Conference on Cyber Conflict
Proceedings 2010
C. Czosseck and K. Podins (Eds.)
CCD COE Publications, 2010, Tallinn, Estonia.

OPTIMIZING IT SECURITY COSTS BY EVOLUTIONARY ALGORITHMS

Toomas Kirt[a,1], Jüri Kivimaa[b,2]

[a] *University of Tartu, Estonia,* [b] *CCD COE, Tallinn, Estonia*

Abstract: One of the most critical issues in IT security is to establish a cost-effective framework for cyber protection against possible threats. The overall security framework is divided into security activity areas, which can have a number of protection levels. Each level of one security activity area provides certain confidence and also requires some expenditure. As the budget level is predefined a critical question remains how to find out an adequate security profile for a certain cost level. As the behavior of cyber attackers and cyber security threats are continuously changing, there should not be just one model to construct an effective security mechanism but rather a variety of changing alternatives. Several methods have been proposed for cost optimization but they are limited by providing only one alternative. In this paper we propose an evolutionary approach as an alternative for optimizing IT security costs and for finding variants of security profiles for every cost level. Higher variability of security profiles will make the security organization more resistant to changing cyber attacks.

Keywords: graded security model, information security metrics, information security requirements, evolutionary computing, genetic algorithms

1 University of Tartu, Institute of Public Law, Teatri väljak 3, Tallinn, 10143, ESTONIA, Email: Toomas.Kirt@ut.ee.

2 Cooperative Cyber Defence Centre of Excellence, Filtri Street 12, Tallinn, 10132, ESTONIA, Email: Jyri.Kivimaa@ccdcoe.org

INTRODUCTION

We have the challenge of ensuring information security under conditions of uncertainty: how can organizations determine appropriate measures to enhance cyber security and allocate resources most efficiently? For finding out an optimal amount of resources a security costs function is proposed, where the total cost of security for a system is based on the cost of system security investments plus the cost of damage and cost of recovery from any security incidents (Olovsson, 1992). Despite the fact that the cost function also includes indirect costs in this study we take into account only the direct costs of security investments. Usually, available resources are limited and therefore it is needed to optimize applied security measures to achieve the highest attainable confidence level. The security framework is divided into several security activity areas that can have a number of levels providing certain confidence. As the number of security activity areas increases the number of different combinations of security measures or profiles grows exponentially. For finding an optimal security profile several optimization methods are used, such as a brute force optimizer and a discrete dynamic programming method (Kivimaa, 2009; Ojamaa, Tyugu, & Kivimaa, 2008).

It is argued that the dynamic programming may have some problems related to independence of security activity areas and additivity and therefore the solutions may not to be optimal (Kivimaa, 2009). This additivity restriction also limits the search space and it is difficult to find out alternative security profiles that provide the same level of confidence. Therefore our aim is to apply an additional method to find out whether the solutions are adequate and also identify alternative security profiles for a certain cost level. We decided to use an evolutionary algorithm as a universal method for complex optimization in many fields. Genetic algorithms are also actively used in IT security and intrusion detection systems (e.g. Li, 2004; Sinclair, Pierce, & Matzner, 1999).

Evolutionary algorithms are based on a Darwinian natural selection process and form a class of population-based stochastic search algorithms (Dracopoulos, 2008; Eiben & Smith, 2003; Holland, 1975). In the evolutionary process for all the individuals representing candidate solutions some perturbations (e.g. crossover, mutations) are applied to generate variation and thereafter a selection procedure, based on the value of a fitness function, is enforced. The selection mechanism prefers individuals that are the best candidates for the solution of the optimization problem. To maintain the variation in population in our experiments the population was divided into subgroups and the selection process was performed within a group. This measure helped to avoid the optimization process to fall into a local optimum and provided better results. To solve the optimization task we have established an evolutionary framework and applied it to the IT security cost/confidence data consisting of 9 se-

curity areas (CyberProtect, see Table 1). In the following optimization tasks we had two goals: to minimize the costs and to maximize the integral security confidence.

This paper is divided into four main parts. In the first part the security model and the data is described that we use in our optimization tasks. Next we introduce the basis of evolutionary algorithm. Thereafter the results of optimization are given. Finally the results are discussed and conclusions are made.

1. SECURITY MODEL

The main challenge in IT security is to ensure required information security under conditions of uncertainty. To achieve the goal an organization has to define adequate security levels and to determine appropriate measures for increasing cyber security and allocating resources most efficiently. Usually certain risk assessment methods are used for performing detailed risk analysis. For small and medium size enterprises the detailed risk analysis is relatively expensive and also the available resources for IT security are limited. Therefore a simpler version of the security model is needed which provides possibility to achieve maximum possible confidence with limited resources.

Table 1. IT security costs/confidence data. 9 security measures

Security measure \ level		Level 0	Level 1	Level 2	Level 3
1. User Training	Cost	0	4	8	12
	Confidence	0	30	50	65
2. Redundant Systems	Cost	0	8	10	12
	Confidence	0	40	70	95
3. Access Control	Cost	0	1	2	4
	Confidence	0	40	70	95
4. Antivirus	Cost	0	2	4	7
	Confidence	0	60	80	95
5. Backup	Cost	0	1	2	4
	Confidence	0	40	70	95
6. Disconnection	Cost	0	2	4	7
	Confidence	0	40	60	75
7. Encryption	Cost	0	2	4	7
	Confidence	0	60	80	95
8. Firewall	Cost	0	2	4	7
	Confidence	0	30	50	65
9. Intrusion Detection	Cost	0	1	2	4
	Confidence	0	25	45	60

In this research we rely on the graded security model, which is an improved and combined version of two security methodologies: the US DoE graded security methodology (best practice security methodology to specify needed security measures for needed security levels; DOE, 1999) and Estonian governmental data classification (metrics to specify needed security level; ISKE, 2009). "The system includes knowledge modules (rule sets) in the form of decision tables for handling expert knowledge of costs and confidence, as well as for selecting security measures for each security group depending on the required security level." [Kivimaa, et. al., 2009] Basic ideas of graded security are presented as a decision table – information security activities areas/their realization levels and information security requirements/their levels in a dependency matrix. As an example a very simple (9 security subareas) decision table/dependency matrix is given in the Appendix.

The example used in the experiments of this paper is an educational security framework CyberProtect version 1.1 (CyberProtect, Table 1). It determines how hardware/software/firmware can be secured based on nine security activity/measure groups and their high/middle/low level realization of costs and confidence. The cost in this example covers only the costs of security investments and is given in conventional units. The confidence level is in the scale of 0···100 and the value is provided as an expert opinion. Each security measure can have a certain level that determines required resources to achieve confidence. The baseline security methodologies define conventional goals of security as confidentiality (C), integrity (I), availability (A), and mission criticality (M). For each goal a finite number of security levels have been determined. For example, four levels 0, 1, 2, 3 for representing required security and protection can be used, where the lowest level 0 denotes unnecessary of special protective measures. [Kivimaa, et. al., 2009]

We can formulate an optimization problem as follows: find the abstract security profile with the best (highest) value of confidence for given amount of resources. As we have a limited amount of available resources r our goal is to achieve a maximum security level

$$S_{\max} = \sum_{i=1}^{n} a_i q_{\max i}$$

where $q_{\max i}$ is maximum security confidence of the i-th group of security activity areas and a_i is the weight of the i-th group

$$\sum_{i=1}^{n} a_i = 1$$

We have an optimization problem with two goals: to minimize resources on the interval $[r_{min}; r_{max}]$ and to maximize security, guaranteeing at least the levels prescribed by a given security class. We are going to solve this problem by finding a function that gives an abstract security profile that has maximum value of a security confidence function given by the weighted mean security for any given value of resources on the interval $[r_{min}; r_{max}]$. The task of the optimization application is to find the best combination of security measure levels that provides the maximum confidence at a cost level.

In previous experiments mainly two optimization algorithms were used to solve our task – one of them was a brute force optimizer and the other one was based on a Pareto optimality (Pareto frontier or Pareto set) and discrete dynamic programming method (Ojamaa, et al, 2009). This problem can be solved by means of building a Pareto optimality trade-off curve that explicitly shows the relation between used resources and security confidence. Then, knowing the available resources, one can find the best possible security level that can be achieved with the resources and specify the security measures to be taken.

For n security measures groups and k levels for information security requirements/goals we have totally k^n abstract security profiles to be considered. The number of security measures groups may be in practice up to 30 or even more and in Estonian data classification a 4-level version for security goals is used. This gives a number of abstract security profiles: 4^{30}.

With the brute force method we must do rk^n computations and with the dynamic programming method r^2kn (r is number of possible values of resources, k is the number of security levels, n is number of security measures groups). For example, if we have a 100 budget points curve for 25 security subareas then it takes ~10 seconds to calculate it with the Pareto optimality & dynamic programming and by the Brute Force method it would take ~10 years to calculate (Kivimaa, 2009).

To use Pareto optimality and dynamic programming in optimization security activities areas/security measures groups must be not dependent from each other's and their security measures to realize their levels must be additive. Independency in IT security activities is quite problematic for some security areas, but in first approximation it is acceptable if we use certain specific logic of description (for example, the IT security experts/specialists training costs are included into the costs of concrete security activities areas/areas levels and some other analogical principles might be followed).

The second weakness of dynamic programming is that it has some difficulties in finding alternative security profiles for a certain optimal cost/confidence level. To get over those weaknesses and to measure adequacy of the dynamic programming

we decided to use an evolutionary algorithm as an alternative method. We expect that the evolutionary approach is not stuck to such limitations and can provide results with a quite reasonable time.

2. EVOLUTIONARY ALGORITHMS

An evolutionary algorithm is a population-based stochastic search algorithm. The basic principle is to iteratively generate random variation within individuals of population, that represents the candidate solution to the problem, and to select the fittest candidates that provide the best solution to the task in hand. The view that random variation provides the mechanism for discovering new solutions (Michalewicz & Fogel, 2004) was inspired by the process of natural evolution.

The idea of using Darwinian principles of evolution to solve some combinatorial optimization problems arose with the invention of computers. Afterwards several approaches were developed like evolutionary programming (Fogel, Owens, & Walsh, 1966) and genetic algorithms (Holland, 1975) in the early stage of the study of evolutionary algorithms. Now there are a wide variety of approaches that can be described as belonging to the field of evolutionary computing. The algorithms used in the field are termed as evolutionary algorithms (Dracopoulos, 2008). The most important characteristics of evolutionary algorithms are as follows:

- *Representation.* Each candidate solution to the problem in hand is represented as an individual. The characteristics of the individual are encoded by genes. The set of individuals form a population.
- *Fitness.* The quality of a candidate solution is measured by a fitness function. The fitness function is used to measure how good an individual is. Fitter solutions have a higher probability to survive and to contribute their characteristics to offspring.
- *Variation.* Variation operators (e.g. crossover, mutations) are applied to the individuals that modify the population of solutions dynamically.
- *Selection.* The average fitness is improved over time as a selection mechanism is applied and the fittest individuals are selected for the next generation (survival of the fittest).

The basis of an evolutionary algorithm is simple. First, a population of initial candidate solutions has to be generated randomly. Thereafter iteratively a number of variation generation operators are applied and new generations are selected based on the fitness values of individuals.

2.1 ALGORITHM

There are several modifications proposed to the basic algorithm and we have adapted some aspects of cooperative co-evolutionary algorithms (see Machado, Tavares, Pereira, & Costa, 2002; Potter & De Jong, 2000). In this approach the problem is decomposed into subcomponents that represent potential components to the global problem (see more details in Selection). As the problem in hand was not very complex we decided to decompose a population P into S subpopulations Ps instead of decomposing a problem. The aim was to maintain variety within the population as a whole.

The algorithm can be defined then as follows:

- for each subpopulation S do:
  - Initialize population Ps(0)
  - Evaluate all individuals from Ps(0)
- While termination condition not met repeat:
  - For each subpopulation S do:
    - Apply crossover and mutation operators to individuals of Ps(t) and obtaining a set of offspring Os(t)
    - Evaluate individuals from Os(t)
    - Combine Ps(t) and Os(t) obtaining Ps(t+1)

During the evaluation the fitness value (average confidence level) of an individual is found. The fittest from the ordered set of parents and offspring are selected for the next generation.

2.2 REPRESENTATION

How to choose a suitable genetic representation of an individual is a key issue in evolutionary computing. Each individual has two representations: phenotype (outside) and genotype (inside). Object forming possible solutions within the original problem context are referred as phenotypes, while their encoding, that is, the individuals within the evolutionary algorithm, are called genotypes (Eiben & Smith 2003). Phenotypic characteristics of the candidate solution are encoded by individual's genotype. The genes are the functional units to carry inherited information and they can be arranged in chromosomes. In evolutionary algorithm a chromosome can be a string of symbols or a vector of numerical variables (Gen & Lin, 2008). The complete inherited information is called a genome.

Genotype contains inherited information to build an individual in phenotype space. In the natural systems the mapping from genotype to phenotype is not direct. In the context of evolutionary algorithms three classes of possible mappings are defined: direct, developmental and implicit (Floreano, Dürr, & Mattiussi, 2008). In a direct representation, there is a one-to-one mapping between the parameter values of the task in hand and the genes that compose the genetic string. In developmental representations which are used mostly in case of large problems the specification of a developmental process is genetically encoded which in turn constructs the desired phenotype. In case of implicit encoding like in biological gene networks, the interaction between the genes is not explicitly encoded in the genome, but follows implicitly from the physical and chemical environment in which the genome is immersed.

In this paper direct mapping is used and each candidate solution is represented as a chromosome consisting of the same amount of genes as the number of security activity areas. Each gene denotes a security level of one security activity area. For example, if there are 3 security levels plus one for the lowest level 0 denoting absence of special protective measures four possible values for one gene (0, 1, 2, 3) can be defined. If there are 9 security activity areas then a chromosome can be G = {1 0 3 2 3 1 2 1 3}.

2.3 FITNESS

The goal of the evolutionary search is defined as a user-specified measure of the quality or the fitness of the individuals. The algorithm is expected to find in the search space an individual with maximum quality or fitness. In our experiments the fitness is measured as a weighted average of confidence levels of security activity areas.

2.4 VARIATION

The initial population is usually generated randomly and therefore it is highly variable. The movement in the search space is based on random changes in chromosomes generated by reproduction and applying several variation operators. The reproduction is carried out with some stochastic mutation and recombination of the parents in order to explore new regions the search space and combine the information carried by each parent (Gen & Lin, 2008).

The main operator to generate variation in population is the crossover. There are introduced several approaches to select parents and to recombine their genetic information. Recombination, the process whereby a new individual is created from the information contained within two parents, is considered to be one of the most

important features in evolutionary algorithms. In the experiments we use the crossover operator called n-point crossover, where the value of n is 2. The basic steps of applying a crossover operator are as follows: first, to select two parents based on some restrictions (if there are any) and next, select segments of genes from both parents to form the genes of an offspring. The second parent is selected randomly from the whole population. An example is illustrated in Figure 1. A segment {4, 3} is taken from one parent and is transferred to the other parent's genetic code.

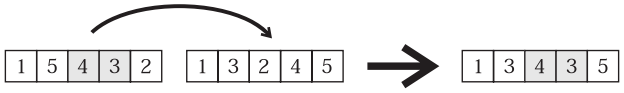

Figure 1. n-point crossover. n = 2.

Several variation operators are used to make variation in population and to move in the search space.

Random mutation is the change of the value of one gene. For example, the value of the first gene {1} is replaced by the new value {3}.

Figure 2. Random mutation of a single gene

Swap operator: selects two genes and swaps them. For example, genes {5} and {3} are selected and swapped.

Figure 3. Swap mutation

Inversion operator: selects a segment of genetic code and reverses order of the genes belonging to it. For example, genes {1 5} are reversed {5 1}.

Figure 4. Inversion mutation

Insertion operator: selects a gene and inserts it in another place. For example, gene {1} is moved to the end of the genetic code.

Figure 5. Insertion mutation

Displacement operator: selects a segment of genetic code and inserts it in another place. For example, genes {1 5} are moved to the end of the genetic code.

Figure 6. Displacement mutation

When mutation operators are applied, the genes are validated whether they are in accordance to the restrictions of the task in hand. When the code does not meet the restrictions it is not used in the further processing.

2.5 SELECTION

The selection is a process to select survivals for the next generation. During each generation, the chromosomes are evaluated, using some measures of fitness. A new generation is formed by selecting some parents and offspring, according to their fitness values, and rejecting others to keep the population size constant.

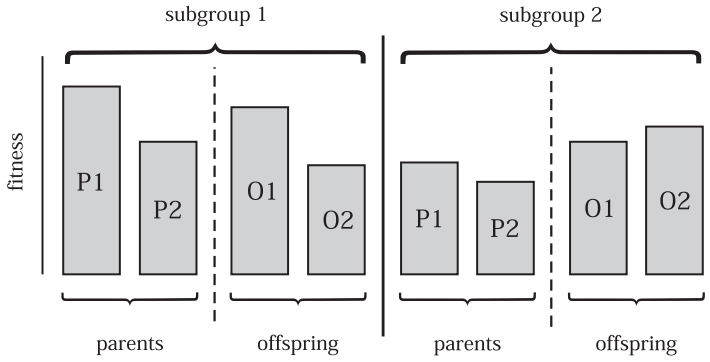

Figure 7. An example of a tournament selection of 2 sub-population consisting of 2 individuals. 4 candidates (2 parents P and 2 offspring O) are competing for selection for next generation within a sub-population

a) After reproduction and mutation a new sets of individuals (offspring) are formed in each subpopulation

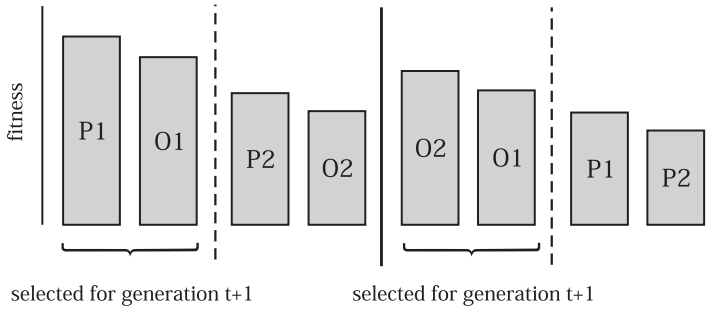

Figure 8. Selection and regrouping of the initial population.

b) For selection the parents and offspring within a sub-group are ordered based on the fitness value and the fittest are selected for the next generation

In this study the selection method is based on the tournament selection strategy, which is deterministic. The tournament selection is effective, because it does not require any global knowledge of the population and it also avoids falling into a local optimum by maintaining variety in the population. This strategy also enhances the search space and allows exploring it in parallel. To perform tournament selection we have to define the tournament size k. The members of a tournament are usually selected randomly, but we use a deterministic strategy where the competing sub-populations are predefined. For example, the tournament or subpopulation size is defined as 2. After reproduction and mutation phase (Figure 7) 4 candidates (2 parents and 2 offspring) compete for being selected for the next generation (Figure 8). The selection is performed locally and therefore the winning members of one tournament may have a weaker fit value than the least-fit members of the other tournament. Further mutations in such a weak subpopulation may reveal some properties of an individual that are needed to reach the global optimum and are not represented in other subgroups.

3. EXPERIMENTS

For experiments we had the IT security cost/confidence data consisting of 9 security activity areas (CyberProtect; see Table 1). The aim of the optimization was to find the highest average confidence level for a given amount of resources. The optimization task is formed as a question (Kivimaa, 2009): “For every possible budget level, what is the maximum confidence one can expect?” In the optimization tasks the amount of resources (budget) was predefined form 1 to max+1. The max value equals the costs of the security measures of the highest level.

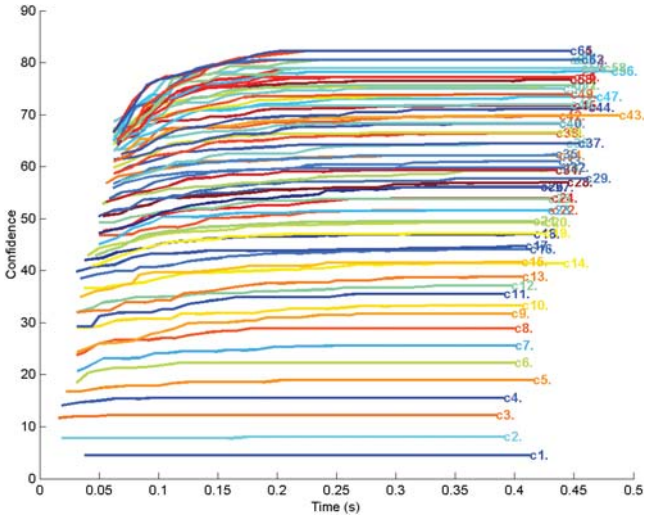

Figure 9. Mean computational time to find optimal confidence value for 9 security areas (mean value of 5 experiments)

The first task was to measure the mean computational time to solve the optimization problem. The second task was to find the cost/confidence optimality curve. The third task was to find out the cost/confidence optimality curve when the optimality was restricted by a security class. The fourth task was to identify adequate and equivalent security profiles for every cost level.

For the results presented in this section we used the following experimental settings: crossover rate 0.49, mutation rate 0.2, swap rate 0.1, inversion rate 0.1, insertion rate 0.1, and displacement rate 0.1. The number of generations was set as 30 and population size 80, and the tournament or subpopulation size was 5. The cost of the highest security level (C3I3A3M3) was 64 units and the optimization was performed for the cost levels from 1 to 65 units. With each cost level 5 experiments were performed. The rates for crossover and mutation operators were selected as the best practice of solving other optimization problems. Despite the optimization tasks are similar the rates might not be the best for solving the security optimization task. Additional computation time is required if either the variation rate is very low or high, as unnecessary calculations are needed to be performed.

As a result the average time for optimization was between 0.4 and 0.45 seconds (Figure 9). The task two was to find the cost/confidence optimality curve (yellow dots in Figure 10). For interpretation a color coding of dots in the curve is used as follows: red dots – all security activities area's security levels are ≤ and at least one is < than required; green dots – all security goals/their required levels are exactly achieved;

yellow dots – at least one security level is less and at least one security level is more than required; blue dots – all security levels are ≥ and at least one security level is > than required. The curve represents the optimal value of weighted mean security confidence depending on the resources that are used.

Table 2. The experimental dependency matr ix of 9 security measures

Security measure	C0	C1	C2	C3	I0	I1	I2	I3	A0	A1	A2	A3	M0	M1	M2	M3
1. User Training	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
2. Redundant Systems	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
3. Access Control	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
4. Antivirus	1	1	2	3	1	1	2	3	1	1	2	3	1	1	2	3
5. Backup	0	1	2	3	0	1	2	3	0	1	2	3	0	1	3	3
6. Disconnection	1	1	2	3	1	1	2	3	1	1	2	3	1	2	3	3
7. Encryption	0	0	1	3	0	1	2	3	0	0	0	0	0	0	2	3
8. Firewall	0	1	2	3	0	1	2	3	0	1	2	3	0	2	3	3
9. Intrusion Detection	0	1	2	3	0	1	2	3	0	1	2	3	0	2	2	3

Next experiments were performed when the limitation of security class was applied. In this study an experimental dependency matrix of connections between security measures and conventional goals of security was used (Table 2). For example, as the security class is defined C1I1A1M1 then the highest level of a security measure is selected and the security configuration is (1, 1, 1, 1, 1, 2, 1, 2, 2). The comparison of confidence values between the security classes C3I3A3M3 and C1I1A1M1 is given in Figure 10. As there are more available resources than are needed to satisfy the restrictions caused by a security class the security measures cannot be weaker than determined by the security class.

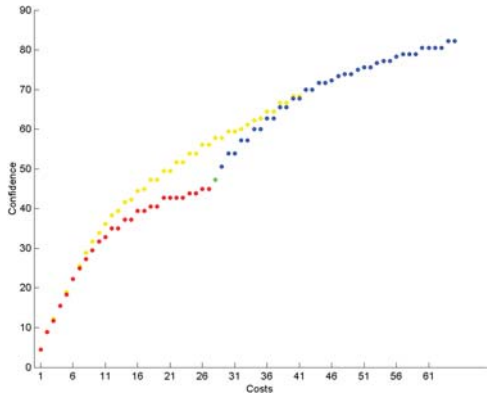

Figure 10. Costs/confidence optimality curve using security-class limitation. Security class C3I3A3M3 versus C1I1A1M1. Optimal security configuration (1, 1, 1, 1, 1, 2, 1, 2, 2)

The final task was to obtain different security profiles. To find out different security profiles we ran experiments 35 times for every cost level. An extract of the results is given in Table 3. For example, when 34 unit of money was available (budget restriction) then 5 equivalent security profiles were found.

Table 3. Equivalent security profiles for every cost/confidence level in case of 9 security measures. An excerpt

				Security measure								
No.	Money	Costs	Confidence	1	2	3	4	5	6	7	8	9
…												
88	34	34	62,22	1	4	4	2	4	2	3	3	3
89	34	34	62,22	1	4	4	3	4	2	2	3	3
90	34	34	62,22	1	4	4	3	4	3	2	2	3
91	34	34	62,22	1	4	4	2	4	3	3	2	3
92	34	34	62,22	1	4	4	2	4	3	2	3	3
93	35	35	62,78	2	1	4	3	4	4	3	3	4
94	35	35	62,78	2	1	4	3	4	3	4	3	4
95	35	35	62,78	2	1	4	3	4	3	3	4	4
96	35	35	62,78	2	1	4	4	4	3	3	3	4
97	36	36	64,44	1	4	4	3	4	3	3	2	3
98	36	36	64,44	1	4	4	2	4	3	3	3	3
99	36	36	64,44	1	4	4	3	4	2	3	3	3
100	36	36	64,44	1	4	4	3	4	3	2	3	3
…												

4. CONCLUSIONS

The aim of the study was to evaluate whether the evolutionary approach is applicable to the security of the cost/confidence optimization task and whether it allows us to generate equivalent security profiles for every cost level. As a result we could conclude that the evolutionary approach is viable for such tasks. The results indicated that the evolutionary algorithm was fast enough to provide results and turned out to be more flexible than the discrete dynamic programming method. The evolutionary approach provided results within a reasonable time limit and the cost/confidence optimization of 9 security activity areas took 0.4-0.45 seconds (Figure 7). The main advantage of the evolutionary algorithm was that it provided several adequate and equivalent security profiles for every cost level with a reasonable time (see Table 3). As it is noted, there should not be just one model to construct an effective security mechanism but several simple security mechanisms that are attuned to the needs of differing applications and organizations (Wulf & Jones, 2009). Thereby the evolutionary approach might help us to provide a better confidence level.

REFERENCES

- CyberProtect, version 1.1. U. S. Department of Defense, Defense Information Systems Agency. Available at: from http://iase.disa.mil/eta/. [Accessed 1st February 2010]
- Department of Energy, 1999. *Classified Information Systems Security Manual*. Available at: https://www.directives.doe.gov/directives/archive-directives/471.2-DManual-2/at_download/file. [Accessed 1st February, 2010]
- Dracopoulos, D. C., 2008. Evolutionary Learning. In B. Wah (Ed.), *Wiley Encyclopedia of Computer Science and Engineering*. New York: John Wiley & Sons.
- Eiben, A. E., & Smith, J. E., 2003. *Introduction to Evolutionary Computing*. Berlin: Springer.
- Floreano, D., Dürr, P., & Mattiussi, C., 2008. Neuroevolution: From architectures to learning. *Evolutionary Intelligence,* 1(1), 47–62.
- Fogel, L. J., Owens, A. J., & Walsh, M. J., 1966. *Artificial Intelligence Through Simulated Evolution,* John Wiley & Sons: New York.
- Gen, M., & Lin, L., 2008. Genetic Algorithms. In B. Wah (Ed.), *Wiley Encyclopedia of Computer Science and Engineering*. New York: John Wiley & Sons.
- Holland, J. H., 1975. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Cambridge, MA: MIT Press.
- ISKE, 2009. ISKE - three-level IT baseline protection system (Version 5.00). Retrieved February 1, 2010, Available at: http://www.ria.ee/public/ISKE/iske_rakendusjuhend_5_00.pdf. [Accessed 1st March, 2010]
- Kivimaa, J., 2009. Applying a costs optimizing model for IT security. In H. Santos (Ed.), *Proceedings of the 8th European Conference on Information Warfare and Security* (pp. 142–153). Reading, UK: Academic Publishing Limited.
- Kivimaa, J., Ojamaa, A., Tyugu, E., 2009. Graded Security Expert System, *Critical Information Infrastructure protection,* Berlin: Springer.
- Li, W., 2004. Using Genetic Algorithm for network intrusion detection. In *Proceedings of United States Department of Energy Cyber Security Group 2004* Training Conference (pp. 1-8). Kansas City, Kansas.
- Machado, P., Tavares, J., Pereira, F. B., & Costa, E., 2002. Vehicle Routing Problem: Doing it the Evolutionary Way, In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)* (p. 690). New York, USA.
- Michalewicz, Z., & Fogel, D. B., 2004. *How To Solve It: Modern Heuristics*. Berlin: Springer.
- Ojamaa, A., Tyugu, E., & Kivimaa, J., 2008. Pareto-optimal situation analysis for selection of security measures. In *Military Communications Conference: Unclassified Proceedings* (pp. 3224–3230). Piscataway, NJ: IEEE.
- Olovsson, T., 1992. A Structured Approach to Computer Security. In: *Technical Report No. 122*. Göteborg, Sweden: Chalmers University of Technology.
- Potter, M. A., & De Jong, K., 2000. *Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents*. Evolutionary Computation, 8(1), 1–29.
- Sinclair, C., Pierce, L., & Matzner, S., 1999. An application of machine learning to network intrusion detection. In *Proceedings of the 15th Annual Computer Security Applications Conference* (pp. 371-377). Phoenix, AZ.
- Wulf, W. A., & Jones, A. K., 2009. Reflections on Cybersecurity. *Science,* 326, 943–944.

APPENDIX

Table 4. The dependency matrix of 9 security measures

NO	Information Security Goals / Information Security activities areas	Confidentiality Requirements				Integrity Requirements				Availability Requirements				Mission Criticality			
		C0	C1	C2	C3	I0	I1	I2	I3	A0	A1	A2	A3	R0	R1	R2	R3
		Public Data	Data for Internal Use	Confidential Data	Highly Confidential Data	Protection/Detection of Changes is Not Important	Protection/Detection of Unauthorized Changes	Inputter/Changer must be detectable	Inputter/Changer Must be Provable (in court)	Delay of Data Will Not Cause Problems	Availability 90% - allowed ~ one day delay in week	Availability 99% - allowed ~ one hour delay in week	Availability 99,9% - allowed one ~10 minuts delay in week	Data Delay Will Not Cause Significant Consequences	Data Delay Causes Damages in Hundreds of Thousands	Data Delay Causes Damages in Millions or Kroons	Data Delay Causes Hundreds of Millions of Damages
1.	**Access Control**	AC-0	AC-1	AC-2	AC-3	AC-0	AC-1	AC-2	AC-3								AC-4
2.	**User Training**		UT-1	UT-2	UT-3												
3.	**Disconnection** (Data Communications)		DC-1	DC-2	DC-3						DC-1	DC-2	DC-3				DC-4
4.	**Encryption**		CR-1	CR-2	CR-3		CR-1	CR-2	CR-3								
5.	**Intrusion Detection** (Monitoring)	ID-0	ID-1	ID-2	ID-3												
6.	**FireWall** (Perimeter Protection)	FW-0	FW-1	FW-2	FW-3	FW-0	FW-1	FW-2	FW-3								
7.	**Antivirus**		AV-1	AV-2	AV-3		AV-1	AV-2	AV-3								
8.	**Backup and Recovery**						BR-1	BR-2	BR-3		BR-1	BR-2	BR-3				BR-4
9.	**Redundancy** (IT Recovery)										R-1	R-2	R-3				R-4