of courses, related to computers and programming; a significant part of study is a practical work realized on computers.

Since its establishment the Faculty of IAM has been collaborating productively with the Yerevan Research Institute of Mathematical Machines and the Institute for Informatics and Automation Problems of NAS of Armenia (former Computing Centre of Academy of Sciences of Armenian SSR), being the main provider of highly-qualified specialists for these organizations. The Faculty has close scientific contacts with leading educational and scientific centres of Russia, such as M.V. Lomonosov Moscow State University, Dorodnicyn Computing Centre and Steklov Mathematical Institute of the Russian Academy of Sciences. Employees and post-graduate students of the Faculty defend their dissertations in Armenia and also in mentioned above centres in Russia. The Faculty has scientific contacts with the Trier University, Germany.

Lots of the graduates are working in different countries in the fields, where computers are used, that is academic and research institutes, universities, state and government organizations, banks, companies with IT profile etc.

ASSOCIATION RULE MINING WITH N-DIMENSIONAL UNIT CUBE CHAIN SPLIT TECHNIQUE

Levon Aslanyan, Robert Khachatryan

Abstract: *This paper considers the association rule mining problem and provides an alternative to APRIORI algorithm, for solving this problem by splitting up the n-dimensional unit cube vertices into the chains using the well know technique introduced by G. Hansel. We use the further development of this technique given by G. Tonyan, which adds computability on levels of chains and chain splits. A brief description of the software implementation of the introduced alternative approach is given.*

Keywords: *association rule, data mining, unit cube.*

ACM Classification Keywords: *1.5. Pattern recognition, H.2.8 Database applications, Data mining.*

Introduction

Association rule mining problem is one of the main objectives of "data mining" research discipline – /finding the knowledge and regularities in large amounts of experimental data sets/. Given a set $I = \{x_1,...,x_n\}$, consisting of n various elements (items) x_i. Subsets of items (itemset) $X \subseteq I$ are considered, and we say that it is given a k-itemset, when $|X| = k$. Let D is a database of records (transactions) that are subsets of the elements (in this view records are lists of elements, but the same information can be given equivalently by the characteristic 0-1 n-vector of itemset X in I). We assume that the records can be in repetition (considering a multiset), but they are provided with an additional field, which composes the key in the database.

We say that a record $T \in D$ is contributing to set of elements X, if $X \subseteq T$. Association rule is an "if-then" type rule $X \Rightarrow Y$, the fulfillment of which is related with certain conditions. Let X, Y be sets of elements where $X \cap Y = 0$. The ratio of number of all records of D contributing to X, and the number of records of D - is called support of X in D,

$$\sup p(X) = |\{T \in D, X \subseteq T\}| / |D|.$$

Next to this is the concept of support for a rule $X \Rightarrow Y$:

$$\sup p(X \Rightarrow Y) = \sup p(X \cup Y).$$

Another important property of rules is the confidence that is defined as:

$$conf(X \Rightarrow Y) = \sup p(X \cup Y) / \sup p(X),$$

which is the conditional probability that a record contains Y when it is known that it contains X. Practical implementation of association rule mining problem is a subject of intense theoretical and algorithmic studies. It is well known that the problem splits naturally into two stages [AGRAW, 1996]. The first step is the construction of frequent fragments, those that occur in the database with frequencies above the predetermined value of support. The second stage is actually the phase of synthesizing the rules of given confidence, from the set of frequent subsets constructed during the first stage of algorithm.

The most accepted algorithm for synthesis of association rules is APRIORI [AGRAW, 1996]. It builds the set of frequent subsets with a so-called building up method. APRIORI considers one-element subsets, and for them by one run on the database computes their frequencies. Next, it considers all two-element subsets, one-element subsets of which are frequent, and verifies their occurrence in the table. Thus the frequent subsets can be building up to the state when it is including subsets that are not frequent enough. Computational complexity is significant, it is especially important because algorithm must be used on very large data volumes.

Are there any alternative approaches for building rules? There is a huge number of approaches, ideas and algorithm that address this issue. In this paper we propose one new approach, which connects the well-known results from the geometry of the n-dimensional unit cube to the problem of algorithmic generation of association rules with given threshold for rule support and confidence.

A brief characterization of this approach is as follows. n-dimensional unit cube B_n is a regular lattice consisting of 2^n vertices that correspond to binary strings of length *n*, which are usually arranged in layers in the way that on the *k*-th layer there are all those vertices that have *k* units (1 values). Vertices that differ in one coordinate are called adjacent and are connected by an edge. Chain in B_n is a sequence of adjacent vertices. A chain is called growing if it contains at most one vertex in one layer.

G. Hansel showed that B_n can be split into growing chains under certain conditions. Further, he considered the monotone Boolean functions and built an algorithm of optimal recognition of these functions using the constructed chains. Relationship of these constructions with the association rules are that frequent subsets with given parameters correspond to a set of zero value vertices of a monotone Boolean function.

Direct use of this technique of Boolean function recognition is difficult because the constructing and storing the Hansel chains is a problem of algorithmic exponential complexity – in computation, and in memory used.

G. Tonoyan was able to offer a computational approach to the work with chains. This is fundamentally and significantly simplifying the recognition algorithm although the reminding complexity is still very high. The idea is in selecting one particular chain split in the collection of Hansel splits. Then a number of functions are introduced that map chains and their elements to each other. In total, this provides the necessary information to recognize

monotone Boolean functions and eliminates the need in storing the complete structure of Hansel chains. This means sensitive economy of memory versus a small additional computation over tye chain split.

The total aim of this paper is to introduce the necessary chain split and computation technique in terms of problems of search of association rules in large databases. Additionally, it is to take into account one more important feature of the problem for mining association rules. It is known that in data mining the number of considered elements, *n,* - is very large. It is also know that frequent subsets consist of relatively small number of elements. According to this an assumption occurs that there exists a value *k* such that all subsets above this power are not frequent. It turns out that the problem of search of frequent subsets is equivalent to decoding of a special class of monotone Boolean functions, which in turn requires an expansion of the results mentioned above for general Boolean functions, according to some restrictions of the set of functions considered. Extended results are introduced in terms of problem of frequent subsets synthesizing, thus providing the way of determining the set of all maximal (largest by inclusion) frequent subsets, without considering and constructing their sub-subsets. This avoids the part that particularly complicates the building up process.

On geometry of the n-dimensional unit cube

Variable with the only values 0 and 1 (false and true) is called a Boolean variable. n-dimensional Boolean function is a single-valued transformation of the set of all vectors composed by n Boolean variables on to the Boolean set $B = \{0,1\}$. The domain where the Boolean function is given is known as the set of vertices of the n dimensional unite cube B_n that is n-th Cartesian degree of set B. B_n is the set of all binary vectors $\tilde{\alpha} = (\alpha_1,...,\alpha_n)$, which are called vertices or points. Usually, B_n also includes a certain structure, a graph, in which vertices of B_n are placed in horizontal layers, a layer contains all the vertices with a given number of *k* ones, $1 \le k \le n$, and the layers are arranged vertically, starting from the zero layer (at the bottom) to the layer with number n. Layer *k* consists of C_n^k vertices. Two vertices $\overline{\alpha}$ and $\overline{\beta}$ are called adjacent if they differ in exactly one coordinate. These neighboring vertices are connected by an edge in the graph structure of B_n.

Vertices of B_n are organized as follows: a point $\tilde{\alpha} = (\alpha_1,...,\alpha_n)$ of B_n precedes the point $\tilde{\delta} = (\delta_1,...,\delta_n)$ of B_n, if $\alpha_i \le \delta_i$, $1 \le i \le n$. The fact that a point $\tilde{\alpha}$ precedes the point $\tilde{\delta}$ is denoted by $\tilde{\alpha} \le \tilde{\delta}$. If $\tilde{\alpha} \le \tilde{\delta}$ and $\tilde{\alpha} \ne \tilde{\delta}$ then we write $\tilde{\alpha} \prec \tilde{\delta}$. Two different points $\tilde{\alpha}$ and $\tilde{\delta}$ are called comparable if one of the following conditions is true: $\tilde{\alpha} \prec \tilde{\delta}$ or $\tilde{\alpha} \succ \tilde{\delta}$.

It is evident, in general, that to uniquely identify a Boolean function it is necessary to know its values at all points of the n-dimensional unit cube. If the function belongs to some specific class that is narrower than the set of all Boolean functions, then for the unique determination of its values at all points of B_n is not necessary to know in advance the values of function at all points of B_n, and sometimes it is enough to know the values on a subset of B_n. For example, to uniquely identify a symmetric Boolean function of n variables (these functions possess the

same value on each layer of B_n) it is enough to know its values on the set of points from B_n which is intersecting with all layers of B_n .

Boolean function $f(x_1, x_2, ..., x_n)$ is called monotone if from the fact that $\tilde{\alpha} \prec \tilde{\delta}$ it implies that $f(\alpha_1, \alpha_2, ..., \alpha_n) \leq f(\delta_1, \delta_2, ..., \delta_n)$. The class of all monotone Boolean functions of n variables is denoted by M_n . Some geometric properties of monotone Boolean functions are evident. To each function there is a unique set $\hat{f}^0$ of incomparable vertices of B_n , so that $f(\tilde{\alpha}) = 0$ iff $\tilde{\alpha}$ precedes one of these points. Geometrically the area is a union of sbcubes, composed by vetex $\tilde{0}$ and the vertices of $\hat{f}^0$. Another important property is that on growing chains of vertices the function values - 0's and 1's fill two different intervals at most.

Two tasks of recognition about the monotone Boolean functions are rising in different applications. One is the recognition whether $f(\tilde{x}) \in \mathrm{M}_n$, and the second is in deciphering of $f(\tilde{x})$ given that $f(\tilde{x}) \in \mathrm{M}_n$. We address the second topic because of it's identity to the problem of frequent itemset mining.

Suppose that an arbitrary (unknown to us) function $f(\tilde{x}) \in \mathrm{M}_n$ is given by an operator A_f , which gives the value $f(\tilde{\alpha})$ by the given input $\tilde{\alpha} \in B_n$. Given the operator A_f it is required to fully restore the set of values of the function $f(\tilde{x})$. After each call to the operator which resumes the value $f(\tilde{\alpha})$ for the point $(\alpha_1, ..., \alpha_n) \in B_n$ other points of B_n become determined through the monotony property. It is clear that we should strive for optimality of these algorithms that is to minimize the steps of applying to A_f .

Consider the set R of all algorithms that solve this problem. That is, for a monotone Boolean function $f(x_1, x_2, ..., x_n)$ an algorithm from R exploiting the operator A_f restores the complete table of values of $f(\tilde{x})$. Obviously the work of algorithms consist of several stages. Algorithm selects a point $\tilde{\alpha} \in B_n$ and with help of operator A_f computes the value $f(\alpha_1, \alpha_2, ..., \alpha_n)$ (selection). The resulting value of the function at the point $\tilde{\alpha}$ is inserted into the table of computed values of the function. The table is extended by monotonity, which includes determination of all points that can't have 0 or 1 values arbitrarily after knowing the value at $\tilde{\alpha}$ (extension). For example if $f(\tilde{\alpha}) = 1$ then for all points $\tilde{\delta}$ that are higher that $\tilde{\alpha}$ (according to the order of vertices defined above) $f(\tilde{\delta}) = 1$ and the table of values of f is filled in accordingly. Next step is the rule that selects another input for operator A_f and the table of values of f is filled again by monotonity. This process is repeated until the table of values is filled completely.

Obviously a pair <algorithm $r \in R$ and monotone function $f(x_1, x_2, ..., x_n)$ > can be associated with a number $\varphi(r, f)$ that is the number of calls to the operator A_f during recovery of table of values of function $f(x_1, x_2, ..., x_n)$ by the algorithm r .

It is appropriate to evaluate the quality of the algorithms R using function $\varphi(R,f)=\min_r \varphi(r,f)$. We have a condition, $f \in M_n$. The complexity of recognition of class of n-dimensional monotone functions can be characterized by function $\varphi(n)=\varphi(R,M_n)=\max_f \varphi(R,f)$, where the maximum is taken over all monotone functions.

Let us introduce some general terms on function deciphering [KOR,1965]. Suppose we are given a certain class N of Boolean functions and a function f, belonging to this class. The set of points $G(f,N)$ from B_n is called resolving set for the pair (f,N), if from the fact that

the function g belongs to *N*,

values of f and g are the same on the set $G(f,N)$

it follows that $f=g$.

To restore the table of values of a functions it is sufficient to determine the values of function on some of its resolving sets. Resolving set $G(f,N)$ is called a deadlock resolving set for (f,N), if no subset of it is resolving for the pair (f,N).

Let us denote by $H(\tilde{\alpha})$ the set of points $\tilde{\delta}$ satisfying the condition $\tilde{\delta} \succ \tilde{\alpha}$, and by $L(\tilde{\alpha})$ - the set of points $\tilde{\gamma}$ such that $\tilde{\gamma} \prec \tilde{\alpha}$.

The upper zero of monotone function $f(x_1,x_2,...,x_n)$ is the point $\tilde{\alpha}$ from B_n such that $f(\tilde{\alpha})=0$ and $f(\tilde{\delta})=1$ for all points $\tilde{\delta} \in H(\tilde{\alpha})$.

The lower one of a monotone function $f(x_1,x_2,...,x_n)$ is a point $\tilde{\alpha}$ such that $f(\tilde{\alpha})=1$ and $f(\tilde{\gamma})=0$ for any point $\tilde{\gamma} \in L(\tilde{\alpha})$.

Let $Z(f)$ denotes the set of all upper zeros of a monotone function $f(x_1,x_2,...,x_n)$, and $O(f)$, - the set of all lower ones. Each monotone Boolean function has a unique deadlock resolving set that is included in its all resolving sets (mention that this is not the case for other classes, fir instance in class of simmetric Boolean functions that we mentioned above above). This deadlock resolving set for a monotone Boolean function is the set $G(f)=Z(f)\cup O(f)$.

V. Kororbkov has obtained the following result concerning the upper and lower grades of $\varphi(n)$.

Korobkov's Theorem

$C_n^{[n/2]}+C_n^{[n/2]+1} \le \varphi(n) \le bC_n^{[n/2]}(1+\varepsilon_n)$ where $b=\frac{8}{(\sqrt[3]{16}-1)^{3/2}}$ and $\varepsilon_n \to 0$ when $n \to \infty$.

The proof of theorem uses the logic algebra function $h(x_1, x_2, ..., x_n)$ defined as:

$$h(\alpha_1, \alpha_2, ..., \alpha_n) = \begin{cases} 1, if\ [n/2]+1 \le \sum_{i=1}^{n} \alpha_i \le n \\ 0, if\ 0 \le \sum_{i=1}^{n} \alpha_i \le [n/2] \end{cases}$$

It's obvious that in this case $G(h)$ has exactly $C_n^{[n/2]} + C_n^{[n/2]+1}$ points.

Chain split in monotone recognition

Let us give the definition of increasing chain and the property of relative supplement:

1. Increasing chain in the structure of B_n, is a sequence $\tilde{\beta}_1, \tilde{\beta}_2, ..., \tilde{\beta}_k$ of elements of B_n such that $\tilde{\beta}_{i+1}$ is obtained from $\tilde{\beta}_i$ with replacement of one zero in the set of coordinates to the one.

2. Suppose we are given three elements $\tilde{\alpha}_1 \prec \tilde{\alpha}_2 \prec \tilde{\alpha}_3$ forming an increasing chain. Relative supplement of $\tilde{\alpha}_2$ on the interval $[\tilde{\alpha}_1, \tilde{\alpha}_3]$ is the fourth element $\tilde{\beta}$, which forms together with $\tilde{\alpha}_1, \tilde{\alpha}_2, \tilde{\alpha}_3$ a two-dimensional subcube (see Figure 1).

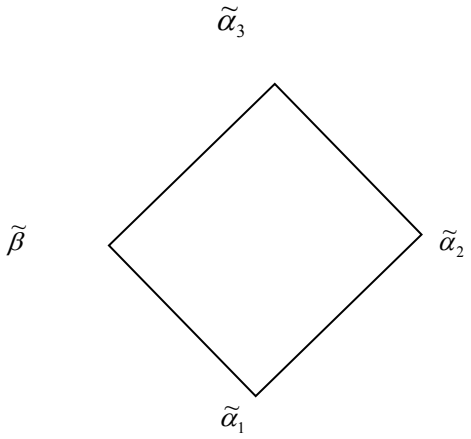


Figure 1

Hansel's Lemma

n dimensional unit cube B_n, endowed with the usual order relation can be covered with a set of $C_n^{[n/2]}$ disjoint increasing chains satisfying the following properties:

- the number of chains of length $n-2p+1$ is equal to $C_n^p - C_n^{p-1}$, where $0 \le p \le [n/2]$. Minimal element of each such chain is a point with p units and $n-p$ zeros and the maximal with p zeros and $n-p$ ones.
- given three elements $\widetilde{\alpha}_1 \prec \widetilde{\alpha}_2 \prec \widetilde{\alpha}_3$ that form an increasing subchain placed on some chain of length $n-2p+1$, then the relative supplement $\widetilde{\alpha}_2$ on the interval $[\widetilde{\alpha}_1, \widetilde{\alpha}_3]$ belongs to a chain of length $n-2p-1$.

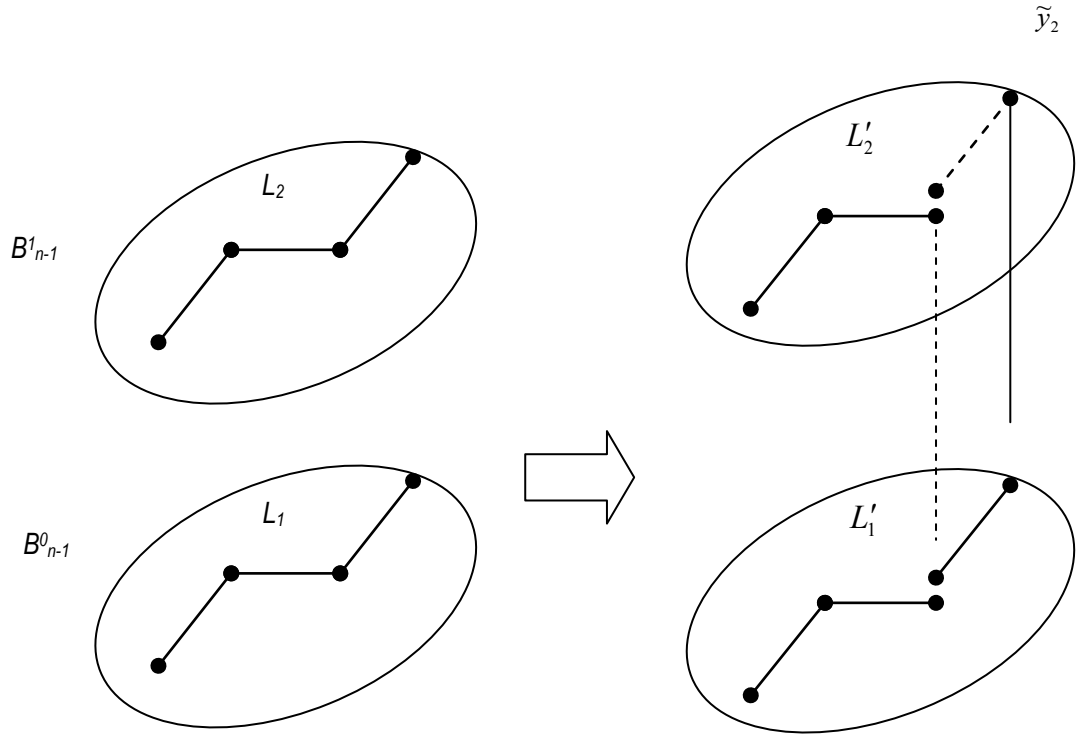


Figure 2.

The proof of this fact is inductive by n. Figure 2 clarifies how the greatest element $\widetilde{y}_2$ of chain L_2 is removed from the L_2 and is added to chain L_1, becoming its new greatest element.

Theorem ([HANS, 1966]**).** Minimal number of operations $\varphi(n)$ of monotone Boolean function's recognition algorithm's is equal to:

$$\varphi(n) = C_n^{[n/2]} + C_n^{[n/2]+1}.$$

Now we bring some more information about the chain computations for the restricted Boolean functions. Let R_n denotes the set of all chains built by and satisfying the conditions 1 and 2 of Hansel's lemma. Consider an element of R_n .The set of all length l chains denote by $[l]_n$ .The optimal algorithm for recovering a monotone Boolean function is denoted by F_0 . The description of F_0 is as follows.

Operator A_f calculates the values of function $f \in M_n$ on the vertices of the shortest chains from $r \in R_n$. These are the chains of length 1 or 2 depending on odd or even values of n . If the values of function f on all elements from set $[l]_n$, $0 \le l \le n$ are known, then according to the monotonity of f , these values are distributed to the set $[l+2]_n$ and according to Hansels's lemma on each chain L of length $l+2$ the values of function f remain unknown for not more than for two vertices. We call these vertices indefinite vertices of L , corresponding to function f . Operator A_f calculates the values of function at these special vertices. This process is applied recursively.

Algorithm F_0 terminates its work, when the values of function *f* are known at all vertices of the chain of length n .

Until now we considered the known means aiming to solve the problem of recovery of monotone Boolean functions $f \in M_n$ with values unknown at all points $\widetilde{\alpha} \in B_n$. Consider a class of monotone Boolean functions of n variables which is narrower than M_n . Suppose that a value k is given, $0 < k < [n/2]$, so that the values of function $f \in M_n$ for vertices of n -dimensional unit cube, which are above the k -th layer equal to 1, and the values are unknown on the reminder area only. Denote this class of function by M_n^k . We consider and adopt the above described technologies in deciphering of this type of monotone Boolean functions.

Theorem. The minimal number $\varphi(n,k)$ of operations of applying to the operator A_f , in recovering the monotone Boolean function's, provided that at all points of n-dimensional cube that are placed higher than some k -th layer, $0 < k < [n/2]$ function equals 1 and on the other points it is unknown, is: $\varphi(n) = C_n^k + C_n^{k-1}$.

Proof. We will show first that $\varphi(n) \ge C_n^k + C_n^{k-1}$ and after that we check $\varphi(n) \le C_n^k + C_n^{k-1}$.

The lower bound will be taken considering some special monotone function:

$$h(\alpha_1, \alpha_2, ..., \alpha_n) = \begin{cases} 1, if\ k \le \sum_{i=1}^{n} \alpha_i \le n \\ 0, if\ 0 \le \sum_{i=1}^{n} \alpha_i \le k-1 \end{cases}$$

We find that $G(h)$, i.e. the deadlock resolving set of function $h(x_1, x_2, ..., x_n)$ contains exactly $C_n^k + C_n^{k-1}$ points as the set of upper zeros are the points of $(k-1)$–th layer of an k-dimensional cube and the set of all lower units will be the points of k-th layer.

We get that $\varphi(n) \geq C_n^k + C_n^{k-1}$.

Now we will prove the upper grade.

It is known that at the points of the cube above the k-th layer function takes the value 1. Consider a chain starting at the k-th layer, we know the values of function at all points on these chains except the starting points, i.e. points located at the k-th layer. By makeing one request for each chain starting from the k-th layer, we will fully recover the function on these chains. Having the values of the function at all points of the chains that start at the k-th layer, we can use the property b) of Hansel's Lemma and property of monotonity to determine the function's values at the points of chains starting at the $k-1$ layer. As a result it turns out that for the chains starting at the $k-1$ layer the functions values remain unknown at no more than two points (see Figure 3).

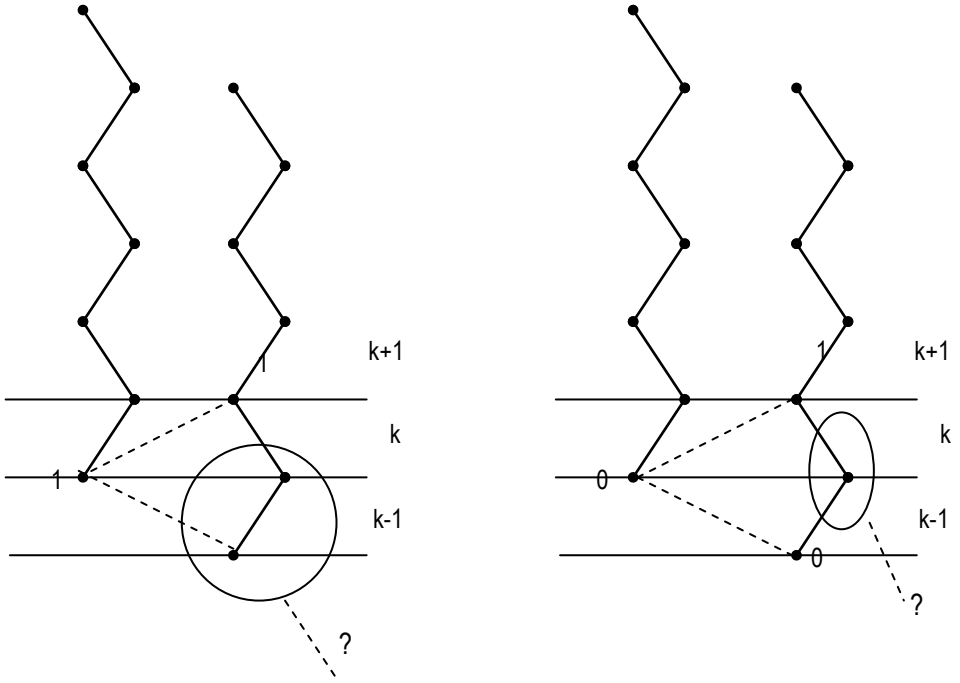


Figure 3.

After performing this procedure for the remaining chains, we conclude that for the complete recognition of a function for the chains starting at the k-th layer is required to determine the function's value only at one point, and that on other chains - at maximum - on two points.

The formula view of the above description is:

φ (n)≤ (the number of chains starting at the k-th layer) +2*(number of other chains starting below the k-th layer).

Let us substitute values using the formula for calculating the number of chains described in the Lemma.

$$\varphi(n) \le C_n^k - C_n^{k-1} + 2(C_n^{k-1} - C_n^{k-2} + ... + C_n^1 - C_n^0 + C_n^0).$$

After making all reductions we receive:

$$\varphi(n) \le C_n^k + C_n^{k-1}.$$

By combining both results we obtain:

$$\left.\begin{array}{l}\varphi(n) \le C_n^k + C_n^{k-1} \\ \varphi(n) \ge C_n^k + C_n^{k-1}\end{array}\right\} \Rightarrow \varphi(n) = C_n^k + C_n^{k-1}.$$

Chain computation

Now we proceed to the collation and analysis of existing knowledge about the chain decomposition of B_n and calculations on chains. Major results here are obtained by G. Tonoyan. These results are connected with the problem of recovering any Boolean function, but they are more universal and applicable to solve other similar problems. Therefore, as a result of our analysis, we will figure out so-called tools i.e. independent procedures that perform particular local operations on the chains, and by the consistent application of which can be solved more global recognition type problems. Specific application will be synthesing of association rules based on the chain split algorithms.

A vertex $\widetilde{\alpha} \in B_n$ is called l-upper zero of function $f \in M_n$, if $f(\widetilde{\alpha}) = 0$ and for any vertex $\widetilde{\beta}$, $\widetilde{\beta} \in \bigcup_{i=2\{n/2\}}^{l} [i]_n = L_l$ from fact that $\widetilde{\beta} \succ \widetilde{\alpha}$ implies that $f(\widetilde{\beta}) = 1$. Here $\{x\}$ denotes the fractional part of x, and $2\{n/2\}$ is to denote the minimal light of chain in chain split, which equals 0 (an isolated point chain) for even n, and 1 for odd n. Remind that $[i]_n$ denotes the set of chains that have length i.

In a similar way a vertex $\widetilde{\alpha} \in B_n$ we call l-lower one of function $f \in M_n$ if $f(\widetilde{\alpha}) = 1$ and for any $\widetilde{\beta} \in L_l$ from the fact that $\widetilde{\beta} \prec \widetilde{\alpha}$ implies that $f(\widetilde{\beta}) = 0$.

We will introduce some definitions that similar but differ somewhat from those already applied.

Let chain $L = (\widetilde{\alpha}_{l+1} \prec ... \prec \widetilde{\alpha}_2 \prec \widetilde{\alpha}_1)$ belongs to R_n. We will say that the length of L is equal to l and $\widetilde{\alpha}_i$ is the i-th vertex of chain L, for $1 \le i \le l+1$.

Let $\widetilde{\alpha} \in L$ and $0 \le k \le \|\widetilde{\alpha}\| - \|\widetilde{\alpha}_{l+1}\|$. Denote by $\widetilde{\alpha}_{(-k)}$ the vertex $\widetilde{\gamma} \in L$ such that $\| \overline{\gamma} \| = \| \overline{\alpha} \| - k$. Through $\widetilde{\alpha}_{(+k)}$ for $\widetilde{\alpha} \in L$ is denoted the vertex $\widetilde{\beta} \in L$ for which $\|\widetilde{\beta}\| = \|\widetilde{\alpha}\| + k$, $0 \le k \le \|\widetilde{\alpha}_1\| - \|\widetilde{\alpha}\|$.

We assume that $\widetilde{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_n) \in B_n$ satisfies property Θ if in $(\alpha_1, \alpha_2, ..., \alpha_n)$, for all $k, 1 \le k \le n$, the number of zero coordinates among the first k coordinates are not less than the number of its unit coordinates.

We assume hat $\widetilde{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_n) \in B_n$ satisfies property Θ' if $(\overline{\alpha}_1, \overline{\alpha}_2, ..., \overline{\alpha}_n)$ satisfies property Θ, where $\overline{\alpha}_i = \begin{cases} 0, & \alpha_i = 1 \\ 1, & \alpha_i = 0 \end{cases}$.

Vertex, which is obtained from $\widetilde{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_n)$ by change of coordinates $\alpha_{i_1}, \alpha_{i_2}, ..., \alpha_{i_s}$ to $\overline{\alpha}_{i_1}, \overline{\alpha}_{i_2}, ..., \overline{\alpha}_{i_s}$ respectively, is denoted as $\widetilde{\alpha}(i_1, i_2, ..., i_s)$.

Finally, a set of numbers $K(\widetilde{\alpha}) = (K_1(\widetilde{\alpha}), K_2(\widetilde{\alpha}), ..., K_n(\widetilde{\alpha}))$ will be associated with vertex $\widetilde{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_n) \in B_n$, where:

$$K_1(\overline{\alpha}) = \begin{cases} 2, & \alpha_1 = 0 \\ 1, & \alpha_1 = 1 \end{cases}$$

$$K_i(\overline{\alpha}) = \begin{cases} K_{i-1}(\overline{\alpha}) + 1, & \alpha_i = 0 \\ K_{i-1}(\overline{\alpha}) - 1, & K_{i-1}(\overline{\alpha}) \ge 2,\ \alpha_i = 1 \\ 1 & K_{i-1}(\overline{\alpha}) = \alpha_i = 1 \end{cases}$$

Tool 1 "VERTEX SECUENTIAL NUMBER ON THE CHAIN"

Vertex $\widetilde{\alpha} \in L$ of chain $L \in R_n$ is the $K_n(\widetilde{\alpha})$-th vertex of the chain L.

At the computational level we are given a binary array of memory of length n as input, and use a work numeric array of memory of length n, where numbers saved do not exceed the n. To fill the area by values $K_i(\tilde{\alpha})$ simple comparison operations and $\pm$ are used and the number of this operations is linear by n.

Tool 2 "NEIGHBOUR VERTEX SECUENTIAL NUMBER ON THE CHAIN"

For a vertex $\tilde{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_n) \in B_n$ such that $\alpha_{i_0} = 0$ (where $1 \le i_0 \le n$) we have:

1. $K_n(\tilde{\alpha}(i_0))$ *is equal to* $K_n(\tilde{\alpha})$ *if and only if there exist a* j, $j \ge 1$, *such that* $K_{i_0+j}(\tilde{\alpha}) = 1$ *and*
2. $K_n(\tilde{\alpha}(i_0))$ *equals to* $K_n(\tilde{\alpha}) - 1$ *or* $K_n(\tilde{\alpha}) - 2$ *in other cases.*

Tool 3 "LENGTH OF CHAIN ADJASENT TO THE VERTEX NEIGHBOUR TO THE GIVEN ONE"

Vertices adjacent (neighbouring) to vertices of chain L with length l from set R_n belong to chains with lengths $l-2$, l, or $l+2$ from the same set.

Tool 4 "CLIMBING UP"

If there exists a value r, such that $K_r(\tilde{\alpha}) = 1$ and $K_s(\tilde{\alpha}) > 1$ for $r < s < n$, then

$$\tilde{\alpha}_{(+1)} = \begin{cases} \tilde{\alpha}(r+1) & for \quad r < n \\ \phi & for \quad r = n \end{cases}$$

and $\tilde{\alpha}_{(+1)} = \tilde{\alpha}(1)$ if the r mentioned doesn't exist.

Let $H(\tilde{\alpha}) = (n_1, n_2, ..., n_s)$, where $n_1, n_2, ..., n_s$ are all the numbers that satisfy the property $K_{n_i - 1}(\tilde{\alpha}) = K_{n_i}(\tilde{\alpha}) = 1$, $1 \le i \le s, n_1 > n_2 > ... > n_s$, and let $H(\tilde{\alpha}) = \phi$ if such configuration doesn't exist.

Tool 5 "CLIMBING DOWN"

Vertex $\tilde{\alpha}_{(-k)}$ can be determined as follows: if $H(\tilde{\alpha}) = \phi$ then

$$\tilde{\alpha}_{(-k)} = \begin{cases} (0, \alpha_2, ..., \alpha_n) & when\ \alpha_1 = 1\ and\ k = 1 \\ \phi & in\ other\ cases \end{cases},$$

and when $H(\widetilde{\alpha}) = (n_1, n_2, ..., n_s)$, then

$$\widetilde{\alpha}_{(-k)} = \begin{cases} \widetilde{\alpha}(n_k, ..., \alpha_1) & when\ k \le s \\ \widetilde{\alpha}(1, n_s, ..., \alpha_1) & when\ \alpha_1 = 1\ and\ k = s+1 \\ \phi & in\ other\ cases \end{cases}.$$

The set of k-th vertices of all chains with length l from set R_n is denoted by $R(n,l,k)$.

Tool 6 "ALL LOWER VERTICES"

If $R(n,l,l+1) \neq 0,$ then $R(n,l,l+1) = \{\widetilde{\alpha} \in B_n\} / \|\widetilde{\alpha}\| = (n-l)/2\ and\ \widetilde{\alpha}\ satisfies\ property\ \Theta\}.$

Tool 7 "ALL UPPER VERTICES"

$$R(n,l,1) = \{\widetilde{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_n) \in B_n\} / (\overline{\alpha}_n, ..., \overline{\alpha}_1) \in R(k+1, l, l+1).$$

Tool 8 "THE CHAIN OF A RELATIVE SUPPLIMENT VERTEX"

Let $\widetilde{\beta} = (\beta_1, \beta_2, ..., \beta_n)$ and $\widetilde{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_n)$ are respectively first and k-th, $1 < k \le l+2$, vertices of chain L of length $l+2$ from set R_n. $\widetilde{\alpha}_1 \prec \widetilde{\alpha}_2 \prec \widetilde{\alpha}_3$ is a chain, $\widetilde{\alpha}_1, \widetilde{\alpha}_2 \in L$, $\widetilde{\alpha}'$ is relational supplement of $\widetilde{\alpha}$ regarding to $\widetilde{\alpha}_1$ and $\widetilde{\alpha}_2$, and $H(\widetilde{\beta}) = (n_1, n_2, ..., n_s)$.

Vertex $\widetilde{\alpha}'$ is the $k-1$-th vertex of l length chain, first vertex of which is

$$\widetilde{\gamma} = \begin{cases} \widetilde{\beta}(n_k), & k \le s \\ \widetilde{\beta}(1), & k = s+1 \end{cases}.$$

Consider $\widetilde{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_n) \in B_n$ and some set $A \subset B_n$. Introduce the following notations:

$$\vec{\alpha} = \begin{cases} \{\widetilde{\alpha}(k),...,\widetilde{\alpha}(n)\},\ \alpha_k = ... = \alpha_n = 1\ and\ \alpha_{k-1} = 0\ or\ \phi \\ \phi, \quad \alpha_n = 0 \end{cases},$$

$$\overleftarrow{\alpha} = \begin{cases} \{\widetilde{\alpha}(1),...,\widetilde{\alpha}(k)\},\ \alpha_1 = ... = \alpha_k = 0\ and\ \alpha_{k+1} = 1\ or\ \phi \\ \phi, \quad \alpha_1 = 1 \end{cases},$$

$\vec{A} = \{\vec{\alpha} / \widetilde{\alpha} \in A\}$ *and* $\overleftarrow{A} = \{\overleftarrow{\alpha} / \widetilde{\alpha} \in A\}$.

The set of all vertices adjacent to set A is denoted as $(A)_*$ (lower neighbours) and $(A)^*$ (higher neighbours).

Tool 9 "LOWER AND HIGHER ADJACENT VERTICES"

$(R(n,l,l+1))_* = \vec{R}(n,l,l+1)$ and $(R(n,l,1))^* = \overleftarrow{R}(n,l,1)$.

Association rule mining alternatives through the chain split technique

At this point we are given a set of tools that create and use chain split and computations in terms of recognizing of monotone Boolean functions $f \in M_n$ so that values are unknown at all points $\widetilde{\alpha} \in B_n$. We also mentioned that specifically we will consider classes of monotone Boolean functions, narrower than the class M_n of all monotone Boolean functions. Suppose that the values of the function at the points of n-dimensional unit cube, which are above a certain k-th layer ($0 < k < [n/2]$) function takes the value 1, and at other points its value is unknown. We consider our dedicated tools to recognise this type of monotone Boolean functions. We will consider three different options for use of tools.

1. We know that above the layer k function f takes constantly the value 1. Let us consider points of k-th layer. For each such point we will define the length of the chain on which it is situated. We also determine which element of chain it is – its position counting from the start. For this purposes we will use the procedure of calculating $K_n(\widetilde{\alpha})$ for each point of k-th layer. According to Tool 1 vertex $\widetilde{\alpha} \in L, L \in R_n$ is the $K_n(\widetilde{\alpha})$-th vertex of chain L. Then we can determine the length of chain as follows. It is known that the point $\widetilde{\alpha}$ has k ones and $n-k$ zeros, where $0 \le k \le n$ and it is known that $\overline{\alpha}$ is the i-th vertex of its chain. It is easy to calculate that the first (biggest, upper) vertex of the chain must contain $(k+i-1)$ ones, and accordingly $n-(k+i-1)$ zeros. Taking into account the symmetry of chains relative to the middle layers, we can calculate that the last vertex of the chain consists of $n-(k+i-1)$ ones and $(k+i-1)$ zeros. Since the chain is obtained by changing one zero in the coordinates of the vertices to one at each crossing of a layer, the length of

chain will be equal to the difference between the number of ones in the first and last vertices of the chain, i.e. for the chain $L \in R_n$ we find that its length is equal to $l = n - 2(k + i - 1)$.

From all points of the k-th layer we mark those that are the last vertices of their chains. For these vertices the length of the chains that pass these vertices is $n - k - k = n - 2k$. With the help of the operator A_f the values of function at these points are determined. Next, consider chains of length $l + 2$ and their lower parts that are below the layer k. According to the symmetry of chains we have that chain of length $l + 2$ end at $(k - 1)$-th layer. Let us consider all points of the *(k-1)*-th layer. Using Tool 6 we will construct the set $R(n, l + 2, l + 3)$ that is the set of all last points of chains with length $l + 2$. $R(n, l + 2, l + 3) = \{\widetilde{\alpha} \in B_n / \|\widetilde{\alpha}\| = (n - l - 2)/2\}$ and $\widetilde{\alpha}$ satisfies the property $\Theta\}$. For each vertex $\widetilde{\alpha} \in R(n, l + 2, l + 3)$ after applying $\widetilde{\alpha}_{(+1)}$ $l + 2$ times we find the appropriate first vertex $\widetilde{\beta}$ of chain that contains $\widetilde{\alpha}$ with and have length $l + 2$. We also find the $l + 2$-nd vertex of considered chain, i.e. the point $\widetilde{\delta}$ that is preceds $\widetilde{\alpha}$. This point is situated at k-th layer and for its determination it is sufficient to apply $\widetilde{\alpha}_{(+1)}$ once. Now let us turn to Tool 8. Let us $\widetilde{\beta} = (\widetilde{\beta}_1, \widetilde{\beta}_2, ..., \widetilde{\beta}_n)$ and $\widetilde{\delta} = (\widetilde{\delta}_1, \widetilde{\delta}_2, ..., \widetilde{\delta}_n)$ are respectively the first and the k-th, $1 < k \le l + 2$, vertices of chain L of length $l + 2$ that belongs to R_n. $\widetilde{\alpha}_1 \prec \widetilde{\delta} \prec \widetilde{\alpha}_2$ is a chain and $\widetilde{\alpha}_1, \widetilde{\alpha}_2 \in L$, $\widetilde{\delta}'$ is the relative supplement of $\widetilde{\delta}$ in regard to $\widetilde{\alpha}_1$ and $\widetilde{\alpha}_2$, and $H(\widetilde{\beta}) = (n_1, ..., n_s)$ is a vector as defined in Tool 5. In this conditions $\widetilde{\delta}'$ is $(k - 1)$-th vertex of a chain with length l, the first vertex of which is

$$\widetilde{\gamma} = \begin{cases} \widetilde{\beta}(n_k), & k \le s \\ \widetilde{\beta}(1), & k = s + 1 \end{cases}$$

For each matched point $\widetilde{\beta}$, that is the point that precedes $\widetilde{\alpha}$, we find the corresponding point $\widetilde{\gamma}$ as described above. From each $\widetilde{\gamma}$ according to the rule $\widetilde{\alpha}_{(-k)}$ we define the point $\widetilde{\delta}'$, which is a supplement to point $\widetilde{\delta}$. Point $\widetilde{\delta}$ is the last vertex of a chain with length l and is placed on the k-th layer. The value of function at this point has already been calculated by the operator A_f. We extend this value by the property of monotonity to the points of chain with length $l + 2$. Remaining unknown points of chain with length $l + 2$ will be defined using the operator A_f.

At the general step, we consider a chain of length $l + m$ and find its first and last vertices, then is finding the corresponding first vertex of chain with length $l + m - 2$, and its last point, which is supplement for the penultimate point of a chain with length $l + m$. At this stage the values of function at all points of the chain with

length $l+m-2$ are already known, and it remains only to extend these values to the chains with length $l+m$ and using the operator A_f to compute the value of function on the remaining unknown points.

2. As in the previous paragraph we will start the function recognition from the k-th layer. We define the points of k-th layer, which are the last points for chains of length l and applying to A_f we calculate the values of the function at these points. We now turn to the chains of length $l+2$, and build the set $R(n,l+2,l+3)$. According to Tool 7:

$$R(n,l+2,1)=\{\widetilde{\alpha}=(\alpha_1,\alpha_2,...,\alpha_n)\in B_n\ /(\overline{\alpha}_n,\overline{\alpha}_{n-1},...,\overline{\alpha}_1,)\in R(n,l+2,l+3)\}.$$

By apply $\widetilde{\alpha}_{(-(l+2))}$ to each vertex $\widetilde{\alpha}\in R(n,l+2,1)$ finding $(l+2)$- nd vertex of chain with length $(l+2)$, which is the relational supplement for $(l+1)$-st vertex of chain with length l, the first vertex of which is obtained by the Tool 8.This method of finding the first vertex makes the task easier compared with the method described in paragraph 1, because of there, for determining the first vertex was necessary to pass step by step through the whole chain starting from the last vertex using $\widetilde{\alpha}_{(+1)}$. In general these methods are identical.

3. Consider another way of recognition, which differs from the first two in the initial notation of the problem. Consider an n-dimensional unit cube B_n, each vertex $\widetilde{\alpha}=(\alpha_1,\alpha_2,...,\alpha_{n)}$ of which is composed of ones and zeros, where 0 indicates that the α_i-th element is involved in the transaction, and 1 - no. Value of the function on the vertex $\widetilde{\alpha}=(\alpha_1,\alpha_2,...,\alpha_{n)}$ is equal to 1 if the set is frequent and 0 otherwise. In these definitions the frequent sets move to the top of the cube, and the function to be recognized takes the following form: on points of B_n that are placed below a certain k-th layer, where $n/2<k\le n$ function takes value 0, and on the rest of cube its values are unknown. All other assumptions and definition of the problem remain unchanged. The recognition process starts from chains that begin at the k-th layer. All points of k-th layer are considered, and for these points corresponding values $K_n(\widetilde{\alpha})$ are calculated. Those points are selected for which $K_n(\widetilde{\alpha})=1$, i.e. points, which are the first points for the chains starting at the k-th layer. Then the lengths of these chains are calculated and the value of function at these points are determined by applying to the operator A_f. We assume the length of chains we consider is l. After that, we consider chains of length $l+2$, i.e., starting at the $(k+1)$-th layer. For this we consider all points of the $k+1$- th layer and select those who have $K_n(\overline{\alpha})$ = 1. The second points of chains with length $l+2$ are found using operation $\widetilde{\alpha}_{(-1)}$. Relational supplements of these points are situated on chains of length l, and are the first points of these chains, and the function values of which we already know. Relative supplements are determined by Tool 8. We extend the function values by monotonity

to the points of chains with length $l+2$. Values of the function at the points remaining unknown are calculated with the help of the operator A_f. We continue the consideration of chains until no unknown points remain.

Software Implementation

For the practical implementation of the above-mentioned methods in systems of association rules mining, a software application was initiated. For this reason an open programming environment (open source) was identified, which is dedicated to the tasks of data analysis, known as "Orange Canvas". This system includes algorithms of commonly used Data Mining techniques i.e. classification, clustering, decision trees, association rules searching, etc. As an algorithm for association rules mining it includes some modification of the Apriori algorithm.

As noted above, we have proposed an alternative approach for solving the problem of association rules mining based on monotone Boolean functions, n-dimensional cube and its coverage with Hansel's chains. Without using the results achieved by Tonoyan it is mostly impossible to make effective program implementation of the proposed method, since for large n operational resources of regular computers are not sufficient to store the current computational data. The reason for this is that for Hansel's technique requires to build the cube completely, as well as to build its chain coverage and keeping it in memory. The application of Tonoyan's Tools allows instead of permanent storage of a cube and its coverage, to calculate the required points of the cube i.e. the first and last vertices of the chains, adjacent vertices of a given point, relative supplement of a given point at a certain interval, etc.

For complete data processing, sometimes it is not enough to apply the association rules technique. It was therefore decided not only to create a software implementation of the algorithm, but to introduce it into the existing Orange Canvas, where a number of data analysis approaches are implemented already.

Bibliography

[KOR,1965] B. K. Korobkov. On monotone functions of algebra of logic, Problemy Kibernetiki, Nauka, Moscow, v. 13, pp. 5-28 (1965).

[AS, 1979] L. Aslanyan. Isoperimetry problem and related extremal problems of discrete spaces, Problemy Kibernetiki, Nauka, Moscow, v. 36, pp. 85-126 (1976).

[HANS, 1966] G. Hansel. Sur le nombre des functions booleennes monotones de n variables, C.R. Acad. Sci. Paris, 262, serie A (1966), pp. 1080-1090.

[TON, 1976] G. P. Tonoyan. Chain decomposition of n dimensional unit cube and reconstruction of monotone Boolean functions, JVM&F, v. 19, No. 6, 1532-1542 (1976).

[AGRAW, 1996] R. Agrawal, H. Mannila, R. Srikant, H. Toivone and A. I. Verkamo. Fast discovery of association rules. In U. M. Fayyed, G. Piatetsky-Sharpiro, P. Smyth, and R. Uthurusamy, editors, Advances in knowledge Discovery and Data Mining, AAAI/MIT press, pp. 307-328, 1996.

[KOT, 2006] S. Kotsiantis, D. Kanellopoulos. Association Rules Mining: A Recent Overview, GESTS International Transactions on Computer Science and Engineering, Vol.32 (1), 2006, pp. 71-82.

[AS, 2008] L. Aslanyan and R. Khachatryan. Association rule mining inforced by the chain decomposition of an n-cube, Mathematical Problems of Computer Science, XXX, 2008, ISSN 0131-4645.

[KACH, 2008] R. Khachatryan. Data Mining algorithmic alternatives, in "Proceedings of the Young Scientists Conference on Perspectives for development of molecular and cellular biology" , ISBN 978-5-8080-0724-6, pp. 54-55, Yerevan, 2008.

[ASL-SAH, 2009) L. Aslanyan, H. Sahakyan, Chain split and computations in practical rule mining, International Book Series "Information Science and Computing" , Book 8, "Classification, Forecasting, Data Mining", ISSN: 1313-0455, 132-135(2009).

[ASL-SAH, 2010] L. Aslanyan, H. Sahakyan. On structural recognition with logic and discrete analysis, NIT 2010, Madrid Technical University, 28 Sept. – 2 Oct. 2010, International Journal "Information Theories & Applications", volume 17, number 1, 3-9(2010).

[SAH-ASL, 2010] H. Sahakyan, L. Aslanyan. Chain Split of Partial Ordered Set of k-Subsets, International Book Series "Information Science and Computing" , Book 18, "New Trends in Information Technologies", ITHEA, ISSN: 1313-0455, 55-65(2010).

Authors' Information

Levon Aslanyan *– Head of Department, Institute for Informatics and Automation Problems, P.Sevak St. 1, Yerevan 14, Armenia, e-mail: lasl@.sci.am*

Robert Khachatryan *– Researcher, Institute for Informatics and Automation Problems, P.Sevak St. 1, Yerevan 14, Armenia, e-mail: robert_khachatryan@hotmail.com*

APPROXIMATION GREEDY ALGORITHM FOR RECONSTRUCTING OF (0,1)-MATRICES WITH DIFFERENT ROWS

Hasmik Sahakyan

Abstract: *Existence and reconstruction issues are considered for a class of (0,1)-matrices with different rows. Intending to define quantitative characteristics, maximization of which lead to matrices with different rows in case when the later exist, we consider the number of pairs of different rows.A greedy algorithm is introduced for this purpose and then its optimality is proven for local steps.*

Keywords: *(0,1)-matrices, greedy algorithms*

ACM Classification Keywords: *F.2.2 Nonnumerical Algorithms and Problems:Computations on discrete structures*

Introduction

Matrices with (0,1) elements and prescribed row and column sums is a classical object which appears in many branches of applied mathematics. In combinatorics, such matrices used to encode hypergraphs with prescribed degrees of verticesand related structures, see, for example [LintWilson, 2001]. In statistics, (0,1) matrices with prescribed row and column sums are known as binary contingency tables, see [ChenDiaconisHolmesLiu, 2005]. In Discrete Tomography (0,1) matrices serve for representation of discrete sets. The projections of a matrix in the horizontal and vertical directions correspond to the row and column sums of the matrix. There is a known result of Ryser - a necessary and sufficient condition for a pair of vectors being the row and column sums of a (0,1)-matrix ([Ryser, 1957]). (0,1) matrices with prescribed row and column sums and with special geometrical properties/constraints imposed, are addressed for example in [DurrChrobak, 1999], [BarcucciDel LungoNivatPinzani, 1996], [Woeginger, 2001].

We will consider another additional requirement on (0,1) matrices with given row and column sums - the requirement of non repetition of rows. Such a requirement on rows has its origin in terms of the n dimensional unit cube. Vertices of the cube are presented as n-tuples of 0,1 values, and in this way a vertex subset has been presented as a (0,1)-matrix, where rows correspond to vertices. Row sums indicate the layers of the cube containing the corresponding vertices. Let $R=(r_1,\cdots,r_m)$ and $S=(s_1,\cdots,s_n)$ denote the row and column sum vectors of a $(0,1)$-matrix of size $m\times n$. Then i-th column sum identifies the number of vertices in the vertex subset with 1 value in i-th position. Now existence of a $(0,1)$ matrix is equivalent to the existence of m vertices situated in r_1-th, r_2-th, etc. r_m-th layers such that s_1 vertices/tuplescontain 1 on the first position, s_2 verticescontain 1 on the second position, etc. and s_n contain 1 on the n-th position. In other words s_i and $m-s_i$ are the partition sizesof the vertex subset on i-th direction. In case when the placement of vertices is not

important and we are interested just in existence of a vertex subset with given partition sizes – we search out a subclass of matrices with column sums $S = (s_1, \cdots, s_n)$ and with m rows which are all different.

Both cases (with or without $R = (r_1, \cdots, r_m)$) are known as algorithmically open problems. The current research is focused on the second one and considers issues of *existence* and *construction* of matrices in a constructive way. A greedy algorithm is proposed and then proven its optimality in local steps. Nevertheless there are examples of matrices showing non optimality of the algorithm globally.

$(0,1)$ matrices with different rows

Consider a $(0,1)$-matrix of size $m \times n$. Let $R = (r_1, \cdots, r_m)$ and $S = (s_1, \cdots, s_n)$ denote the row and column sum vectors of the matrix respectively, and let $U(R,S)$ be the class of all $(0,1)$-matrices with row sum R and column sum S.A necessary and sufficient condition for the existence of a $(0,1)$ matrix of the class $U(R,S)$ was found by Ryser. Another result of Ryser is the definition of an interchange operation to be a transformation which replaces the 2x2 submatrix $\begin{bmatrix} 10 \\ 01 \end{bmatrix}$ of a matrix into the 2x2 submatrix $\begin{bmatrix} 01 \\ 10 \end{bmatrix}$ or vice versa. Clearly an interchange (and hence any sequence of interchanges) does notchange the row and column sum vectors of a matrix, and therefore transforms a matrix in $U(R,S)$ into another matrix in $U(R,S)$. Ryser proved that given $A, B \in U(R,S)$ there is a sequence of interchanges which transforms A into B.

We are interested in a subclass of $U(R,S)$ where all the rows of matrices are different and in particular we will consider the class $U(S)$ of all $(0,1)$-matrices with column sum $S = (s_1, \cdots, s_n)$ and with m rows which are all different.

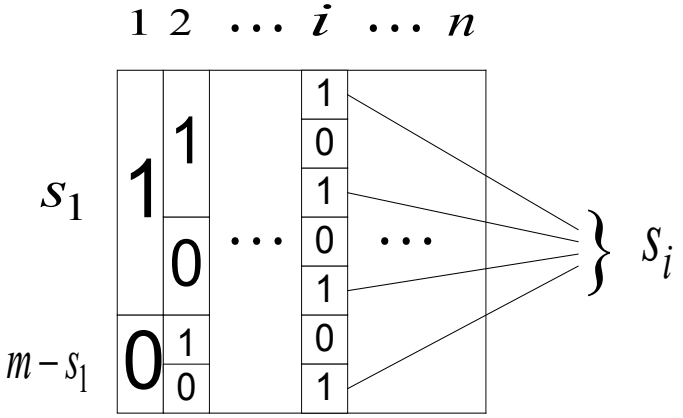
Now we formulate two versions of the problem: decision (P1) and optimization (P2).

(P1) Existence of a $(0,1)$ ***matrix with the given column sum and with different rows***

Remain that no polynomial algorithms are known for solving (P1)and it is known as an open problem. The combinatorial origin is the hypergraph degree sequence problem.

First we define an interchange operation for the class $U(S)$. Let usdefine itin analogous to Ryser's way:the interchange operation replaces the 2x1 submatrix $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ into the 2x1 submatrix $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ or vice versa. Clearly not every interchange operation is admissible: it keeps column sums but it may induce repeating rows and thus may transform a matrix in $U(S)$ into a matrix out of $U(S)$.We call this - simple interchange operation. However performing all interchanges within a pair of rows will keep a matrix within the class. But it is simply the interchange of rows.

Let $S=(s_1,\cdots,s_n)$, and $A\in U(S)$. Applying row interchanges we can transform A into another matrix of $U(S)$ where s_1 ones form an interval in the first column and situated in $1,\cdots,s_1$ rows. Then in the same way we can transform the obtained matrix into another one where s_2 ones compose two intervals in the second column (say $s_{2,1}$ and $s_{2,2}$ lengths, $s_2=s_{2,1}+s_{2,2}$) situated in $1,\cdots,s_{2,1}$ and $s_1+1,\cdots,s_1+s_{2,2}$ rows respectively, and so on. So in each column we will get alternating 1 and 0 intervals (possibly of 0 lengths) as illustrated in the figure below. Rows i and j taken from different intervals are differing; rows within the same interval coincide with each other.



We will call this construction a matrix of *partitioned form*. Starting from some column (at least it is the n-th column) all columns consist of all one length intervals. Depending on partitioning of s_i there can be obtained different matrices of partitioned form in the same class $U(S)$.

Note: two different matrices of partitioned form of the same class $U(S)$ can be transformed into each other applying a sequence of simple interchange operations. Concluding, - given $A,B\in U(S)$ there is a sequence of interchanges which transforms A into B.

So with the help of row interchanges each matrix in $U(S)$ is transformable into a partitioned form. Therefore if the class $U(S)$ is not empty then it contains at leastone matrix of this form. Then it is reasonable to find solution among the matrices of partitioned form. It can be realized constructing a matrix column by column and providing s_i ones in i-th column. If in n-th column we get all one length intervals, then the matrix will not have coinciding rows. During the construction of each column the partitioning of intervals of the previous column can be arbitrary keeping only s_i as the sum of all 1 intervals. But it is reasonable to follow a goal which will lead to the required matrix – with different rows.Let us assume that the partitioning of intervals aims to maximize some quantitative characteristics, which leads to the matrices with different rows in case when the later exists.

We will consider one of such characteristics - the number of pairs of different rows,– it was first considered in [S, 1995].

(P2) $(0,1)$ ***matrices with maximum number of pairs of differing rows***

Let $\Im(S)$ define the class of $(0,1)$-matrices of size $m \times n$, having $S = (s_1, \cdots, s_n)$ as its column sum vector. In this way $U(S) \subseteq \Im(S)$. For a given $A \in \Im(S)$ let $D(A)$ denote the number of pairs of differing rows of A.

Consider the following optimization problem:

P2:Find $A_{opt} \in \Im(S)$ such that $D(A_{opt}) = \max_{A \in \Im(S)} D(A)$.

Obviously C_m^2 is the lowest upper bound for $D(A)$ and it is achievable only for matrices of $U(S)$. Therefore if $U(S)$ is not empty, then a solution of optimization problem (P2)on $\Im(S)$ will serve also as a solution of existence problem(P1) for $U(S)$: $A_{opt} \in U(S)$.

Thus(P2) is not easier than (P1).

We will introduce an approximation algorithm for solving P2. Further we will prove that the algorithm is optimal in local steps.

Greedy approach for solving (P2)

The greedy heuristic is the most used heuristics for optimization problems. The general approach is as follows: repeatedly execute a procedure which minimizes (maximizes) the increase of the objective function. In some cases such a strategy guarantees the optimal solution.

Given $S = (s_1, \cdots, s_n)$. The goal is to construct a matrix $A_{opt} \in \Im(S)$ such that $D(A_{opt}) = \max_{A \in \Im(S)} D(A)$.

Now we describe an algorithm G that constructs a matrix column by column: starting from the first one and adding a column in each step.The objective function is D: $A \in \Im(S) \rightarrow$ number of pairs of differing rows; and the goal is a matrix with the greatest possible value of D.Let A_G denote the constructed by G matrix and let $\Delta D_k(A_G)$ denote the increase of objective function during the k-th stepof G.

Assume without loss of generality that $s_i \geq m - s_i$, $i = 1, \cdots n$.

Algorithm *G*

Step1. Construction of the first column: it consists of s_1 ones placed in the first s_1 rows-positions followed by $m - s_1$ zeros in others. Two intervals is the result: – the s_1-length interval of ones, and the $(m - s_1)$-length interval of zeros. We denote these intervals by $d_{1,1}^G$ and $d_{1,2}^G$. Hereafter the first sub-index will indicate the number of column and the second – the number of interval within the column. Intervals with odd numbers contain all ones, and intervals with even number contain all zeros. So construction of the first column is in unique way:

$$\begin{cases} d_{1,1}^G + d_{1,2}^G = m \\ d_{1,1}^G = s_1 \end{cases}$$

At this point we get $d^G_{1,1} \cdot d^G_{1,2} = s_1 \cdot (m - s_1)$ pairs of differing (by the first position) rows andthus: $\Delta D_1(A_G) = d^G_{1,1} \cdot d^G_{1,2}$.

Let we have constructed the first $k-1$ columns. In general, $(k-1)$-th column consists of 2^{k-1} intervals filled by ones and zeros accordingly. Since among thema presence of 0-length intervalsis possible and they can not be used anymore, let assume that ($k-1$)-th column consists of p non-zero length intervals denoted by $d^G_{k-1,1}, d^G_{k-1,2}, \cdots, d^G_{k-1,p}$. Recall that the rows coincide within the intervals and differ otherwise. If in some column j we get all one length intervals, then at this moment non repetition of all rows, and hence the maximum number of pairs of different rows is already provided. Further constructionsare arbitrary.

Step k. During this step each $d^G_{k-1,i}$ length interval will be partitioned into $d^G_{k-1,i,0}$ and $d^G_{k-1,i,1}$ length intervals filled by zeros and ones respectively: $d^G_{k-1,i,} = d^G_{k-1,i,0} + d^G_{k-1,i,1}$ such that

$\sum_{i=1}^{p} d^G_{k-1,i,0} = m - s_k$ and $\sum_{i=1}^{p} d^G_{k-1,i,1} = s_k$. The increase of objective function during the k-th step is:

$$\Delta D_k(A_G) = \sum_{i=1}^{p} d^G_{k-1,i,1} \cdot d^G_{k-1,i,0}.$$

We will realize partitions having a goal to minimize length differences of intervals.

The idea is in following: if $s_k = m - s_k, k = 1, \cdots n$, then in each step we would split every interval into 2 equal (± 1) parts and fill by zeros and ones respectively which will lead to all one length intervalsin logarithmic number (minimum possible [Knuth,]) steps.Furthermore,among all integer partitions of $d^G_{k-1,i}: d^G_{k-1,i} = d^G_{k-1,i,1} + d^G_{k-1,i,0}$,the largest product $d^G_{k-1,i,1} \cdot d^G_{k-1,i,0}$ is achieved when $d^G_{k-1,i,1} = d^G_{k-1,i,0}$ .Thus following this strategy would bring to the goal, but in general at each step *k*we have $s_k - (m - s_k)$ extra ones. Trying to be closer to equal lengths of intervals we1) distribute the extra $s_k - (m - s_k)$ ones among intervals keeping a "homogeneous" distribution; and then 2) split the remaining intervals into 2 equal parts– puttingequal number of zeros and ones.

Further we will show that this will satisfy the optimization criterion, - the maximum number of new (i, j) pairs of different rows in each step.

Now describe the process in detail.

Let $r_k = s_k - (m - s_k)$ and assume that there are l odd length intervals among the intervals of $(k-1)$-th column. It is easy to check that r_k and l have the same parity and hence $r_k - l$ is even number.Construction of the k-th column is in 2 phases: distribution of r_k "extra" ones during the first, and distribution of remaining ones during the second phases.

Phase 1.

a) $r_k \le l$

Chose arbitrary r_k intervals among the l odd intervals and put an 1 in each.

b) $r_k > l$

All l odd intervals get an 1. After this we put two by two ones in intervals starting from the intervals of even length then altering from odd to even, and continuing cyclically until all r_k ones have been exhausted. If during the process some short intervals have been filled, they do not participate any longer.It is worth to mention that after putting l ones on odd intervals, we get all even lengths, and $r_k - l$ is even as well, so in this way the process of distribution is correct.After this phase there remain equal numbers of zeros and ones.

Phase 2.

a) $r_k \le l$

Half of the remaining $l - r_k$ odd intervals get one 0, others – one 1, after that all intervals have been split into equal parts and receive equal number of zeros and ones.

b) $r_k > l$

all intervals have been split into equal parts and receive equal number of zeros and ones.

Let c_i denote the difference between the distributedones and zeros on i interval: $c_i = d^{G_1}_{k-1,i,1} - d^{G_1}_{k-1,i,0}$, $i = 1, \cdots, p$. Now we will estimate c_i.

The case of $r_k \le l$ is simple:

$$c_i = \begin{cases} 0, & \text{for all even length intervals} \\ 1, & \text{for } (l + r_k)/2 \text{ odd length intervals} \\ -1, & \text{for } (l - r_k)/2 \text{ odd length intervals} \end{cases}$$

Suppose that in case of $r_k > l$, t complete cycles were performed during the two by two distributions of ones. Let D denote the maximum length of those intervals filled during this process. Thus all even intervals of $(\ge D)$-lengths received at least D extra ones ($(< D)$-lengths are filled). Concerning odd intervals - $(\ge D+1)$-lengths received at least $D+1$ extra ones ($(< D+1)$-lengths are filled). Now let d denote the amount of extra ones (above D) received by each not filled interval. Remain that both D and d are even numbers. Thus after t complete cycles, all even intervals of $(> D)$-lengths receive $D + d$ and all odd intervals of $(> D)$-lengthsreceive $D + d + 1$ extra ones. Remaining extra ones (denote this amount by r') is not enough for a next complete cycle. Continue distribution starting from even intervals. Suppose their number is p_1.

Consider cases.

(1) $0 < r' < 2p_1$

Choose $r'/2$ intervals among p_1 and distribute 2 ones on each.

(2) $r' = 2p_1$

All p_1 intervals get 2 ones.

(3) $r' > 2p_1$

All p_1 intervals get 2 ones. Among odd intervals choose $(r' - 2p_1)/2$ and distribute remaining $r' - 2p_1$ ones by two.

Coming back to estimation of c_i, the picture is as follows:

1. $0 < r' < 2p_1$

$$c_i = \begin{cases} D + d + 1, & \text{for odd length intervals} \\ D + d \text{ or } D + d + 2 & \text{for even length interval} \end{cases}$$

2. $r' = 2p_1$

$$c_i = \begin{cases} D + d + 1, & \text{for odd length intervals} \\ D + d + 2 & \text{for even length interval} \end{cases}$$

3. $r' > 2p_1$

$$c_i = \begin{cases} D + d + 1 \text{ or } D + d + 3 & \text{for odd length intervals} \\ D + d + 2 & \text{for even length interval} \end{cases}$$

Resuming: - $\max_{i,j} | c_i - c_j | \leq 2$ for all i, j pairs of even $(\geq D)$-length and odd $(\geq D+1)$-length intervals.

Now the k-th step is completely described.

Thus on k-th column the lengths are the following:

$d^{G_1}_{k-1,i,1} = \frac{d^{G_1}_{k-1,i} + c_i}{2}$, $d^{G_1}_{k-1,i,0} = \frac{d^{G_1}_{k-1,i} - c_i}{2}$, $i = 1, \cdots, p$ filled by 1 and 0 respectively. Each of the intervals may be of 0-length.

Note. (1) and (3) cases may lead to not uniqueness of constructions. Choice of $r'/2$ even intervals in (1) and $(r' - 2p_1)/2$ odd intervals in (3) will cause branching during the first phase.

The goal is to prove that all branches maximize the increase of objective function – pairs of differing rows – in each local step.

Local Optimality

Theorem

(1) Each step of thealgorithm G is optimal: it provides the maximum increase of the objective function – pairs of differing rows;

(2) All optimal constructions of each column are those according to G.

Proof.

(1) Let we have p non zero intervals in $(k-1)$-th column denoted by: $d_{k-1,1}, d_{k-1,2}, \cdots, d_{k-1,p}$, $k = 2, \cdots, n$. Assume thatduring k-th step of G the the i-th interval of $d_{k-1,i}$ length is partitioned into the $d^G_{k-1,i,0}$, $d^G_{k-1,i,1}$ length intervals filled by zeros and ones respectively; and let $d_{k-1,i,1}$, $d_{k-1,i,0}$ bethe corresponding lengths of intervals obtainedas a result of the optimal partition provided by some algorithm during its k-th step, where $\sum_{i=1}^{p} d_{k-1,i,0} = m - s_k$ and $\sum_{i=1}^{p} d_{k-1,i,} = s_k$.

Thus $\Delta D_k(A_G) = \sum_{i=1}^{p} d^G_{k-1,i,1} \cdot d^G_{k-1,i,0}$ and $\Delta D_k(A) = \sum_{i=1}^{p} d_{k-1,i,1} \cdot d_{k-1,i,0}$ are the corresponding increases of the objective function. We intend to prove that $\Delta D_k(A_G) \geq \Delta D_k(A)$.

$$\Delta D_k(A_G) - \Delta D_k(A) = \sum_{i=1}^{p} \left(d^G_{k-1,i,1} \cdot d^G_{k-1,i,0}\right) - \sum_{i=1}^{p} \left(d_{k-1,i,1} \cdot d_{k-1,i,0}\right) =$$

$$\sum_{i=1}^{p} \left(\frac{(d^G_{k-1,i} + c_i)}{2} \cdot \frac{(d^G_{k-1,i} - c_i)}{2} \right) - \sum_{i=1}^{p} \left(d_{k-1,i,1} \cdot d_{k-1,i,0}\right) =$$

$$\frac{1}{4} \sum_{i=1}^{p} \left((d^G_{k-1,i})^2 - (c_i)^2 - 4 \cdot d_{k-1,i,1} \cdot d_{k-1,i,0} \right) =$$

$$\frac{1}{4} \sum_{i=1}^{p} \left(\left(d_{k-1,i,1} + d_{k-1,i,0}\right)^2 - (c_i)^2 - 4 \cdot d_{k-1,i,1} \cdot d_{k-1,i,0} \right) = \frac{1}{4} \sum_{i=1}^{p} \left(\left(d_{k-1,i,1} - d_{k-1,i,0}\right)^2 - (c_i)^2 \right)$$

We denote by α_i the length differences for i-th interval provided by algorithms G and A: $\alpha_i = d_{k-1,i,1} - d^G_{k-1,i,1}$, $i = 1, \cdots, p$. Obviously $d_{k-1,i,0} - d^G_{k-1,i,0} = -\alpha_i$. Hence $d_{k-1,i,1} = \alpha_i + d^G_{k-1,i,1}$ and $d_{k-1,i,0} = -\alpha_i + d^G_{k-1,i,0}$, which implies:

$d_{k-1,i,1} - d_{k-1,i,0} = 2\alpha_i + (d^G_{k-1,i,1} - d^G_{k-1,i,0}) = 2\alpha + c_i$.

Notice that $\sum_{i=1}^{p}\alpha_i = 0$ as $\sum_{i=1}^{p} d_{k-1,i,1} = \sum_{i=1}^{p} d^G_{k-1,i,1} = s_k$.

Thus

$$\Delta D_k(A_G) - \Delta D_k(A) = \frac{1}{4}\sum_{i=1}^{p}\left((c_i + 2\alpha_i)^2 - (c_i)^2\right) = \frac{1}{4}\sum_{i=1}^{p}\left(4\alpha_i c_i + 4\cdot(\alpha_i)^2\right) = \sum_{i=1}^{p}\left(\alpha_i c_i + (\alpha_i)^2\right)$$

and so

$$\Delta D_k(A_G) - \Delta D_k(A) = \sum_{i=1}^{p}\left(\alpha_i c_i + (\alpha_i)^2\right) \quad (1)$$

Consider cases.

1. $\max_{1\le i,j\le p} |c_i - c_j| \le 2$.

Let $\alpha_{j_1} \ge 0,\cdots,\alpha_{j_t} \ge 0,\quad \alpha_{j_{t+1}} < 0,\cdots,\alpha_{j_p} < 0$. For simplification of notations assume that the first are non negative:

$\alpha_1 \ge 0,\cdots,\alpha_t \ge 0,\quad \alpha_{t+1} < 0,\cdots,\alpha_p < 0$. $\sum_{i=1}^{p}\alpha_i = 0$ implies $\sum_{i=1}^{t}\alpha_i = -\sum_{i=t+1}^{p}\alpha_i$.

Thus

$\Delta D_k(A_G) - \Delta D_k(A) = c_{i_0}\cdot\sum_{i=1}^{t}\alpha_i + c_{i_0}\cdot\sum_{i=t+1}^{p}\alpha_i + \sum_{i=1}^{p}(\alpha_i)^2$, where $c_{i_0} = \min_{1\le i\le t} c_i$ and $c_{j_0} = \max_{t+1\le i\le p} c_i$.

We get $c_{i_0}\cdot\sum_{i=1}^{t}\alpha_i - c_{j_0}\cdot\sum_{i=1}^{t}\alpha_i + \sum_{i=1}^{p}(\alpha_i)^2 = (c_{i_0} - c_{j_0})\cdot\sum_{i=1}^{t}\alpha_i + \sum_{i=1}^{p}(\alpha_i)^2 \ge -2\cdot\sum_{i=1}^{t}\alpha_i + \sum_{i=1}^{p}(\alpha_i)^2 =$

$-\sum_{i=1}^{t}\alpha_i - \sum_{i=1}^{t}\alpha_i + \sum_{i=1}^{p}(\alpha_i)^2 = -\sum_{i=1}^{t}\alpha_i + \sum_{i=t+1}^{p}\alpha_i + \sum_{i=1}^{p}(\alpha_i)^2$.

This is non negative since $\alpha_1,\cdots,\alpha_p$ are integers.The proof is completed for case 1.

2. $\max_{1\le i,j\le p} |c_i - c_j| \le 2$ condition may be broken when filled intervals appeared during the complete cycles.

Assume that $d^G_{k-1,i_1},\cdots,d^G_{k-1,i_h}$ are lengths of the filled intervals. So $c_{i_1} = d^G_{k-1,i_1},\cdots,c_{i_h} = d^G_{k-1,i_h}$ takeplace.Thus $d^G_{k-1,i_j} \le D$ for even d^G_{k-1,i_j} and $d^G_{k-1,i_j} \le D+1$ for odd d^G_{k-1,i_j} , $j \in \overline{1,h}$. For remaining intervals: $\max_{i,j\ne i_1,\cdots i_h} |c_i - c_j| \le 2$ is true; and $c_i \ge c_{i_j}$, for $i \ne i_j$, $j \in \overline{1,h}$.

Assume that $\alpha_1 \ge 0,\cdots,\alpha_t \ge 0,\quad \alpha_{t+1} < 0,\cdots,\alpha_p < 0$. Notice that $\alpha_{i_1},\cdots,\alpha_{i_h}$ can not be positive numbers ($\alpha_i = d_{k-1,i,1} - d^G_{k-1,i,1},\quad i = 1,\cdots,p$).

It follows from (1) that

$$\Delta D_k(A_G) - \Delta D_k(A) = \min_{1\le i\le t} c_i \cdot \sum_{i=1}^{t}\alpha_i + \sum_{i=t+1}^{p}\alpha_i c_i + \sum_{i=1}^{p}(\alpha_i)^2 \ge$$

$$\min_{1\le i\le t} c_i \cdot \sum_{i=1}^{t}\alpha_i + \sum_{j=1}^{h}\alpha_j c_j + \max_{\substack{t+1\le i\le p\\ i\ne i_1,\cdots,i_h}} c_i \cdot \sum_{\substack{i=t+1\\ i\ne i_1,\cdots,i_h}}^{p}\alpha_i + \sum_{i=1}^{p}(\alpha_i)^2 = \quad (\text{let } c_{i_0} = \min_{1\le i\le t} c_i \text{ and } c_{j_0} = \max_{\substack{t+1\le i\le p\\ i\ne i_1,\cdots,i_h}} c_i)$$

$$\sum_{j=1}^{h}\alpha_j c_j + c_{i_0}\cdot\sum_{i=1}^{t}\alpha_i + c_{j_0}\cdot\sum_{i=t+1}^{p}\alpha_i - c_{j_0}\cdot\sum_{j=1}^{h}\alpha_j + \sum_{i=1}^{p}(\alpha_i)^2 =$$

$$\sum_{j=1}^{h}\left(\alpha_j\cdot(c_j - c_{j_0})\right) + (c_{i_0} - c_{j_0})\cdot\sum_{i=1}^{t}\alpha_i + \sum_{i=1}^{p}(\alpha_i)^2 .$$

Thus

$$\Delta D_k(A_G) - \Delta D_k(A) = \sum_{j=1}^{h}\left(\alpha_{i_j}\cdot(c_{i_j} - c_{j_0})\right) + (c_{i_0} - c_{j_0})\cdot\sum_{i=1}^{t}\alpha_i + \sum_{i=1}^{p}(\alpha_i)^2 \qquad (2)$$

$(c_{i_0} - c_{j_0})\cdot\sum_{i=1}^{t}\alpha_i + \sum_{i=1}^{p}(\alpha_i)^2 \ge 0$ since $|c_{i_0} - c_{j_0}| \le 2$ and α_i are integers.At the same time $\sum_{j=1}^{h}\left(\alpha_{i_j}\cdot(c_{i_j} - c_{j_0})\right) \ge 0$, since $\alpha_{i_1},\cdots,\alpha_{i_h} \le 0$ and $c_{i_1},\cdots,c_{i_h} \le c_{j_0}$.

Therefore $\Delta D_k(A_G) \ge \Delta D_k(A)$.

Thus we have proven that (1) all branches of algorithm G are optimal in local steps: they provide maximum number of differing rows. Now we prove the converse: (2) all optimal partitions of intervals in each local step appear as a result of algorithm G.

(2) Suppose that $d^G_{k-1,1},\cdots,d^G_{k-1,p}$ are lengths of p intervals ofthe $(k-1)$-th column of the matrix. Let assume that $d^G_{k-1,i,1}$, $d^G_{k-1,i,1}$ are lengths of partitions of the i-th interval appearing as a result of algorithm G at the k-th step, and $d_{k-1,i,1}$, $d_{k-1,i,0}$ are lengths of optimal partitions provided by some algorithm at the k-th step.Let $\Delta D_k(A)$ and $\Delta D_k(A_G)$ denote the increase of objective function at the k-th step according to the optimal partition and partition by algorithm G respectively.It follows from (1) $\Delta D_k(A_G) = \Delta D_k(A)$. All we need to prove that the given optimal partition coincides with some brunch of G.

In (1) we have: $\Delta D_k(A_G) - \Delta D_k(A) = \sum_{i=1}^{p}\left(\alpha_i c_i + (\alpha_i)^2\right)$ where $\alpha_i = d_{k-1,i,1} - d^G_{k-1,i,1}$, $i = 1,\cdots,p$. Hence

$$\sum_{i=1}^{p}\left(\alpha_i c_i + (\alpha_i)^2\right) = 0 \qquad (3)$$

Consider possible cases.

1. $\max\limits_{1\le i,j\le p} | c_i - c_j | \le 2$.

Let

$c_{j_1} = \cdots = c_{j_{t_1}} = c$,

$c_{j_{t_1+1}} = \cdots = c_{j_{t_2}} = c+1$,

$c_{j_{t_2+1}} = \cdots = c_{j_p} = c+2$.

Again for simplification of notifications let assume that

$c_1 = \cdots = c_{t_1} = c$,

$c_{t_1+1} = \cdots = c_{t_2} = c+1$,

$c_{t_2+1} = \cdots = c_p = c+2$:

Putting all this into (3) we get

$$c\cdot\sum_{i=1}^{t_1}\alpha_i + (c+1)\cdot\sum_{i=t_1+1}^{t_2}\alpha_i + (c+2)\cdot\sum_{i=t_2+1}^{p}\alpha_i + \sum_{i=1}^{p}(\alpha_i)^2 =$$

$$c\cdot\left(\sum_{i=1}^{t_1}\alpha_i + \sum_{i=t_1+1}^{t_2}\alpha_i + \sum_{i=t_2+1}^{p}\alpha_i\right) + \sum_{i=t_1+1}^{t_2}\alpha_i + 2\cdot\sum_{i=t_2+1}^{p}\alpha_i + \sum_{i=1}^{p}(\alpha_i)^2 = \sum_{i=t_1+1}^{t_2}\alpha_i + 2\cdot\sum_{i=t_2+1}^{p}\alpha_i + \sum_{i=1}^{p}(\alpha_i)^2 = 0$$

$\sum\limits_{i=1}^{p}\alpha_i = 0$ implies $\sum\limits_{i=t_2+1}^{p}\alpha_i = -\left(\sum\limits_{i=1}^{t_1}\alpha_i + \sum\limits_{i=t_1+1}^{t_2}\alpha_i\right)$.

Hence

$$\sum_{i=t_1+1}^{t_2}\alpha_i + \sum_{i=t_2+1}^{p}\alpha_i - \sum_{i=1}^{t_1}\alpha_i - \sum_{i=t_1+1}^{t_2}\alpha_i + \sum_{i=1}^{p}(\alpha_i)^2 = -\sum_{i=1}^{t_1}\alpha_i + \sum_{i=t_2+1}^{p}\alpha_i + \sum_{i=1}^{p}(\alpha_i)^2 = 0$$

α_i are integers and so $|\alpha_i| \le 1$ for $i \in \overline{1,p}$;and $\alpha_i = 0$ for $i \in \overline{t_1+1, t_2}$ .Thus $\alpha_i = \begin{cases} 1 \\ 0 \end{cases}$ for $i \in \overline{1,t_1}$; $\alpha_i = \begin{cases} -1 \\ 0 \end{cases}$ for $i \in \overline{t_2+1, p}$ where equal numbers of positive and negativeones present. Without loss of generality we assume that no 0 α_i are there. Thus $d_{k-1,i,1} - d^G_{k-1,i,1} = 1$ and hence $d_{k-1,i,0} - d^G_{k-1,i,0} = -1$ for $i \in \overline{1,t_1}$; and

$d_{k-1,i,1} - d^G_{k-1,i,1} = -1$ and hence $d_{k-1,i,0} - d^G_{k-1,i,0} = 1$ for $i \in \overline{1, t_2 + 1, p}$. It follows that the $d^G_{k-1,i,1}$-length interval can not be filled by ones when $i \in \overline{1, t_1}$. Notice also that the lengths of the remaining $i \in \overline{t_1 + 1, t_2}$ intervals are the same in both partitions.

At the same time

$d^G_{k-1,i,1} - d^G_{k-1,i,0} = c$ for $i \in \overline{1, t_1}$ and

$d^G_{k-1,i,1} - d^G_{k-1,i,0} = c + 2$ for $i \in \overline{t_2 + 1, p}$.

Since c and $c + 2$ have the same parity it follows that all intervals in both groups come from either even or odd parts. Since as mentioned above $d^G_{k-1,i,1}$-length intervals are not filled by ones when $i \in \overline{1, t_1}$, then algorithm G had more than one choices, - so we had either (1)or (3)cases. Intervals that receive $c + 2$ extra ones are those that during the last stage of phase 1, received 2 new ones. These 2 ones could be distributed among any other intervals which in our case received c extra ones.

Now all we have to show that this choice made by G coincides with the optimal. Indeed:

$d_{k-1,i,1} - d_{k-1,i,0} = d^G_{k-1,i,1} + 1 - d^G_{k-1,i,0} + 1 = c + 2$ for $i \in \overline{1, t_1}$; and

$d_{k-1,i,1} - d_{k-1,i,0} = d^G_{k-1,i,1} - 1 - d^G_{k-1,i,0} - 1 = c$ for $i \in \overline{1, t_2 + 1, p}$.

So we get intervals where differences between ones are zeros are the same, thus have the same distribution.

2. $\max\limits_{1 \le i, j \le p} | c_i - c_j | \le 2$ condition may be broken if there exist intervals filled during the complete cycles. Let $d^G_{k-1,i_1}, \cdots, d^G_{k-1,i_h}$ are lengths of these intervals. For them $c_{i_1} = d^G_{k-1,i_1}, \cdots, c_{i_h} = d^G_{k-1,i_h}$ is true. Thus $d^G_{k-1,i_j} \le D$ for even d^G_{k-1,i_j}; and $d^G_{k-1,i_j} \le D + 1$ for odd d^G_{k-1,i_j}, where $j \in \overline{1, h}$. For remaining intervals $\max\limits_{i, j \ne i_1, \cdots i_h} | c_i - c_j | \le 2$ take place and $c_i \ge c_{i_j}$, for $i \ne i_j$ and $j \in \overline{1, h}$.

First we show that for the mentioned group of intervals $\alpha_{i_j} = 0$, $j \in \overline{1, h}$.

Recall (2)

$$\Delta D_k(A_G) - \Delta D_k(A) \ge \sum_{j=1}^{h} \left(\alpha_{i_j} \cdot (c_{i_j} - c_{j_0}) \right) + (c_{i_0} - c_{j_0}) \cdot \sum_{i=1}^{t} \alpha_i + \sum_{i=1}^{p} (\alpha_i)^2$$

If c_{j_0} appears among $c_{i_1}, \cdots, c_{i_h}$ then it will imply $c_{i_0} \ge c_{j_0}$ as $c_i \ge c_{i_j}$, for $i \ne i_j$ and $j \in \overline{1, h}$ and hence in (2) we will geta positive summand:

$(c_{i_0} - c_{j_0}) \cdot \sum_{i=1}^{t} \alpha_i + \sum_{i=1}^{p} (\alpha_i)^2 \geq \sum_{i=1}^{p} (\alpha_i)^2 > 0$ in case if a non zero α_i presents. Therefore $\Delta D_k(A_G) > \Delta D_k(A)$ due to $\sum_{j=1}^{h} \left(\alpha_{i_j} \cdot (c_{i_j} - c_{j_0})\right) \geq 0$. But this is acontradiction.

Thus $c_{i_j} - c_{j_0} < 0$ for $i \neq i_j$ and $j \in \overline{1,h}$.

If for some j, $\alpha_{i_j} < 0$ (it can not be positive) then in (2) we geta positive summand $\sum_{j=1}^{h} \left(\alpha_{i_j} \cdot (c_{i_j} - c_{j_0})\right) > 0$.

Therefore $\Delta D_k(A_G) - \Delta D_k(A) > 0$ as $c_{i_j} - c_{j_0} < 0$ and $(c_{i_0} - c_{j_0}) \cdot \sum_{i=1}^{t} \alpha_i + \sum_{i=1}^{p} (\alpha_i)^2 \geq 0$. This again leads to contradiction since $\Delta D_k(A)$ assumed optimal. Therefore $\alpha_{i_j} = 0$, $j \in \overline{1,h}$.

For simplification let $\alpha_j = 0$, $j \in \overline{1,h}$.

Suppose that

$$c_{h+1} = \cdots = c_{t_1} = c,$$

$$c_{t_1+1} = \cdots = c_{t_2} = c + 1,$$

$$c_{t_2+1} = \cdots = c_p = c + 2.$$

Putting into (3) we get:

$$c \cdot \sum_{i=h+1}^{t_1} \alpha_i + (c+1) \cdot \sum_{i=t_1+1}^{t_2} \alpha_i + (c+2) \cdot \sum_{i=t_2+1}^{p} \alpha_i + \sum_{i=1}^{p} (\alpha_i)^2 = 0$$

$$c \cdot \left(\sum_{i=h+1}^{t_1} \alpha_i + \sum_{i=t_1+1}^{t_2} \alpha_i + \sum_{i=t_2+1}^{p} \alpha_i \right) + \sum_{i=t_1+1}^{t_2} \alpha_i + 2 \cdot \sum_{i=t_2+1}^{p} \alpha_i + \sum_{i=1}^{p} (\alpha_i)^2 = 0$$

We have that $\sum_{i=h+1}^{p} \alpha_i = 0$ and $\sum_{i=t_2+1}^{p} \alpha_i = -\left(\sum_{i=h+1}^{t_1} \alpha_i + \sum_{i=t_1+1}^{t_2} \alpha_i \right)$.

Thus

$$\sum_{i=t_1+1}^{t_2} \alpha_i + \sum_{i=t_2+1}^{p} \alpha_i - \left(\sum_{i=h+1}^{t_1} \alpha_i + \sum_{i=t_1+1}^{t_2} \alpha_i \right) + \sum_{i=1}^{p} (\alpha_i)^2 = 0$$

$$\sum_{i=t_2+1}^{p} \alpha_i - \sum_{i=h+1}^{t_1} \alpha_i + \sum_{i=1}^{p} (\alpha_i)^2 = 0$$

α_i are integers which imply that $|\alpha_i| \le 1$ for $i \in \overline{h+1, p}$; and $\alpha_i = 0$ for $i \in \overline{t_1+1, t_2}$. Hence $\alpha_i = \begin{cases} 1 \\ 0 \end{cases}$ for $i \in \overline{h+1, t_1}$, $\alpha_i = \begin{cases} -1 \\ 0 \end{cases}$ for $i \in \overline{t_2+1, p}$, and equal numbers of positive and negative ones present.

The same reasoning as in the previous case will bring that the intervals are the result of a branch of G.

The theorem is proved.

However algorithm G does not provide the global optimum.

Consider the following example: $m = 13, n = 8, S = (11, \cdots, 11)$. All possible realizations of G leave two coinciding rows (see the matrix in left side in figure 1 below). However there exist a matrix in $U(S)$. In the matrix in side in figure 1, the fifth column does not correspond to any realization of G.

$A =$

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	0	1	1
1	1	1	1	0	1	1	1
1	1	1	1	0	1	1	0
1	1	1	0	1	1	1	1
1	1	1	0	1	1	1	0
1	1	0	1	1	1	1	1
1	1	0	1	1	1	0	1
1	0	1	1	1	1	1	1
1	0	1	1	1	1	0	1
0	1	1	1	1	1	1	1
0	1	1	1	1	0	1	1

$A^G =$

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	0
1	1	1	1	1	1	0	1
1	1	1	1	1	0	1	1
1	1	1	1	0	1	1	1
1	1	1	0	1	1	1	1
1	1	1	0	1	1	1	0
1	1	0	1	1	1	1	1
1	1	0	1	1	1	0	1
1	0	1	1	1	1	1	1
1	0	1	1	1	0	1	1
0	1	1	1	1	1	1	1
0	1	1	1	0	1	1	1

Figure 1

Further research concern the study of global properties of the given algorithm and evaluation of results, which is out of scope of the current paper.

Bibliography

[LintWilson, 2001] J.H. van Lint and R.M. Wilson, A Course in Combinatorics. Second edition, Cambridge University Press, Cambridge, 2001

[ChenDiaconisHolmesLiu, 2005], Y. Chen, P. Diaconis, S.P. Holmes, and J.S. Liu, Sequential Monte Carlo methods for statistical analysis of tables, Journal of the American Statistical Association 100 (2005), 109–120,

[DurrChrobak, 1999] Durr Ch., Chrobak M., Reconstructing hv-convex polyominoes from orthogonal projections, Information Processing Letters 69 (1999) pp. 283-291.

[BarcucciDel LungoNivatPinzani, 1996] E. Barcucci, A. Del Lungo, M. Nivat, and R. Pinzani, Reconstructing convex polyominoes from horizontal and vertical projections, Theoret. Comput.Sci., 155:321{347, 1996.

[Woeginger, 2001] G.J. Woeginger, The reconstruction of polyominoes from their orthogonal projections, Inform. Process.Lett., 77:225{229, 2001.

[Ryser, 1966] H. J. Ryser, Combinatorial Mathematics, 1966.

[Knuth, 1973] D. Knuth, TheArt of Computer Programming, vol.3. Sorting and Searching, Addison-Wesley Publishing Company, 1973.

[Sahakyan, 1995] H. Sahakyan, Hierarchical Procedures with the Additional Constraints, II Russia, with participation of NIS Conference, Pattern Recognition and Image Analysis: New Information Technologies, Ulianovsk, 1995, pp.76-78.

Authors' Information

Hasmik Sahakyan *– Leading Researcher, Institute for Informatics and Automation Problems, NAS RA, P.Sevak St. 1, Yerevan 14, Armenia, e-mail: hasmik@ipia.sci.am*

IMPLEMENTATION OF DICTIONARY LOOKUP AUTOMATA FOR UNL ANALYSIS AND GENERATION

Igor Zaslawskiy, Aram Avetisyan, Vardan Gevorgyan

Abstract*: In this paper we present some research results and propose solutions for natural language string lookup techniques. In particular a fast method is suggested for searching dictionary entries for possible matches of sentence words without using relational databases or full dictionary load into machine random access memory. Such approach is essential for minimizing the speed dependency from dictionary size and available machine resources as well as for the scalability of the analyzer software. The mentioned is based on an implementation of Aho-Corasick [Aho, Corasick, 1977] automata with a number of optimizations in the indexing and lookup algorithm.*

Keywords*: UNL, natural language processing, dictionary lookup, indexing, search, XML, pattern matching machine, string matching algorithm, information search*

ACM Classification Keywords*: F.2.2 Non-numerical Algorithms and Problems – Pattern matching, Sorting and searching, I.2.7 Natural Language Processing - Text analysis, Language parsing and understanding, I.7.2 Document Preparation – Index generation, Markup languages, H.3.1 Content Analysis and Indexing – Dictionaries, Indexing Methods, H.3.3 Information Search and Retrieval - Retrieval models, Search process.*

Introduction

UNL (Universal Networking Language) is a meta-language representing semantic information *[Uchida, Zhu, 2005]*. Its main purpose is to store "the meaning" of natural language texts in a language independent format. Each sentence in UNL is a directed linked graph. Unlike natural languages, UNL expressions are less ambiguous. In the UNL semantic networks, nodes represent concepts, and arcs represent relations between concepts. These concepts are called "Universal words" (UWs). The UWs' connections are called "relations". They specify the role of each word in the sentence *[Uchida, Zhu, 2005].*

Many UNL centers and the UNDL Foundation itself have created a number of tools for working with UNL and Natural Language (NL) resources. In this paper we are highlighting some aspects of the development of tools for natural language text analysis and generation. The core tools of the project are the NL Analyzer and NL Generator developed by the UNDL Foundation *[Uchida, Zhu, 2005].*

The aim of the paper

The NL analyzer uses several types of resources to build semantic UNL graphs of NL sentences: NL-UNL dictionary, transformation and disambiguation rule sets. These resources are being used for semi or fully

automatic transformation during the analysis process. Many tools were created for working with UNL-NL and NL-UNL dictionaries, rules and UNL documents, however in the work process of mentioned tools some difficulties arise connected with the increase of the volumes of resources. As the amount of data is growing, the queries to relational databases require more time to return. In this paper we present a specific dictionary lookup tool which was built based on some search algorithms tuned to match specificity of UNL resources and to get better performance.

Finite-state Machine Implementations

Nowadays the finite-state machine implementation algorithms for string lookup are considered giving comparatively effective solutions for the better lookup performance. In particular machines based on algorithms like Boyer–Moore, Knuth–Morris–Pratt and Aho–Corasick are considered, they are widely used for string lookup purposes.

One of the most efficient and known string lookup algorithms is the Knuth–Morris–Pratt string searching algorithm (KMP) *[Knuth, Morris, Pratt, 1977]*. This algorithm searches for occurrences of strings in text. The interesting point in this algorithm is that when a mismatch occurs, the word itself embodies information to determine where the next match can begin, thus bypassing re-examination of previously matched characters.

Another efficient solution is the Aho–Corasick string matching algorithm invented by Alfred V. Aho and Margaret J. Corasick. It has been proved that the running time of the Aho-Corasick Algorithm (ACA), where ***m*** is the length of the text ***T***, ***n*** is the total (cumulative) length of all patterns in ***P***, and ***k*** is the total number of matches of ***P*** in ***T***, is ***O(n + m + k)***. If we compare the work of ACA with the native exact matching algorithm then allowing that ***T*** is searched once for each of the ***z*** patterns in ***P*** and a single search can run in ***O(m)*** time, there are ***z*** iterations of ***T***, and ***O(n)*** amount of work spent looking at the patterns. This results in a total running time of ***O(n + mz)***, which is significant amount of time compared to the linear search time of the ACA. Clearly, the ACA is more efficient than naive exact set matching algorithms *[Spreen, Van Slyke, 2010]*.

Similar to KMP algorithm Boyer-Moore(BM) string matching algorithm is widely used in string lookup purposes. The execution time of the BM algorithm, while still linear in the size of the string being searched, can have a significantly lower constant factor than many other search algorithms: it doesn't need to check every character of the string to be searched, but rather skips over some of them. Generally the algorithm gets faster as the key being searched for becomes longer. Its efficiency derives from the fact that with each unsuccessful attempt to find a match between the search string and the text it's searching, it uses the information gained from that attempt to rule out as many positions of the text as possible where the string cannot match. In 1991 it was proved that the BM time complexity O(m) can have in worst case ***3m*** comparisons, while in best case only ***m/n*** comparisons *[Boyer, Moore, 1977]*.

However it must be noted that both Boyer–Moore and Knuth–Morris–Pratt string search algorithms perform lookup in plain text strings, while Aho-Corasick algorithm is capable to search in text documents represented as lists of strings (which in our case can be the NL-UNL Dictionary). Thus the ACA implementation was chosen as a more suitable solution for NL-UNL Dictionary lookup.

Aho-Corasick Automaton

Let $D = \{y_1, y_2, \dots, y_k\}$ be a finite set of strings that we shall call ***words*** which will be the headwords of dictionary entries that will be included in the pattern matching machine. And let ***x*** be an arbitrary string that we shall call ***text string*** (natural language sentence). Our problem is to find all the substrings of ***text string*** which are ***words*** in ***D***. Substrings may overlap or include each other.

The pattern matching machine for ***D*** is a machine that takes as input the ***text string*** and returns all occurrences of text string substrings that are ***words*** in ***D***. Thus, if we create a pattern matching machine for a dictionary and give to its input a natural language sentence, we shall receive all the dictionary entries that the NL Analyzer may need for the further UNL generation. *[Aho, Corasick, 1977]*

NL-UNL Dictionary structure

Like the conventional dictionaries, NL-UNL dictionaries have simple structure, e.g. [HEADWORD]{UW ID} "UW" (ATTRIBUTES) <L,P,F>; (more simplified : [HEADWORD]<WORD DESCRIPTION >). *[Uchida, Zhu, 2005][UNDL, 2007]*

In NL Analysis the keywords for lookup can be any combinations of letters, digits and special characters, which means that the string matching machine must be constructed to provide maximum speed for searching any type of character combinations. For compilation we created a custom XML-like syntax, the example below illustrates the compiled dictionary syntax.

```
<root>
	<_h_00000182>
		<_e_00000164>
			<e>[he]{UW ID} "UW" (ATTRIBUES) <L,P,F>;</e>
			<_r_00000106>
				<e>[her]{UW ID} "UW" (ATTRIBUES) <L,P,F>;</e>
				<_s_00000047>
					<e>[hers]{UW ID} "UW" (ATTRIBUES) <L,P,F>;</e>
				</_s>
			</_r>
		</_e>
	</_h>
	<_s_00000082>
		<_h_00000064>
			<_e_00000046>
				<e>[she]{UW ID} "UW" (ATTRIBUES) <L,P,F>;</e>
			</_e>
		</_h>
	</_s>
	<_u_00000222>
		<_s_00000204>
			<e>[us]{UW ID} "UW" (ATTRIBUES) <L,P,F>;</e>
			<_h_00000146>
				<_e_00000128>
					<_r_00000110>
						<e>[usher]{UW ID} "UW" (ATTRIBUES) <L,P,F>;</e>
						<_s_00000049>
							<e>[ushers]{UW ID} "UW" (ATTRIBUES) <L,P,F>;</e>
						</_s>
					</_r>
				</_e>
			</_h>
		</_s>
	</_u>
</root>
```

Example 1.

Example 1 illustrates a finite state string matching machine compiled from a dictionary containing entries:

[he] {UW ID} "UW" (ATTRIBUTES) <L,P,F>;

[her] {UW ID} "UW" (ATTRIBUTES) <L,P,F>;

[hers] {UW ID} "UW" (ATTRIBUTES) <L,P,F>;

[she] {UW ID} "UW" (ATTRIBUTES) <L,P,F>;

[us] {UW ID} "UW" (ATTRIBUTES) <L,P,F>;

[usher] {UW ID} "UW" (ATTRIBUTES) <L,P,F>;

[ushers] {UW ID} "UW" (ATTRIBUTES) <L,P,F>;

There are several reasons why we use "XML-like" syntax instead of valid XML. The first reason is that a valid XML document cannot contain special characters or digits as node elements (e.g. < ' > or < 1 > are invalid). The second reason is that in a valid XML document the opening and closing tag contents should be identical, while in our document each opening tag contains an eight digit number describing the count of characters inside the tag; this number enhances the keyword lookup speed by allowing the program to identify the position of the closing tag. It is unreasonable to keep that number in closing tag, because it makes the document larger.

ACA Construction

The first step of ACA implementation is the creation of the Data Object Model (DOM); that is a tree containing all the dictionary entries in a structure described below (*Fig1*). The second step is the exporting of DOM into a "XML-like" text form file (*Example 1*).

DOM constructing algorithm (DOMCA) iterates through the dictionary entries fed in a text file or database. All entries are being inserted into the same DOM by the *DOMCA*. During the process *DOMCA* receives a new entry and separates the headword from the whole entry string, splits the headword into characters and starting from the root element of the current DOM inserts the characters into the tree. If the character of headword has not been inserted into that level of the tree in previous steps, a new node object representing that character is being inserted into that level. After that the pointer shifts to the next character of the headword and repeats same steps considering as a start level element the previous character node that was inserted. If the node already exists, nothing is being inserted, the program only shifts the pointer to the next character of the headword and repeats the same steps considering as a start level element the previous character node that was found inserted in previous level. The program repeats these steps recursively until the pointer reaches the end of the headword and after that inserts a new node containing information about the whole entry string. As a result, the final output will be a single DOM tree with characters as intermediate nodes and dictionary entries as final nodes (*Fig1*).

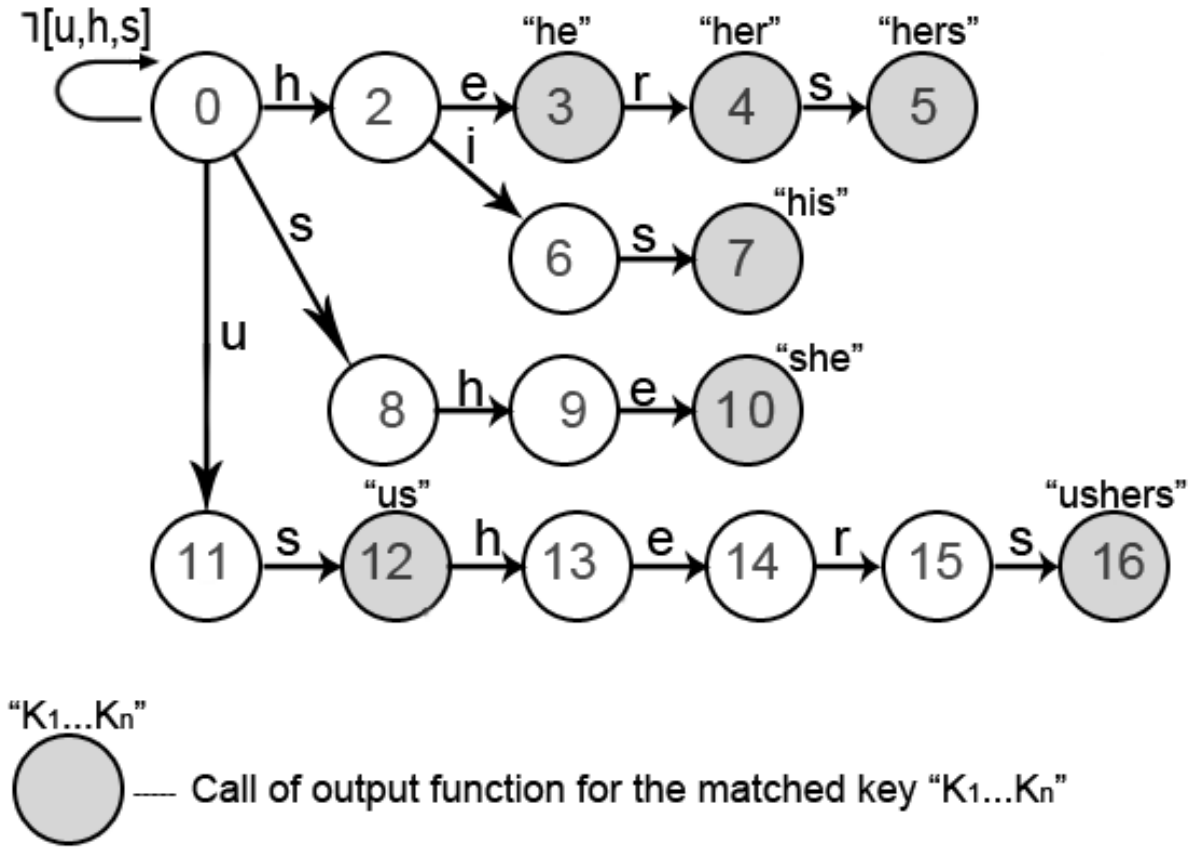


Fig1

.

Exporting DOM into a Text Document

After the pattern matching machine has been created and presented as a DOM, we need to export it into a text document for storing and further usage. We use a function that receives a DOM node element to its input and returns its content as a string including the string values of sub-nodes which in their turn are presented as text string by calling the same function recursively until there are no sub-level nodes left. The final string is an XML-like document (*Example 1*) that represents the DOM (*Fig1*) in a final text form, which will be later used in keyword lookup process.

Dictionary Lookup Using the ACA

String lookup algorithm is applied to a text string ***T*** and a dictionary DOM which is constructed as a directed graph described above.

Let $\boldsymbol{T}$ be a word $\boldsymbol{T} = \sigma_1, \sigma_2, \ldots, \sigma_L$. We define a *goto function* giving for every state k and every letter σ a new state $\boldsymbol{g}(k, \sigma)$ obtained by applying the letter σ (graph edge) in the directed graph DOM to the state k (graph node). For example in *Fig. 1*, the edges labeled from 0 to 11 indicate that $\boldsymbol{g}(0, u) = 11$, $\boldsymbol{g}(0, s) = 8$ and so on. Whenever there is no edge σ for the node k in the graph the *goto* function $g(k, \sigma)$ reports *fail* result.

In our Machine we also define an ***output*** function returning the dictionary entries that match the word generated by the current state (may be empty).

At the beginning the string lookup algorithm considers the words $\boldsymbol{g}(\ldots\boldsymbol{g}(\boldsymbol{g}(0, \sigma_1), \sigma_2)\ldots,\sigma_k)$ such that $k \le L$ while the symbol *fail* does not arise during the process. Every time when the function $\boldsymbol{output}(s_k) \neq empty$ we add its returned entry to the output list of entries. If the failure symbol *fail* has appeared during the word lookup process $\boldsymbol{g}(\ldots\boldsymbol{g}(\boldsymbol{g}(0, \sigma_1), \sigma_2)\ldots,\sigma_L)$ the lookup process for the word $\boldsymbol{T} = \sigma_1, \sigma_2, \ldots, \sigma_L$ stops and starts again considering as the lookup word $\boldsymbol{T} = \sigma_2, \ldots, \sigma_L$, and so on. The list of all output entries obtained during these iterations will be the final output of the String Lookup Algorithm.

Initially, the current state of the machine is the start state ***0*** and the first symbol of the keyword string is the current input symbol. The machine then processes the keyword string by making one operating cycle on each symbol of the keyword string. *[Aho, Corasick, 1977]*

Development of Dictionary Lookup Program Using ACA

The text file of the created pattern matching machine may be very large depending on dictionary size, thus, instead of loading the whole document into the machine's memory, the program reads the document bytes one by one.

Let us assume that we have a keyword w and need to find all the dictionary entries with headwords that match as a whole word or as substring to the w.

The lookup program receives the keyword w and splits it into character array $K = \{ k_1, k_2, \ldots , k_n \}$. On each iteration of $i = \{1, 2, \ldots , n\}$ loop the program searches for an opening tag "<_k_i>" or "<e>" ("<e>" tag indicates a dictionary entry or entry set and is different from "<_e>" tag) . When an opening tag "<e>" is found the program calls the ***output("$k_1k_2\ldots k_{i-1}$")*** function to print the content of the "<e>…</e>" tag which is a match for the keyword substring "$k_1k_2\ldots k_{i-1}$", and continues search, if a tag "<_k_i>" is found, meaning that the i-th character of the keyword has matched, the next character of the keyword becomes the current character ($i = i+1$) and the operations repeat (e.g. for the keyword "ushers", if the "<e>" was found in tag representing the second symbol of the keyword, ***output("us")*** function will be called and the found entries in <e> tag will be printed as the matches for a sub-keyword "us"). If no tag "<_k_i>" is found on the i-th step, the iteration stops and returns the printed values.

After each circle while $|w| > 1$, we remove the first character of k keyword and run lookup program with the new keyword $w' =$ "$k_2k_3\ldots k_n$" again.

For example if the base keyword is "ushers", the keywords that will be sent to the lookup program input are {*w* = "ushers", *w'* = "shers", *w''* = "hers", *w'''* = "ers", *w''''* = "rs", *w'''''* = "s"}. See *Table 1* for the returned results after each call of lookup program.

Keyword	Matches
"ushers"	**"us", "usher", "ushers"**
"shers"	**"she"**
"hers"	**"he", "her", "hers"**
"ers"	**-**
"rs"	**-**
"s"	**-**

Table 1.

Lookup Time Optimization

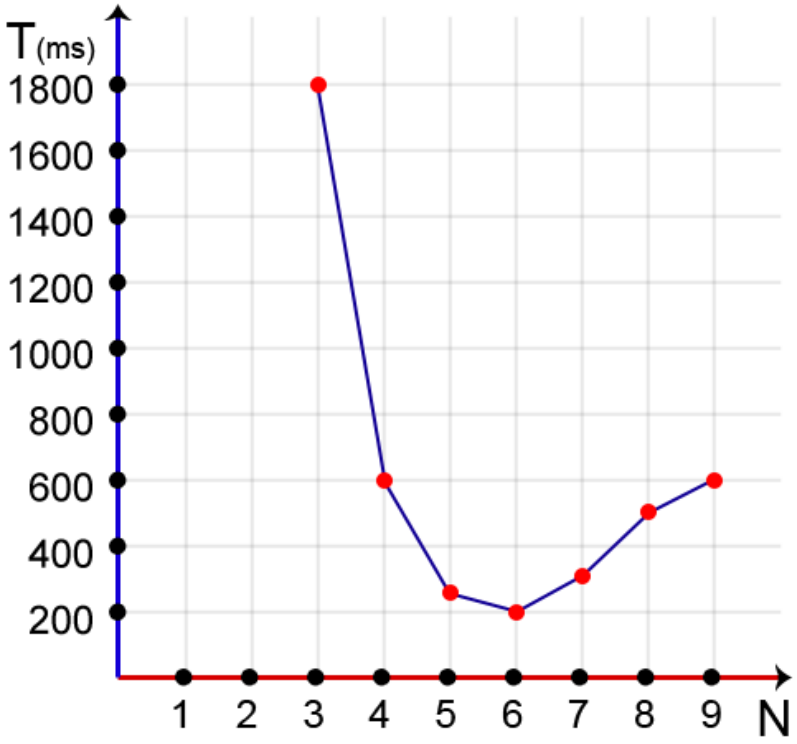
Several enhancements were done in order to minimize the time required for lookup.

1. It was decided to create a mapping file that will contain the byte addresses of the first 6 characters of all headwords of dictionary entries used in pattern matching machine creation. Thus for a keyword $K = \{ k_1, k_2, \ldots, k_n \}$ $(n>6)$, the program can find the byte indexes for sub-keywords {"k_1" , "k_1k_2" , "$k_1k_2k_3$" , "$k_1k_2k_3\ k_4$" , "$k_1k_2k_3k_4k_5$" , "$k_1k_2k_3k_4\ k_5\ k_6$"} from the mapping file and retrieve all dictionary entries that may be allocated in those byte addresses of the dictionary file. After that the program starts the dictionary lookup starting from the "$k_1k_2k_3k_4k_5k_6$" index position with a keyword "$k_7k_8\ldots k_n$". If n ≤ 6, no lookup will be called at all, this significantly minimizes the time required for process. The limit of 6 characters was chosen for mapping as the most optimal number in mapping file size and lookup time relation. By increasing the length of mapped keyword strings more time is required for loading the mapping file, less time for lookup and vice versa.

   Graph 1 illustrates the efficiency of lookup algorithm in required time in milliseconds (T) and mapping characters limit (N) relation. This graph is based on the average of test results held on a PC with dual core CPU of 2.0 GHz frequency and 2 GB of RAM. 10 sentences in English of average 180-200 characters each were chosen as keyword strings for testing with English-UNL dictionaries containing 100'000 to 400'000 entries. According to the graph, mapping character limit of 6 is the optimal number producing the best speed results.

2. When exporting the dictionary string matching machine into a text file, all character nodes are being ordered alphabetically. During the lookup process, when a character tag is found but does not match the

current keyword character, the program is supposed to find the next opening tag in that level. But considering the fact that the tags are stored in alphabetical order, the program compares the byte codes of the current keyword character and the found tag character, if the tag character code is less than the keyword character code, we can be sure that there will be no further character tags in that level of tree that will match the current character of keyword, thus the failure function is being called, avoiding the further lookup actions that will not return any match. Also to ensure that the program will not miss any entry tag by performing this action, the entry tags are being placed before all character tags of each level of the tree.



Graph 1.

Results and Conclusion

As it was mentioned before there are other alternative ways of dictionary lookup to the string matching machines and those methods were also tested and compared to the implemented ACA results. One of the most robust search engines nowadays is the Apache Lucene: an open source project created in Java and currently being released under the Apache Software License. Currently, Lucene is considered of the leading tools for search and is being used as a basis in many powerful search engines. It uses similar search indexing and lookup approaches, thus was chosen for our comparison. *Table 2* illustrates the lookup results gained by using MySQL

database, Apache Lucine and our ACA implementation. Below are listed 5 randomly selected sentences that were used in comparison results (*Table 2*). These results were later confirmed by a bigger test corpus.

1. "The flexibility and opportunities that UNL gives are enormous, so we decided to create a project that will be a pioneering effort to invest on this technology into real world applications."

2. "The flexibility and opportunities that UNL gives are enormous."

3. "The flexibility and opportunities."

4. "Cyanogenmod is free of charge, but let's face it - it takes time and effort from Cyanogen to make it happen, time he could be using to work a salaried position, but instead is working on getting you the ROM you love, and doing it without asking anything in return."

5. "Depending on the current state of your handset, there are basically three different ways to upgrade to the latest CyanogenMod version."

Sentence No.	Matched entries	ACA lookup time	Lucine lookup time	MySQL lookup time
1	111	355 ms	1093 ms	2512 ms
2	55	154 ms	152 ms	215 ms
3	36	42 ms	50 ms	162 ms
4	120	131 ms	789 ms	7300 ms
5	86	90 ms	149 ms	2048 ms

Table 2.

Thus, we can say that the implemented search engine performance is robust and resolves the performance issues on standard desktop machines.

Bibliography.

[Uchida, Zhu, 2005] Uchida H., Zhu M., "The Universal Networking Language (UNL) specifications" version 7, UNDL Foundation, June 2005.

[Avetisyan, 2009] Avetisyan A., "Some approaches to the generation of sentences in natural language from UNL" Institute of Informatics and Automation Problems of NAS of RA, 2009.

[Aho, Corasick, 1977] Alfred V. Aho, Margaret J. Corasick, "Efficient String Matching: An Aid to Bibliographic Search" Communications of the ACM, 1977.

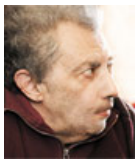
[Knuth, Morris, Pratt, 1977] D. Knuth, James H. Morris Jr, V. Pratt, "Fast pattern matching in strings", SIAM Journal on Computing, 1977.

[Spreen, Van Slyke, 2010] T. Spreen, A. Van Slyke, "Exact-Set Matching with the Aho-Corasick Automaton" University of Victoria, 2010.

[Boyer, Moore, 1977] R. S. Boyer, J. S. Moore, "A Fast String Searching Algorithm" Communications of the ACM, 1977.

[UNDL, 2007] UNDL Foundation, www.unlweb.net/wiki/, 2010

Authors' Information



Igor Zaslawskiy *– Head of lab., Chief sci. worker, Institute of Informatics and Automation Problems of NAS RA, 1, P. Sevak str., Yerevan, Armenia, 0014, e-mail: zaslav@ipia.sci.am,*

Major Fields of Scientific Research: Mathematical logic and the automated logic conclusion

Aram Avetisyan *– Junior sci. worker, Institute of Informatics and Automation Problems of NAS RA, 1, P. Sevak str., Yerevan, Armenia, 0014, e-mail: a.avetisyan@undlfoundation.org*

Major Fields of Scientific Research: Mathematical logic and the automated logic conclusion

Vardan Gevorgyan *– CEO, "VTGSoftware" Ltd., 52a, Sose str., Yerevan, Armenia 0019,*

e-mail: vgevorgyan@vtgsoftware.com

COMPARISON OF PROOF SIZES IN FREGE SYSTEMS AND SUBSTITUTION FREGE SYSTEMS

Anahit Chubaryan, Hakob Nalbandyan

***Abstract**: It is known that the minimal number of the steps in a proof of a tautology in a Frege system can be exponentially larger than in a substitution Frege system, but it is an open problem whether Frege systems can polynomially simulate substitution Frege systems by sizes. Many people conjecture that the answer is no. We prove that the answer is **yes**. As a bridge between substitution Frege systems and Frege systems we consider the Frege systems, augmented with restricted substitution (single renaming) rule. We prove that Frege systems with single renaming rule polynomially simulate by size Frege systems with substitution rule without any restrictions, and Frege systems without substitution rule polynomially simulate Frege systems with single renaming rule both by steps and by size.*

***Keywords**: Frege system, proof complexity, depth-restricted substitution rule, k-bounded substitution rule, polynomial simulation, exponential speed-up.*

***ACM Classification Keywords**: F.4.1 Mathematical Logic and Formal Languages, Mathematical Logic, Proof theory*

1. Introduction

It is well known that the investigations of the propositional proof complexity are very important due to their relation to the main problem of the complexity theory: $P \stackrel{?}{=} NP$.

One of the most fundamental problems of the proof complexity theory is to find an efficient proof system for propositional calculus. There is a wide spread understanding that polynomial time computability is the correct mathematical model of feasible computation. According to the opinion, a truly "effective" system must have a polynomial size, p(n) proof for every tautology of size n. In [Cook, Reckhow, 1979] Cook and Reckhow named such a system, a *super system*. They showed that if there exists a super system, then NP = coNP.

It is well known that many systems are not super. This question about Frege system, the most natural calculi for propositional logic, is still open. It is interesting how efficient can be Frege systems augmented with new, not sound rules, in particular, Frege systems with different modifications of substitution rules.

In the field of proof complexity the relation of Frege systems ($\mathcal{F}$-systems) to Frege systems with substitution (S$\mathcal{F}$-systems) has been discussed in [Cejtin, Chubaryan, 1975], [Krajicek, 1989], [Buss, 1995]. It has been proved that there exist tautologies which have n line substitution Frege proofs, but which require Frege proofs of 2^{cn} lines for some constant c. It has also been proved, that substitution Frege proofs of these tautologies can be transformed

into Frege proofs only with quadratic increase of size [Buss, 1995], [Chubaryan, 2000]. It is an open problem whether Frege systems can polynomially simulate substitution Frege systems [Buss, 1995].

In [Chubaryan, 2000] a special construction of substitution Frege proofs is described. By their transformation into Frege proofs the maximum (exponential) increase in the number of lines is obtained, although increase in size is at most polynomial (it seems, the latter size increase is maximal). The paper [Chubaryan, 2000] reveals also some important properties of substitution Frege proofs, which can be simulated by Frege systems.

In [Chubaryan et al. 2008], [Chubaryan, Nalbandyan, 2009] the substitution rules with two different restrictions are introduced:

a) if for any constant $k \geq 1$ we allow substitution instead of occurrences of no more than k different variables at a time (k – bounded substitution)

b) if for any constant $d \geq 0$ we allow substitution of formulas, depth of which is no more than d (d – depth restricted substitution).

For every type restriction in [Chubaryan et al. 2008] and [Chubaryan, Nalbandyan, 2009] it is proved that:

1) the minimal numbers of steps (the minimal sizes) of the proofs of tautology in any two restricted substitution Frege systems are polynomially related

2) the minimal sizes of the proofs of tautology in without restrictions substitution Frege system and in restricted substitution Frege system are also polynomially related

3) the minimal number of steps of a tautology in restricted substitution Frege system can be exponentially larger than in the system with substitution rule without restrictions**.**

Here it is proved that:

4) the minimal number of steps of a tautology in Frege system without substitution rule can be exponentially larger than in Frege system with restricted substitution rule

The question about the increase of sizes by transformation in the case 4) was also open.

Here we consider the substitution rule with double restriction: 1 – bounded (single) and 0 – depth (renaming).

We prove that Frege systems with such double restricted substitution rule and Frege systems without substitution rule are polynomially equivalent both by steps and by size.

2. Preliminary

We shall use generally accepted concepts of Frege system and Frege system with substitution.

A Frege system $\mathcal{F}$ uses a denumerable set of propositional variables, a finite, complete set of propositional connectives; $\mathcal{F}$ has a finite set of inference rules defined by a *figure* of the form $\frac{A_1 A_2 \ldots A_m}{B}$ (the rules of inference with zero hypotheses are the axioms schemes); $\mathcal{F}$ must be sound and complete, i.e. for each rule of inference $\frac{A_1 A_2 \ldots A_m}{B}$ every truth-value assignment, satisfying A_1, A_2, ..., A_m, also satisfies B, and $\mathcal{F}$ must prove every tautology.

A substitution Frege system S$\mathcal{F}$ consists of a Frege system $\mathcal{F}$ augmanted with the substitution rule with inferences of the form $\frac{A}{A\sigma}$ for any substitution σ = $\begin{pmatrix} \varphi_{i_1} & \varphi_{i_2} \dots \varphi_{i_s} \\ p_{i_1} & p_{i_2} \dots p_{i_s} \end{pmatrix}$, s≥ 1, consisting of a mapping from propositional variables to propositional formulas, and $A\sigma$ denotes the result of applying the substitution to formula A, which replaces each variable in A with its image under σ. This definition of substitution rule allows to use the simultaneous substitution of multiple formulas for multiple variables of A without any restrictions. The substitution rule is not sound.

If the depths of formulas φ_{i_j} (1 ≤ j ≤ s) are restricted by some fixed d (d ≥ 0), then we have *d-restricted substitution rule* and we denote the corresponding system by S$^d\mathcal{F}$. 0-restricted substitution rule is named renaming rule.

If for any constant k≥ 1 we allow substitution instead of occurrences of no more than k different variables at a time, then we have *k – bounded substitution rule*. The k – bounded substitution Frege system S$_k\mathcal{F}$ consists of a Frege system augmented with the k – bounded substitution rule.

We use also the well-known notions of proof, proof complexities and p – simulation given in [1]. The proof in any system Φ (Φ-proof) is a finite sequence of such formulas, each being an axiom of Φ, or is inferred from earlier formulas by one of the rules of Φ.

The total number of symbols, appearing in a formula φ, we call size of φ and denote by | φ |.

We define ℓ – complexity to be the size of a proof (= the total number of symbols) and *t* – complexity to be its length (= the total number of lines).

The minimal ℓ – complexity (*t* – complexity) of a formula φ in a proof system Φ we denote by $l_\varphi^\Phi(t_\varphi^\Phi)$.

Let Φ_1 and Φ_2 be two different proof systems.

Definition 1. The system Φ_2 p-ℓ-simulates Φ_1 ($\Phi_1 \prec_\ell \Phi_2$), if there exists a polynomial p( ) such that for each formula φ, provable both in Φ_1 and Φ_2, we have $l_\varphi^{\Phi_2} \le p(l_\varphi^{\Phi_1})$.

Definition 2. The system Φ_1 is p-ℓ-equivalent to system Φ_2 ($\Phi_1 \sim_\ell \Phi_2$), if Φ_1 and Φ_2 p-ℓ-simulate each other.

Similarly p-*t*-simulation and p-*t*-equivalence are defined for *t* – complexity.

Definition 3. The system Φ_2 has exponential ℓ-speed-up (*t*-speed-up) over the system Φ_1, if there exists a sequence of such formulas φ_n, provable both in Φ_1 and Φ_2, that $l_{\varphi_n}^{\Phi_1} > 2^{\theta(l_{\varphi_n}^{\Phi_2})}$ $\left(t_{\varphi_n}^{\Phi_1} > 2^{\theta(t_{\varphi_n}^{\Phi_2})}\right)$.

In this paper we compare under the p-*t*-simulation relation S$^0\mathcal{F}$ and $\mathcal{F}$, S$_1\mathcal{F}$ and $\mathcal{F}$, and under p-*t* (p-ℓ)-simulation $S_1^0\,\mathcal{F}$ and $\mathcal{F}$.

For proving the main results we use also the notion of *essential subformulas*, introduced in [Chubaryan et al. 2008], and the notion of τ – set of subformulas, introduced in [Cejtin, Chubaryan, 1975].

Let F be some formula and Sf(F) is the set of all non-elementary subformulas of formula F.

For every formula F, for every φ ∈ Sf(F) and for every variable p $(\mathrm{F})_{\varphi}^{p}$ denotes the result of the replacement of the subformulas φ everywhere in F with the variable p. If φ ∉ Sf(F), then $(\mathrm{F})_{\varphi}^{p}$ is F.

We denote by Var(F) the set of variables in F.

Definition 4. Let p be some variable that p ∉ Var(F) and φ ∈ Sf(F) for some tautology F. We say that φ is an essential subformula in F iff $(\mathrm{F})_{\varphi}^{p}$ is non-tautology.

We denote by Essf(F) the set of essential subformulas in F.

If F is minimal tautology, i.e. F is not a substitution of a shorter tautology, then Essf(F) = Sf(F).

The formula φ is called determinative for the $\mathcal{F}$-rule $\frac{A_1,A_2,\ldots,A_m}{B}$ (m ≥ 1) if φ is essential subformula in formula A_1 & (A_2 & … & (A_{m-1} & A_m) …) ⊃ B. By the Dsf(A_1, …, A_m, B) the set of all determinative formulas for rule $\frac{A_1,A_2,\ldots,A_m}{B}$ is denoted.

We say that the formula φ is important for some $\mathcal{F}$-proof (S$\mathcal{F}$ -proof) if φ is essential in some axiom of this proof or φ is determinative for some $\mathcal{F}$-rule.

In [Chubaryan et al. 2008] the following statement is proved.

Proposition 1. Let F be a minimal tautology and φ ∈ Essf(F), then in every S$\mathcal{F}$-proof of F, in which the employed substitution rules are

$$\frac{A_1}{A_1\sigma_1}; \frac{A_2}{A_2\sigma_2}; \ldots; \frac{A_r}{A_r\sigma_r}$$

either φ must be important for this proof or it must be the result of the successive employment of the substitutions $\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_s}$ for $1 \le i_1, i_2, \ldots, i_s \le r$ in any important formula.

τ – set of subformulas for some formula F with the logical connectives &, ∨, ⊃ and ¬ is defined as follows:

τ(F) = {F} ∪ τ_1(F), where

τ_1(F) = Ø, if F is propositional variable

τ_1(F_1 & F_2) = τ(F_1) ∪ τ(F_2), if F = F_1 & F_2

τ_1(F_1 ∨ F_2) = τ(F_1) ∩ τ(F_2), if F = F_1 ∨ F_2

τ_1(F_1 ⊃ F_2) = τ(F_2) \ τ(F_1), if F = F_1 ⊃ F_2

τ_1(¬F_1) = $\overline{\tau(\mathrm{F}_1)}$, if F = ¬$F_1$

In [Nalbandyan, 2010] the following 3 auxiliary statements are proved.

1. For every minimal tautology F τ(F) ⊆ Essf(F).
2. For every formula F if subformula φ ∈ τ(F), then every occurrence of φ in F is positive.

The notions of *positive (negative)* occurrence of some subformula in the formula are well known (see, for example [Buss, 1995]), as well as *0-1 numeration* of subformulas in formula.

3. If formula F has only the connectives ⊃ and ¬, then the number of every subformula from the set τ(F) is in the form $(\underbrace{11\ \ldots\ 1}_{n})$ for the corresponding n.

The main results, connected with comparison of different substitution rule modifications, are the followings.

Theorem 1.

1. given arbitrary $k_1 \geq 1$ and $k_2 \geq 1$ $S_{k_1}\mathcal{F} \sim_{\ell} S_{k_2}\mathcal{F}$ $S_{k_1}\mathcal{F} \sim_{t} S_{k_2}\mathcal{F}$
2. given arbitrary $k \geq 1$ $S_k\mathcal{F} \sim_{\ell} S\mathcal{F}$
3. given arbitrary $k \geq 1$ $S\mathcal{F}$ has exponential t -speed-up over the system $S_k\mathcal{F}$
4. given arbitrary $k \geq 1$ $S_k\mathcal{F}$ has exponential t -speed-up over the system $\mathcal{F}$.

Theorem 2.

1. given arbitrary $d_1 \geq 1$ and $d_2 \geq 1$ $S^{d_1}\mathcal{F} \sim_{\ell} S^{d_2}\mathcal{F}$ $S^{d_1}\mathcal{F} \sim_{t} S^{d_2}\mathcal{F}$
2. given arbitrary $d \geq 0$ $S^d\mathcal{F} \sim_{\ell} S\mathcal{F}$
3. given arbitrary $d \geq 0$ $S\mathcal{F}$ has exponential t -speed-up over the system $S^d\mathcal{F}$
4. given arbitrary $d \geq 0$ $S^d\mathcal{F}$ has exponential t -speed-up over the system $\mathcal{F}$.

The proofs of the points 1., 2., 3. for both theorems are given in [Chubaryan et al. 2008] and [Chubaryan, Nalbandyan, 2009], [Chubaryan et al. 2009] accordingly. Note that the proofs of the points 1. and 2. are based on the result of Buss [Buss, 1995], who proved that renaming Frege systems p-ℓ-simulate Frege systems with substitution without any restrictions.

The proof of point 4. for k = 1 from Theorem 1., using the formulas

$\varphi_n = p_1 \supset (\overbrace{p_2 \supset (p_2 \supset \ldots \supset (p_2}^{n} \supset p_1) \ldots)$, is given in [Cejtin, Chubaryan, 1975], where only single substitution rule is considered, therefore the proof for every k ≥ 1 follows from point 1.

To prove the statement of point 4. from Theorem 2. we show that for the formulas

$$\Psi_n = (p_1 \supset p_1) \,\&\, (p_2 \supset p_2) \,\&\, (p_3 \supset p_3) \,\&\, \ldots \,\&\, (p_n \supset p_n)$$

are true the following results

$$t_{\Psi_n}^{S^0\mathcal{F}} = O(\log_2 n) \text{ and } t_{\Psi_n}^{\mathcal{F}} = \Omega(n).$$

Really, the formula Ψ_n can be derived in $S^0\mathcal{F}$ as follows:

1. $p_1 \supset p_1$
2. $p_2 \supset p_2$ (renaming $\binom{p_2}{p_1}$)

3. $(p_1 \supset p_1)$ & $(p_2 \supset p_2)$ $(\frac{A,B}{A \,\&\, B}$ rule)
4. $(p_3 \supset p_3)$ & $(p_4 \supset p_4)$ (renaming $\binom{p_3 \quad p_4}{p_1 \quad p_2}$)
5. $((p_1 \supset p_1)$ & $(p_2 \supset p_2))$ & $((p_3 \supset p_3)$ & $(p_4 \supset p_4))$ $(\frac{A,B}{A \,\&\, B}$ rule)

⋮

2k + 1. $(p_1 \supset p_1)$ & ..... & $(p_{2^k} \supset p_{2^k})$ $(\frac{A,B}{A \,\&\, B}$ rule)

2k + 2. $(p_{2^k+1} \supset p_{2^k+1})$ & ..... & $(p_{2^{k+1}} \supset p_{2^{k+1}})$ (renaming $\binom{p_{2^k+i}}{p_i}$)

2(k+1) + 1. $(p_1 \supset p_1)$ & ..... & $(p_{2^{k+1}} \supset p_{2^{k+1}})$ $(\frac{A,B}{A \,\&\, B}$ rule)

On the other hand τ – set of Ψ_{2^k} has 2^{k+1} formulas, and therefore from above auxiliary statements 1. and 2., follows that the number of steps in $\mathcal{F}$ - proof of Ψ_{2^k} must be no less than $c \cdot 2^k$ for some c, depending only from choice of system $\mathcal{F}$.

Note that if we compare the sizes of $\mathcal{F}$ - proof and $S_1\mathcal{F}$ - proof for φ_n and the sizes of $\mathcal{F}$ - proof and $S^0\mathcal{F}$ - proof for Ψ_n, then we obtain only polynomial increase.

3. Main result

Here we will consider the Frege system augmented with double restrictions substitution rule(single renaming). According to above notations, such system must be denoted by $S_1^0\mathcal{F}$.

In [Cook, Reckhow, 1979] it is proved that every two Frege systems are polynomially equivalent both by size and by length, therefore without loss of generality we assume that $\mathcal{F}$ is a Frege system, whose language contains only the connectives ⊃ and ¬.

The axiom-schemas are:

1. $A \supset (B \supset A)$
2. $(A \supset B) \supset ((A \supset (B \supset C)) \supset (A \supset C))$
3. $(\neg A \supset B) \supset ((\neg A \supset \neg B) \supset A)$

and inference rule is Modus ponens.

The main result of the paper is the following statement.

Main Theorem.

$$S\mathcal{F} \sim_\ell \mathcal{F}$$

First we will prove that $S_1^0\mathcal{F} \sim_\ell \mathcal{F}$ and obtain the statement of the Main Theorem as a corollary.

Let us recall some notions in addition.

Note that every Φ-proof has an associated graph with nodes, labeled by formulas, and edges from A to B if formula B is the result of applying of some inference rule to A (perhaps with another formulas). A proof is said to be tree-like if the associated graph is tree. It is obvious, that any derivation can be made tree-like by merely repeating parts of the derivation if need be. It is also obvious that by such natural transformation we have tree-like proof, the steps of which can be much more than the steps in original proof-sequence, nevertheless in [Krajicek, 1994] a transformation of proof-sequence into tree-like proof is suggested such that the following statement is hold:

for every Frege system there exists a polynomial p() such that for every tautology φ if n is the steps of its proof in the sequence form, then steps of its tree-like proof is no more than p(n).

Without making some important corrections in the proof of this statement, we prove

Lemma 1. There exists a polynomial p() such that for every tautology φ if n is the number of steps of its $S_1^0\mathcal{F}$-proof in the sequence form, then the number of steps of its tree-like $S_1^0\mathcal{F}$-proof is no more than p(n).

In [Cejtin, Chubaryan, 1995] a natural method of transformation of a given S$\mathcal{F}$-proof into $\mathcal{F}$-proof is described. This method is following: let some formula ψ of S$\mathcal{F}$-proof be inferred from φ by substitution rule, i.e. there is a substitution σ such that ψ = φσ. To prove the formula ψ in $\mathcal{F}$ we have to repeat the proof of φ, applying the substitution σ to all formulas of this proof.

As the sequence of successive substitution is closed under composition, then described transformation method must be applied in the case when both formulas φ and φσ are used for the inference of some next formulas in the given S$\mathcal{F}$-proof, and therefore, as it is pointed in introduction, the number of steps of $\mathcal{F}$-proof can be much more than the number of steps of S$\mathcal{F}$-proof, but if S$\mathcal{F}$-proof is in tree-like form, then the number of formulas in corresponding $\mathcal{F}$-proof is no more than in S$\mathcal{F}$-proof, so we obtain the following statements.

Lemma 2. $S_1^0\mathcal{F} \sim_t \mathcal{F}$

Now we must compare the size of the proofs for arbitrary formula in $S_1^0\mathcal{F}$ and in $\mathcal{F}$.

Let us recall the notion of right-chopping proof, introduced in [Nurijanyan, 1981]. For Intuitionistic and Minimal (Johansson's) Logic there is proved the following statements:

If the axiom $F_1 \supset (F_2 \supset (\dots \supset (F_m \supset G) \dots)$ and the formulas $F_1, F_2, \dots, F_m$ are used from the minimal (by steps) derivation of formula G by the successive applying of the rule modus ponens, then $m \leq 2$, i.e. the length of branch, going to right and upwards from every node of the corresponding graph, is no more than 2. Such graph and hence the corresponding proof are called right-chopping.

The analogous statement for classical Hilbert style systems is not valid, but using 1) the method of Nurijanyan, 2) proved in [Cejtin, Chubaryan, 1975] fact that every formula from τ – set of arbitrary derivable formula must be an element from τ – set at least of one axiom from this derivation, and 3) that the τ – sets of our axioms are the following:

1. $\tau(A \supset (B \supset A)) = \{A \supset (B \supset A), B \supset A\}$
2. $\tau((A \supset B) \supset ((A \supset (B \supset C)) \supset (A \supset C))) = \{(A \supset B) \supset ((A \supset (B \supset C)) \supset (A \supset C)),$
   $(A \supset (B \supset C)) \supset (A \supset C), A \supset C\}$
3. $\tau((\neg A \supset B) \supset ((\neg A \supset \neg B) \supset A)) = \{(\neg A \supset B) \supset ((\neg A \supset \neg B) \supset A), (\neg A \supset \neg B) \supset A)\},$

we obtain the following statements.

Lemma 3. Every $\mathcal{F}$-proof of a formula φ can be transformed into right-chopping proof of φ, the t-complexity of which is no more than t-complexity of original proof.

Note that the depth of occurrence of each formula from τ – set of any above axioms is no more than 2.

In [Buss, 1995] it is showed that a Frege proof for a formula can be transformed into new one, where the symbol-size is related to line-size and the depth of the original proof. Recall that the depth of the proof is the maximum and/or depth of a formula ψ, occurring in the proof.

More precise result is the following:

a depth d Frege proof with m lines can be transformed into a depth d Frege proof with $O(m^d)$ symbols.

Using 1) the main idea of the proof of this result, 2) above remark about depth of formulas from τ – set of axioms, 3) the possibility of evaluating of the sizes for interpolants and two contrary formulas, deduced from counter-factural hypothesis in right-chopping proof, we obtain the following statement.

Lemma 4. If t is the number of steps in right-chopping $\mathcal{F}$-proof of tautology φ, then the size of this proof is no more than $t^3 \cdot |\varphi|$.

Now we can proof the following

Main Lemma. $S_1^0\mathcal{F} \sim_\ell \mathcal{F}$

Really, let we have some $S_1^0\mathcal{F}$ proof of arbitrary tautology φ with the size L. It is obvious that $|\varphi| < L$ and t-complexity of this proof also is < L. At first we transform this proof into tree-like proof, then into $\mathcal{F}$-proof, and finally into right-chopping $\mathcal{F}$-proof.

Using the statements of above Lemmas, we can state that there is a polynomial p(), depending only on the choice of Frege system such that $\ell_\varphi^{\mathcal{F}} = O(p(L))$.

So $\mathcal{F}$ p-ℓ-simulates $S_1^0\mathcal{F}$. The reverse p-ℓ-simulation is obvious.

Now the proof of Main Theorem follows from the results of Theorem 1, Theorem 2 and Main Lemma.

Bibliography

[Cook, Reckhow, 1979] S.A.Cook, A.R.Reckhow, "The relative efficiency of propositional proof systems", Journal of Symbolic Logic, vol. 44, pp. 36-50, 1979.

[Cejtin, Chubaryan, 1975] G.Cejtin, A.Chubaryan, "On some bounds to the lengths of logical proofs in classical propositional calculus" (in Russian), Trudy Vycisl.Centra AN Arm SSR i Yerevan Univ. 8, pp. 57-64, 1975.

[Krajicek, 1989] J.Krajicek, "Speed-up for propositional Frege systems via generalizations of proofs", Commentationes Mathematicae Univ. Carolinae, vol. 30, pp. 137-140, 1989.

[Krajicek, 1994] J.Krajicek, "Lower bounds to the size of constant-depth propositional proofs", Journal of Symbolic Logic, vol. 59, pp. 73-86, 1994.

[Buss, 1995] S.R.Buss, "Some remarks on lengths of propositional proofs", Arch. Math. Logic, vol. 34, pp. 377-394, 1995.

[Chubaryan, 2000] A.A.Chubaryan, "Comparison of proof sizes in systems and substitution systems of Frege", Izvestiya NAN Armenii, Matematika, vol. 35, No. 5, pp. 21-29, 2000 and Journal of CMA (AAS), vol. 35, No. 5, 2000.

[Chubaryan et al. 2008] An.Chubaryan, Arm.Chubaryan, S.Aleksanyan, "Comparison of the complexities in Frege proofs with Different Substitution Rules", Mathematical Problems of Computer Science, Yerevan, vol. 30, pp. 36-39, 2008.

[Chubaryan, Nalbandyan, 2009] A.Chubaryan, H.Nalbandyan, "Comparison of efficiency of Frege systems with different modification of substitution rule" (in Russian), DNAN, RA Prikladnaja matematika, vol. 109, No. 3, pp. 208-213, 2009.

[Chubaryan et al. 2010] An.Chubaryan, Arm.Chubaryan, H.Nalbandyan, "On the sizes in Frege proofs and substitution Frege proofs", ASL ESM Logic Colloquium 2010, Paris, p. 7.

[Nalbandyan, 2010] H.Nalbandyan, "Efficiency of depth-restricted substitution rules", Mathematical Problems of Computer Science, vol. 33, pp. 5-10, 2010.

[Chubaryan et al. 2009] An.Chubaryan, Arm.Chubaryan, H.Nalbandyan, "Comparison of Frege systems with restricted substitution rules", CSIT, Yerevan, pp. 31-32, 2009.

[Nurijanyan, 1981] A.Nurijanyan, "On the some condition of proofs in Intuitionistic and Minimal propositional calculi", Sbornik "Molodoy nauchnij sotrudnic", YGU, Yerevan, vol. 2(34), pp. 42-50, 1981.

Author's Information

Anahit Chubaryan *– Doctor of sciences, full professor of the Department of Informatics and Applied Mathematics, Yerevan State University, 1. A.Manoogyan, 0025 - Yerevan, Armenia; e-mail:* achubaryan@ysu.am*.*

Major Fields of Scientific Research: Mathematical Logic, Proof Theory, Proof complexity.

Hakob Nalbandyan *– PhD Student of the Institute for Informatics and Automation Problems, National Academy of Sciences of Armenia, 1. P.Sevak, 0014 - Yerevan, Armenia; e-mail:* hakob_nalbandyan@yahoo.com*.*

Major Fields of Scientific Research: Mathematical Logic, Proof Theory, Proof complexity.

SOME PROPERTIES IN MULTIDIMENSIONAL MULTIVALUED DISCRETE TORUS

Vilik Karakhanyan

Abstract: *Current research concerns the following issues: n-dimensional discrete torus generated by cycles of even length is considered; the concept of standard arrangement in the torus is defined and some basic properties of this arrangement are investigated. The issues considered are similar to discrete isoperimetry constructions, being related to concept of neighbourhood in terms of linear arrangements of vertices. Considered are the basic properties of solving the discrete isoperimetry problem on torus.*

Keywords*: discrete torus, standard arrangement.*

ACM Classification Keywords*: G.2.1 Discrete mathematics: Combinatorics*

Introduction

Isoperimetric problems appeared in the variational calculus of Euler and Lagrange as a classic mathematical issue. The similar considerations in discrete spaces are completely different in research technologies which can be easily seen by early works on isoperimetry [1-6]. The difference and the problem novelty appear on the boundary of the subset considered to be isoperimetric. First results were delivered in terms of linearization of domain elements. After this studies appeared example problems that exempt this property. And appeared one more model, - with cylindrical coordinates like the torus [5-6]. In which extend the rules of [1, 3] are extendable to this domain? This is the main topic of study of current investigation.

Basic Definitions

For any integers $1 \le k_1 \le k_2 \le \cdots \le k_n < \infty$ the multivalued n-dimensional torus $T^n_{k_1 k_2 \cdots k_n}$ has been defined as the set of vertices: $T^n_{k_1 k_2 \cdots k_n} = \{(x_1, x_2, \cdots, x_n) / \ -k_i + 1 \le x_i \le k_i,\ x_i \in Z, 1 \le i \le n\}$, where two vertices $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ of $T^n_{k_1 k_2 \cdots k_n}$ are considered neighbours, if they differ by exactly one coordinate for which either $| x_i - y_i | = 1$; or the values equal $-k_i + 1$ and k_i respectively. The sum and difference of these vectors has been defined in the following way: $z = x \pm y = (x_1 \pm y_1, x_2 \pm y_2, \cdots, x_n \pm y_n) = (z_1, z_2, \cdots, z_n)$, where $-k_i + 1 \le z_i \le k_i$ and $z_i \equiv (x_i \pm y_i) (\mathrm{mod}\ 2k_i)$.

We will consider discrete isoperimetric problem for the torus. First let us define the concept of interior and boundary vertices for subsets of $T^n_{k_1k_2\cdots k_n}$.

Definition 1 For a given subset $A \subseteq T^n_{k_1k_2\cdots k_n}$ we say that a vertex $x \in A$ is an interior point of A, if all its neighbouring vertices belong to A. Otherwise $x \in A$ is called a boundary vertex of A. We denote by $B(A)$ and $\Gamma(A)$, respectively, the subset of all interior and boundary points of A.

Discrete isoperimetric problem

Given an integer a, $0 \le a \le |T^n_{k_1k_2\cdots k_n}|$. Determine a subset $A \subseteq T^n_{k_1k_2\cdots k_n}$, $|A| = a$, that have the largest number of interior points among all subsets of size a:

$$|B(A)| = \max_{\substack{A' \subseteq T_{k_1k_2\cdots k_n} \\ |A'|=a}} |B(A')|.$$

Sets, being the solution of the discrete isoperimetric problem, we call optimal.

In case when $k_1 = k_2 = \cdots = k_n = 1$, $T^n_{k_1k_2\cdots k_n}$ becomes n-dimensional unit cube E^n; the solution of the discrete isoperimetric problem in E^n is given in [1-4]. In case of $k_1 = k_2 = \cdots = k_n = \infty$, the solution is given in [5]. Notice that in [4,5] the subset of boundary vertices of $A \subseteq T^n_{k_1k_2\cdots k_n}$ is defined as the set of vertices from $T^n_{k_1k_2\cdots k_n} \setminus A$, that have at least one neighbouring vertex from A.

We denote by $||x||$ the norm of a vertex $x = (x_1, x_2, \ldots, x_n)$ where $||x|| = \sum_{i=1}^{n} |x_i|$, and denote by $\rho(x,y)$ the distance between the vertices $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ where $\rho(x,y) = ||x - y||$.

Now we define the concepts of sphere and envelope with a given centre and radius.

Definition 2 The set $S^n(x,k) = \{ y \in T^n_{k_1k_2\cdots k_n} / \rho(x,y) \le k \}$ is called a sphere with the centre $x \in T^n_{k_1k_2\cdots k_n}$ and radius k, and the set $O^n(x,k) = \{ y \in T^n_{k_1k_2\cdots k_n} / \rho(x,y) = k \}$ is the envelope with centre x and radius k.

Let $e_i = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ denote the unit vector of i-th direction, where $\alpha_i = 1$ and $\alpha_j = 0$ for $j \ne i$, and let $\tilde{1}$ and $\tilde{0}$ be the vectors with all 1 and all 0 coordinates respectively: $\tilde{1} = (1,1,\ldots,1)$ and $\tilde{0} = (0,0,\ldots,0)$.

For any subset $A \subseteq T^n_{k_1k_2\cdots k_n}$ and any $i, 1 \le i \le n$ and $j, -k_i + 1 \le j \le k_i$ we make the following designation:

$$A + je_i = \{ x + je_i / x \in A \}.$$

We will consider partition of $T^n_{k_1k_2\cdots k_n}$ (respectively partition of $A \subseteq T^n_{k_1k_2\cdots k_n}$) on i-th direction, $1 \le i \le n$ and j-th value, $-k_i + 1 \le j \le k_i$ and will denote by $T^n_i(j)$ (respectively by $A_i(j)$):

$$T_i^n(j) = \{x = (x_1, x_2, \ldots, x_n) \in T^n_{k_1 k_2 \cdots k_n} \ / \ x_i = j\},$$

$$A_i(j) = \{x = (x_1, x_2, \ldots, x_n) \in A \ / \ x_i = j\} = A \cap T_i^n(j).$$

Notice that the intersections of the sphere $S^n(x,k)$ and the envelope $O^n(x,k)$ with the (n-1)-dimensional torus $T_i^n(x_i + j)$, are respectively the sphere and envelope with the centre $x + je_i$ and radius $k - |j|$ in $T_i^n(x_i + j)$. We make the following designations:

$$S_i^n(x + je_i, k - |j|) = \{y \in S^n(x,k) \ / \ y_i = x_i + j\} = S^n(x,k) \cap T_i^n(x_i + j);$$

$$O_i^n(x + je_i, k - |j|) = \{y \in O^n(x,k) \ / \ y_i = x_i + j\} = O^n(x,k) \cap T_i^n(x_i + j),$$

where in case of $k - |j| < 0$ these sets are empty: $S_i^n(x + je_i, k - |j|) = O_i^n(x + je_i, k - |j|) = \varnothing$.

It is clear that $T^n_{k_1 k_2 \cdots k_n} = \bigcup_{j=-k_i+1}^{k_i} T_i^n(j)$, $A = \bigcup_{j=-k_i+1}^{k_i} A_i(j)$, $S^n(x,k) = \bigcup_{j=-k_i+1}^{k_i} S_i^n(x + je_i, k - |j|)$,

$$O^n(x,k) = \bigcup_{j=-k_i+1}^{k_i} O_i^n(x + je_i, k - |j|),$$

for each i, $1 \le i \le n$.

For each $A_i(j)$ in the partition of $A = \bigcup_{j=-k_i+1}^{k_i} A_i(j)$ we denote by $B(A_i(j))$ and $\Gamma(A_i(j))$, respectively, the subsets of its interior and boundary vertices in $(n-1)$ -dimensional torus $T_i^n(j)$.

For any vertex $x = (x_1, x_2, \ldots, x_n)$ of $T^n_{k_1 k_2 \cdots k_n}$, we denote by $|x|$ and $\delta(x)$ the vectors $|x| = (|x_1|, |x_2|, \ldots, |x_n|)$ and $\delta(x) = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, where $\alpha_i = 1$ for $x_{n-i+1} > 0$ and $\alpha_i = 0$ for $x_{n-i+1} \le 0$.

In general, for n-dimensional vectors $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ with nonnegative integer coordinates, we say that the vector x lexicographically precedes y (written by $x \prec y$), if there is a number r, $1 \le r \le n$, such that $x_i = y_i$ for $1 \le i < r$ and $x_r < y_r$.

Now we order the vertices of the torus $T^n_{k_1 k_2 \cdots k_n}$ as follows:

vertex x precedes vertex y written by $x \Leftarrow y$), if and only if

1. $\|x\| < \|y\|$, or
2. $\|x\| = \|y\|$ and $\delta(y)$ lexicographically precedes $\delta(x)$, or
3. $\|x\| = \|y\|$, $\delta(x) = \delta(y)$ and $|y|$ lexicographically precedes $|x|$.

It is easy to check that this ordering between the vertices of the torus $T^n_{k_1k_2\cdots k_n}$ is a linear order.

The first a vertices of the torus $T^n_{k_1k_2\cdots k_n}$ by the above determined liner order we call standard arrangement of cardinality a, $0 \le a \le |T^n_{k_1k_2\cdots k_n}|$.

Basic properties of the standard arrangement

In this section we investigate the basic properties of the standard arrangement.

Theorem 1. If a set A is the standard arrangement in $T^n_{k_1k_2\cdots k_n}$, then its interior vertices precede the boundary vertices.

Proof. Prove that if $(x_1, x_2, \ldots, x_n) = x \in B(A)$ and $y \Leftarrow x$, then $(y_1, y_2, \ldots, y_n) = y \in B(A)$. It suffices to show that $y \pm e_i \in A$ for $i, 1 \le i \le n$. Clearly, $y + e_i \in A$ for $y_i < 0$ or $y_i = k_i$ and $y - e_j \in A$ for $y_j > 0$, such that $\| y + e_i \| < \| y \|$ and $\| y - e_j \| < \| y \|$, i.e. $y + e_i \Leftarrow y$ and $y - e_j \Leftarrow y$. On the other hand, it is clear that $y + e_i \Leftarrow y - e_i$, if $y_i = 0$. Consequently, to complete the proof, it suffices to show that $y \oplus e_i \in A$, for $y_i \ne k_i$, where $y \oplus e_i = \begin{cases} y + e_i, \text{ if } y_i > 0, \\ y - e_i, \text{ if } y_i \le 0 \end{cases}$. Starting from the definition of the linear ordering $\Leftarrow$, consider the following three cases:

I. $\| y \| < \| x \|$.

If $\| x \| = \sum_{i=1}^{n} k_i$, then $A = B(A) = T^n_{k_1k_2\cdots k_n}$, and then the vertex y is also interior. And if $\| x \| \ne \sum_{i=1}^{n} k_i$, then there exists i_0, that $x_{i_0} \ne k_{i_0}$. Then $\| x \oplus e_{i_0} \| > \| x \| \ge \| y \oplus e_i \|$ for each $i, 1 \le i \le n$, that is $y \oplus e_i \Leftarrow x \oplus e_{i_0}$. Hence $y \oplus e_i \in A$, for $1 \le i \le n$.

II. $\| y \| = \| x \|$ and $\delta(x) \prec \delta(y)$.

Let $\delta(x) = (\alpha_1, \alpha_2, \cdots, \alpha_{n-r}, 0, \cdots)$ and $\delta(y) = (\alpha_1, \alpha_2, \cdots, \alpha_{n-r}, 1, \cdots)$. If there is a number i_0 that

- $x_{i_0} > 0$ and $x_{i_0} \ne k_{i_0}$, or
- $x_{i_0} \le 0$ and $i_0 < r$, or
- $x_{i_0} \le 0$, $x_{i_0} \ne -k_{i_0} + 1$ and $i_0 \ge r$, then

for every $y_i \ne k_i$, the vertex $y \oplus e_i$ precedes the vertex $x \oplus e_{i_0}$, as $\| x \oplus e_{i_0} \| = \| y \oplus e_{i_0} \|$ and $\delta(x \oplus e_{i_0}) \prec \delta(y \oplus e_{i_0})$. Hence $y \oplus e_i \in A$, when $y_i \ne k_i$.

Otherwise, if $x_i = k_i$ for $1 \le i < r$, $x_r = -k_r + 1$ and $x_i = k_i$ or $x_i = -k_i + 1$ for $r < i \le n$ then from the condition $\| x \| = \| y \|$ we find:

$$\sum_{i=1}^{r-1}(k_i - |y_i|) + (k_r - 1 - y_r) + \sum_{i=r+1}^{n}(|x_i| - |y_i|) = 0 \qquad (1)$$

Since $k_i - |y_i| \ge 0$ for $1 \le i < r$, $k_r - 1 - y_r \ge -1$ and $|x_i| - |y_i| \ge 0$, for $r < i \le n$ (according to condition $\delta(x) \prec \delta(y)$), then it follows from (1) that the following cases take place:

a) $y_i = k_i$ for $1 \le i < r$, $y_r = k_r - 1$ and $x_i = y_i$ for $r < i \le n$; then it is clear, that for $i > r$ vertex $y \oplus e_i$ precedes vertex $x \oplus e_i$, and $y + e_r = x - e_r$, therefore $y \oplus e_i \in A$ for any $y_i \ne k_i$;

b) $y_i = k_i$ for $1 \le i \le r$, and there is a unique number $i_0 > r$ such, that $|x_{i_0}| = |y_{i_0}| + 1$ and $x_i = y_i$, for $r < i \le n, i \ne i_0$, then the vertex $y \oplus e_i$ is preceded by the vertex $x \oplus e_i$, for $i > r, i \ne i_0$, and $y \oplus e_{i_0} = x - e_r$, so again $y \oplus e_i \in A$, for any $y_i \ne k_i$;

c) there is a unique number $i_1 < r$ such that $|y_{i_1}| = k_{i_1} - 1, y_i = k_i$ for $r \le i \le n, i \ne i_1$, and $x_i = y_i$ for $r < i \le n$. In this case $y \oplus e_{i_1} = x - e_r$, and vertex $y \oplus e_i$ is preceded by the vertex $x \oplus e_i$ for $r < i \le n$. Hence $y \oplus e_i \in A$, when $y_i \ne k_i$.

III. $\| y \| = \| x \|$, $\delta(y) = \delta(x)$ and $|x| \prec |y|$.

Suppose that $x_i = y_i$ for $1 \le i < r \le n$, and $|x_r| < |y_r|$.

In this case if $y_i = -k_i + 1$, then $y - e_i \in A$, since $y - e_i \Leftarrow x \oplus e_r$. When $y_i \ne k_i$ and $y_i \ne -k_i + 1$, if

- $1 \le i \le r$, or
- $i > r$ and $|y_r| - |x_r| > 1$, or
- $i > r$, $|y_r| - |x_r| = 1$ and there is a number $i_0 > r$, that is $x_{i_0} \ne -k_{i_0} + 1$ *or* $x_{i_0} \ne k_{i_0}$,

then $y \oplus e_i \in A$, since in the first case, the vertex $y \oplus e_i$ precedes the vertex $x \oplus e_i$, in the second case – precedes the vertex $x \oplus e_r$, and the third – precedes the vertex $x \oplus e_{i_0}$.

Otherwise, if $|y_r| - |x_r| = 1$, $x_i = -k_i + 1$ *or* $x_i = k_i$ for $i > r$, then from the condition $\| x \| = \| y \|$ we find

$$\sum_{i=r+1}^{n}(|x_i| - |y_i|) = 1. \qquad (2)$$

Since $\delta(x)=\delta(y)$ then $|x_i|-|y_i|\geq 0$, for any $i>r$. Then (2) implies that there exists a unique number $i_0>r$, such that $|x_{i_0}|=|y_{i_0}|+1$, and for $i>r, i\neq i_0$, $x_i=y_i=k_i$ or $x_i=y_i=-k_i+1$. It is clear that $y\oplus e_{i_0}=x\oplus e_r$.

Thus, we proved that the vertex $y\pm e_i$ belongs to A, for any i, $1\leq i\leq n$, i.e. vertex y is an interior vertex of the set A. The theorem is proved.

Corollary 1. If A and C are standard arrangements in the $T^n_{k_1k_2\cdots k_n}$ and $|A|\geq|C|$, then $B(A)\supseteq B(C)$ and $A_i(j)\supseteq C_i(j)$ for any i and j, $1\leq i\leq n$, $-k_i+1\leq j\leq k_i$.

For a subset A of $T^n_{k_1k_2\cdots k_n}$, we denote by $O(A)=\{x\in T^n_{k_1k_2\cdots k_n} / \rho(x,y)\leq 1 \quad for\ some \quad y\in A\}$.

Then the following statement takes place:

Lemma 1. If A is a standard arrangement, then $O(A)$ is a standard arrangement too.

Proof. Let A is the standard arrangement. Suppose that $x\in O(A)$ and $y\Leftarrow x$. All we have to show is that y belongs to $O(A)$. In case when $x\in A$, the proof is obvious. Now assume that $x\notin A$. Then there exists a vertex $x^1\in A$ and a direction i_0, that $x=x^1+e_{i_0}$, where $0\leq x^1_{i_0}<k_{i_0}$, or $x=x^1-e_{i_0}$ where $x^1_{i_0}\leq 0$. It is clear, that in both cases, $\delta(x^1)=\delta(x)$ or $\delta(x^1)\prec\delta(x)$ and $O(A)\supseteq S^n(\tilde{0},\|x^1\|)$. Next we will find such a vertex that belongs to A and is located at distance one from the vertex y.

Since $y\Leftarrow x$ then, according to the definition of ordering $\Leftarrow$, the following three cases are possible:

Case I. $\|y\|<\|x\|$;

Case II. $\|y\|=\|x\|$ and $\delta(x)\prec\delta(y)$, where $\delta(x)=(\alpha_1,\alpha_2,\ldots,\alpha_{n-r},0,\ldots)$, $\delta(y)=(\alpha_1,\alpha_2,\ldots,\alpha_{n-r},1,\ldots)$;

Case III. $\|y\|=\|x\|$, $\delta(y)=\delta(x)$ and $|x|\prec|y|$, where $x_i=y_i$ for $1\leq i<r\leq n$ and $|x_r|<|y_r|$.

In the first case, it is obvious that $y\in S^n(\tilde{0},\|x^1\|)$, therefore y belongs $O(A)$ too.

In case II, if there is a number i_1 that $y_{i_1}\neq 0$, for $i_1<r$, or $y_{i_1}\neq 0$ and $y_{i_1}\neq 1$, when $i_1\geq r$, then vertex $y-e_{i_1}$ precedes x^1 in case of $y_{i_1}>0$, and in case when $y_{i_1}<0$, vertex $y+e_{i_1}$ precedes x^1 since

- $\|y-e_{i_1}\|=\|x^1\|$ and $\delta(x^1)\prec\delta(y-e_{i_1})$ when $y_{i_1}>0$;
- $\|y+e_{i_1}\|=\|x^1\|$ and $\delta(x^1)\prec\delta(y+e_{i_1})$ when $y_{i_1}<0$.

Thus, in case of $y_{i_1} > 0$, the vertex $y - e_{i_1} \in A$ and in case of $y_{i_1} < 0$ the vertex $y + e_{i_1}$ belongs to A, hence $y \in O(A)$. Otherwise, if $y_i = 0$, $1 \le i < r$, $y_r = 1$ and $y_i = 0$ or $y_i = 1$ for $r < i \le n$, then the conditions $\| y \| = \| x \|$ and $\delta(x) \prec \delta(y)$ imply the existence of a unique number i_2 such that the following conditions hold:

a) $i_2 \ne r$, $x_{i_2} = -1$, $x_r = 0$ *and* $x_i = y_i$ for any $i \ne i_2, r$, or

b) $i_2 = r$, $x_{i_2} = -1$ *and* $x_i = y_i$ for any $i \ne i_2$, or

c) $i_2 < r$, $x_{i_2} = 1$, $x_r = 0$ *and* $x_i = y_i$ for any $i \ne i_2, r$, or

d) $i_2 > r$, $x_{i_2} = 2$, $x_r = 0$ *and* $x_i = y_i$ for any $i \ne i_2, r$.

These conditions in their turn imply that if $i_0 \ne i_2$, then $y - e_{i_0} \Leftarrow x^1$, and if $i_0 = i_2$, then $y - e_r = x^1$. So the vertex $y - e_{i_0}$ or the vertex $y - e_r$ belongs to A, hence, $y \in O(A)$.

In case III, if $| y_r | - | x_r | > 1$, or $| y_r | - | x_r | = 1$ and there is such number $i_3 > r$ that $y_{i_3} \ne 0, 1$, then vertex x^1 is preceded by $y - e_{i_4}$ in case of $y_{i_4} > 0$, or is preceded by $y + e_{i_4}$ in case of $y_{i_1} < 0$, (where $i_4 = r$ for $| y_r | - | x_r | > 1$ and $i_4 = i_3$ for $| y_r | - | x_r | = 1$), as

$$\| y - e_{i_4} \| = \| x^1 \|, \ \delta(y - e_{i_4}) = \delta(x^1) \text{ and } | x^1 | \prec | y - e_{i_4} |, \text{ for } y_{i_4} > 0,$$

$$\| y + e_{i_4} \| = \| x^1 \|, \ \delta(y + e_{i_4}) = \delta(x^1) \text{ and } | x^1 | \prec | y + e_{i_4} |, \text{ for } y_{i_4} < 0.$$

Thus, either vertex $y - e_{i_4}$ or vertex $y + e_{i_4}$ belongs to A, and hence $y \in O(A)$.

Now consider the cases when $| y_r | - | x_r | = 1$ and $y_i = 0$ or $y_i = 1$, for any $i, r < i \le n$.

In both cases it follows from the conditions $\| y \| = \| x \|$ and $\delta(x) = \delta(y)$, that there is a unique number $i_5 > r$ such, that

- $x_{i_5} = -1$, $y_{i_5} = 0$ and $x_i = y_i$, for $i \ne i_5, r$, or
- $x_{i_5} = 2$, $y_{i_5} = 1$ and $x_i = y_i$, for $i \ne i_5, r$.

Now if $i_5 \ne i_0$, then either vertex $y - e_{i_0}$ (for $y_{i_0} > 0$) or vertex $y + e_{i_0}$ (for $y_{i_0} < 0$) precede the vertex x^1.

If $i_5 = i_0$, then vertex $y - e_r$ coincides with x^1 when $y_r > 0$, or vertex $y + e_r$ coincides with x^1 for $y_r < 0$. Therefore, again we get a vertex belonging to A and located at distance one from the vertex y. Hence $y \in O(A)$. Lemma is proved.

Now let us derive some properties of the standard arrangement in the torus.

Let A be a standard arrangement. Consider its partitions by i-th direction, $1 \le i \le n$: $A = \bigcup_{j=-k_i+1}^{k_i} A_i(j)$.

Further in this section we will prove the following properties:

1^0. each subset $A_i(j)$ is a standard arrangement of $T_i^n(j)$;

2^0. for any j, $1 \le j < k_n$, $A_n(-j) + e_n \subseteq B(A_n(-j+1))$;

3^0. for any j, $1 < j \le k_n$, $A_n(j) - e_n \subseteq B(A_n(j-1))$;

4^0. for any j, $0 \le j < k_n$, $A_n(-j) \supseteq A_n(j+1) - (2j+1)e_n \supseteq B(A_n(-j))$

or $A_n(-j) = T_n^n(-j)$ and $A_n(j+1) = T_n^n(j+1) \setminus \{(k_1, k_2, \cdots, k_{n-1}, j+1)\}$;

5^0. for any j, $0 < j < k_n$, $A_n(j) \supseteq A_n(-j) + 2je_n \supseteq B(A_n(j))$ or $A_n(j) = T_n^n(j)$ and

$A_n(-j) = T_n^n(-j) \setminus \{(k_1, k_2, \cdots, k_{n-1}, -j)\}$.

1^0. This property is obvious.

2^0. Let, $x = (x_1, x_2, \cdots, x_{n-1}, -j) \in A_n(-j)$, where $1 \le j < k_n$. Then obviously

- $\| x + e_n \| < \| x \|$,
- $\| x + e_n + e_i \| < \| x + e_n \|$, when $x_i < 0$ or $x_i = k_i$,
- $\| x + e_n - e_i \| < \| x + e_n \|$, when $x_i > 0$,
- $\| x + e_n - e_i \| = \| x \|$ and $\delta(x) \prec \delta(x + e_n - e_i)$ when $x_i = -k_i + 1$,
- $\| x + e_n \oplus e_i \| = \| x \|$, $\delta(x) \prec \delta(x + e_n \oplus e_i)$ and $|x| \prec |x + e_n \oplus e_i|$, when $x_i \ne -k_i + 1, k_i$,
- $\| x + e_n + e_i \| = \| x \|$ and $\delta(x) \prec \delta(x + e_n + e_i)$ when $x_i = 0$.

From these conditions follows that $x + e_n \Leftarrow x$ and $x + e_n \pm e_i \Leftarrow x$, for any $i, 1 \le i \le n-1$. As A is standard arrangement, then $x + e_n \in A$ and $x + e_n \pm e_i \in A$ for any $i, 1 \le i \le n-1$, i.e, $x + e_n$ is the interior vertex of the subset $A_n(-j+1)$.

3^0. Can be proved in similar way.

4^0. Let, $x = (x_1, x_2, \cdots, x_{n-1}, j+1) \in A_n(j+1)$, where $0 \le j < k_n$. Then $x - (2j+1)e_n \Leftarrow x$, as $\| x - (2j+1)e_n \| < \| x \|$. Therefore, the vertex $x - (2j+1)e_n$ belongs $A_n(-j)$, i.e., $A_n(-j) \supseteq A_n(j+1) - (2j+1)e_n$. On the other hand, if $y = (y_1, y_2, \cdots, y_{n-1}, -j) \in B(A_n(-j))$ and $A_n(-j) \ne T_n^n(-j)$, then there exist such number i_0, $1 \le i_0 \le n-1$, that $y_{i_0} \ne k_{i_0}$ and $y \oplus e_{i_0} \in A$. Then

vertex $y+(2j+1)\mathbf{e}_n$ precedes the vertex $y \oplus \mathbf{e}_{i_0}$, since $\| y+(2j+1)\mathbf{e}_n \| = \| y \oplus \mathbf{e}_{i_0} \|$ and $\delta(y \oplus \mathbf{e}_{i_0}) \prec \delta(y+(2j+1)\mathbf{e}_n)$. Therefore, the vertex $y+(2j+1)\mathbf{e}_n$ belongs A, that is, $A_n(j+1)-(2j+1)\mathbf{e}_n \supseteq B(A_n(-j))$, when $A_n(-j) \neq T_n^n(-j)$. And if $A_n(-j)=T_n^n(-j)$, then all the vertices $y=(y_1, y_{2,} \cdots, y_{n-1}, j+1)$ with norm less than or equal to $\sum_{i=1}^{n-1} k_i + j$ precede the vertex $y=(k_1, k_{2,} \cdots, k_{n-1}, -j)$, and only the vertex $y=(k_1, k_{2,} \cdots, k_{n-1}, j+1)$ from $T_n^n(j+1)$ might not belong to $A_n(j+1)$.

5^0. The proof is similar.

As a corollary from the above properties we get the following lemma.

Lemma 2. If A is standard arrangement and $|A| < |T^n_{k_1 k_2 \cdots k_n}| - 1$ then

$$|B(A)| = |A| - |A_n(0)| - |A_n(1)| + |B(A_n(-k_n+1))| + |B(A_n(k_n))|.$$

In general, a set $A \subseteq T^n_{k_1 k_2 \cdots k_n}$ possessing the above properties 1^0 – 5^0 of standard arrangement, might be itself non standard arrangement. However, we have

Theorem 2. Let the partition of some set $A \subseteq T^n_{k_1 k_2 \cdots k_n}$ satisfies the following conditions: $A_n(j+1) = O(A_n(j)+\mathbf{e}_n)$, when $j<0$, and $A_n(j) = O(A_n(j+1)-\mathbf{e}_n)$, for $j \geq 1$, then if there is a number j_1, $0 \leq j_1 \leq k_n - 1$, that either

a) $A_n(-j_1) = S_n^n(-j_1\mathbf{e}_n, r+1)$, $A_n(j_1+1) = S_n^n((j_1+1)\mathbf{e}_n, r) \cup S_1$, $S_1 \subseteq O_n^n((j_1+1)\mathbf{e}_n, r+1)$ and $A_n(j_1+1)$ is standard arrangement in $T_n^n(j_1+1)$, and $r=0$, when $j_1 < k_n - 1$, or

b) $A_n(-j_1) = S_n^n(-j_1\mathbf{e}_n, r) \cup S_0$, $A_n(j_1+1) = S_n^n((j_1+1)\mathbf{e}_n, r)$, $S_0 \subseteq O_n^n(-j_1\mathbf{e}_n, r+1)$ and $A_n(-j_1)$ is standard arrangement in $T_n^n(-j_1)$, and $r=0$, when $j_1 < k_n - 1$,

then A is standard arrangement in $T^n_{k_1 k_2 \cdots k_n}$.

Proof.

Consider case a). According to the conditions of the theorem and by Lemma 1, we have:

1. for any j, $-k_n+1 \leq j \leq k_n$, $A_n(j)$ is standard arrangement of $T_n^n(j)$, and $A_n(-j) = A_n(j) = \varnothing$, for $j > j_1+1$ *if* $j_1 < k_n - 1$;

2. $A = S^n(\tilde{0}, r+j_1+1) \cup A''$, where A'' is a set of vertices $z=(z_1, z_2, \cdots, z_n)$, for which $\| z \| = r+j_1+2$ and $z_n > 0$ (note that for $A''=\varnothing$ theorem is obvious, therefore further will assume that $A'' \neq \varnothing$);

3. for any $j \le 0$ the latest vertex of the subset $A_n(j)$ precedes the latest vertex of the subsets $A_n(j-1)$ in $T^n_{k_1k_2\cdots k_n}$;

4. for any $j \ge 1$ the latest vertex of the subset $A_n(j)$ precedes the latest vertex of the subset $A_n(j+1)$ in $T^n_{k_1k_2\cdots k_n}$.

Let $x = (x_1, x_2, \cdots, x_n)$ is the latest vertex of the set A (clearly, that $x \in A$" , i.e. $x_n > 0$), and $y = (y_1, y_2, \cdots, y_n) \in T^n_{k_1k_2\cdots k_n}$ is an arbitrary preceding x vertex in the $T^n_{k_1k_2\cdots k_n}$. We must show, that $y \in A$. First we notice that if $y_n = x_n$, then the vertex y precedes x in $T^n_n(x_n)$ too. Then also, y belongs to $A_n(x_n)$, so as $x \in A_n(x_n)$ and $A_n(x_n)$ is standard arrangement in $T^n_n(x_n)$. Therefore, further, we assume, that $y_n \ne x_n$. Since the vertex y precedes vertex x, then the following three cases are possible:

Case I. $|| y || < || x ||$;

In this case $|| y || \le r + j_1 + 1$, i.e. $y \in S^n(\tilde{0}, r + j_1 + 1)$, it means that $y \in A$.

Case II. $|| y || = || x ||$ and $\delta(x) \prec \delta(y)$.

Let $\delta(x) = (\alpha_1, \alpha_2, \ldots, \alpha_{n-i_0}, 0, \ldots)$ and $\delta(y) = (\alpha_1, \alpha_2, \ldots, \alpha_{n-i_0}, 1, \ldots)$. It is clear, that $y_n > 0$, as $x_n > 0$.

If $1 \le y_n < x_n$, then from the condition $|| y || = || x ||$ there exists such i_1 that $| x_{i_1} | - | y_{i_1} | < 0$, and if $i_1 \ne i_0$, then $|| x - e_n \oplus e_{i_1} || = || y ||$ and $\delta(x - e_n \oplus e_{i_1}) \prec \delta(y)$, and if $i_1 = i_0$ is a unique number for which $| x_{i_1} | - | y_{i_1} | < 0$, then or $| x_{i_1} | - | y_{i_1} | < -1$ (that means that $x_{i_1} \ne -k_{i_1} + 1$ so again $|| x - e_n \oplus e_{i_1} || = || y ||$ and $\delta(x - e_n \oplus e_{i_1}) \prec \delta(y)$), or $| x_{i_1} | - | y_{i_1} | = -1$ and then $x - e_n \oplus e_{i_1} = y$.

Thus we can always find a vertex $x^1 = x - e_n \oplus e_{i_1}$, that either, $y \Leftarrow x^1$ or $y = x^1$ and $x^1 \in A$, as, according to the conditions of the theorem, $x - e_n \in B(A_n(x_n - 1))$. Repeating this process again at $k = x_n - y_n$, we find a vertex $x^k = x^{k-1} - e_n \oplus e_{i_k}$ such that $y \Leftarrow x^k$ or $y = x^k$, and $x^k \in A_n(y_n)$.

Since $A_n(y_n)$ is standard arrangement, then $y \in A_n(y_n)$.

If $y_n > x_n \ge 1$ and if we assume that $y \notin A_n(y_n)$, then $x \Leftarrow z \Leftarrow y$ (according to property 4), where z is the latest vertex of the set $A_n(y_n)$, which contradicts the supposition $y \Leftarrow x$. Therefore $y \in A_n(y_n)$.

Case III. $\| y \| = \| x \|$, $\delta(y) = \delta(x)$ and $| x | \prec | y |$;

Suppose that $x_i = y_i$ for $1 \le i < r_0 \le n$, and $| x_{r_0} | < | y_{r_0} |$.

If $1 \le y_n < x_n$, then the condition $\| y \| = \| x \|$ and $\delta(y) = \delta(x)$ imply

- $| y_{r_0} | - | x_{r_0} | \ge 2$, or
- $| y_{r_0} | - | x_{r_0} | = 1$ and there exists such number r_1, $r_1 > r_0$ and $r_1 \ne n$, that $| y_{r_1} | - | x_{r_1} | > 0$, or
- $| y_{r_0} | - | x_{r_0} | = 1$, $x_n - y_n = 1$ and $x_i = y_i$, if $i \ne r_0, n$.

Then it is clear that in the first case $y \Leftarrow x - e_n \oplus e_{r_0}$, as $\| y \| = \| x - e_n \oplus e_{r_0} \|$, $\delta(y) = \delta(x - e_n \oplus e_{r_0})$, $| x - e_n \oplus e_{r_0} | \prec | y |$, in the second case $y \Leftarrow x - e_n \oplus e_{r_1}$, so as well $\| y \| = \| x - e_n \oplus e_{r_1} \|$, $\delta(y) = \delta(x - e_n \oplus e_{r_1})$ and $| x - e_n \oplus e_{r_1} | \prec | y |$, and in the third case $y = x - e_n \oplus e_{r_0}$. Thus there is always such vertex $x^1 = x - e_n \oplus e_{i_1}$ that $y \Leftarrow x^1$ or $y = x^1$, where $i_1 = r_0$ or $i_1 = r_1$, and $x^1 \in A$, under the conditions of the theorem. Repeating this process again at $k = x_n - y_n$, we find such vertex $x^k = x^{k-1} - e_n \oplus e_{i_k}$, that $y \Leftarrow x^k$ or $y = x^k$, and $x^k \in A_n(y_n)$. Since $A_n(y_n)$ is the standard arrangement in the $T_n^n(y_n)$, then $y \in A_n(y_n)$.

If $y_n > x_n \ge 1$, then again $y \in A_n(y_n)$, since otherwise (as in the case II) we would get $x \Leftarrow y$, that would contradict the condition $y \Leftarrow x$.

The proof is completed for the case a). Case b) can be proved in similar way.

Consider a subset $A \subseteq T^n_{k_1 k_2 \cdots k_n}$ and its partitions: $A = \bigcup_{j=-k_i+1}^{k_i} A_i(j)$. We replace each $A_i(j)$ by standard arrangement in the $T_i^n(j)$ of the same cardinality, and this transformation is called N_i - normalization of the set A in respect to i - th direction. The resulting set we denote by $N_i(A)$.

Below we formulate one more property of the standard arrangement which is a generalization of Lemma 4 of [1] and further will be used proving the optimality of the standard arrangement of n - dimensional torus.

Lemma 3. If the standard arrangement is the optimal subset in the $(n-1)$ - dimensional torus and $A \subseteq T^n_{k_1 k_2 \cdots k_n}$ is an arbitrary set, then for any i, $1 \le i \le n$

$$| B(N_i(A)) | \ge | B(A) |.$$

For a Boolean vector $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ the set $\alpha(T^n_{k_1 k_2 \cdots k_n}) = \{ x \in T^n_{k_1 k_2 \cdots k_n} \ / \ \delta(x) = \alpha \}$ will be called α - part of the torus $T^n_{k_1 k_2 \cdots k_n}$. In general for an arbitrary

subset $A \subseteq T^n_{k_1k_2\cdots k_n}$, $\alpha(A) = \{x \in A \,/\, \delta(x) = \alpha\}$. It is clear that $T^n_{k_1k_2\cdots k_n} = \bigcup_{\alpha \in E^n} \alpha(T^n_{k_1k_2\cdots k_n})$ and all α-parts the torus are isomorphic. Notice also that α-parts of $T^n_{k_1k_2\cdots k_n}$ are arranged according to order $\Leftarrow$.

For two vertices $x = (x_1, x_2, \cdots, x_n)$ and $y = (y_1, y_2, \cdots, y_n)$ of $\alpha(T^n_{k_1k_2\cdots k_n})$, we define their sum as follows:

$x + y = (x_1 + y_1, x_2 + y_2, \cdots, x_n + y_n) = (z_1, z_2, \cdots, z_n)$,
where $x_i + y_i \equiv z_i (\text{mod } k_i)$, $1 \le z_i \le k_i$ for $\alpha_i = 1$ and $-k_i + 1 \le z_i \le 0$ for $\alpha_i = 0$, for any i, $1 \le i \le n$.

In the α-part of $T^n_{k_1k_2\cdots k_n}$ we define sphere and envelope with the centre $x \in \alpha(T^n_{k_1k_2\cdots k_n})$ and radius k in the following way: $S^n_\alpha(x,k) = \{y = x + \sum_{i=1}^{n} (-1)^{1+\alpha_i} r_i e_i \,/\, \sum_{i=1}^{n} r_i \le k\}$ and $O^n_\alpha(x,k) = S^n_\alpha(x, k) \setminus S^n_\alpha(x, k-1)$, where r_i are non-negative integers, for any i, $1 \le i \le n$.

For any subset of α-parts of $T^n_{k_1k_2\cdots k_n}$, $A \subseteq \alpha(T^n_{k_1k_2\cdots k_n})$, the subset of interior vertices is defined as follows:
$B_\alpha(A) = \{x \in A \,/\, S^n_\alpha(x,1) \subseteq A\}$.

It is easy to check that the linear order $\Leftarrow$ between the vertices in each α-part of $T^n_{k_1k_2\cdots k_n}$ coincides with a diagonal sequence defined in [6], and its each initial segment is again called a standard arrangement.

It is proven in [6] that if A is the standard arrangement in $\alpha(T^n_{k_1k_2\cdots k_n})$, and $C \subseteq \alpha(T^n_{k_1k_2\cdots k_n})$ is an arbitrary set of cardinality $|A|$, then $|B_\alpha(A)| \ge |B_\alpha(C)|$.

Now we prove a statement, referring to the standard arrangements in α-parts, which is a generalization of Lemma 3 in [1].

Lemma 4. If A, E, F *and* C are such standard arrangements in the α-part of $T^n_{k_1k_2\cdots k_n}$, that $|A| \ge |E| \ge |F| \ge |C|$, $|A| + |C| = |E| + |F|$ and either A or C are a sphere in α-part, then

$$|B_\alpha(A)| + |B_\alpha(C)| \ge |B_\alpha(E)| + |B_\alpha(F)|.$$

Since all α-parts of the torus $T^n_{k_1k_2\cdots k_n}$ are isomorphic, then without loss of generality we will consider only the first α-part, that is $\alpha = (1, 1, \cdots, 1) = \tilde{1}$. Then the partitions of sets $S^n_\alpha(x,k)$ and $O^n_\alpha(x,k)$ by i-th direction are:

$$S_\alpha^n(x,k)=\bigcup_{j=0}^{k_i-1}S_{\alpha,i}^n(x+je_i,k-j),$$

$$O_\alpha^n(x,k)=\bigcup_{j=0}^{k_i-1}O_{\alpha,i}^n(x+je_i,k-j),$$

where $S_{\alpha,i}^n(x+je_i,k-j)\subseteq\alpha(T_i^n(x_i+j))$, $O_{\alpha,i}^n(x+je_i,k-j)\subseteq\alpha(T_i^n(x_i+j))$ and $S_{\alpha,i}^n(x+je_i,k-j)=O_{\alpha,i}^n(x+je_i,k-j)=\varnothing$ for $k-j<0$.

It is easy to see that if A is the standard arrangement in α - part, then all $A_1(j)$, except maybe one, are spheres in the $(n-1)$-dimensional α - part and

$$B_\alpha(A)=\bigcup_{j=1}^{k_1}B_\alpha(A_1(j))$$

or (3)

$$B_\alpha(A)=\bigcup_{j=1}^{k_1}B_\alpha(A_1(j))\setminus\{(j_0,k_2,k_3,\cdots,k_n)\}$$

when $A_1(j_0)=\alpha(T_1^n(j_0))$ and $A_1(j_0+1)\neq\alpha(T_1^n(j_0+1))$.

Indeed, if x is the latest vertex of the set A and $x\in A_1(j_1)$, then any vertex $y=(j,y_2,y_3,\cdots,y_n)$, such that $\|y\|<\|x\|$ or $\|y\|=\|x\|$ and $j>j_1$, precedes x, and when $\|y\|>\|x\|$ or $\|y\|=\|x\|$ and $j<j_1$, none of the vertices $y=(j,y_2,y_3,\cdots,y_n)$ precede the vertex x. Consequently,

$$A_1(j)=\begin{cases}S_{\alpha,1}^n(\tilde{1}+(j-1)e_1,\|x\|-n-j) & , \quad if \quad 1\le j<j_1\\ S_{\alpha,1}^n(\tilde{1}+(j-1)e_1,\|x\|-n-j+1), & if \quad j_1<j\le k_1\\ S_{\alpha,1}^n(\tilde{1}+(j_1-1)e_1,\|x\|-n-j_1)\cup S, & if \quad j=j_1,\\ \quad where\ \ S\subseteq O_{\alpha,1}^n(\tilde{1}+(j_1-1)e_1,\|x\|-n-j_1+1)\end{cases}$$

It follows that $B_\alpha(A_1(j))+e_1\subseteq A_1(j+1)$ for any $j,1\le j<k_1$, except perhaps the one j_0, for which $A_1(j_0)=\alpha(T_1^n(j_0))$ and $A_1(j_0+1)\neq\alpha(T_1^n(j_0+1))$.

Now we prove the Lemma 4 .

The proof is by induction on n. For $n=1$ the proof is obvious. Suppose two standard arrangements are given in α - part: $A=S_\alpha^n(\tilde{1},k)\cup A'$ and $C=S_\alpha^n(\tilde{1},r)\cup C'$, where . Consider the partition of these sets by the first direction:

$$A = \bigcup_{j=1}^{k_1} A_1(j) = (\bigcup_{j=1}^{j_0-1} S^n_{\alpha,1}(\tilde{1} + (j-1)e_i, k-j+1)) \cup S^n_{\alpha,1}(\tilde{1} + (j_0-1)e_1, k-j_0+1) \cup$$
$$\cup A^1_1 \cup (\bigcup_{j=j_0+1}^{k_1} S^n_{\alpha,1}(\tilde{1} + (j-1)e_1, k-j+2)),$$

$$C = \bigcup_{j=1}^{k_1} C_1(j) = (\bigcup_{j=1}^{j_1-1} S^n_{\alpha,1}(\tilde{1} + (j-1)e_i, r-j+1)) \cup S^n_{\alpha,1}(\tilde{1} + (j_1-1)e_1, r-j_1+1) \cup$$
$$\cup C^1_1 \cup (\bigcup_{j=j_1+1}^{k_1} S^n_{\alpha,1}(\tilde{1} + (j-1)e_1, r-j+2)),$$

where $A^1_1 \subset O^n_{\alpha,1}(\tilde{1} + (j_0-1)e_i, k-j_0+2)$, $\varnothing \neq C^1_1 \subseteq O^n_{\alpha,1}(\tilde{1} + (j_1-1)e_i, r-j_1+2)$, and $S^n_{\alpha,i}(\tilde{1} + je_1, a) = S^n_{\alpha,1}(\tilde{1} + (j-1)e_1, a+1) + e_1$ for $a = \sum_{i=2}^{n} k_i$.

Two cases are possible:

Case I. $k - j_0 > r - j_1$ or $k - j_0 = r - j_1$ and $|A^1_1| \geq |C^1_1| > 0$.

In this case, if we remove some number of vertices from the subset C^1_1 of the set $C_1(j_1)$ and add the same number of new vertices to the set $A_1(j_0)$ so that the newly formed subsets $C^1_1(j_1)$ and $A^1_1(j_0)$ also are standard arrangements in $\alpha(T^n_1(j_1))$ and $\alpha(T^n_1(j_0))$, where at least one of them was a sphere, then by property (4.4) and the induction supposition, the total number of interior vertices of the obtained sets

$$A^1 = (A \setminus A_1(j_0)) \cup A^1_1(j_0) \text{ and } C^1 = (C \setminus C_1(j_1)) \cup C^1_1(j_1)$$

will not decrease.

In the next step, instead of subsets $C_1(j_1)$, and $A_1(j_0)$ we consider

- the subsets $C_1(j_1+1)$ and $A^1_1(j_0)$, where on the first step the $C^1_1(j_1)$ was a sphere, or
- the subsets $C^1_1(j_1)$ and $A_1(j_0-1)$, where on the first step the $A^1_1(j_0)$ was a sphere, or
- the subsets $C_1(j_1+1)$ and $A_1(j_0-1)$, where on the first step $C^1_1(j_1)$ and $A^1_1(j_0)$ were the spheres,

and apply the above transfer of the vertices. This process continues until at least one of the sets A and C becomes a sphere.

Case II. $k - j_0 < r - j_1$ or $k - j_0 = r - j_1$ and $|A^1_1| < |C^1_1|$.

In this case, first of all we remove from subset A' of set A a certain number of vertices and add the same number of new vertices to the set C, so that one of the sets A and C will be sphere, and at each step this transfer takes place between some of the subsets $A_1(j)$, $j \geq j_0$, and $C_1(j)$, $j \leq j_1$. Therefore, by the induction assumption, the total number of internal vertices can only increase. Received after this transformation sets A^1 and C^1 can only be of two kinds:

a) $C^1 = S^n_\alpha(\tilde{1}, r+1) = \bigcup_{j=1}^{k_1} S^n_{\alpha,1}(\tilde{1} + (j-1)e_1, r-j+1)$,

$$A^1 = S^n_\alpha(\tilde{1}, k) \cup A'' = (\bigcup_{j=1}^{j_2-1} S^n_{\alpha,1}(\tilde{1} + (j-1)e_1, k-j+1)) \cup$$

$$\cup (S^n_{\alpha,1}(\tilde{1} + (j_2-1)e_1, k-j_2+1) \cup A^2_1) \cup (\bigcup_{j=j_2+1}^{k_1} S^n_{\alpha,1}(\tilde{1} + (j-1)e_1, k-j+2)),$$

b) $A^1 = S^n_\alpha(\tilde{1}, k) = \bigcup_{j=1}^{k_1} S^n_{\alpha,1}(\tilde{1} + (j-1)e_1, k-j+1)$,

$$C^1 = S^n_\alpha(\tilde{1}, r) \cup C'' = (\bigcup_{j=1}^{j_3-1} S^n_{\alpha,1}(\tilde{1} + (j-1)e_1, r-j+1)) \cup$$

$$\cup (S^n_{\alpha,1}(\tilde{1} + (j_3-1)e_1, r-j_3+1) \cup C^2_1) \cup (\bigcup_{j=j_3+1}^{k_1} S^n_{\alpha,1}(\tilde{1} + (j-1)e_1, r-j+2)).$$

In case a) it is clear that $k - j_2 + 1 < r + 1$. Hence, if instead of sets A^1 and C^1 we take the set

$$A^2 = (\bigcup_{j=1}^{k-r} S^n_{\alpha,1}(\tilde{1} + (j-1)e_1, k-j+1)) \cup (\bigcup_{j=k-r+1}^{k_1} S^n_{\alpha,1}(\tilde{1} + (j-1)e_1, k-j+2)),$$

$$C^2 = \bigcup_{j=1}^{r-k+j_2-1} S^n_{\alpha,1}(\tilde{1} + (j-1)e_1, r-j+1) \cup (S^n_{\alpha,1}(\tilde{1} + (r-k+j_2-1)e_1, k-j_2+1) \cup C^2_1)$$

$$\cup (\bigcup_{j=r-k+j_2+1}^{k_1} S^n_{\alpha,1}(\tilde{1} + (j-1)e_1, r-j+2)),$$

where $|C^2_1| = |A^2_1|$ and $C^2_1(r-k+j_2)$ is the standard arrangement in the $\alpha(T^n_1(r-k+j_2))$, it is obvious that $|A^2| + |C^2| = |A^1| + |C^1|$, and by (3) $|B_\alpha(A^2)| + |B_\alpha(C^2)| = |B_\alpha(A^1)| + |B_\alpha(C^1)|$.

So, if one of the sets A^2 and C^2 is sphere, then the lemma is proved; otherwise we come to the case I, since $r+1 > k - j_2 + 1$.

In case b), since $k - j_0 < r - j_1$, $j_0 \le k_1$, $j_3 \le j_1$, then $r - k_1 + 1 \le k - k_1 + 1 \le r - j_3 + 1$. Hence, if instead of sets A^1 and C^1 take the sets

$$C^2 = (\bigcup_{j=1}^{r-k+k_1} S^n_{\alpha,1}(\tilde{1} + (j-1)e_1, r-j+1)) \cup (\bigcup_{j=r-k+k_1+1}^{k_1} S^n_{\alpha,1}(\tilde{1} + (j-1)e_1, r-j+2)),$$

$$A^2 = \bigcup_{j=1}^{k-r+j_3-1} S^n_{\alpha,1}(\tilde{1}+(j-1)e_1, k-j+1) \cup (S^n_{\alpha,1}(\tilde{1}+(k-r+j_3-1)e_1, r-j_3+1) \cup$$

$$\cup A^2_1) \cup (\bigcup_{j=k-r+j_3+1}^{k_1} S^n_{\alpha,1}(\tilde{1}+(j-1)e_1, k-j+2)),$$

where $|C^2_1| = |A^2_1|$ and $A^2_1(k-r+j_3)$ is the standard arrangement at the $\alpha(T^n_1(k-r+j_3))$, then $|A^2| + |C^2| = |A^1| + |C^1|$ and, by (3),

$$|B_\alpha(A^2)| + |B_\alpha(C^2)| = |B_\alpha(A^1)| + |B_\alpha(C^1)|.$$

So, if one of the sets A^2 and C^2 is sphere, then the lemma is proved; otherwise we come to the case I, since $k-k_1 < r-j_3$. The proof is completed.

Bibliography

[1] Aslanyan L.H., Karakhanyan V. M., Torosyan B. E., On the compactness of Subsets of Vertices of the n-dimensional unit cube, Dokl. Akad. Nauk SSSR,241, N 1 (1978), pp. 11-14, Translation in Soviet Math, Dokl., American Mathematical Society, v.10, 4, 1978, pp. 781-785.

[2] Aslanyan L.H, The discrete isoperimetric problem and the related extremal problems in discrete spaces (in Russian), "Problemy Kibernetiki", 36, Moscow, 1979, pp. 85-127.

[3] Harper L., H., Optimal numberings and isoperimetric problems on graphs, Journal of Combinatorial theory, 1 (1966), pp. 385-393.

[4] Katona G., The Hamming-sphere has minimum boundary, Studia Scient. Math. Hungarica, 10 (1975), pp. 131-140.

[5] Wang D. L., Wang P., Discrete isoperimetric problems, SIAM J. Appl. Math., Vol. 32, N4 (1977), pp. 860-870.

[6] Wang D. L., Wang P., External configurations on a discrete torus and a generalization of the generalized Macauley theorem, SIAM J. Appl. Math., Vol.33, N1 (1977), pp.55-59.

Authors' Information

Vilik Karakhanyan *– Senior Researcher, Institute for Informatics and Automation Problems, NAS RA, P.Sevak St. 1, Yerevan 14, Armenia*

ON THE STRUCTURE OF MAXIMUM INDEPENDENT SETS IN BIPARTITE GRAPHS

Vahagn Minasyan

Abstract: *In this paper it is shown that for bipartite graphs the structure of the family of maximum independent sets can be described constructively, in the following sense. For a bipartite graph there are some "basic" maximum independent sets, in terms of which any maximum independent set can be described, in the sense that there is one-to-one correspondence between a maximum independent set and an irreducible combination of these "basic" maximum independent sets. König's theorem states that there is duality between the cardinalities of maximum matching and minimum vertex cover. Viewing the mentioned structure, in this paper it is shown that another duality holds, which is between the sets rather than their cardinalities. We believe that this duality is not just of theoretical interest, but it also can yield to a usable algorithm for finding a maximum matching of bipartite graph. In this paper we do not present such algorithm; instead we mention what approaches we plan to use in further works to obtain such algorithm.*

Keywords: *bipartite graph, maximum independent set, distributive lattice, duality.*

ACM Classification Keywords:*G.2.1 Discrete mathematics: Combinatorics*

Introduction

Let $G = (W, E)$ be a graph, where W is the set of vertices, and E is the set of edges. Two vertices $u, v \in W$ are said to be *adjacent* with each other, if $\{u, v\} \in E$; otherwise they said to be *independent*. A set of vertices is called an *independent set*, if any two vertices of it are independent. For instance,a set consisting of one vertex is an independent set. A*maximum independent set* (*MMIS*) is one with the maximum cardinality among all independent sets (don't be confused with the *maximal independent set*, which is an independent set, no proper superset of which is an independent set). The cardinality of MMIS-es of G is denoted by$\alpha(G)$. A set of vertices in G, such that each edge of G is adjacent with some vertex in that set, is called a *vertex cover*. For instance, the set of all vertices of G is a vertex cover. A *minimum vertex cover* is one with minimum cardinality among all vertex covers;that cardinality is denoted by $\beta(G)$. It is easy to see that each vertex cover is a complement of some independent set and vice-versa, so the complement of any MMIS is a minimum vertex cover and vice-versa. Thus we get$\alpha(G) + \beta(G) = |W|$. The concepts of independent set and vertex cover are related with the concept of *matching*, which is a set of edges, no distinct two of which share a common vertex. For instance, a set consisting of one edge is a matching. A *maximum matching* is one with maximum cardinality among all matchings; that cardinality is denoted by $\gamma(G)$. Note, that for a given matching and a given vertex cover, each edge of the matching is "covered" by some vertex of the vertex cover, and different edges are covered with different vertices.This means that the cardinality of matching doesn't exceed the cardinality of the vertex cover, so

$\gamma(G) \leq \beta(G)$. $G$is called *bipartite*, if its vertices can be decomposed into two independent sets U and V, such that $W = U \cup V$ and $U \cap V = \emptyset$. A bipartite graph is denoted as $G = (U, V, E)$, where U and V are said to be its *parts*. König's theorem [Harary, 1969] states, that for a bipartite graph it holds$\gamma(G) = \beta(G)$. For a general graph the problem of finding a MMIS is NP-hard [Karp, 1972], however for various specific classes of graphs, including bipartite graphs, there are polynomial-time algorithms [Harary, 1969].In some applications [Johnson, 1988] it is needed to deal with all MMIS-es of the graph, so it is of both theoretical and practical interest to describe the family of all MMIS-es and to show how to construct not just any, but some particular MMIS. Usually this is done simply by generating all MMIS-es of the graph, and for some specific classes of graphs there are algorithms which generate all MMIS-es with polynomial-time delay between two successive outputs [Kashiwabara, 1992] (note, that in some cases the number of MMIS-es to be generated is potentially exponential, and there are various notions on what to consider a "polynomial-time" algorithm for problems of this kind [Johnson, 1988]). In this paper we describe the family of MMIS-es of a bipartite graph by describing its structure, rather than generating all MMIS-es. We show that MMIS-es of a bipartite graph form a *distributive lattice* with respect to simple set operations (see the preliminaries regarding lattice theory bellow in this section), and we show how to obtain that lattice. After it is done, various queries can be performed, and generating all MMIS-es is one of them. The classic solution of the problem of finding just one MMIS of a bipartite graph is by Ford-Fulkerson algorithm [Ford, Fulkerson, 1962], which provides a MMIS of bipartite graph $G = (U, V, E)$, performing $O(|E||W|)$ operations in worst case, where $W = U \cup V$. There is an optimization of this approach, called Hopcroft-Karp algorithm [Hopcroft, Karp, 1973], which provides a MMIS performing $O\left(|E|\sqrt{|W|}\right)$ operations in worst case. In this paper we also discuss the problem of providing an algorithm, which obtains a MMIS of bipartite graph while sequentially handling its vertices. Here we show that to do this, it is preferable to obtain the *greatest* (in the sense of the lattice of MMIS-es) MMIS, rather than just any MMIS.In some sense, the greatest MMIS corresponds to the intersection of all MMIS-es. In this paper we prove that a sort of duality holds between the intersection of all MMIS-es and the union of all maximum matchings. Besides this duality is of theoretical interest, we also believe, that it can yield to a usable algorithm which provides a MMIS of bipartite graph. In this paper we do not provide such algorithm; instead we mention what approaches we plan to use in order to obtain that result in further works. Next in this section we give some preliminaries regarding distributive lattices.

There are two equivalent definitions for lattices [Birkhoff, 1948]. Let$L$ be a carrier set. In terms of partially ordered sets*lattice* is a pair $(L, \preccurlyeq)$, where $\preccurlyeq$ is such partial order on L, that every two elements have infimum and supremum in$L$. In terms of abstract algebra, *lattice* is a triple $(L,\vee,\wedge)$, where $\vee$ and $\wedge$ are such binary operations on L, that for all $a, b, c \in L$ it holds:

$a \vee a = a$ and $a \wedge a = a$,

$a \vee b = b \vee a$ and $a \wedge b = b \wedge a$,

$a \vee (b \vee c) = (a \vee b) \vee c$ and $a \wedge (b \wedge c) = (a \wedge b) \wedge c$,

$a \vee (a \wedge b) = a$ and $a \wedge (a \vee b) = a$,

distributive lattice is one where the following property also holds:

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c) \text{and} a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c).$$

Operations $\vee$ and $\wedge$ are called *join* and *meet* respectively. The equivalence of mentioned two definitions can be checked by showing that if a lattice is defined as a partially ordered set, than one can define join and meet operations on it as $a \vee b = \sup\{a, b\}$ and $a \wedge b = \inf\{a, b\}$, and if lattice is defined as an abstract algebra, then one can define a partial order on it as$a \preccurlyeq b$ if and only if $a \vee b = b$ (or $a \wedge b = a$). For instance, the family of all subsets of a set is a lattice, where join and meet operations are respectively the union and intersection of subsets; as these operations are distributive with respect to each other, then the mentioned lattice is distributive as well. Actually, not only the family of all subsets, but any ring of subsets (i.e. a family of subsets, which is closed with respect to the union and intersection operations) is a distributive lattice. Birkhoff's representation theorem [Birkhoff, 1948] states that the opposite claim is also true, i.e. each distributive lattice is isomorphic to some ring of subsets. Another example of a distributive lattice is the set of natural numbers with operations of taking the least common multiple and the greatest common divider as join and meet operations respectively. Note, that the corresponding partial order is the divisibility of the numbers, and as one divides any number, and any number divides zero, then they are the least and the greatest elements of the lattice respectively. It may not be the case for infinite lattices, but any finite lattice has the least and the greatest elements. For a lattice $(L,\vee,\wedge)$, an element $x \in L$ is called *join-irreducible*, if it is not the least element, and if for all $a, b \in L$, $a \vee b = x$ implies $a = x$ or $b = x$. It is known [Birkhoff, 1948], thateach element of a distributive lattice has only one irreducible representation as a join of join-irreducible elements of that distributive lattice. In this sense, a distributive lattice can be considered as given, if its join-irreducible elements are given.

In the next section we show that MMIS-es of a bipartite graph form a distributive lattice with respect to simple set operations and show how to find the lowest and the join-irreducible elements of that lattice. In the next section we discuss the problem of obtaining a MMIS of bipartite graph while sequentially handling its vertices and show that a sort of duality holds between the intersection of all MMIS-es and the union of all maximum matchings. Finally we provide a short conclusion of this paper and mention the further works.

The lattice of MMIS-es

Let $G = (U, V, E)$ be a bipartite graph. Here we will define join and meet operations on MMIS-es of G and will show that the family of all MMIS-es of G is a distributive lattice with respect to that operations. First we need some notations.We denote by $\mathcal{M}$ the family of all MMIS-es of G. For a set of vertices $S \subseteq U \cup V$ we denote $S_U = S \cap U$and $S_V = S \cap V$; we will call these sets projections of S on U and V respecvely. Also for any set of edges $X \subseteq E$ and set of vertices $S \subseteq U \cup V$, we will denote by $X(S)$ the set of vertices of G, where each vertex is adjacent with some vertex of S by an edge of X. Now we claim that ifM_1 and $M_2$are MMIS-es of G, then $M_{1_U} \subseteq M_{2_U}$implies $M_{2_V} \subseteq M_{1_V}$ and vice-versa:

Claim 1: for all $M_1, M_2 \in \mathcal{M}\, M_{1_U} \subseteq M_{2_U}$ if and only if $M_{2_V} \subseteq M_{1_V}$.

Indeed, as M_1 and M_2 are MMIS-es, then M_{1_V} is the set of all vertices in V, which are independent with M_{1_U}, and M_{2_V} is the set of all vertices in V, which are independent with M_{2_U}, thus if $M_{1_U} \subseteq M_{2_U}$, then only some vertices of M_{1_V} are also independent with M_{2_U}, so we get $M_{2_V} \subseteq M_{1_V}$. The opposite direction of the claim can be proved identically. We define a partial ordered set $(\mathcal{M}, \preccurlyeq)$ as follows:

$$\text{for all } M_1, M_2 \in \mathcal{M} \text{ we define } M_1 \preccurlyeq M_2 \text{ if } M_{1_U} \subseteq M_{2_U}. \tag{1}$$

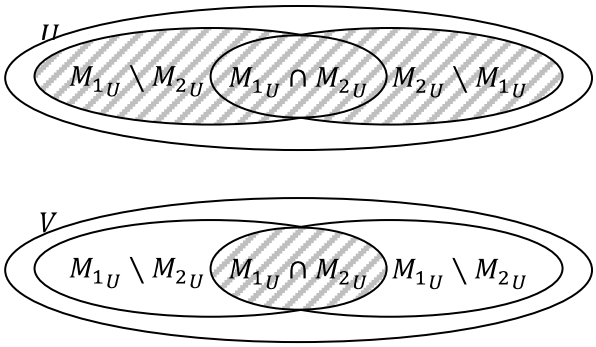


Figure 1a

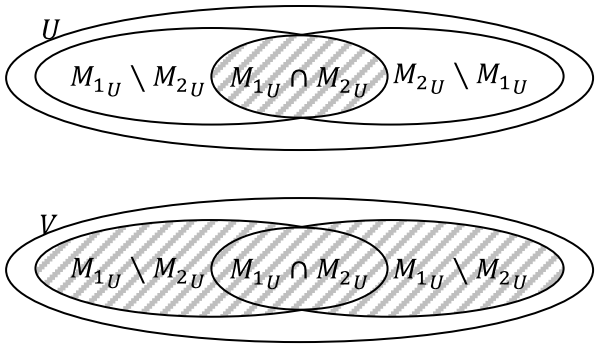


Figure 1b

So we have defined a partial order on MMIS-es according to their projections on U; from Claim 1 it follows, that we would get the dual partial order of one we got, if we define it according to the projections on V. In this sense, the partial ordered set $(\mathcal{M}, \preccurlyeq)$ is invariant with respect to the parts of G, though we define it with respect to U. Now let M_1 and M_2 be MMIS-es of G. We define join and meet operations for M_1 and M_2 as follows:

$$M_1 \vee M_2 = (M_{1_U} \cup M_{2_U}) \cup (M_{1_V} \cap M_{2_V}) \tag{2}$$

and

$$M_1 \wedge M_2 = (M_{1_U} \cap M_{2_U}) \cup (M_{1_V} \cup M_{2_V}): \tag{3}$$

At Figure 1a the shaded part corresponds to $M_1 \vee M_2$ and at Figure 1b the shaded part corresponds to $M_1 \wedge M_2$. We will show that $(\mathcal{M}, \preccurlyeq)$ is a distributive lattice with respect to these operations. First we show, that:

Claim 2: $\mathcal{M}$ is closed with respect to operations defined at (2) and (3); i.e. $M_1 \vee M_2$ and $M_1 \wedge M_2$ are MMIS-es.

Note, that $M_1 \vee M_2$ and $M_1 \wedge M_2$ are independent sets, as any vertex in $M_1 \cap M_2$ is independent with any vertex in $M_1 \cup M_2$, so to prove this claim we need to show that $|M \vee N| = |M \wedge N| = \alpha(G)$. From (2) and (3) it follows, that $(M_1 \vee M_2) \cup (M_1 \wedge M_2) = M_1 \cup M_2$and $(M_1 \vee M_2) \cap (M_1 \wedge M_2) = M_1 \cap M_2$.

Note, that for any two sets A and B we have $|A| + |B| = |A \cup B| + |A \cap B|$. So we get $|M_1 \vee M_2| + |M_1 \wedge M_2| = |(M_1 \vee M_2) \cup (M_1 \wedge M_2)| + |(M_1 \vee M_2) \cap (M_1 \wedge M_2)| = |M_1 \cup M_2| + |M_1 \cap M_2| = |M_1| + |M_2| = 2\alpha(G)$ Thus, we got $|M_1 \vee M_2| + |M_1 \wedge M_2| = 2\alpha(G)$. As $M_1 \vee M_2$ and $M_1 \wedge M_2$ are independent sets, then we also have $|M \vee N| \leq \alpha(G)$ and $|M \wedge N| \leq \alpha(G)$, so we get $|M \vee N| = |M \wedge N| = \alpha(G)$, which proves the claim. Note that $(M_1 \vee M_2)_U = M_{1_U} \cup M_{2_U}$, so from (1) it follows, that $M_1 \vee M_2$ is the supremum of M_1 and M_2 in $(\mathcal{M}, \preccurlyeq)$. Similarly it can be show that $M_1 \wedge M_2$ is the infimum of M_1 and M_2in $(\mathcal{M}, \preccurlyeq)$. Thus, we have shown that $(\mathcal{M},\vee,\wedge)$ is a lattice; now we will show, that:

Claim 3: $(\mathcal{M},\vee,\wedge)$is distributive.

Indeed, from Claim 2 it follows that that the family of projections of all MMIS-es on U is closed with respect to the union and intersection operations, and thus, as it is mentioned above, forms a distributive lattice with respect to them (the same holds for the projections on V). Now note that the bijection $M_U \leftrightarrow M$ is an isomorphism between that lattice and $(\mathcal{M},\vee,\wedge)$, so the last is also distributive. Also note, that the bijection $M_V \leftrightarrow M$ yields to the lattice $(\mathcal{M},\wedge,\vee)$, which is the dual of $(\mathcal{M},\vee,\wedge)$; this is because at (1) we have defined the partial order on $\mathcal{M}$ with respect to U. So we have proved that the family of all MMIS-es of a bipartite graph forms a distributive lattice with respect to join and meet operations defined by (2) and (3). As it is mentioned before, each distributive lattice is described by its least and join-irreducible elements. Next in this section we show how to find these elements for $(\mathcal{M},\vee,\wedge)$.

Let $G = (U, V, E)$ be a bipartite graph and $(\mathcal{M},\vee,\wedge)$ be the lattice of its MMIS-es. Let M be a MMIS of G and Z be a maximum matching of G. As it is mentioned before, the complement of M is a minimum vertex cover; we will denote by N. Also we will denote by K the set of vertices which are not adjacent with Z. We claim, that:

Claim 4: each edge of Z is adjacent with exactly one vertex of N, and each vertex of N is adjacent with some edge of Z (see Figure2).

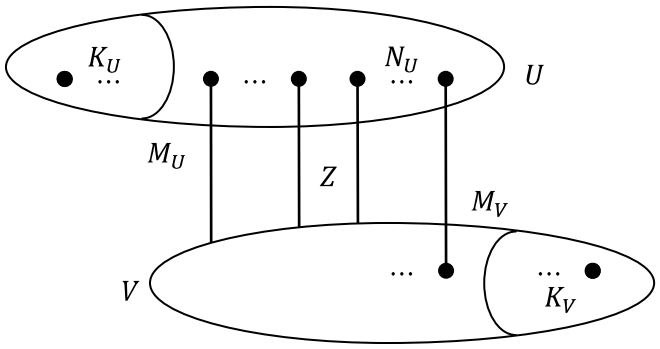


Figure 2

Indeed, as N is a vertex cover, then any edge of Z is adjacent with some vertex of N, and as Z is a matching, then different edges of Z are adjacent with different vertices of N. Note that Z is a maximum matching, and N is a minimum vertex cover, so by König's theorem we have $|Z| = |N|$. This means, that there are no vertices in N, which are not adjacent with an edge of Z, and that there are no edges of Z which are adjacent with two vertices of N (as otherwise there would be less edges in Z then there are vertices in N). So the claim is proved.

Note, that from this claim it follows, that for any MMIS M and for any maximum matching Z it holds $K \subseteq M$ (see Figure 2), where K is the set of vertices which are not adjacent with Z. Now let $S \subseteq U$be a set of vertices of G. If there are some MMIS-es containing S, then from the definition of lattice $(\mathcal{M},\vee,\wedge)$ it follows, that there is the least among them. Next we will describe how to find it.

From Claim 4 it follows, that:

if there is a MMIS which contains S, then it also contains $Z\big(E(S)\big)$; we will denote it by $(ZE)(S)$.

Indeed, if MMIS Mcontains S, then $S \subseteq M_U$ and obviously $E(S) \subseteq N_V$ (see Figure 2). From the other hand, Claim 4 states, that each edge of Z is adjacent with exactly one vertex of N, so no edge of Z connects a vertex of N_U with a vertex of N_U; this means, that if $T \subseteq N_V$ then $Z(T) \subseteq M_U$, which proves the claim (see Figure 2). From this claim it follows, that if there is a MMIS which contains S, then it also contains $(ZE)^i(S)$ for any $i \geq 0$. We will denote $\langle S\rangle = (ZE)^k(S)$, where k is the least integer, such that $(ZE)^k(S) = (ZE)^{k+1}(S)$ (obviously such k exists). We claim, that:

there exists a MMIS containing S if and only if for some maximum patching Z it holds $E(\langle S\rangle) \cap K_V = \emptyset$, and if it holds, then the least MMIS containing S is the following: $K_U \cup \langle S\rangle \cup \big(V \setminus E(\langle S\rangle)\big)$.

To be short, in the proof of this claim we will denote $A = K_U \cup \langle S\rangle$and $B = E(\langle S\rangle)$. From the denotation of $\langle S\rangle$ it follows, that there is no edge, which connects a vertex of A with a vertex of $V \setminus B$, and there is no edge of Z which connects a vertex of $U \setminus A$ with a vertex of B. Note, that if $B \cap K_V = \emptyset$, then all vertices in B are

adjacent with Z, and as there is no edge of Z between B and $U \setminus A$, then $|B| + |U \setminus A| = |Z|$. From König's theorem it follows, that in this case $A \cup (V \setminus B)$ is a MMIS. Otherwise, if $B \cap K_V \neq \emptyset$, then $\langle S \rangle$ is adjacent with some vertex of K, and from Claim 4 it follows, that there is no MMIS containing S. So we have proved this claim. Based on this claim it is easy to describe an algorithm, which provides the least MMIS of G containing the given set of vertices $S \subseteq U$, if such MMIS exists. The algorithm takes as input the graph G, a maximum matching of it and a set of vertices $S \subseteq U$; if G has a MMIS containing S, then it provides the least of such MMIS-es, and otherwise it reports that no MMIS of G contains S:

Algorithm 1:

A1 denote by Z the given maximum matching of G,
denote by K the set of vertices which are note adjacent with Z,

A2 set $A = K_U \cup S$ and $B = \emptyset$,

A3 set $B = E(A)$,

A4 if $B \cap K_V \neq \emptyset$, then report, that G has no MMIS containing S and exit,

A5 if B didn't get greater, then provide $A \cup (V \setminus B)$as the least MMIS of G containing S and exit,

A6 set $A = Z(B)$,

A7 go to step A3.

Note that this algorithm performs $O(|E|)$ operations in the worst case. Also note that in order to find the least MMIS of G, we can find a maximum matching Z and call Algorithm 1 for set K_U, where K is the set of vertices which are not adjacent with Z, as any MMIS of G contains K. Obviously, this algorithm can be also used to find the greatest MMIS of G. As it is mentioned in the proof of Claim 3, the projections of all MMIS-es on U are closed with respect to the union and intersection operations, thus they form a ring of subsets, and the bijection $M_U \leftrightarrow M$ is an isomorphism between that ring and $(\mathcal{M},\vee,\wedge)$. This means, that the join-irreducible elements of $(\mathcal{M},\vee,\wedge)$ are the isomorphic images of the join-irreducible elements of the ring of projections, so for any $u \in U$, the least MMIS containing u is join-irreducible in $(\mathcal{M},\vee,\wedge)$. Thus, by Algorithm 1 one can find all join-irreducible elements of $(\mathcal{M},\vee,\wedge)$. For two partially ordered set X and YY^X denotes the partial ordered set all isotonic functions from X to Y, where for two isotonic functions θ_1 and $\theta_2\theta_1 \preccurlyeq \theta_2$ if for all $x \in X\theta_1(x) \preccurlyeq \theta_2(x)$ [Birkhoff, 1948]. Birkhoff's representation theorem states, that if L is a distributive lattice and A is the partially ordered set consisting of its join-irreducible elements, then $L \cong \mathbf{2}^{\check{A}}$, where $\mathbf{2}$ denotes the chain with length 1 and $\check{A}$ denotes the dual of A. From this theorem it follows, that each element of L has exactly one irreducible representation as join of join-irreducible elements of L. In this sense, the family of all MMIS-es of G can be considered as obtained, if for all $u \in U$ the least MMIS containing u, as well as the least MMIS of G are obtained.

Duality between the intersection of all MMIS-es and union of all maximum matchings

Let $G = (U, V, E)$ be a bipartite graph. In the previous section we have shown, that the MMIS-es of G form a distributive lattice with respect to join and meet operations defined at (2) and (3). This lattice has the greatest and

the least elements, so in this sense there are the greatest and the least MMIS-es in G; we will denote them respectively by $\bigvee M$ and $\bigwedge M$. We will also denote the union and the intersection of all MMIS-es respectively by $\bigcup M$and $\bigcap M$. Note, that from (2) and (3) it follows, that:

$$\bigvee M = (\bigcup M_U) \cup (\bigcap M_V) \text{and} \bigwedge M = (\bigcap M_U) \cup (\bigcup M_V). \quad (4)$$

We will say, that "a new vertex u' is being added" to bipartite graph $G = (U, V, E)$, bearing in mind that we obtain a "new" bipartite graph G', which parts are $U \cup \{u'\}$ and V, and which edges are the edges of$E$ in addition with some "new" edges, which connect u' with some vertices of V. We claim, that:

Claim 5: while "adding a new vertex" u' to G, $\alpha(G)$ increments if and only if u' is independent with $\bigvee M$.

Obviously, either $\alpha(G') = \alpha(G) + 1$ or $\alpha(G') = \alpha(G)$. Note, that if u' is independent with $\bigcap M_V$, then $(\bigvee M) \cup \{u'\}$ is the greatest MMIS of G'. Otherwise, i.e. if u' is adjacent with some vertex of $\bigcap M_V$, then no MMIS of G is independent with u', so $\alpha(G') = \alpha(G)$. Thus the claim is proved. Next we will describe an algorithm, which obtains the greatest MMIS of G' based on the greatest MMIS of G. Note, that by so we will provide an algorithm, which sequentially handles vertices of a bipartite graph and provides a MMIS of it. As Claim 7 states, if $\alpha(G') = \alpha(G) + 1$, then the greatest MMIS of G' can be easily obtained. Otherwise, i.e. if $\alpha(G') = \alpha(G)$, then we have, that $\bigvee M$ is a MMIS of G', but it may not be the greatest one.If we obtain a maximum matching of G', then by Algorithm 1, we can obtain the greatest MMIS of G'.

We claim that:

each vertex of $\bigcap M_V$ is not adjacent with some maximum matching of G.

Let Z be a maximum matching of G, and $v \in \bigcap M_V$. If v is not adjacent with Z, then the claim is proved; otherwise, let $u \in \bigcap N_U$ be a vertex, such that $(u, v) \in Z$ (see Figure 3). As it is denoted at (4), $\bigvee M = (\bigcup M_U) \cup (\bigcap M_V)$ is the greatest MMIS of G, so no MMIS of G contains u. This means, that while calling Algorithm 1 for $\{u\}$, it stops at step A3 by finding a vertex $w \in K_V$ and a path from u to w, which has odd length, and even edges of which are edges of Z (see Figure 3).

Note that if we remove from Z the edge (u, v) and the even edges of the path found by Algorithm 1, then add to Z the odd edges of that path found by Algorithm 1, then we will get a maximum matching of G, which is not adjacent with v (see Figure 3). Thus the claim is proved. Note, that if $\alpha(G') = \alpha(G)$ (i.e. if u' is adjacent with some vertices in $\bigcap M_V$), then by König's theorem we have $\gamma(G') = \gamma(G) + 1$, so in this case each "new" edge belongs to some maximum matching of G'.

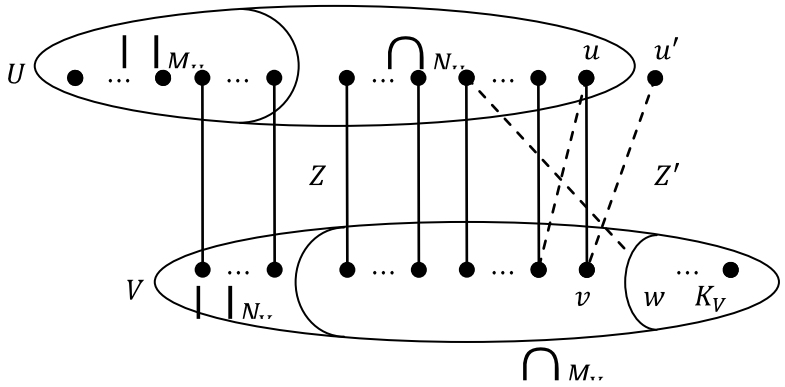


Figure 3

Now we will describe an algorithm, which provides the greatest MMIS of G' based on the greatest MMIS of G. The algorithm takes as input the graph G, the greatest MMIS of it, a maximum matching of it and the "new" vertex; it provides a maximum matching and the greatest MMIS of the "new" graph G':

denote by M the given greatest MMIS of G,

denote by Z the given maximum matching of G,

denote by u' the given "new" vertex,

denote by K the set of vertices which are note adjacent with Z,

if u' is independent with M, then set $M' = M \cup \{u'\}$, set $Z' = Z$ and go to step A8,

if u' is adjacent with some vertex $v \in K_V$, then set $Z' = Z \cup \{(u', v)\}$ and go to step A7,

pick a vertex $v \in M_V \setminus K_V$, which is adjacent with u' and denote by u the vertex for which $(u, v) \in Z$,

call Algorithm 1 for $\{u\}$, denote by p the path it finds to some vertex $w \in K_V$ and set $K' = K \setminus \{w\}$,

set $Z' = Z$, remove even edges of p from Z', add odd edges of p to Z' add (u', v) to Z',

call Algorithm 1 for K'_V and set M' to the set it provides,

provideM' as the MMIS of G' and Z' as a maximum matching of G'.

As Algorithm 1 performs $O(|E|)$ operations in the worst case, then Algorithm 2 also performs $O(|E|)$ operations in the worst case. This means, that the algorithm which sequentially handles vertices of a bipartite graph and for each vertex calls Algorithm 2, performs $O(|E||W|)$ operations in the worst case, where $W = U \cup V$. Next we

show that a sort of duality holds between the intersection of all MMIS-es and the union of all maximum matchings. We believe that this duality can yield to a more efficient algorithm which provides the greatest MMIS of G' based on one of G, then Algorithm 2 is. However in this paper we do not provide such algorithm;instead we mention what approaches we plan to use in order to obtain that result in further works.

Let $G = (U, V, E)$ be a bipartite graph, M be a MMIS of G and Z be a maximum matching of G. As it is mentioned before, the complement of M is a minimum vertex cover; we will denote by N. Also we will denote by L the set of vertices which are adjacent with Z, and by K the set of vertices which are not adjacent with Z (i.e. K is the complement of L). As it is mentioned above, from Claim 5 it follows, that for each MMIS M and for each maximum matching Z it holds$K \subseteq M$ and $N \subseteq L$ (see Figure 2). This means that:

$$\cup K \subseteq \cap M \text{ and } \cup N \subseteq \cap L, \tag{5}$$

where the union and intersection operations are taken trough all MMIS-es and trough all maximum matchings. Next we will show that equality holds in (4). Note, that if $\{A\}$ is a family of subsets, and if the complement of subset A is denoted by B, then $\cap B$ is the complement of $\cup A$. Taking into account this and (5), on Figure 4 we schematically illustrate relations between sets K, L, M and N.

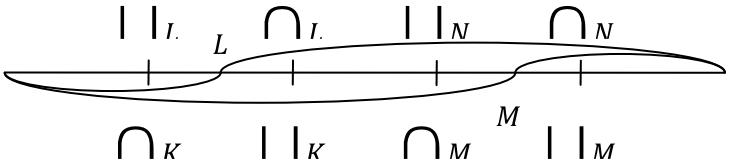


Figure 4

On Figure 4 the horizontal line corresponds to the set of vertices of graph G, sets noted in the same column are the complements of each other, sets noted at the bottom row are listed in increasing order (i.e. the right one is a superset of one on the left) and sets noted at the top row are listed in decreasing order (i.e. the right one is a subset of one on the left). Note that from Claim 8 it follows that $(\cap L) \cap (\cap M) = \emptyset$, so the following duality holds:

$$\cup K = \cap M \text{and } \cup N = \cap L. \tag{6}$$

(4) states that $\vee M = (\cup M_U) \cup (\cap M_V)$, where $\vee M$ is the greatest MMIS of G, so in some sense the duality (6) describes the relation between the greatest MMIS and all maximum matchings of G. Now let G' be a "new"

bipartite graph obtained by adding a new vertex $u'$to G, and $\bigvee M'$ be the greatest MMIS-es of G'. Next we will discuss the problem of providing $\bigvee M'$ based on $\bigvee M$. Analogically with the notations of G, we will denote by$\bigcup M'$ and $\bigcap M'$ respectively the union and the intersection of all MMIS-es of G'**.** We will also denote by K' a set of vertices of G', which are not adjacent with some maximum matching of G', and by $\bigcup K'$ we will denote the union of all such sets. (6) states that $\bigcap M' = \bigcup K'$. As it follows from Claim 7, if u' is independent with $\bigcap M_V$, then $\bigvee M' = \bigvee M \cup \{u\}$, so next we will discuss the case when u' is adjacent with some vertices $D \subseteq \bigcap M_V$. As it is mentioned before, in this case $\bigvee M$ is a MMIS of G', so $\bigcap M'_V \subseteq \bigcap M_V$. We claim that:

Claim 6:for$v \in V$ it holds$v \in \bigcap M'$ if and only if $D \cap (\bigcup_{v \in K} K) \neq \emptyset$.

Indeed, $\bigcap M' = \bigcup K'$, and v belongs to some K' if and only if there is a maximum matching in G' which is not adjacent with v. This holds if and only if there is a matching in G which is not adjacent with v and is adjacent with some vertex in D. It is easy to see, that this proves the claim. From this claim it follows, that for a vertex $v \in \bigcap M_V$ in order to find out whether $v \in \bigcap M'$ or not, we can check if $\bigcup_{v \in K} K$ has a common vertex with D or not. Thus based on Claim 9 we can provide an algorithm which obtains the greatest MMIS of G' based on the greatest MMIS of G, however performing the mentioned check for all $v \in \bigcap M_V$ cannot be performed efficiently, if we just roughly hold the family of subsets $\{K\}$ and generate $\{K'\}$. We believe that the family of subsets $\{K\}$ has some properties based on which $\{K'\}$ can be obtained efficiently and the check whether $\bigcup_{v \in K} K$ has a common vertex with D or not can be performed efficiently as well. In the next section we conclude this paper and mention about further works.

Conclusion and further works

In the second section of this paper we have shown that the family of MMIS-es of a bipartite graph forms a distributive lattice with respect to join and meet operations defined at (2) and (3). This result is not just of theoretical interest, as in applications where it is required to obtain all MMIS-es of a bipartite graph, the join-irreducible elements of the mentioned lattice can be obtained using Algorithm 1 in $O(|E||W|)$ time, and the obtained structure describes the family of all MMIS-es in the sense of Birkhoff's representation theorem. Algorithm 1 also can be used to obtain the greatest MMIS, as well as any MMIS containing the given set of vertices. In the third section of this paper we present Algorithm 2, which obtains the greatest MMIS of bipartite graph G' based on the greatest MMIS of G in $O(|E|)$ time, where G' is a bipartite graph obtained by "adding" a new vertex to bipartite graph G. Next in that section we prove that duality (6) holds between the MMIS-es and maximum matchings of a bipartite graph. We believe that this duality is not just of theoretical interest, and it can yield to a usable algorithm which sequentially handles vertices of a bipartite graph and maintains the greatest MMIS of it. Claim 9 shows how the duality (6) can be used in order to obtain the greatest MMIS of G' based on the greatest MMIS of G, however in this paper we do not provide an efficient technique of implementing the results of that claim. We believe, that such technique can be obtained using some properties of the family of subsets $\{K\}$, which is the family of sets of vertices which are not adjacent to some maximum matching. In further works we plan to obtain such technique.

Bibliography

[Birkhoff, 1948] G. Birkhoff. Lattice Theory: Revised Edition. American Mathematical Society. 1948.

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

[Karp, 1972] R. Karp.Reducibility Among Combinatorial Problems. Complexity of Computer Computations, pp. 85-103. 1972.

[Ford, Fulkerson, 1962] L. Ford, D. Fulkerson. Flows in Networks.Princeton University Press, 113. 1962.

[Hopcroft, Karp, 1973] J. Hopcroft,R. Karp. An n5/2algorithm for maximum matchings in bipartite graphs, SAIM Journal of Computing, 2 (4), pp. 225-231. 1973.

[Johnson, 1988] D. Johnson and M. Yannakanis.On Generating all Maximal Independent sets. Information Processing Letters, 27, pp. 119-123. 1998.

[Kashiwabara, 1992] T. Kashiwabara and S. Masuda, K. Nakajima and T. Fujisawa.Generation of Maximum Independents Sets of a Bipartite Graph and Maximum Cliques of a Circular-Arc Graph. Journal of Algorithms, 13, pp. 161-174. 1992.

Authors' information

Vahagn Minasyan *– Postgraduate student, Yerevan State University, faculty of Informatics and Applied Mathematics, department of Discrete Mathematics and Theoretical Informatics, Armenia, 0025, Yerevan, 1st Alex Manoogian; e-mail: vahagn.minasyan@gmail.com*

INTELLIGENT AGENTS AND PROTOCOLS

Levon H. Aslanyan, David A. Karapetyan

***Abstract**: We study interaction protocols of software agents in an Intelligent Agent Server system, which employs software agents in regard to different applied problems. Models of agent interaction protocols are proposed. The protocols are evaluated for utility, implementation and applicability. The logical level of system is designed and implemented algorithmically. The test application is the intrusion detection problem.*

***Keywords**: saftware agent, secure communication, inteligence.*

***ACM Classification Keywords**: D.2.11 Software Architectures.*

Introduction

The main goal of the research project NetInt (Networked Intelligence) is the design of a distributed application software system to operate in computing networks. The use of software mobile and intelligent agents enables the system to solve a variety of applied problems, such as network management and maintenance, network dynamic optimization and security control. Agents' communication and interaction become a major technical issue of such systems. Agents, which behave autonomously, change their locations. New agents are appearing and others may stop their functioning. Proper communication in this case requires a complete algorithmic model. Similar to this is the known PKI system for security. NetInt is a complex mobile environment which is under the control of a set of servers, where the security reasons are analysed and implemented by means of practical cryptography. The communication system provides functionality, related to data bases (sniffing, log files) and data mining type of analysis and decision support. Typical applications considered are the network management issues and intrusion detection into the systems[1].

NetInt is an extension of SPARTA (Security Policy Adaptation Reinforced Through Agents) system designed within the 5th Framework Programme (FP5) of European Community Framework Programme for Research, Technological Development and Demonstration, 2000-2001.

System Architecture

The system is basically organized as follows: NetInt agent platforms, or more simple Agent servers, are installed on a number of computers binded together to organize a network. The computers on which the servers are installed are called nodes. Agent servers produce agents as well as permit and manage agents' access to system resources and their utilization. Mobile agents travel from one host to another and perform the thread of executions. They may assemble and swap information, analyze it and later interchange the analysis results. In this way they try to synchronize the system parameters, reveal pathologies in the system and try to eliminate them.

NetInt agent environment is a dialogue system, implemented in Java programming language, which supports transferability of software code across any Java Virtual Machine containing operating system.

Fig. 1 outlines NetInt agent-based system implemented in OMG MASIF standard[2].

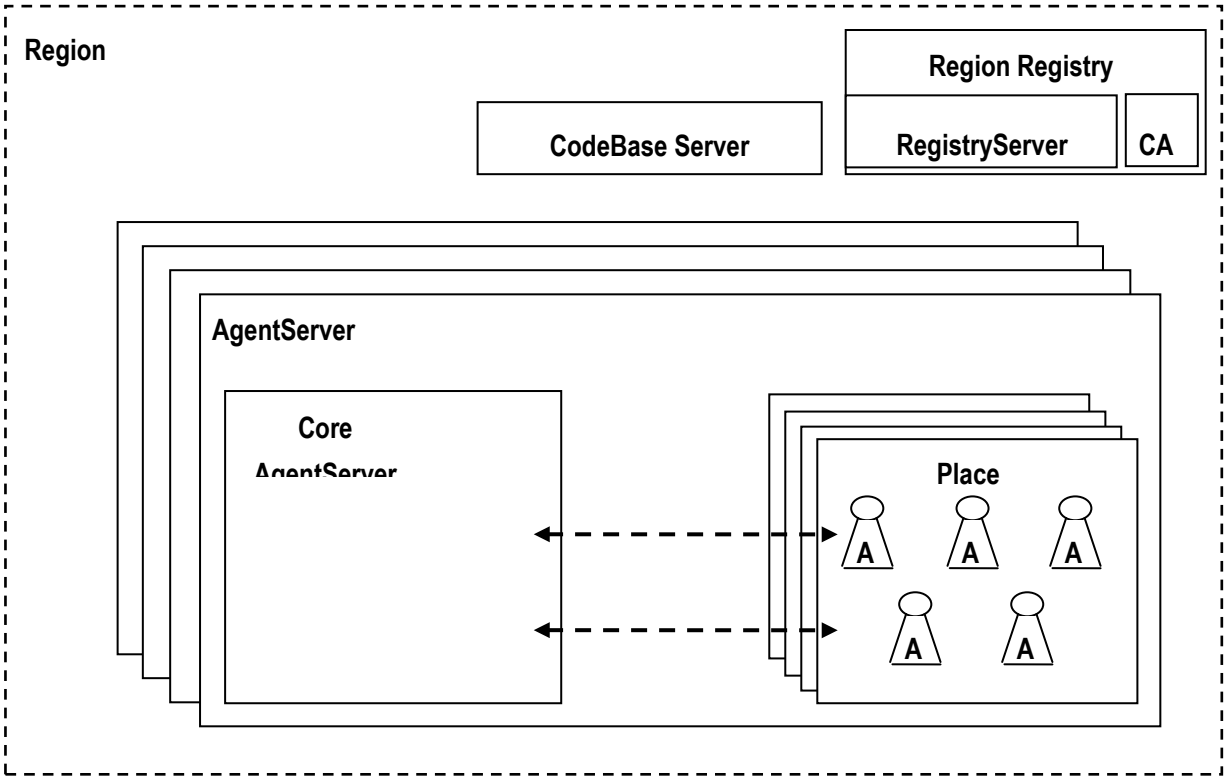


Fig. 1 An outline of functionality of NetInt agent-based system

NetInt agent system involves 3 subsytems:

a. ***AgentServer (AS),*** - the server is basically intended to provide tools to create, run, receive and transfer agents.

b. ***RegistryServer (RS)*** - registration subsystem, where several service delivering subsystems and their components are registered. This subsystem also provides information on necessary agents and agent subsystems. It incorporates internal Certificate Authorities (CA) center as well.

c. ***CodeBaseServer (CBS***) - this server is mainly designed to provide agent software codes. Whenever there is a need to run an agent, the system seeks the agent software code on the local computer. In case the code is not found, the system requires the code from **CodeBaseServer[3].**

A number of different communicators may exist, that support different transfer technologies (e.g. plain socket connections, RMI or Email). Each Home and Agent Server has at least one communicator present, which the agent can use for jumping. The agent itself does not necessarily know about the mode of transport and most of the time will not be informed. When an agent state has been successfully transferred, the agent's code is loaded from the agent's code base. The code base itself consists of all locally available classes and references to available Code Base Servers. When code for an agent is locally available, it is taken from there, otherwise loaded from a CBS. Sending an agent over the network has a number of security implications, which are touched in the following.

To combat threats appearing during the transferring stage, the architecture uses two main mechanisms, namely encryption and code signing. Before the agent's state is sent over the network by the communicator, the system encrypts it. The agent server only accepts digitally signed class files to prevent malicious code from being inserted into the agent system. This prevents the agent platform from running modified code as long as the digital signature is not compromised.

Fig. 2 below gives UML scheme of basic classes that are used in NetInt agent-based system.

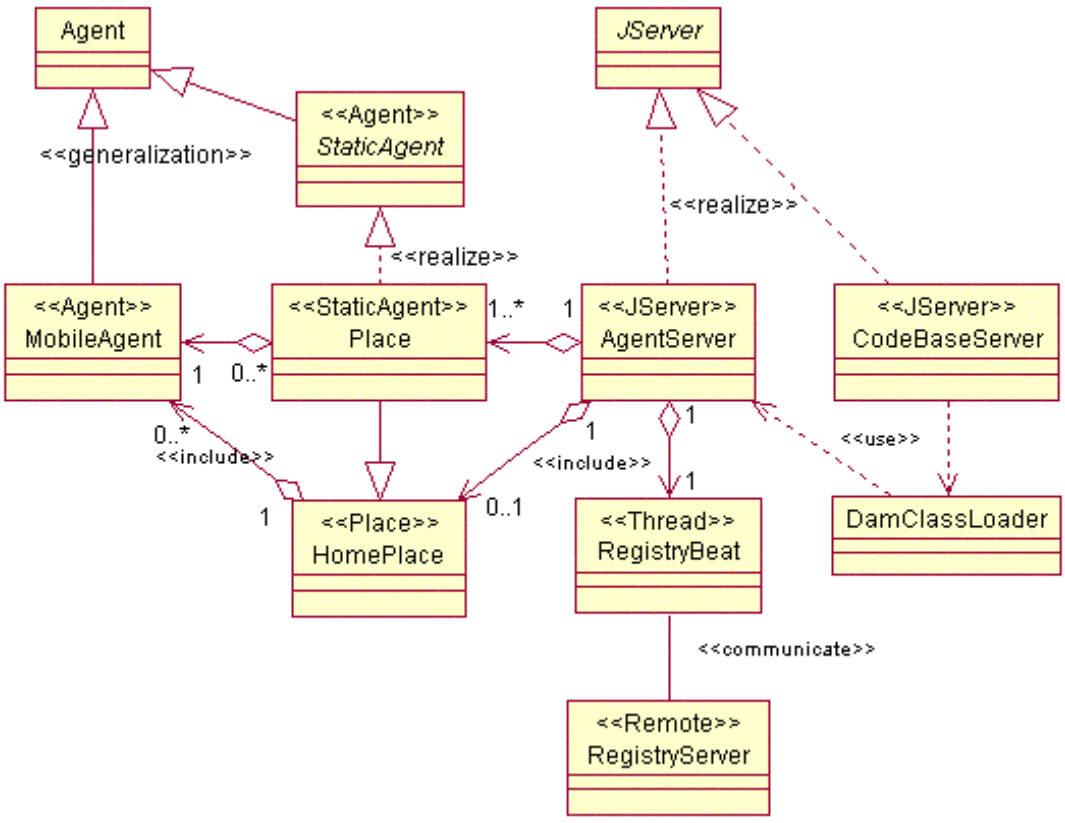


Fig. 2 UML scheme of basic classes used in NetInt agent-based system

Sub-environments in the systems interact via Java RMI communication facility. The latter allows a more effective usage of object-oriented programming (OOP) resources.

Software agents, in accordance with the base concept, should posses the following characteristics to ensure the proper functioning of the whole system:

- **Autonomicy** – agen't ability to act by autonomous, i.e. without meddling of other person or program;
- **Mobility** – agent's ability to travel within the network to search information necessary for task execution;
- **Interoperability** – equal possibilities to interoperate between various software agents;
- **Liability** – the ability of an agent to perform the thread of execution for which the agent is liable for;
- **Flexibility** – agent's ability to act in response to the changes of execution environment

Agent interaction is the major feature that we address when we describe an agent community. Interaction means establishment a form of two-way dynamic communications between two or more agents, trying to reach a mutually acceptable agreement. The term *interaction protocol* is used in reference to sets of rules that guide interactions. In a simple interaction protocol the agents elaborate, accept, or reject proposals.

Agent interaction and coordination in a multi-agent system are based on procedure of exchanging packets among agents. Fig. 3 depicts a pattern of data exchange among agens in an agent-based system:

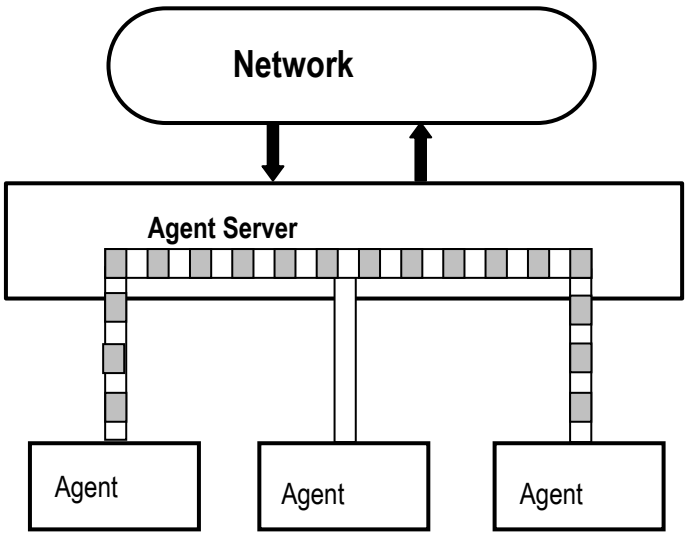


Fig.3 Data exchange diagram in an agent-based system

Data package transferring emerges in situations, when an agent tries to communicate to another agent or to the system. Communication is to be differentiated into the internal (in a local computer and in one Agent Server), local (in a LAN), and global (in a TCP/IP area). In the process of these connections sender determines the address of the counterpart whom it chooses to send a message: it establishes a session with the intended one and sends it a generated packet. In other cases communication needs to apply to the Registry Server.

In the following we address the technical aspects of problems dealing with establishing communications and data exchange between two participants. They can be categorized into that type of problems, which are concerned with achieving secure communications in a medium which is untrustworthy and subject to tampering by potential intruders. These interactions and the consequent need for security, regarding a range of security features and levels, vary widely from application to application. Such issues will be the focus of this study in a context of communication security, which in turn is an issue related to agent community protection.

Within the context of agent-to agent (global) communication, we suppose to obey the general security requirements, including:

- **Confidentiality** – Ensuring that no communication between two parties is revealed to the third party, i.e. that no one can read the message except the intended receiver.
- **Authentication** – The process of proving one's identity. The recipient of a message should be able to securely authenticate its origin; moreover the third party (the intruder) won't be able to act with another name.
- **Integrity** – Assuring the receiver that the received message has not been altered in any way from the original. during transmission, i.e. the intruder would not be able to trap or fake the message.
- **Nonrepudiation** – The sender of message should be unable to deny having sent the message, i.e. the sender should be able to prove that the message is precisely the one issued by it.

Further we shall consider generic public-key encryption algorithms. They basically employ a pair of keys for each participant: one of the keys is designated the *public key* and may be advertised as widely as the owner wants. The other key is designated the *private (secret) key* and is never revealed to another party. It is used to decrypt messages. Suppose participant A wants to send a message to B. Denote A's public key by $pk(A)$ and secret key by $sk(A)$. A message M is encrypted first, by encryption algorithm E that uses the message M and the public key $pk(A)$. The encrypted message (cipher text) $E_{pk(A)}\{M\} = \{M\}_{pk(A)} = C$ is decrypted with the algorithm D using the appropriate secret key $D_{sk(A)}\{C\} = \{C\}_{sk(A)} = M$.

There are several problems in the use of cryptographic systems, such as the problem of key distribution and secure transfer of keys via public net, establishment of intact communication sessions, etc. The crucial role in performing these actions is assigned to keys, however still not a little part has precise organization and application of communication protocols. A *communication /cryptographic/ protocol* is a system of carefully defined actions designed so that it provides interactions between two or more communicating parties according to one or another set of functional /encryption/ requirements. If we refer to cryptographic algorithms as algebraic and logical units that require appropriate theoretical background for proving their security, then cryptographic protocols present to be systems that are subjected to logical analysis to prove their security against malicious actions of the third party. To do these things, we need to make use of appropriate formalisms, such as model checking and

analysis. It is easier to detect possible attacks a priori knowing their scenario and having the protocol secureity proof against the attacks. Security provision of a Protocol requires checking for occurence of all events, which is a time and resource consuming procedure. CSP (communicating sequential processes) and FDR (failures-divergence refinemen) prove to be fine tools to implement such analysis, where the first is an appropriate tool to create formalism, and the second to perform global analysis. CSP is used below in analysis of synthesised protocols, and some known results by FDR are taken into account.

An Outline of Crypto Protocols

We shall now schematically explore several cryptographic protocols:

1. *Simpe communication protocol:*

   Suppose A knows the public key $pk(B)$ of B and wants to communicate a message M to B:
   A -> B : $\{M\}_{pk(B)}$ *(-> means sending (address : message))*
   ($\{M\}_{pk(B)}$ means M encrypted by the key $pk(B)$)

   a. A encryptes message M as $\{M\}_{pk(B)} = C$ and sends it to B.

   b. B decryptes the encryted text C using his private key $sk(B)$.

2. *Public-key distribution protocol*:

   A -> CA : B
   CA -> A : $\{pk(B)\}_{sk(CA)}$

   A is willing to obtain B's public key from Certificate Authorities (CA) center.
   a. A sends B's name to CA's database, which implies that A needs to get B's public key.
   b. CA encryptes $pk(B)$ by its secret key $sk(CA)$, i.e. signs it and sends to A.
   c. A decryptes the message from CA using CA's key $pk(CA)$.

   Several options are available here. B can send A its public key on his own initiative, i.e. B can send the key signed or encrypted, if it knows A's public key.
   For generality we shall also present key pair distribution protocol performed at session set-up. Since at this stage the channel is completely unsecure and the only known fact is CA center's public key, then distribution of the key pair could be performed as follows:

   Suppose A needs to receive an encryption key pair from CA.
   A -> CA : A
   CA -> A : $\{\{pk(A), sk(A)\}_L\}_{sk(CA)}$

a. A requests the key pair from CA.

b. CA generates the public and secret key pair $pk(A)$ $sk(A)$, encrypts it with any key L of length n and sends it to A signed with his secret key $sk(CA)$.

c. A is assumed to know the key L, and recovers its key pair from the encrypted message upon its receipt.

3. *Session set-up protocol:*

A -> B: $\{k\}_{pk(B)}$

Suppose again that A has learnt (in a way) B's public key $pk(B)$:

a. To initate a session A generates a key k at random to be used with a symmetric cryptographic algorithm. Then A encrypts k with $pk(B)$ and sends it to B.

b. B recovers its session key from the encrypted message.

c. Both parties encrypt messages during the session using their symmetric encryption key which is already known to each of them.

Interaction Protocols in Multi-agent Systems

The above-mentioned cryptographic protocols (there are a greatly many such protocols) are exploited in agent-based systems to acheive secure data exchange between agents. They guide and manage every interaction between agents. We propose interaction protocols that provide secure communications between agents, agent servers in a multi-agent system, based on the cryptographic protocols mentioned above. A commentary on each protocol follows:

1. *After loading, AS agent server addresses to registry server RS for registration and to receive its public and secret key pair*. The protocol proceeds as follows.

Suppose that the public key *pk(CA)* of *CA* and its general properties are known and accessible to agent community.

AS -> CA : AS
CA -> AS : $\{\{pk(AS),sk(AS)\}_L\}_{pk(CA)}$

a. AS asks CA for the key pair.

b. CA generates the public and secret key pair $pk(AS)$ $sk(AS)$, encrypts it with the key L of length n and sends it to AS signed with his secret key $sk(CA)$.

c. AS recovers its key pair by decrypting the received message with the keys L and $pk(AC)$.

2. *Associating or binding an encryption key pair* to agent *A* by AgentServer (AS) at agent creation stage.

AS -> CA : A
CA -> AS : {pk(A), sk(A)}$_{pk(AS)}$
AS -> A : {pk(A), sk(A)}

a. *AS* asks *CA* for receiving *A*'s key pair.
b. *CA* generates key pair *pk(A), sk(A)* and encrypes it by using *AS*'s public key, and sends it encrypted back to *AS.*
c. *AS* recovers the key pair from the encrypted message and passes it to agent *A*.

3. *Session set-up protocol between agents A and B*

Suppose *A* is a registered agent in *AS* agent server and is willing to establish session with agent *B*, having no knowledge of its public key and location.

A -> AS : LOC(B)
AS have B (case of internal communication of agents)
AS -> A : {LOC(B)}
A -> B : {k}
B -> A : {k}
SESSION

Else (global communication)
AS -> CA : LOC(B) (at this stage LOC(B) is just a text string as we see it)
CA -> AS :{LOC(B), pk(B)}$_{sk(CA)}$
AS -> A : {LOC(B), pk(B)}
A -> B : {k}$_{pk(B)}$
SESSION
END

a. *A* asks *AS* for B's address.
b. Agent server *AS* checks whether *B* is located in its environment, if yes, then it sends this address to *A*.
c. *A* sends *B* a key k generated at random which is referred to as the session key.
d. B decrypts the message and gets the session key.
e. Both parties encrypt messages using symmetric encryption key constructed at the session set-up, which is already know to each of the parties.

No encryption problem is created in this case, since messages are exchanged within a certain system i.e. massages do not go through the non-secure network, consequently no third party can reveal their secrets

We now consider the case when agent *B* is located on another agent server, denoted as AS_1 .

a. Agent *A* asks *AS* for B's address.

b. *AS* checks agent *B*'s location and if it is not located within its environment, runs public-key distribution protocol to ask registry server (*RS*) for B's public key and address.

c. *RS* creates a massage containing B's address and its public key, signs it with secret key *sk(CA)* and sends to *AS*.

d. *AS* decrypts the message and sends its contents to *A*.

e. *A* generates a key *k* (session key) at random, encrypts it with *pk(B)* and sends it to *B*.

f. The same steps a.b.c.d. are performed by *B* to learn *A*'s address from agent server AS_1.

g. *B* generates its own session key k_1 and sends it together with the key *k*, previously sent by *A*, back to *A* encrypted with *pk(A)*.

h. Upon receipt *A* decrypts the message and recovers the session key k_1, which sends back to *B*.

i. Both parties authenticate each other and encrypt messages using symmetric encryption key pair *k*, k_1 established at the session set-up, which is already known to each of them.

4. *Agent A* ***from agent server AS to agent server AS***$_1$ *transfer protocol*

A -> AS: MOV(AS_1)

AS -> CA : LOC(AS_1)

CA -> AS : {LOC(AS_1), pk(AS_1)}$_{sk(CA)}$

AS -> AS_1 : {k}$_{pk(AS1)}$

AS_1 -> CA : LOC(AS)

CA -> AS_1 : {LOC(AS), pk(AS)}$_{sk(CA)}$

AS_1 -> AS : {k, k_1}$_{pk(AS)}$

AS -> AS_1 : {k_1}$_{pk(AS)}$

SESSION

a. Agent *A* asks *AS* for transfer to AS_1.

b. AS runs public-key distribution protocol to ask registry server (*RS*) for AS_1's address and public key.

c. *RS* creates a massage with AS_1's address and its public key, signs it using secret key *sk(CA)* and sends to *AS*.

d. *AS* decrypts the message, generates a session key *k* at random, encrypts it with AS_1's public key *pk(AS_1)* and sends it to AS_1.

e. AS_1 repeats the steps taken in b.c.d. to request *RS* for *AS*'s address and public key

f. AS_1 generates its own session key k_1 and sends it together with the key *k*, previously sent by *AS*, back to *AS* encrypted with *pk(AS)*.

g. Upon receipt *A* decrypts the message and recovers the session key k_1 which sends to *B*.

h. Both parties authenticate each other and encrypt messages using session symmetric encryption key, which is already known to each of them.

i. Finally *AS* sends agent *A* to AS_1, and upon reception AS_1 registers its new address in *RS*.

Comment: There is a slight weakness in security here. For example an agent, which has been removed and added again, can be removed yet another time by a replay attack. We choose not to do anything about this rather minor problem, which could be solved by including the nonce into the part, which is signed. It is the responsibility of local CA to make sure that no certificate is added more than once. The methods of CA are secure enough, such that they could in principle be remotely callable.

Comment: The methods of protocols have a significant overhead. This is mainly because they involve a high number of cryptographic calculations. This is no big problem since they are very rare events. The methods of the NetInt must however be optimized for speed. Jumping is a special case because it involves a considerable overhead, but a common operation should preferably be in the framework. The jumping mechanisms in NetInt are optimized for security and flexibility. If they should also be optimized for speed, protocols for negotiating shared keys between hosts with much communication should be included. In this way secure highways can be made available on the most used jumping paths. All these optimizations are possible but non-essential and rather complicated to manage, and have therefore been left out.

Comment: Like any other cryptoprotocol algorithms the proposed solutions also require protocol security proving. *CSP* and *FDR* were applied for checking security properties of such algorithms[4]. A check on Session protocol performed by using these tools revealed that not only the session key but also the sender name should be transferred to achieve the necessary level of protocol security.

To test a protocol like this one with FDR we have to build models of well-behaved nodes (Alice and Bob) and an intruder (Cameron) and see how the latter can interfere with the former.

As we have already said, the encryptions and similar will be modelled as symbolic objects: we create an appropriate data type to contain them. It consists of various constants we will need, public-key (PK) and symmetric-key encryption constructions and a sequencing construct.

datatype fact = Sq. Seq(fact) |
PK. (fact , fact) |
Encrypt. (fact, fact) |
Alice | *Bob* | *Cameron* |
Na | *Nb* | *Nc* |
pkA | *pkB* | *pkC* |
skA | *skB* | *skC* |
AtoB | *BtoA* | *Cmessage*

The type fact contains various collections of constants, which can be collected together into sets for later use.
The three identities used are those of two nodes we will later treat as reliable, plus one (Cameron) for the intruder to assume when it acts as another party.

nodes= {Alice, Bob, Cameron}

publickey = {pkA, pkB, pkC}

secretkey = {skA, skB, skC}

nonces = {Na, Nb, Nc}

sessmess = {AtoB, BtoA, Cmessage}

Almost all possible actions for both parties are modeled. For example, both Alice and Bob can either act as an initiator of the protocol (Send) or as the responder (Resp).

User(id,ns) = if ns == <> then STOP else
Send(id,ns) [] Resp(id,ns)

In a similar manner intruder actions are constructed.

Equally, we would expect the intruder to be unable to learn the secrets AtoB and BtoA, since these are never revealed to Cameron deliberately. The spy or intruder can hear whatever passes between Alice and Bob, can interact with them as Cameron, and can intercept and fake messages. Such data is too bulky to be covered here and much more details could be found in [5]

Based on check results appropriate changes have been made in the protocol (not only the session key but also the sender name should be transferred), which makes a good background to assure that parties could engage in an intact communication over a non-secure communications media, without running a risk of intrusion.

NetInt Software System

NetInt system is implemented in Java programming language in WINDOWS operating system environment. Moreover, exploitation of Java programming allows NetInt software system to be executed on any operating system containing Java Virtual Machine. The proposed NetInt agent environment is an automated dialogue system that enables direct communication of the user with the system; consequently availability of an appropriate Interface (see Fig 4) is required.

NetInt agent system management interface includes the following main parts. The left part of the interface presents AgentServer of current node with its agents and places. The main tools that enable management (run/remove) of agents in the system are located at the top. The bottom part of the interface provides data on the agent or place such as agent name, type, its creation time and the name of its creator, run time, current state of the system, user descriptions, etc. The right part of Interface displays RegistyServer of NetInt, i.e. the whole NetInt agent system.

The type *Agent* is used to create and transport agents in NetInt system. All agents in the network are inherited from this type. The basic methods responsible for transportation of agents are *getNextLocation*, *getLocation* and*, move,* as well *run* method which is called during initiation stage.

The *Crypto_Methods* type is implemented for generating and distributing the public and secret keys as well as for encrypting data. The following functions *KeyPairGeneration, GetKeys Message_Encryption, Message_Decryption, RSASignature, Message_Encryption*_and_*Sign, SignatureVerification*, respectively, are used in *Crypto_Methods* to perform these procedures.

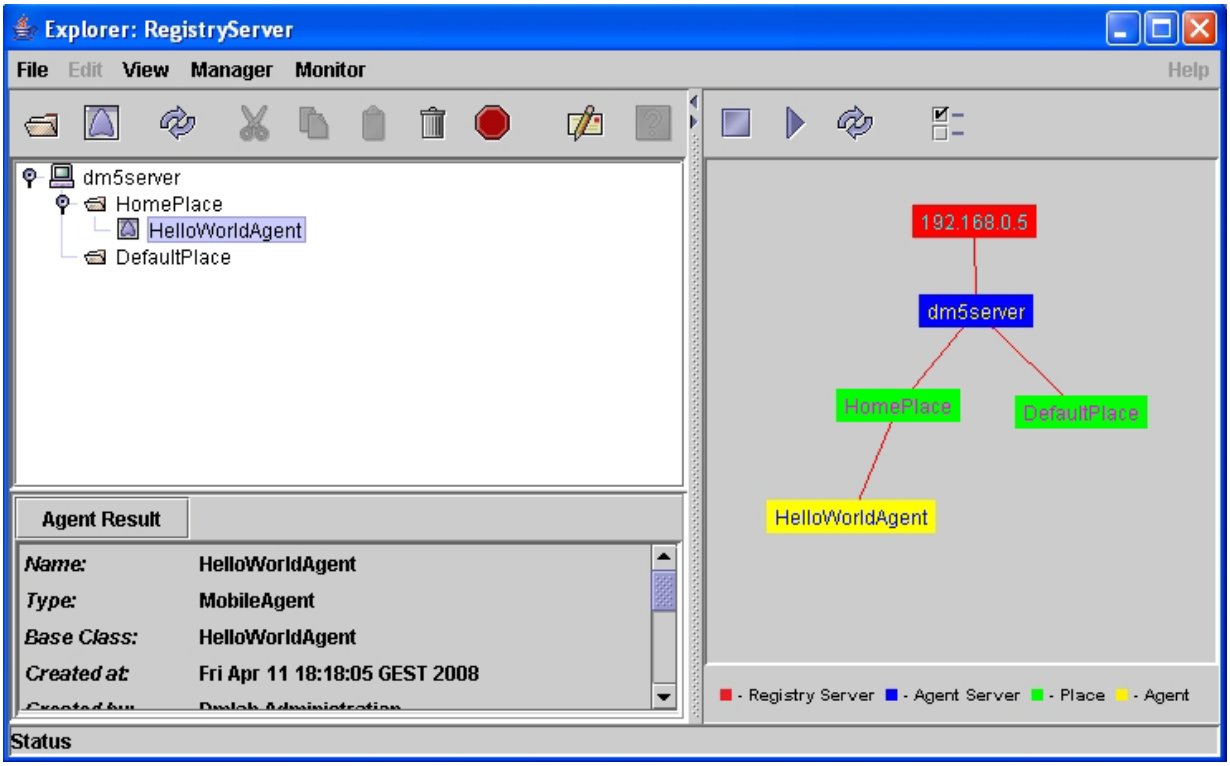


Fig. 4 Interface of NetInt agent system managment

Bibliography

Aslanyan L., Margaryan K., Sahakyan H., Data analysis algorithms in network protection systems, III International Conference on "Digital Information Processing and Control in Extreme Situations", Minsk., May 28-30 2002, ISBN: 985-6453-80-1, pp. 221-225.

D. Milojicic, M. Breugst, MASIF - The OMG Mobile Agent System Interoperability Facility. Mobile Agents - Second International Workshop, MA '98 (Stuttgart, Germany, September 1998).

D. A. Karapetyan, Intelligent Agent Server (NetInt) System, Mathematical Problems of Computer Science 25, pp. 64-70, 2006.

C. A. R. Hoare, Communicating sequential processes, Prentice Hall, 1985.

G. Lowe, Breaking and fixing the Needham-Schroeder public-key protocol using FDR, Proceedings of TACAS '96, Springer LNCS 1055, 1996.

Authors' Information

Levon Aslanyan *– Head of Department, Institute for Informatics and Automation Problems, P.Sevak St. 1, Yerevan 14, Armenia, e-mail: lasl@.sci.am*

David Karapetyan *– Researcher, Institute for Informatics and Automation Problems, P.Sevak St. 1, Yerevan 14, Armenia, e-mail: k_davidam@yahoo.com*

Table of contents of IJ ITA Vol. 17, No.: 2