Design of Virtual Local Area Network Scheme Based on Genetic Optimization and Visual Analysis*

Igor Saenko[1] and Igor Kotenko[1,2†]
[1]*Laboratory of Computer Security Problems*
St.Petersburg Institute for Informatics and Automation (SPIIRAS),
39, 14 Liniya, St.Petersburg, 199178, Russia
{ibsaen, ivcote}@comsec.spb.ru
[2]*St.Petersburg National Research University of Information Technologies, Mechanics and Optics,*
49, Kronverkskiy prospekt, St.Petersburg, Russia

Abstract

The paper considers an approach to genetic optimization of Virtual Local Area Network (VLAN) scheme using the developed software — VLAN scheme design tool. Authors suggest a formal statement of the problem of VLAN scheme optimization, which solution can improve the reliability and security of operation of corporate computer networks. The paper shows that the problem considered is related to one of the forms of Boolean Matrix Factorization. A number of improvements were implemented in the proposed genetic algorithm, concerning the formation of initial population, kind of fitness function, coding chromosomes, and operation of crossing and mutation. The VLAN scheme design tool allows to solve the problem by genetic optimization, forms a visual representation of the progress of solving the problem and provides an estimation of the genetic algorithm. Experimental results show the proposed genetic algorithm has high effectiveness.

Keywords: VLAN, VLAN design, Boolean Matrix Factorization, genetic algorithms, visualization.

1 Introduction

Sustainable, reliable and secure exchange of data between computers in local area computer networks is a very important research and development problem. To solve this problem new tools and methods are regularly being developed and installed. Many different factors can be attributed among the possible threats that have considerable impact on the quality of information exchange in local area networks. One of the most dangerous factors is the unauthorized actions of internal users (insiders) aimed to access adjacent computer nodes. These threats may lead not only to leakage of confidential information on individual nodes, but also to the failure of the entire computer network, for example, through Denial of Service attacks. Known means of access control used in local area networks are not always able to prevent such threats. An additional level of protection for local area networks, which can provide strict regulation of data flows between computers (under the condition that its implementation does not result in a large expenses for performance and cost) is desirable. And it is essential in some cases, for example, in critical infrastructures.

One of possible ways to create such level of protection in local area networks is the Virtual Local Area Network (VLAN) technology [2, 3]. It allows to divide a network into fragments of logically related computers, called *virtual computer networks* or, for simplicity, virtual subnets. The scheme

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, volume: 5, number: 4, pp. 86-102

*This paper is an extended version of the work originally presented at the 8th International Symposium on Intelligent Distributed Computing (IDC'14), Madrid, Spain, September 2014 [1].

†Corresponding author: Tel: +7(812) 328-71-81, Web: `http://www.comsec.spb.ru/`

determining distribution of computers on virtual subnets is called as *VLAN scheme*. In the local area network implementing VLAN technology, data exchange between two computers is possible if and only if they belong to the same virtual subnet. The number of computers in the same virtual subnet is not limited. In an extreme case, all computers in the local area network can be in the same virtual subnet. But then there is no need to use VLAN, as data exchange is allowed between each pair of computers.

Besides VLAN, there are other known approaches to delimitate the data exchange on the network level. In particular, access lists can be used. However, this approach can be considered as less flexible than VLAN, as if there is a need to change the access scheme, the network administrator must take appropriate actions on remote workstations that may not be possible. In contrast to this approach, all actions for configuring the VLAN scheme are carried out centrally, because it is stored in one place, for example, on an Ethernet switch or a router.

Despite the undoubted merits of using virtual subnets to improve sustainability of information flows in local area networks, the problem of VLAN scheme design is insufficiently discussed in the scientific literature. Mainly empirical recommendations are used now to design VLAN schemes. According to these recommendations the network should contain virtual subnets according to the functional subsystems selected in the organization. The virtual subnets are usually formed around various application servers and computers that have the right to work with these servers. However, this approach makes it impossible to exchange data between computers with different servers. This is largely due to the lack of formal statements of the VLAN optimization problem and special VLAN design tools.

One of the goals of our research is to develop a new approach to the VLAN scheme design, which is based on the formal statement of the optimization problem and its subsequent decision. As the initial data, we use the matrix of required connectivity of computers in the network. Boolean matrix of belonging computers to virtual subnets acts as a problem variable. Criterion for searching a solution is the requirement to minimize the number of virtual subnets with the full implementation of the required connectivity matrix. Analysis of relevant works shows that this problem can be assigned to one of the varieties of Boolean Matrix Factorization (BMF) problems. For this reason, it is NP complete problem and requires heuristics solution.

It is known that one of the very effective and universal methods of solving such problems is genetic algorithms. They are most suited to solve NP complete optimization problems with Boolean variables. Some works are known, in which genetic algorithms were used for BMF. However, the direct application of these methods to optimize the VLAN scheme is impossible, and development of some improvements in genetic algorithms is necessary. Getting them is one of the objectives of the paper.

Genetic algorithms are customizable methods for solving optimization problems. Their effectiveness depends on the parameters used, for example, population size, crossover and mutation probability, etc. In order to determine how effective the application of the genetic algorithm from parameters in the VLAN scheme design, the special software tool, called VLAN Scheme Design Tool (VSDT), was developed. This tool not only finds a solution to the problem, but also has advanced facilities for visualization that help identify important data dependencies.

The main theoretical contribution of the paper can be defined as a further research and developments in two areas: virtual networks and genetic optimization. In the fist area we formulate the problem of virtual subnets design and show that it is one of the BMF forms. In the second area we determine a number of improvements, necessary to implement in the genetic algorithm the criterion of minimizing the number of virtual subnets under the conditions of achieving the desired logical connectivity matrix. This paper is an extended version of the paper presented on IDC'2014 [1]. The difference of the paper from the conference paper [1] is in two main improvements — theoretical and practical. We consider in more details properties of trivial solutions and the formalization of the optimization criterion, besides a new technique to perform the crossing operation (as a two dimensional crossover) is proposed. We also outline more deeply the issues of the VSDT architecture and user interfaces. The evaluation of

experiments and discussion is expanded.

The paper is organized as follows. Section 2 presents an overview of related work. Section 3 reveals the necessary mathematical foundations. Section 4 discusses the genetic algorithm and its improvement. Section 5 outlines the developed tool and the results of experiments. Section 6 contains the conclusions and directions for further research.

2 Related work

Analysis of relevant works primarily address the results in the area of decomposition of Boolean matrices. These results are necessary to better understand the scale of complexity of the mathematical problem which is being solved. Among these works we highlight the papers written by P. Miettinen et al. [4, 5, 6] as the most interesting. They propose the mathematical methods for solving some variants of the BMF problems. These works examine the BMF problems as a special case of the Non-Negative Matrix Factorization (NMF) problems and prove that they all belong to the class of NP complete problems.

Another group of works (H. Lu et al. [7, 8]) indicates that some problems in computer security can be described using the BMF problem statement. These problems include, for example, the task to generate the role-based access control scheme, known as the Role Mining Problem (RMP).

This idea is outlined and implemented in our works [9, 10]. Here the RMP problem is formalized by a mathematical problem statement to find two binary matrices "users-roles" and "role-resources". The multiplication of these matrices gives the required matrix "users-resources". The number of roles in the problem should be minimal. To solve the problem an approach is proposed that is based on improved genetic algorithms. The main innovations are multi-chromosomal coding for solutions [9] and implementation of an additional stage for data preprocessing [10]. Multi-chromosomal coding is stipulated by the fact that the chromosomes, which encode the binary matrices, have various lengths. Therefore the third chromosome is introduced, which manages the length of the chromosomes.

However, the use of genetic algorithms developed to solve the RMPs is impossible for solving the BMF problems, as the solution of the BMF problems is to find not two, but only one Boolean matrix. Unlike the case of the RMPs for the problem, discussed in the paper, there is no need for both multi-chromosomal coding and data pre-processing in the genetic algorithm. In our case, as it will be shown below, all individuals in the genetic algorithm, discussed in the paper, have chromosomes of the same length, because so-called trivial solutions are used to generate the initial population of the individuals.

Among the works that offer different heuristics to solve the BMF problems, we highlight the paper of A. Janecek et al. [11]. This work proposes bio-inspired optimization techniques to solve this class of problems. This paper discusses a set of different techniques, including genetic algorithms, particle swarm optimization, differential evolution, etc. However, as the comparison of all investigated algorithms shows, the genetic algorithms demonstrate the highest speed and accuracy in solving the BMF problems.

V. Snasel et al. [12, 13] also propose to use genetic algorithms to solve the BMF problems. They suggest to use the columns of one of the binary matrices, that is determined in the BMF problem, as the genes of chromosomes. The Euclidean distance between the searched and required matrices is applied a the goal function. However, an algorithm, that is considered in the work, helps you find two different binary matrices. Therefore, it cannot be used for solving the task of virtual subnets design.

Several papers propose using data mining methods to create virtual networks. For example, the methods of cluster analysis intended for mobile ad hoc networks of small and middle dimensions are investigated in [14]. For large dimension, this approach is less efficient than genetic algorithms. In [14] genetic algorithms were proposed to optimize the access control schemes in local area networks. Here the authors use both folder sharing and virtual subnet creation mechanisms. The matrix of computers

logical connectivity is considered in [14] as the solution of the problem. The current paper considers this matrix as source data. For this reason, the genetic algorithm, developed in [14] cannot be used to design virtual subnets.

Thus, the analysis of relevant works shows, firstly, that genetic algorithms should be considered as sufficiently effective methods to solve the problem of virtual subnets optimization, and, secondly, the known genetic algorithms require further improvement.

3 Mathematical background

In this section we consider the mathematical statement of the problem. Suppose that there are n computers in a network. The diagram of allowed flows between these computers is defined by Boolean matrix $\mathbf{A}[n,n]$. If $a_{ij} = 1(i, j = 1,...,n)$ then the exchange between computers i and j is allowed. Otherwise, this exchange is not possible.

Suppose that we have formed k virtual subnets in the network. Each of these subnets combines two or more computers. Let us set the distribution of computers on subnets by using the matrix $\mathbf{X}[n,k]$. If $x_{ij} = 1(j = 1,\ldots,k)$ then the computer i belongs to the subnet j. Otherwise subnet j does not include computer i.

The matrix $\mathbf{A}$ plays a role of initial data. The matrix $\mathbf{X}$ is the result of solving the problem. We will show that between Boolean matrices $\mathbf{A}$ and $\mathbf{X}$ there is the following relationship:

$$\mathbf{A} = \mathbf{X} \otimes \mathbf{X}^{\top}, \tag{1}$$

where $\mathbf{X}^{\top}$ is a transposed matrix $\mathbf{X}$, symbol $\otimes$ stands for Boolean matrix multiplication, which is a form of matrix multiplication based on the rules of Boolean algebra. Boolean matrix multiplication allows getting the elements of matrix $\mathbf{A}$ by the following expression: $a_{ij} = \vee_{j=1}^{n}(x_{ij} \wedge x_{ji})$.

Consider the following example. Let $n = 5, k = 3$. There are virtual subnets in the computer network as shown in Figure 1. Subnet 1 includes workstations WS1, WS2 and WS4, subnet 2 — WS2, WS3 and WS5, subnet 3 — WS1, WS4 and WS5.

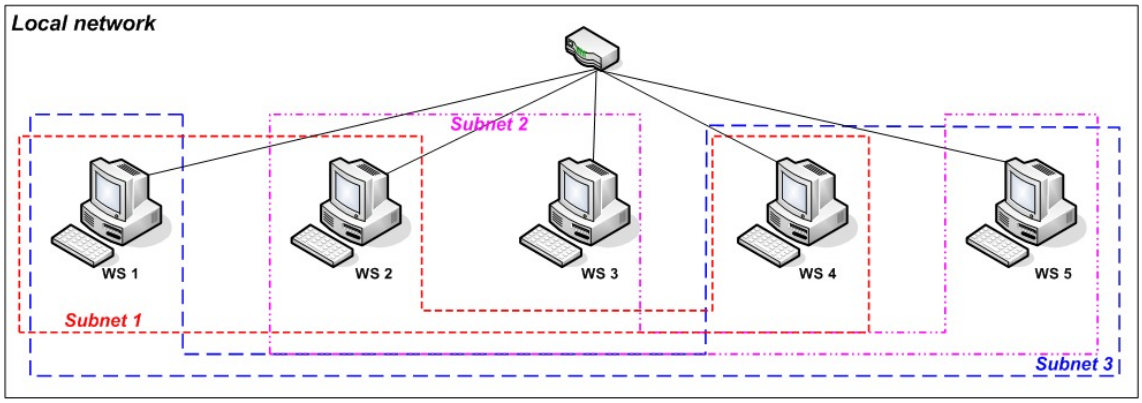
Figure 1 also shows how to formally represent the distribution workstations over virtual subnets, using Boolean matrix "hosts — subnets".

In addition, the figure shows what will be the logical connectivity of computers, if virtual subnets are organized according to the scheme that is displayed by the matrix "hosts — subnets". This information in the formal form is represented in the matrix "hosts — hosts". This matrix shows that only WS2 and WS5 are able to communicate with all computers in the network. For WS2 it is because merging subsets of computers in subnets, where WS2 belongs (subnets 1 and 2), provides a full set of computers in the local network. WS5 is in a similar situation but with subnets 2 and 3.

Easy to see that the matrix "hosts — subnets", represented in Figure 1, plays the role of the matrix $\mathbf{X}$, and the matrix "hosts — hosts" is a matrix $\mathbf{A}$.

The relationships between the matrices $\mathbf{A}$ and $\mathbf{X}$ are as follows:

$$\mathbf{X} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \mathbf{X}^{\top} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix},$$



Subnets

Hosts	1	2	3
1	1	0	1
2	1	1	0
3	0	1	0
4	1	0	1
5	0	1	1

Matrix "hosts - subnets"

Hosts

Hosts	1	2	3	4	5
1	1	1	0	1	1
2	1	1	1	1	1
3	0	1	1	0	1
4	1	1	0	1	1
5	1	1	1	1	1

Matrix "hosts - hosts"

Figure 1: Distribution of workstations over virtual subnets (example)

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

It is easy to see that the matrix $\mathbf{A}$ is symmetric.

Now we will show that this problem is a kind of BMF problems. The BMF problem is to find Boolean matrices $\mathbf{W}$ and $\mathbf{H}$ that are related with given Boolean matrix $\mathbf{A}$ by the following equation:

$$\mathbf{A} = \mathbf{W} \otimes \mathbf{H}, \tag{2}$$

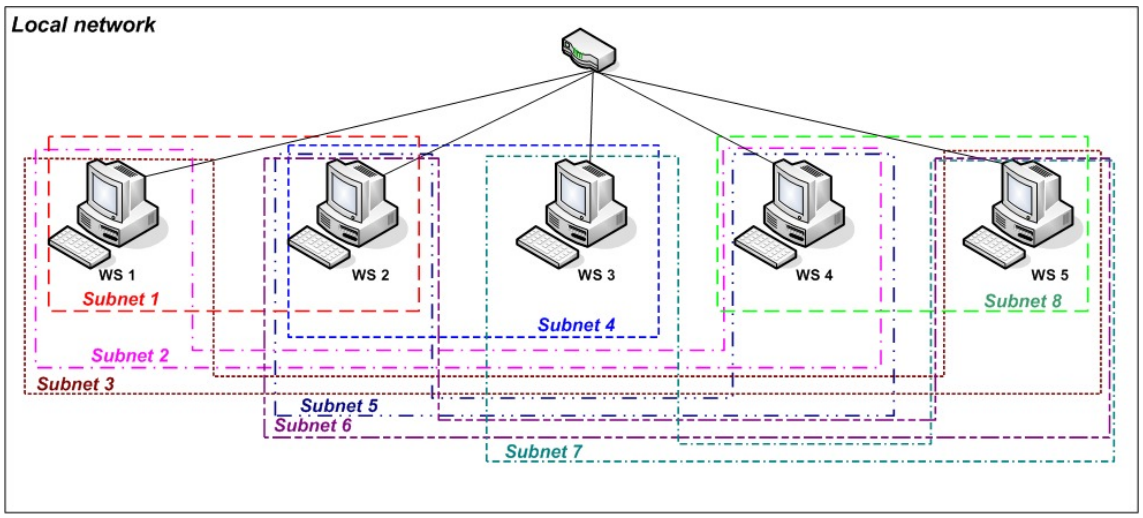
where $\mathbf{A} = \mathbf{A}[n,m], \mathbf{W} = \mathbf{W}[n,k]$ and $\mathbf{H} = \mathbf{H}[k,m]$.

Comparing (1) and (2), we can notice that the equation (1) can be seen as a special case of the equation (2) when the next two conditions are met. The first condition is the equality $m = n$. The second condition takes the following form:

$$w_{ij} = h_{ji} \text{ for any } i = 1,\ldots.n \text{ and } j = 1,\ldots,k. \tag{3}$$

From the fact that the problem discussed is a variant of BMF problems, it follows that the problem is NP-complete. However, because of conditions (3), the direct application of the BMF methods for solving the problem is difficult. Hence our proposal is to solve this problem by heuristic methods, namely, genetic algorithms.

However, for this purpose, the optimization criterion should be defined. Let us pay attention to the fact that in the task (1) the value k can be arbitrary. Now we show how we can restrict it from above. For this purpose we will split the network on subnets, where each subnet includes only two computers i and j when $a_{ij} = 1$. Such distribution of workstations over virtual subnets we will call as binary.



Subnets

Hosts	1	2	3	4	5	6	7	8
1	1	1	1	0	0	0	0	0
2	1	0	0	1	1	1	0	0
3	0	0	0	1	0	0	1	0
4	0	1	0	0	1	0	0	1
5	0	0	1	0	0	1	1	1

Matrix "hosts - subnets"

Hosts

Hosts	1	2	3	4	5
1	1	1	0	1	1
2	1	1	1	1	1
3	0	1	1	0	1
4	1	1	0	1	1
5	1	1	1	1	1

Matrix "hosts - hosts"

Figure 2: Example of binary distribution of workstations over virtual subnets

For the example above, the binary distribution of workstations over virtual subnets looks as shown in Figure 2. In this case the matrix $\mathbf{X}$ has five rows and eight columns in accordance with the matrix "hosts — subnets", presented in this figure.

The matrix $\mathbf{X}^\top$, respectively, has eight rows and five columns. However, Boolean multiplication of these matrices still results in a matrix $\mathbf{A}$ (it is not hard to check). The order of the columns can be arbitrary. Matrix $\mathbf{A}$ corresponds to the matrix "hosts — hosts" presented in Figure 2.

We will call such a matrix $\mathbf{X}$ as the *trivial solution*. In other words, the trivial solution corresponds to the distribution of workstations, which is shown in Figure 2.

It is easy to see that a trivial solution has the following properties:

- the number of columns in the matrix $\mathbf{X}$ is equal to the number of '1'-values in the matrix $\mathbf{A}$ located above the main diagonal (we denote this value as M);
- there is just two '1'-values in each column of the matrix $\mathbf{X}$;
- the number of trivial solutions is equal to the number P of permutations of columns in the matrix $\mathbf{X}$ (it is obvious that $P = 2^M$).

Analyzing the above properties, we can make the following conclusions:

- trivial solutions cannot be considered as good. This is due to two aspects. First, the number of virtual subnets in a trivial solution is very large and is equal to M. Secondly, the number of trivial decisions is very large and is equal to $P = 2^M$;

- the number of columns in the matrix $\mathbf{X}$ in the course of solving the problem (1) can be limited atop to the value M;

- we see that for the example shown in Figure 1 there is a solution, where the number of virtual subnets smaller than M (here we have $k = 3$). If the matrix $\mathbf{X}$ has M columns, then $k = 3$ corresponds to the number of nonzero columns in this matrix;

- it is believed that the smaller k, the better solution of the problem. The smaller k, the less the complexity of administrative work and higher the network availability.

Therefore, we can propose the *optimization criterion* of the problem as the *minimum value of k under full coincidence* of matrices $\mathbf{A}$ and $\mathbf{X} \otimes \mathbf{X}^{\top}$.

Formally, this criterion can be represented as follows:

$$\begin{cases} k \to \min, \\ k = 1, \ldots, M; M = \left| \{ a_{ij} | a_{ij} = 1, i > j \} \right|, \\ \mathbf{X}[n,k] \otimes \mathbf{X}[n,k]^{\top} = \mathbf{A}[n,n]; A = ||a_{ij}||. \end{cases}$$

4 Genetic algorithm

At present, genetic algorithms as a method of solving various optimization problems are developed well enough. There are many works that address various schemes to build genetic algorithms [15, 16, 17]. In the paper we focus on the following steps in the genetic algorithm:

1. *Determining*, which shows why one solution better than the other solution.

2. *Encoding possible solutions of the problem.* According to the terminology of genetic algorithms the encoded solution is called *individual*. Every solution is encoded using a string consisting of characters or numbers. This string is called *chromosome*. One element in the chromosome is called a *gene*. A set of individuals is called *population*.

3. *Forming the initial population.* A genetic algorithm starts with the creation of an initial population. As rule, this process takes place in a random order, but the number N of individuals in a population is permanent. In this step, the evaluation of all individuals in the population is performed by using a fitness function, and individuals are sorted in descending order by the value of the fitness function.

4. The operation of *crossing* is performed. During this operation the pairs of individuals, which play the role of *parents*, are selected randomly from the population. Then their chromosomes are divided into two parts in the same point. The new individuals, called *descendants*, are formed through the exchange of parts of parent's chromosomes.

5. The *mutation* is performed. During the mutation some individuals are randomly selected from the population, and then they change their genes.

6. The *selection* is performed. During the selection, from the population, which is added with the descendant and mutated individuals, the "worst" individuals are deleted that have the lowest values of the fitness function. The population size remains the same.

7. The algorithm terminates if the iteration limit is reached or any individual receives the highest value for its fitness function. In this case, the individual with the highest fitness function is considered as an optimization problem solution. Otherwise, go to step 3.

In order to use the genetic algorithm to solve the problem discussed, it is necessary to determine the fitness function and coding decisions.

In the known papers on the application of genetic algorithms for solving the BMF problem, the fitness

function was based on the Euclidean distance between the matrices $\mathbf{A}$ and $\mathbf{W} \otimes \mathbf{H}$:

$$F = \left(\sqrt{\sum_i \sum_j (a_{ij} - w_{ij} h_{ji})^2} \right)^{-1},$$

Under full matching of these matrices the Euclidean distance is equal 0, and the fitness function has the maximum value.

In our case only one Euclidean distance between $\mathbf{A}$ and $\mathbf{X} \otimes \mathbf{X}^\top$, which is calculated as $\sqrt{\sum_i \sum_j (a_{ij} - x_{ij} x_{ji})^2}$, is not sufficient. The trivial solutions may be obtained from this one criterion. We must also take into account the requirement that in the matrix $\mathbf{X}$ the number of columns k should be minimal. As a result, we propose the following type of fitness functions:

$$F = \left(\alpha k + \beta \sqrt{\sum_i \sum_j (a_{ij} - x_{ij} x_{ji})^2} \right)^{-1}, \tag{4}$$

where α and β are the weights that determine the direction of the solution search. The condition $\alpha \ll \beta$ between these coefficients (for example, $\alpha = 1$ and $\beta = 10$) guarantees that first of all we want to look for solutions which matrices $\mathbf{A}$ and $\mathbf{X} \otimes \mathbf{X}^\top$ are the same, then we will look for solutions with smaller values of k. The opposite condition $\alpha \gg \beta$ ensures primarily search of solutions, which firstly reduce the value of k, and then supplies the equality of matrices $\mathbf{A}$ and $\mathbf{X} \otimes \mathbf{X}^\top$.

Now, let us consider the order to create chromosomes. To do this, we propose to use as genes of chromosomes the vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})$. The vector $\mathbf{x}_i$ plays the role of a column of the matrix $\mathbf{X}$. Let us set the number of genes in the chromosome equal M. As it was mention above, the number M specifies the number of columns in a trivial solution. More columns make no sense.

Finally, in order to increase the convergence of the genetic algorithm we will make another proposal, which deals with the formation of the initial population. We will form a half of the initial population from possible trivial decisions. Thus the individuals with a high value of the fitness function value $F = 1/k$ will be artificially included into the initial population. If we get $P \geq N/2$, then a half of the initial population will consist of trivial solutions. Otherwise P individuals are trivial solutions, and the other $(N \breve{} P)$ individuals will be randomly generated.

Operations of crossing and mutation will be performed in a standard way. However, a column of the matrix $\mathbf{X}$ is used as a gene of the chromosome. This imposes certain peculiarities of the genetic algorithm. In particular, for the crossing operation we propose to undertake the division of parental chromosomes not in the one-dimensional, but in *two-dimensional* mode. The idea of the two-dimensional mode for crossing is illustrated in Figure 3 for two trivial solutions.

Two-dimensional crossing increases the likelihood of the emergence of new '1'- or '0'-elements in the genes of child's chromosomes. Therefore, the speed of the genetic algorithm is improved.

Another feature of the crossing is as follows. When it is executed, descendants with the same genes can be generated. Such is possible, for example, when crossing two trivial solutions that are in the initial population. This means that the same subnet is displayed twice in the matrix $\mathbf{X}$ of such descendant. In this case, one of the columns must be zero. As a result, the number of nonzero columns that corresponds to the total number of subnets is decreased by one.

The mutation is performed in two stages. During the first phase the genes (the columns of the matrix $\mathbf{X}$) are determined which have to be modified. The genes are selected with a probability W_{gen}. Then the value of the corresponding element in the selected column of the matrix $\mathbf{X}$ is inverted with the probability W_{el}.

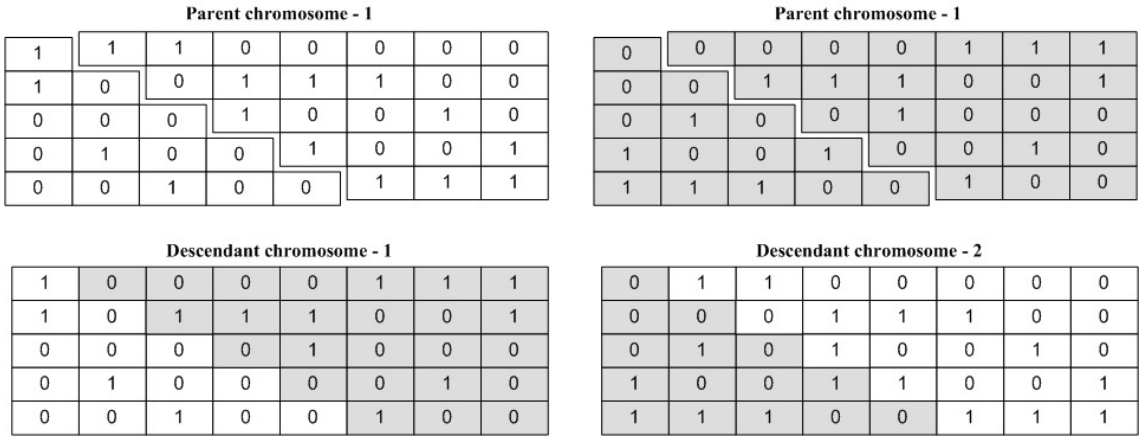


Figure 3: Example of two-dimensional crossing

5 VLAN Scheme Design tool and experimental results

5.1 VLAN Scheme Design Tool architecture

In order to solve the task of VLAN scheme design based on the genetic algorithm, as well as to estimate the speed and accuracy of the genetic algorithm, the VLAN Scheme Design Tool (VSDT) was developed. The VSDT architecture is shown in Figure 4.

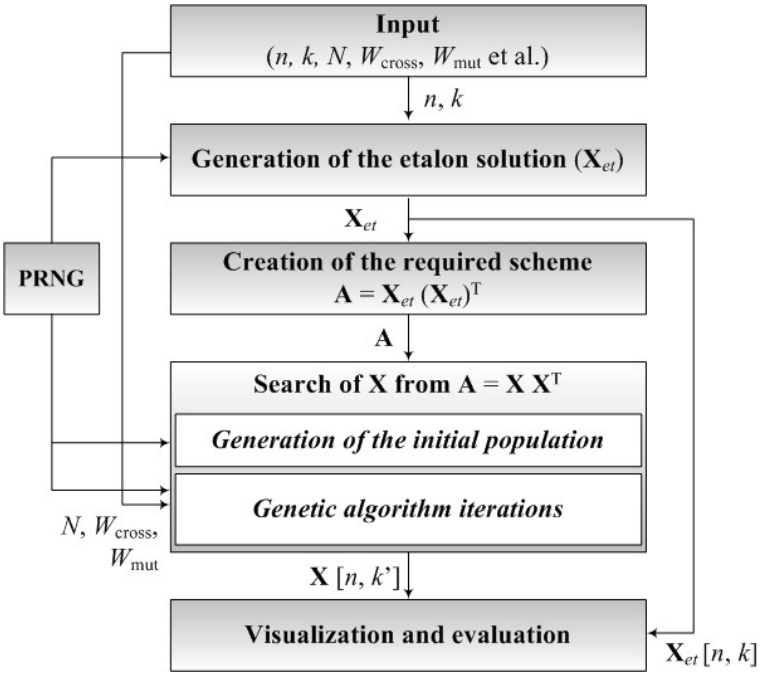


Figure 4: VLAN Scheme Design Tool architecture

The VSDT includes the following modules:

- *Input of initial data,* which enters the total number of computers in the local network (n), the genetic algorithm parameters (population size (N), probability of crossing (W_{cross}), probability of mutation (W_{mut}), etc.), as well as the number of virtual subnets in the optimal access scheme (k);

- *Generation of the etalon solution* ($\mathbf{X}_{et}$), which is used to generate the matrix $\mathbf{A}$ and assess the accuracy of the solution;
- *Generation of the required scheme,* which forms the required logical connectivity matrix $A = \mathbf{X}_{et} \otimes \mathbf{X}_{et}^{\top}$;
- *Search of a solution* of the VLAN scheme optimization problem, which is the solution of the equation $\mathbf{A} = \mathbf{X} \otimes \mathbf{X}^{\top}$ generated by using the proposed genetic algorithm;
- *Visualization and evaluation,* which visualizes the progress of finding a solution and evaluates the genetic algorithm, based on the comparison of $\mathbf{X}$ and $\mathbf{X}_{et}$.

Pseudorandom number generator (PRNG) is also included in VSDT [18]. It is used to generate the etalon solution and to run the genetic algorithm.

The initial data is entered via the VSDT dialog forms. The general view of these forms is shown in Figure 5. The values of the elements of the matrix $\mathbf{A}$, outlined on this figure, correspond to the example depicted in Figure 1.

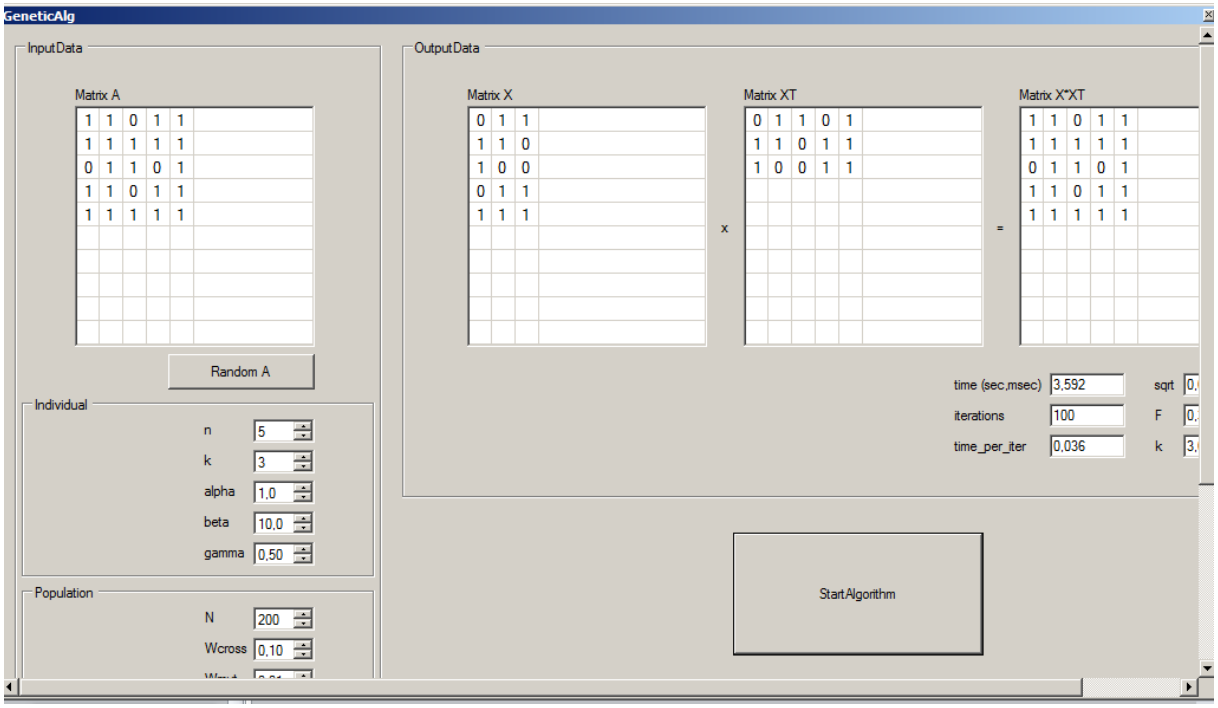


Figure 5: Dialog forms

The initial data is represented by the following parameters:

- *parameters that characterize the individuals in the population* (combined into the block 'Individual'). These parameters include n, k, α ('alpha'), β('beta'), the probability of the value '1' in the matrix $\mathbf{A}$, designated as 'gamma'. The value of gamma 0.5 corresponds to an equiprobable distribution. However, in this case, if $n > 20$, almost all elements of the matrix $\mathbf{A}$ are equal to '1'. To increase the quantity of zero elements in $\mathbf{A}$, the gamma should be less than 0.5;
- *parameters that describe the population* (combined into the block 'Population'). They include the population size N, the probabilities W_cross and W_mut, as well as the number of iterations T.

Formation of the matrix **A** occurs by typing the bottom 'Random A'. The result is displayed in the window 'Matrix A'.

When you click the button 'Start Algorithm', the tool starts to search solutions. As shown in Figure 4, this module includes two blocks — *Generation of the initial population* and *Execution of the genetic algorithm iterations*. Population size in the algorithm is constant and equal to N (by default $N = 200$). One half of the initial population is filled with trivial solutions (an example is shown in Figure 2). The other half is filled with solutions obtained using the PRNG.

Individuals are selected from populations for crossing operations with the probability W_{cross} (by default $W_{cross} = 0.1$). Individuals are selected for mutations with the probability W_{mut} (by default $W_{mut} = 0.01$). The default values of N, W_{cross} and W_{mut} are taken from [12]. However, they can be changed in the module *"Input of initial data"*. The weights used to compute the fitness function also can be changed. By default, these coefficients have the following values: $\alpha = 1, \beta = 10$.

Completion of iterations (and termination of the algorithm) occurs when the limit number of iteration T is reached (by default $T = 1000$). The iteration number, at which the optimal solution is obtained, is determined by visual analysis. This analysis is carried out by using graphs of changing the values of the fitness function and its components, which are formed in the module *"Visualization and evaluation"*.

After completion of the genetic algorithm the matrix $X[n,k']$ is determined. In the VSDT dialog forms this solution appears in the field "Output Data". In addition, this field displays the following values:

- the total time of the algorithm running (time);
- the average time per iteration (time_per_iter);
- the values of the fitness functions (F) and its components — the number of nonzero columns (k) in the solution and Euclidean distance (sqrt).

The behavior of the fitness function and its components during the genetic algorithm running (for the eight best solutions) appears in the form of graphics of dependences on the number of iterations. These graphics are formed in the module *"Visualization and evaluation"*.

An example of visualization of the problem solving progress is represented in Figure 6. In this example $n = 25$ and $k = 5$. As you can see from the figure, the optimal solution to the problem was found on the 220-th iteration. The fitness function reached the maximum value 0.2 and the Euclidean distance was equal to 0.

In general case, after genetic algorithm was terminated, the number of columns k' in the matrix $\mathbf{X}$ is not equal to the value of k in the etalon solution $\mathbf{X}_{et}$. Comparison of $\mathbf{X}$ and $\mathbf{X}_{et}$, fulfilled in the module *"Visualization and evaluation"*, allows to assess the accuracy of the solution using the genetic algorithm.

5.2 Experimental results

The developed VSDT was used to design the VLAN in Enterprise LAN that was physically deployed on three floors of a building. This network includes about 100 computers (servers and workstations). Cisco Catalyst 5000 Series switches were used to store the VLAN scheme. Support for virtual subnets was implemented on the basis of MAC addresses. The required logical connectivity matrix was formed on the basis of access lists that were generated by the network administrator for each workstation. Each access list forms one row in the matrix. By using VSDT the VLAN scheme was formed as for the whole network, as for its fragments consisted of 10, 25, 50 and 75 workstations.

The results obtained were then implemented in the real network. Herewith users have not identified the decrease of the network performance. However, the data flows were completely prevented from unauthorized access to neighboring computers.

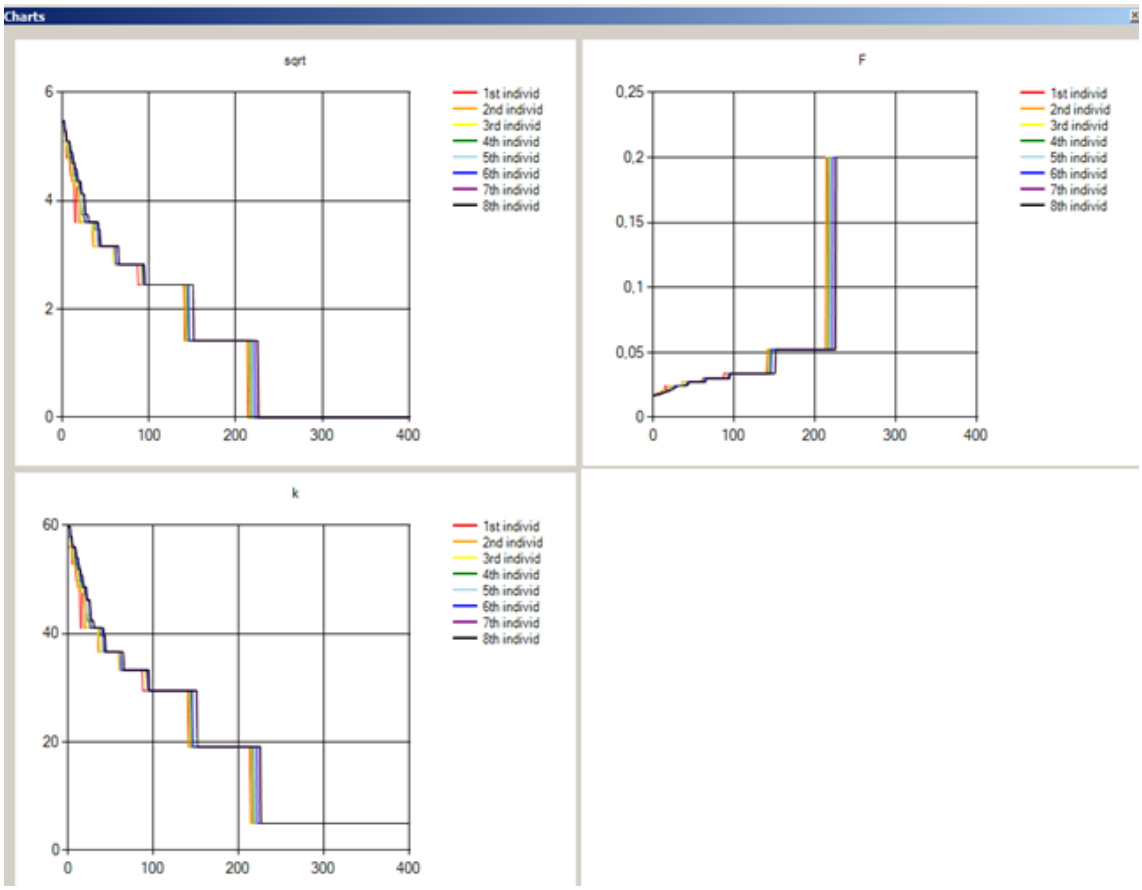


Figure 6: Visual representation of problem solving progress

Thus it can be argued that the security and reliability of the network were increased.

Using VLANS, the experiments were performed to assess the effectiveness of the proposed genetic algorithm for network dimensions mentioned above. The estimation was produced by the following indicators:

- the average number of the algorithm iterations, which were required to find the optimal solution ($\bar{T}$);
- the average duration of a single iteration ($\bar{\tau}_{it}$) for the various dimensions of the problem;
- the accuracy of the algorithm (δ).

The first two indicators characterize the performance of the genetic algorithm.

To determine $\bar{T}$ and $\bar{\tau}_{it}$, five tests were carried out for each pair (n,k). The value k had the following values: $0.2n$, $0.3n$, $0.5n$ and $0.7n$. The computer, on which VSDT run, had the following configuration: processor Intel(R) Atom(TM) CPU N2800, RAM 2 Gb, the operating system Windows 7. VSDT was implemented using C#.

The accuracy δ of the algorithm was evaluated based on the comparison $\mathbf{X}[n,k]$ and $\mathbf{X}_{et}[n,k']$ based on the following assertion: "if $k' \leq k$, then we consider that the algorithm has found the exact solution of the problem, otherwise — approximate". The maximum number of iterations was 1000. To calculate the accuracy of the algorithm the following formula was suggested:

$$\delta = [(n - \max(0; k' - k))/n] \cdot 100\% \tag{5}$$

Table 1: Experimental results

Configuration		Evaluation values		
Number hosts in network (n)	Number virtual subnets (k)	$\bar{T}$	$\bar{\tau_{ij}}$, sec	δ, %
10	2	48.3	0.023	100
10	3	35.6	0.026	100
10	5	25.8	0.028	100
10	7	22.9	0.031	100
25	5	255.8	0.096	100
25	8	183.7	0.13	100
25	13	127.5	0.19	100
25	18	107.2	0.24	100
50	10	1000.0	0.96	98
50	15	811.4	1.19	100
50	25	534.6	1.33	100
50	35	483.4	1.65	100
75	15	1000.0	2.27	91
75	23	1000.0	4.58	95
75	38	1000.0	5.55	97
75	53	1000.0	6.65	98
100	20	1000.0	5.28	78
100	30	1000.0	6.98	85
100	50	1000.0	10.75	90
100	70	1000.0	14.64	92

In this case, if the condition $k' = k$ hold true, then regardless of the n value the accuracy is 100%. If $k' > k$, then the accuracy depends on the relationship between the $(k'^{\smile}k)$ and the n values. So, if $n = 50, k = 10$ and $k' = 11$, then $\delta = 98$.

The results of the speed and accuracy evaluation for the genetic algorithm proposed in the paper are outlined in Table 1.

Analyzing the data presented in Table 1, it is possible to draw the following conclusions. First, under the small dimensions of the problem ($n = 10$ and $n = 25$) the proposed algorithm allowed to generate the solution with maximum accuracy $\delta = 100\%$ in real or near-real-time. Secondly, for large dimensions ($n = 75$ and $n = 100$) it was impossible to obtain the maximum accuracy in the allotted time. However, the accuracy is quite large and ranges from 78 to 98 percent. Finally, when n is constant, the complexity of solving the problem increases when k decreases. It is well illustrated by the $n = 50$. When $k = 15$ and $k = 25$, the most accurate solutions were received; and when $k = 10$, the accuracy of the solution was 98 percent.

In addition, Table 1 reveals the dependence of the average duration of a single iteration $\bar{\tau_{it}}$ on the dimension of the computer network (n). This dependence is shown in Figure 7 for different values of the ratio k/n. We see that in large dimensions of the network the growth of $\bar{\tau_{it}}$ is exponential.

Figure 8 shows the dependence of the average duration of a single iteration on the ratio k/n. We see that in all dimensions this dependence is linear.

In general, the analysis of experimental results, outlined in Table 1, leads to the conclusion that the proposed genetic algorithm has sufficiently high performance for designing virtual computer networks.

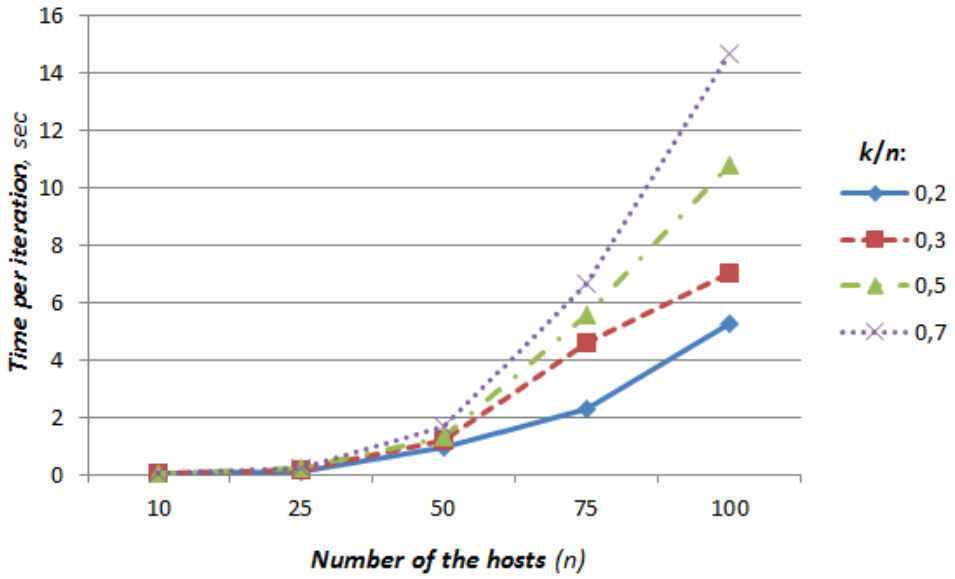


Figure 7: Dependence of the average duration of a single iteration on the dimension of the computer network

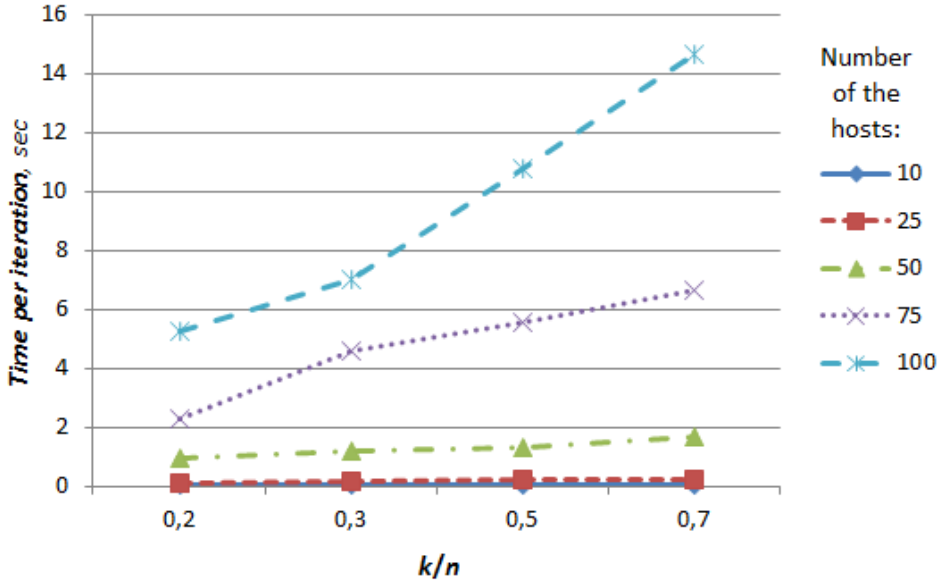


Figure 8: Dependence of the average duration of a single iteration on the ratio k/n

6 Conclusion

The paper has proposed the approach for designing VLAN schemes using the developed software tool VSDT. This tool allows to solve the task of finding a solution of the equation (1) with a minimal number of non-null columns by genetic optimization. In addition, it provides a visual representation of the design progress and evaluates the quality of the genetic algorithm used.

Selection of genetic algorithms as a solution method was due to the following factors. First, it was shown that the task is a kind of BMF, in which both sought-for matrices are the same. For this reason, the problem is the NP-complete and requires to use heuristic methods for a solution. Secondly, some works are known in which genetic algorithms have been used successfully to meet the challenges of the BMF. However, they do not take into account the characteristics of the problem. Therefore, the developed genetic algorithm demanded some improvements that take into account these features.

Key improvements proposed in the paper are as follows:

- the use of trivial solutions to generate an initial population;
- taking into account in the fitness function the criterion of minimum number of virtual subnets;
- the use of columns of the connectivity matrix as genes of the chromosomes which encode the solutions;
- two-dimensional mode for crossing operation in the algorithm;
- two-phase mutation operation in the algorithm;
- zeroing of duplicate columns in the matrix of child solutions, resulting by crossing or mutation.

The VSDT has the following additional features:

- input and correction of the initial data by using on-line forms;
- PRNG-based generation of the desired logical connectivity matrix;
- visual representation of the design progress using the graphs of changes in the fitness function and its components for some of the best solutions;
- assessment of temporal parameters of the genetic algorithm.

Experimental evaluation of speed and accuracy for the proposed genetic algorithm has shown its high efficiency.

6.1 Future Work

Future studies are directed to the application of the proposed approach to the reconfiguration of virtual subnets.

6.2 Acknowledgements

This research is being supported by the grants of the Russian Foundation of Basic Research (13-01-00843, 13-07-13159, 14-07-00697, 14-07-00417), the Program of fundamental research of the Department for Nanotechnologies and Informational Technologies of the Russian Academy of Sciences (contract #2.2), and by Government of the Russian Federation, Grant 074-U01, and State contracts #14.604.21.0033, #14.604.21.0137, #14.604.21.0147 and #14.616.21.0028.

References

[1] I. Saenko and I. Kotenko, "A genetic approach for virtual computer network design," in *Intelligent Distributed Computing VIII, Studies in Computational Intelligence, Proc. of the 8th International Symposium on Intelligent Distributed Computing (IDC'14), Madrid, Spain*. Springer-Verlag, September 2014, pp. 95–105.

[2] "Catalyst 2900 series xl and catalyst 3500 series xl software configuration guide. cisco ios release 12.0(5) wc(1)," Cisco Systems, San Jose, 2001.

[3] "Oracle real application clusters (rac) and oracle clusterware interconnect virtual local area networks (vlans) deployment considerations," An Oracle White Paper, 2012.

[4] P. Miettinen and J. Vreeken, "Model order selection for boolean matrix factorization," in *Proc. of the 17th Conference on Knowledge Discovery and Data Mining (KDD'11), San Diego, USA*. ACM, August 2011, pp. 51–59.

[5] P. Miettinen, "Dynamic boolean matrix factorizations," in *Proc. of the 12th International Conference on Data Mining (ICDM'12), Brussels, Belgium*. IEEE, December 2012, pp. 519–528.

[6] E. Cergani and P. Miettinen, "Discovering relations using matrix factorization methods," in *Proc. of the 22nd International Conference of Information and Knowledge Management (CIKM'13), Burlingame, USA*. ACM, October-November 2013, pp. 1549–1552.

[7] H. Lu, J. Vaidya, V. Atluri, and Y. Hong, "Extended boolean matrix decomposition," in *Proc. of the 9th International Conference on Data Mining (ICDM'09), Miami, Florida, USA*. IEEE, December 2009, pp. 317–326.

[8] H. Lu, J. Vaidya, and V. Atluri, "Optimal boolean matrix decomposition. application to role engineering," in *Proc. of the 24th International Conference on Data Engineering (ICDE'08), Cancun, Mexico*. IEEE, April 2008, pp. 297–306.

[9] I. Saenko and I. Kotenko, "Genetic algorithms for role mining problem," in *Proc. of the 19th International Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP'11), Ayia Napa, Cyprus*. IEEE, February 2011, pp. 646–650.

[10] Igor Saenko and Igor Kotenko, "Design and performance evaluation of improved genetic algorithm for role mining problem," in *Proc. of the 20th International Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP'12), Garching, Germany*. IEEE, February 2012, pp. 269–274.

[11] A. Janecek and Y. Tan, "Using population based algorithms for initializing nonnegative matrix factorization," vol. 6729, pp. 307–316, June 2011.

[12] V. Snasel, J. Platos, and P. Kromer, "On genetic algorithms for boolean matrix factorization," in *Proc. of the 8th International Conference on Intelligent Systems Design and Applications (ISDA'08), Kaohsiung, Taiwan*. IEEE, November 2008, pp. 170–175.

[13] V. Snasel, J. Platos, P. Kromer, D. Husek, R. Neruda, and A. A. Frolov, "Investigating boolean matrix factorization," in *Proc. of the Workshop on Data Mining using Matrices and Tensors (DMMT'08) held in Conjunction with the 14th International Conference on Knowledge Discovery and Data Mining (SIGKDD'08), Las Vegas, USA*. ACM, August 2008, pp. 18–25.

[14] I. Saenko and I. Kotenko, "Genetic optimization of access control schemes in virtual local area networks," vol. 6258, pp. 209–216, September 2010.

[15] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[16] M. Mitchell, *An Introduction to Genetic Algorithms*. The MIT Press, 1998.

[17] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Springer-Verlag, 2003.

[18] E. Barker and J. Kelsey, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. NIST, 2012.

Author Biography

Igor Saenko graduated with honors from St.Petersburg Signal Academy. He obtained the Ph.D. degree in 1992 and the National degree of Doctor of Engineering Science in 2001. He is Professor of computer science and Leading Researcher of the Laboratory of Computer Security Problems of St.Petersburg Institute for Informatics and Automation. He is the author of more than 200 refereed publications and participated in several Russian and international research projects. His main research interests are security policy management, access control, management of virtual computer networks, database management, knowledge modeling, soft and evolutionary computation, information and telecommunication systems.

Igor Kotenko Igor Kotenko graduated with honors from St.Petersburg Academy of Space Engineering and St.Petersburg Signal Academy. He obtained the Ph.D. degree in 1990 and the National degree of Doctor of Engineering Science in 1999. He is Professor of computer science and Head of the Laboratory of Computer Security Problems of St.Petersburg Institute for Informatics and Automation. He is the author of more than 200 refereed publications. Igor Kotenko has a high experience in the research on computer network security and participated in several projects on developing new security technologies. For example, he was a project leader in the research projects from the US Air Force research department, via its EOARD (European Office of Aerospace Research and Development) branch, EU FP7 and FP6 Projects, HP, Intel, F-Secure, etc. The research results of Igor Kotenko were tested and implemented in more than fifty Russian research and development projects. The research performed under these contracts was concerned with innovative methods for network intrusion detection, simulation of network attacks, vulnerability assessment, security protocols design, verification and validation of security policy, etc. He has chaired several International conferences and workshops, and serves as editor on multiple editorial boards.