

Evolutionary Algorithms for Optimal Selection of Security Measures

Jüri Kivimaa¹ and Toomas Kirt²

¹Cooperative Cyber Defence Centre of Excellence, Tallinn, Estonia

²University of Tartu, Tallinn, Estonia,

Jyri.Kivimaa@mil.ee

Toomas.Kirt@ut.ee

Abstract: A very important issue in IT Security or Cyber Security management is to provide cost-efficient security measures to achieve needed or required security goals (mainly CIA - Confidentiality, Integrity, Availability levels). For providing an optimal solution an optimization task with two goals have to be solved – to minimize needed resources and to maximize achievable security. The computational complexity of the optimization task is very high. In previous work a matrix based security model and an optimization framework based on the Pareto optimality and the discrete dynamic programming method has been used. But that solution has a quite important imperfection – there was required independence between security activity areas. That is not appropriate for IT security, as this solution does not follow the quite important principle in IT security – security is like a chain that is only as strong as the weakest link of layered security or defence in depth. The evolutionary optimization, as an alternative optimization tool, removed the independence restriction of the matrix based security model and the dynamic optimization method, but the first implementation of it was slightly slower than the other methods. For improving the performance of the evolutionary optimization we have performed a meta-level optimization of parameters of the algorithm and as a result the speed of optimization is comparable to other optimization techniques. As the evolutionary optimization is independent for all possible budget levels it lead to possibility to use a graph based security model. The graph based security model is a new and dynamical framework for security management. This paper presents how implementation of an evolutionary optimization technique removed the restrictions of independence of security measures and lead to implementation of an efficient graph based security model.

Keywords: graded security model, information security metrics, evolutionary optimization

1. Introduction

One of the most important tasks for IT security management is the optimal use of existing resources and the main idea for our R&D work is to propose to IT Security decision-makers a Graded Security Model (GSM) and a decision support system for this. In papers (Kivimaa, 2009; Kivimaa, Ojamaa, and Tyugu, 2009; Ojamaa, Tyugu, and Kivimaa, 2008) it was shown how to use the GSM for finding optimal solutions based on the Pareto-optimal situation analysis, the discrete dynamic programming method for optimization calculations and weighted average confidence of security activities areas was used as optimization criteria. As it turned out the computational complexity of the optimization task is very high. For example, if to consider that an IT security model has 30-40 activity areas and in each of them has 4 possible implementation levels then there are $4^{30} \div 4^{40}$ possible solutions within to select an optimum. The Brute Force optimization technique requires a couple of years to calculate even one possible budget point.

In (Kivimaa 2009) was also brought up some weaknesses caused from the dynamic programming method. Namely, using dynamic programming in optimization of security activities areas must not be dependent from each other and their levels must be additive. To achieve better solutions in the future it is reasonable to continue GSM development – mainly to collect expert knowledge for the up-to-date model – that is, up-to-date information about security goals, their levels and information security activities areas and their realization levels dependency matrix and up-to-date their levels realization costs and effectiveness's. And, as independent IT security activities is source for quite serious problems, to cover IT security problems in more detail and correct way we have to accept dependencies between lines in Dependencies Matrix - to describe these dependencies in addition to Dependencies Matrix use (find or work out) the IT security or IT security activities areas Dependencies Graph.

Because the independence of security activity areas was required by the Dynamic Programming (DP) method our aim was to apply an alternative method for optimization and we decided to use an evolutionary algorithm as a universal method for complex optimization in many fields. The evolutionary algorithm starts each optimization process from the beginning and therefore it does not have any problems related to independence and additivity.

As the evolutionary optimization is independent for all possible or interesting budget levels and intervals it leads to possibility to use a graph based security model. The graph based security model is a new and dynamical framework for security management. The new graph model gives us possibility to calculate the most needed/wanted reliability for a specific IT security System (also often named as Confidence) and Security Efficiency (SE), which value can be expressed as $SE = \text{Information Value} / \text{Real Losses} = 1 / (1 - \text{Confidence})$.

Our main ideas are:

- Use metrics to determine information systems security requirements - i.e. use high level risk analysis (levels of security goals) as IT security metrics;
- Secure IT systems and their information in an economically rational/optimal manner – i.e. accordingly to data security requirements;
- The important issue in defining and implementing security measures is the economic efficiency of security activities, that is: we want to get the best results for our money - to minimize the costs and to maximize the integral security confidence.

2. Graded security model

The graded security model has been in use for a long time in the high-risk areas like nuclear waste depositories, radiation control etc. (DOE 1999, see also Kivimaa 2009 for details). In IT security is also reasonable to apply a methodology that allows one to select rational security measures based on graded security, and taking into account the available resources, instead of using only hard security constraints prescribed by standards that usually do not include economic parameters - the cost and efficiency of implemented security measures.

The ideas of graded security were used on the US Department of Energy security model (DOE 1999) and on its updated NISPOM version (NISPOM 2006).

In the NISPOM model 14 graded security activities areas are defined and 15÷20 left only on base levels. As the NISPOM model is meant for protection of critical information infrastructure it is obvious that these base levels are the highest possible implementation levels. But for institutions having less critical IT security these NISPOM areas on the base level have different possible implementation levels too – i.e. theoretically they are graded too (look Figure 1).

But the matrix based model has one quite serious limitation – in table we have no good possibilities to consider dependencies between table columns and rows – that is, there is not any good way to describe really existing additive and dependent nature in IT security goals and activities areas (Kivimaa 2009).

2.1 Graph based security model

It is possible to write dependencies between the matrix rows as functions into cells, but much more understandable and comprehensive results (understandable in one look) if we represent collection of rules as a graph structure. At the same we are no more limited to weighted average only, with graph we get possibility to calculate for decision makers some very interesting and important parameters about achieved security level - confidence and security efficiency (in more details look 2.2).

The graded IT security graph is based on the main ideas from the “(People - Process – Technology) and Organization” Business Model for IT security (ISACA 2009). Based on this and the IT security Dependency Matrix (Figure 1), containing security areas and their levels, a Bank IT security Graph (Figure 2) is formed.

There are two important principles in IT security that are based on the graph and are much more visible and understandable:

- A chain is only as strong as the weakest link – in some IT security areas we must have valid reliability level otherwise overall reliability of security system will be 0 (look Figure 2 – mainly people, SW, Power, HW, LAN and AntiMalware) - so called *must-be elements* in the graph (look Figure 2).

- Layered security / defence in depth – we have a lot security activities areas that are parallel to so called must-be areas that make possible to raise reliability of these must-be areas (Figure 2).

| Security Activities | | Reference | Confidentiality | | | | Integrity | | | | Availability | | | | |
|---|---|-----------|-----------------|--------------|--------|------------|---------------------------|---------------------------------|-----------------------------|--------------------------------|------------------------|--------------------------|----------------------------|----------------------------|---|
| | | | C0 | C1 | C2 | C3 | I0 | I1 | I2 | I3 | A0 | A1 | A2 | A3 | |
| Identified Security Goals | | | Public | Confidential | Secret | Top secret | Identification not needed | Non-authorized usage identified | Authorized usage identified | Identification usable in court | Delay measured in days | Delays measured in hours | Delays measured in minutes | Delays measured in seconds | |
| G1: Organization of information security | | | | | | | | | | | | | | | |
| 1 | Security Documentation | SDOC | CIA | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 2 | Risk Assessment and Treatment | RISK | CIA | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 3 | Security Accrediting | SECA | CIA | | | 1 | 2 | | | 1 | 2 | | | 1 | 2 |
| G2: Human resources security | | | | | | | | | | | | | | | |
| 4 | IT Human Resource Management | HRM | CA | 0 | 2 | 3 | 4 | | | | | 0 | 2 | 3 | 4 |
| 5 | Awareness Training | AWT | CA | 1 | 2 | 3 | 4 | | | | | 1 | 2 | 3 | 4 |
| G3: Physical and environmental security | | | | | | | | | | | | | | | |
| 6 | Perimeter Security(Physical Security) | PSEC | C | | 2 | 3 | 4 | | | | | | | | |
| 7 | Communication & Process Support IT Systems (DBserver, failserver, printserver, meilisüsteem, e- | C&P S | CIA | 0 | 2 | 3 | 4 | 0 | 2 | 3 | 4 | 0 | 2 | 3 | 4 |
| 8 | Personnel Working Environment(Physical Security) | ENV | C | | 2 | 3 | 4 | | | | | | | | |
| 9 | Personnel Workplace Equipment | WPE | A | | | | | | | | | 0 | 1 | 2 | 3 |
| 10 | Power | POW | A | | | | | | | | | 0 | 2 | 3 | 4 |
| 11 | Data Centres | DC | CIA | 0 | 2 | 3 | 4 | 0 | 2 | 3 | 4 | 0 | 2 | 3 | 4 |
| G4: Information systems acquisition, development and maintenance | | | | | | | | | | | | | | | |
| 12 | Outsourcing (incl. 3rd parties support) | OUTS | CIA | 0 | 1 | 2 | 3 | 0 | 2 | 3 | 4 | 0 | 2 | 3 | 4 |
| 13 | Purchased SoftWare | SW | CIA | 0 | 2 | 3 | 4 | 0 | 2 | 3 | 4 | 0 | 2 | 3 | 4 |
| 14 | Self-developed SoftWare | DEV | CIA | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 |
| G5: Access control | | | | | | | | | | | | | | | |
| 15 | Access Rights Management | ARM | CA | | 1 | 2 | 3 | | | | | | 1 | 2 | 3 |
| 16 | Network Access Control | NAC | CI | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | | | | |
| 17 | Mobile computing and teleworking | MOB | CA | 0 | 2 | 3 | 4 | | | | | 0 | 2 | 3 | 4 |
| G6: Communications and operations management (ISO 17799) | | | | | | | | | | | | | | | |
| 18 | Internal network security | LAN | CA | 0 | 1 | 2 | 3 | | | | | 0 | 2 | 3 | 4 |
| 19 | External network security (incl. PerimProt, IDS/IPS, ...) | WAN | CA | 1 | 2 | 3 | 4 | | | | | 1 | 2 | 3 | 4 |
| 20 | Malware Handling | AM | CI | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | | | | |
| 21 | Information exchange policies and procedures | ENC | CIA | | 2 | 3 | 4 | | 2 | 3 | 4 | | 2 | 3 | 4 |
| 22 | Transaction Integrity | TINT | I | | | | | | | 1 | 1 | | | | |
| 23 | Data backup and Restoration | BCK | IA | | | | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 24 | Data Archiving | ARCH | IA | | | | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| G7: Compliance | | | | | | | | | | | | | | | |
| 25 | External Regulations | REG | CI | | 1 | 1 | 2 | | 1 | 1 | 2 | | | | |
| 26 | Audit Capability | AUD | CI | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | | | | |
| G8: Information security incident management | | | | | | | | | | | | | | | |
| 27 | Audit Trail | LOG | C | 1 | 2 | 3 | 4 | | | | | | | | |
| 28 | Monitoring(Help Desk) | MON | A | | | | | | | | | | 2 | 3 | 4 |
| 29 | IT Governance(IT quality, fiduciarity & security manage | IT Gov | CIA | 1 | 2 | 3 | 4 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 |
| G9: Business continuity management | | | | | | | | | | | | | | | |
| 32 | Business Continuity Management(main input to ITS R) | BCM | A | | | | | | | | | | 1 | 2 | 3 |
| 30 | IT Systems Recovery(Redundancy jms, sisend BCM'is | ITS R | A | | | | | | | | | | 2 | 3 | 4 |
| 33 | Crisis Management(tagab, et max IT riskid ~5% Panga k | CM | A | | | | | | | | | | 1 | 2 | 3 |
| G10: Asset Management | | | | | | | | | | | | | | | |
| 31 | Asset Management | ASM | A | | | | | | | | | | 2 | 3 | 4 |
| 35 | | | | | | | | | | | | | | | |

Figure 1: IT security dependency matrix for a bank

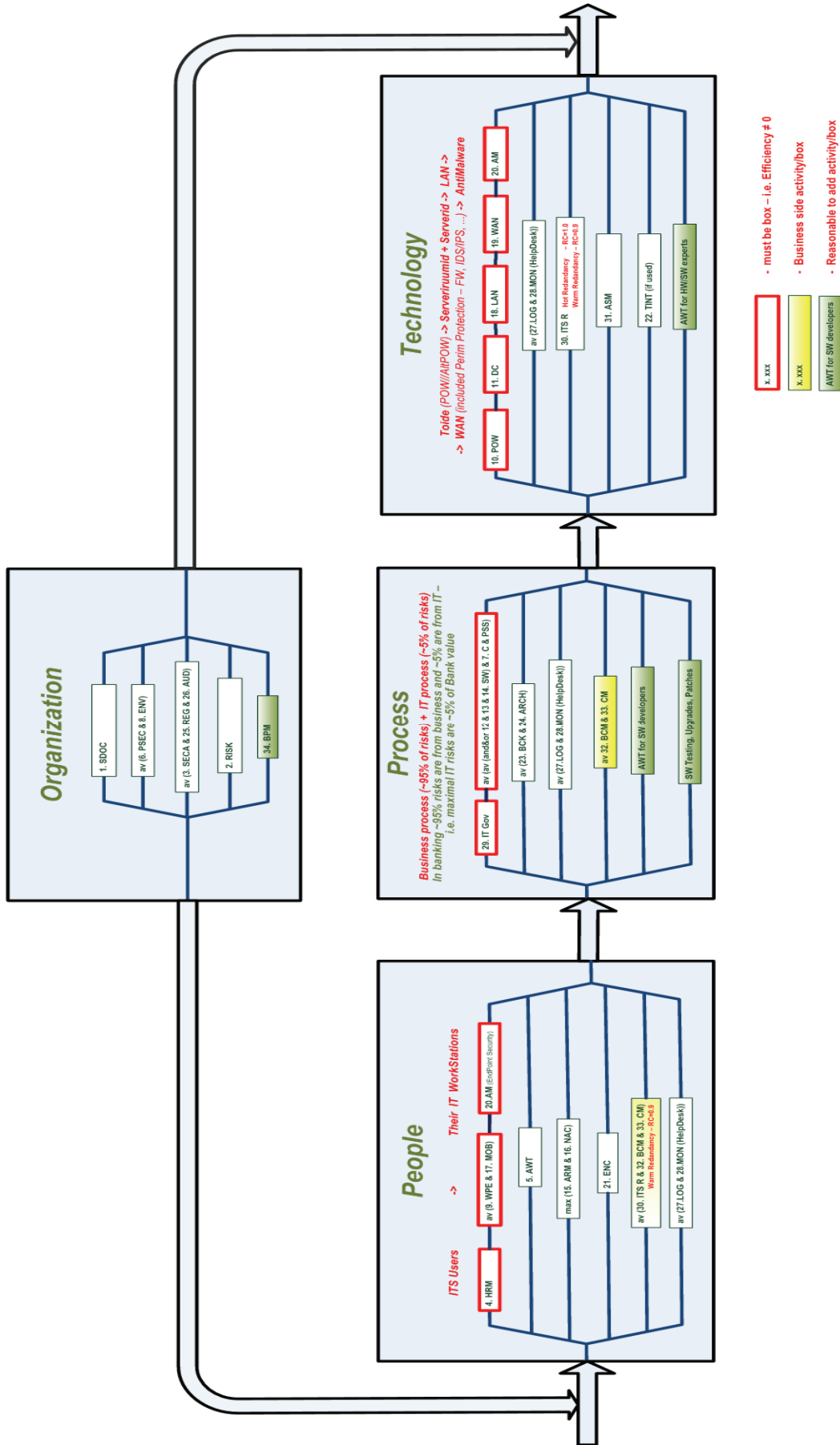


Figure 2: IT security dependency graph for a bank

2.2 Model optimization

We are building a model that binds security measures (grouped by security activities areas) with costs and confidences of achieved the security goals and their levels. We introduce a fitness function that presents by one numeric value the integral confidence of achieved security level. This allows us to formulate a problem of selecting security measures as an optimization problem in precise terms. However, we still have two goals: to minimize the costs and to maximize the integral security confidence. This problem will be solved by means of building a Pareto optimality trade-off curve that explicitly shows the relation between used resources and security confidence (Figure 3).

Knowing the available resources, we can find the best possible security level that can be achieved with the available resources and find the security measures to be taken. From the other side – if the required security level is given we can find the resources needed and the measures that have to be taken. This requires solving an optimization problem for each value of resources.

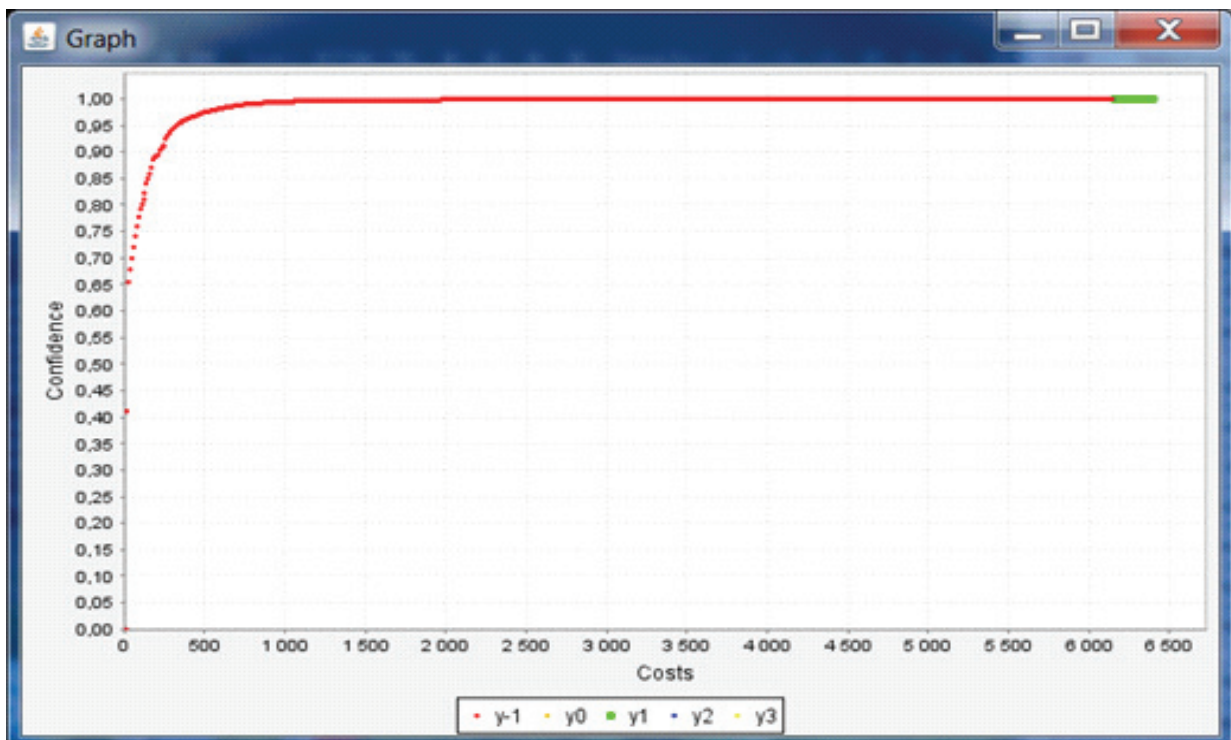


Figure 3: Search of optimal security along resource dimension – Pareto optimality trade-off curve

To calculate Pareto set/curve for GSM we have used/tested three possible optimization techniques:

- Brute Force
- Dynamic Programming
- Evolutionary Algorithms

And all approaches have their pluses and minuses. The first area for problems is calculations time needed for optimization (in more detail look 2.2.1).

Although the Dynamic Programming method is very good way to become free from calculation time problems (optimizations time for medium consumer desktop PC is excellent – minute or two), the DP has quite serious other limitations:

- Security activities areas/security measures groups must be not dependent from each other
- Their levels/security measures to realize their levels must be additive
- Practically impossible to specify alternative and very close optimization results.

The best capabilities has the evolutionary algorithm – it has no problems with dependency/independency, additive/non-additive and matrix/graph, it finds all alternative or very close results for all possible and

interesting cost-levels and the main advantage is that evolutionary optimization starts optimization for all possible and/or interesting budget points from the very beginning. The only possible problem is related to calculations time - the parameters for optimization have to be optimal (in more detail look 2.2.1 and 3.1).

2.2.1 The computational complexity of the optimization task

For comparing three optimization methods we will find calculation times for all three optimization methods for small and medium not IT-critical enterprises (~10 security activities areas) and for bigger IT-critical enterprises (for the Bank ~30 security activities areas):

3. Brute force

We have to calculate and compare qk^n possible variations (q is the number of possible values of security budget levels, n is the number of security measure groups or security activities areas, k is the value of possible implementation levels for security measure group/security activities area, quite prevalently used 3 or 4):

- For 10 security activities areas is required testing of $100 \cdot 4^{10} \approx 100 \cdot 10^6$ variations,
- For 30 security activities areas is required testing of $100 \cdot 4^{30} \approx 100 \cdot 10^{18}$ variations,

In more detailed IT security handling (n) optimization time increase is exponential and if to consider that medium consumer PC can perform optimization for 10 security activities areas (for small and not IT-critical institution, $\sim 100 \cdot 10^6$ calculations and comparisons) in a minute then Brute Force optimization for bigger and IT-critical institution will take hundreds years.

4. Dynamic programming

We have to calculate and compare q^2kn possible variants (q is the number of possible values of security budget levels, n is the number of security measure groups or security activities areas, k is the value of possible implementation levels for security measure group/security activities area, quite prevalently used 3 or 4):

- For 10 security activities areas is required testing of $100 \cdot 100 \cdot 4 \cdot 10 = 0,4 \cdot 10^6$ variations,
- For 30 security activities areas is required testing of $100 \cdot 100 \cdot 4 \cdot 30 = 1,2 \cdot 10^6$ variations.

In more detailed IT security handling optimization time increase is linear and consequently n rise even the magnitude does not lead to any calculations time problems.

5. Evolutional

The number of variants required to calculate/compare by this algorithm is:

$q \cdot \text{Population size} \cdot \text{Number of Generations} \cdot \text{Number of Repeats}$.

And as based on results of meta-level optimization (see 3.1.2) 'Population size' = $n \cdot 3$, 'Number of Generations' = $n \cdot 4$ and 'Number of Repeats' = 3 (q is the number of possible values of security budget levels, n is the number of security measure groups or security activities areas) and optimal number of variants to calculate and compare is $36 \cdot q \cdot n^2$:

- For 10 security activities areas is required testing of $36 \cdot 100 \cdot 10^2 = 0,36 \cdot 10^6$ variations,
- For 30 security activities areas is required testing of $36 \cdot 100 \cdot 40^2 = 3,24 \cdot 10^6$ variations.

For more detailed IT security handling optimization time increase is quadratic and consequently is quite important to use optimal parameters in optimization.

In conclusion:

- Optimization time is critical,
- The Brute Force optimization method is inappropriate for more complex cases,
- The Dynamic Programming based optimization method has not any problems related to calculations time,
- For the Evolutionary method it is important to use the optimal optimization parameters.

5.1.1 GS graph-based model reliability/confidence calculations

The main idea for optimization is to achieve graph's maximal Confidence with minimal Costs – i.e. Pareto set or Pareto frontier for GSM Costs or Confidence.

5.1.2 Reliability (alias confidence) of series systems of "n" identical and independent components

A series system is a configuration such that, if any one of the system components fails, the entire system fails. Conceptually, a series system is one that is as weak as its weakest link. A graphical description of a series system is shown in Figure 4.

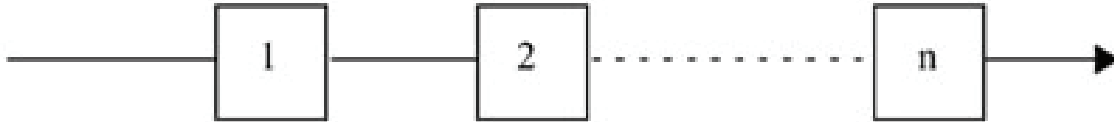


Figure 4: Representation of a series system of "n" components

Engineers are trained to work with system reliability $[R_S]$ concepts using "blocks" for each system element, each block having its own reliability for a given mission time T:

$$R_S = R_1 \times R_2 \times \dots \times R_n \text{ (if the component reliabilities differ, or)}$$

$$R_S = [R_i]^n \text{ (if all } i = 1, \dots, n \text{ components are identical)}$$

A set of n blocks connected in series can be replaced with a single block with the Reliability/Confidence R_S/C_S .

5.1.3 Reliability (alias confidence) of parallel systems

A parallel system is a configuration such that, as long as not all of the system components fail, the entire system works. Conceptually, in a parallel configuration the total system reliability is higher than the reliability of any single system component. A graphical description of a parallel system of "n" components is shown in Figure 5.

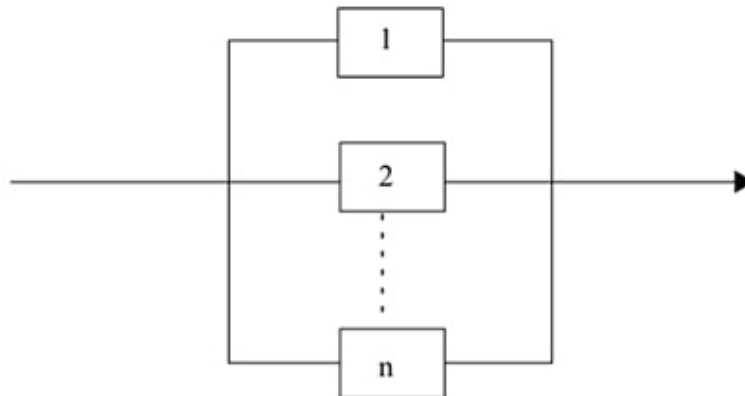


Figure 5: Representation of a parallel system of "n" components

Reliability engineers are trained to work with parallel systems using block concepts:

$$R_S = 1 - (1 - R_i) = 1 - (1 - R_1) \times (1 - R_2) \times \dots \times (1 - R_n); \text{ if the component reliabilities differ, or}$$

$$R_S = 1 - [1 - R]^n; \text{ if all "n" components are identical: } [R_i = R; i = 1, \dots, n].$$

A set of n blocks connected in parallel can be replaced with a single block with the reliability/Confidence R_S/C_S .

By recursively replacing the series and parallel subsystems by single equivalent elements we can obtain the Reliability/Confidence R_S/C_S for entire graph/system.

5.1.4 Specifics for GS graph-based model confidence calculations.

In GSM we have the only so called must-be serial box's and logic „if any one of the system components fails, the entire system fails“ is exact and perfect.

But with parallel components is situation a bit more complicated. For full redundant security activities (for example, HW and Redundant HW) is principle „as long as not all of the system components fail, the entire system works exact, but if we have in parallel must-be security activity area with activities areas trying to improve the must-be activity Confidence (as example HW and Logging/Monitoring) then we have not fully redundant situation – we must bring in Redundancy Coefficient R_c .

Practically $R_c = 1 \div 0,1$ - for full redundancy $R_c = 1$ and parallel to must-be activity with less Redundancy than 0,1 is pointless.

If for full redundancy $C = 1 - (1 - C_{1_mb}) * (1 - C_2) = C_{1_mb} + C_2 (1 - C_{1_mb})$

then bringing in Redundancy Coefficient R_c for Not-Full-Redundant parallel situations

$C = 1 - (1 - C_{1_mb}) * (1 - R_c * C_2)$ or $C = C_{1_mb} + R_c * C_2 * (1 - C_{1_mb})$

By recursively replacing the series (must-be) and parallel subsystems by single equivalent elements we can obtain the Reliability/Confidence R_s/C_s for entire graph/system and the new graph model gives us possibility to calculate for IT managers/decision makers the most needed/wanted reliability for a specific IT security System (also often named as Confidence) and Security Efficiency (SE), which value can be expressed as

$SE = IT\text{-risks} / \text{Real Losses} = 1 / (1 - C_s).$

For example, on Figure 6 SE is produced as a function from IT security activities and measures of costs.

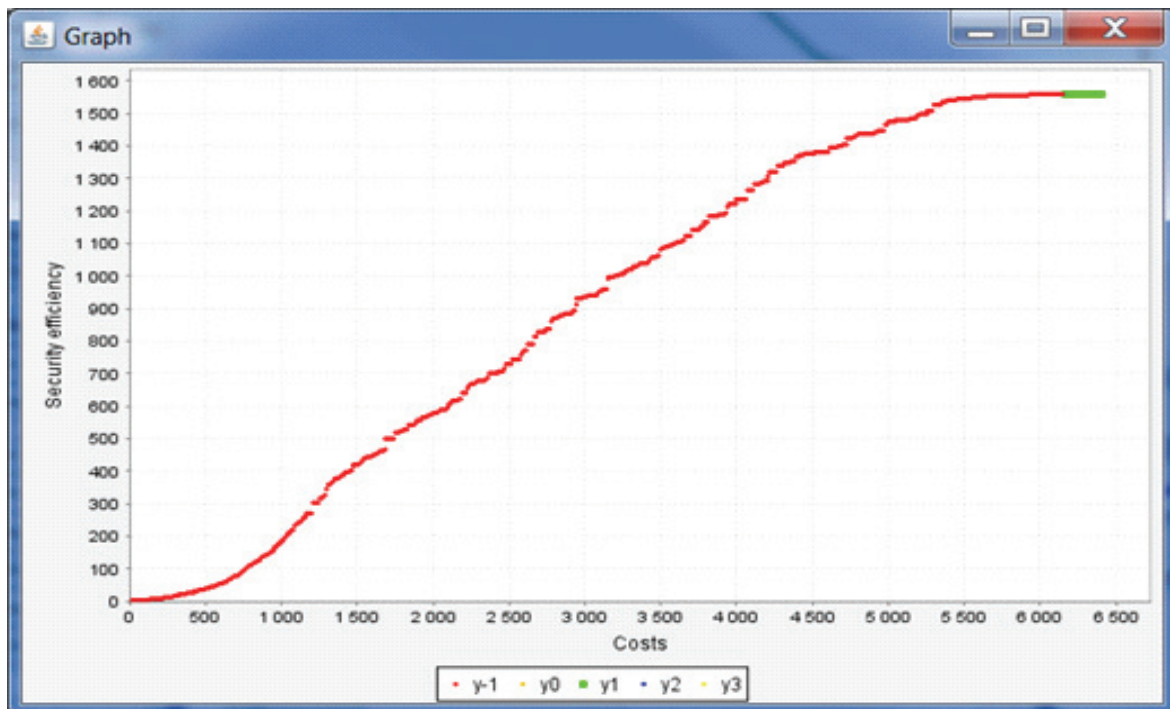


Figure 6: SE = f (costs)

6. Evolutionary algorithms

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). 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 electronic computers. 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:

- Each candidate solution to the optimization problem is represented as an individual. The set of individuals are named as a population.
- The quality of a candidate solution is measured by a fitness function. Fitter solutions have a higher probability to survive and to contribute their characteristics to offspring (next generation).
- Variation operators (e.g., crossover, mutations) are applied to the individuals that modify the population of solutions dynamically.
- 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 for the new generations the fittest individuals are selected.

6.1 Meta-level optimization of evolutionary algorithms

The aim of this work is to optimize the parameters of an evolutionary algorithm. As the optimization process is based on randomness it makes the speed of the problem solving task rather variable. There are no hard and fast rules for choosing appropriate values for the parameters (Cicirello & Smith, 2000). The first scientist, who put a considerable effort into finding parameter values, was De Jong (1975). He tested different values experimentally and concluded that the following parameters give reasonable performance for his test functions: population size 50, crossover 0.6 and mutation rate 0.001 (see also for details Eiben, Hinterding, & Michalewicz, 1999). But those values are suitable for the problem that he had at hand. It has been shown that it is not possible to find parameter values which are optimal for all problem domains (Wolpert, & Macready, 1997) therefore each problem need its own approach and different set of parameters.

A widely practised approach to identify a good set of parameters for a particular class of problem is through experimentations and using the trial-and-error approach. As the evolutionary approach is mostly based on the trial-and-error to move through the search space therefore it would be reasonable to use the evolutionary algorithm itself to optimize its parameters and such approach is called as a meta-level optimization (Cicirello & Smith, 2000). The main weakness of this approach is that it is computationally expensive and takes a lot of time.

There are two ways to improve the performance of the evolutionary algorithm. The strategy can either be static or adaptive (Aine, Kumar, & Chakrabarti, 2006). For static framework, the parameter values are decided at the start of the algorithm and the decision is not revised during runtime. The static model works well when there is little or no uncertainty about the progress of the algorithm. For algorithms where the progress is not predictable and different parameter settings are suitable at different stages, a dynamic monitoring based strategy is preferred. In the dynamic case, the control decision is updated during runtime by monitoring the progress of the algorithm for a particular run. As the IT security costs optimization task is rather stable and does not include many uncertainties, we decided to find out a static set of parameters rather than develop a dynamic framework for parameter changes.

6.1.1 Meta-level optimization set-up

An individual in the optimization task was represented as a vector consisting of 10 elements. The elements represented the adjustable set of parameters: Repeat – how many times to repeat optimization process, Population – population size, Tournament – tournament size (number of individuals in a subset), Generations – a predefined number of generations, Crossover – probability of applying crossover operator (value 0.49 means that in 49% cases the crossover occurs), Mutate – probability of mutation, Swap – probability of swapping, Inversion – probability of inversion, Insertion – probability of insertion,

and Displacement – probability of displacement. During the meta-level optimization process a candidate solution was optimized based on these parameters.

An important question was how to measure the fitness of the meta-level evolutionary optimization. We had two optimization goals, first, to find maximum level of confidence and second, to find it as fast as possible. Therefore we had to combine the measure of confidence and time. As each optimization was repeated r times the value of meta-level fitness function F was calculated as average of fitness of original task minus time:

$$F = \text{sum}(c_i - t_i) / r$$

where c_i is the confidence level and t_i is the calculation time in seconds of i -th experiment (see curve in Figure 7).

6.1.2 Results of meta-level optimization

We performed experiments with the data (Figure 1) consisting of 33 security activity areas. From the original data we formed 6 sets consisting of 13, 17, 21, 25, 29 and 33 areas. The parameters for meta-level optimizer were as follows: population size 75, tournament size 15 and the number of generations 75, crossover rate 0.9 and mutation rate 0.7.

The optimization process took almost two and half days. As we could see on the detailed graph (Figure 7) the fine tuning of the meta-level optimization took some time to find the optimal level.

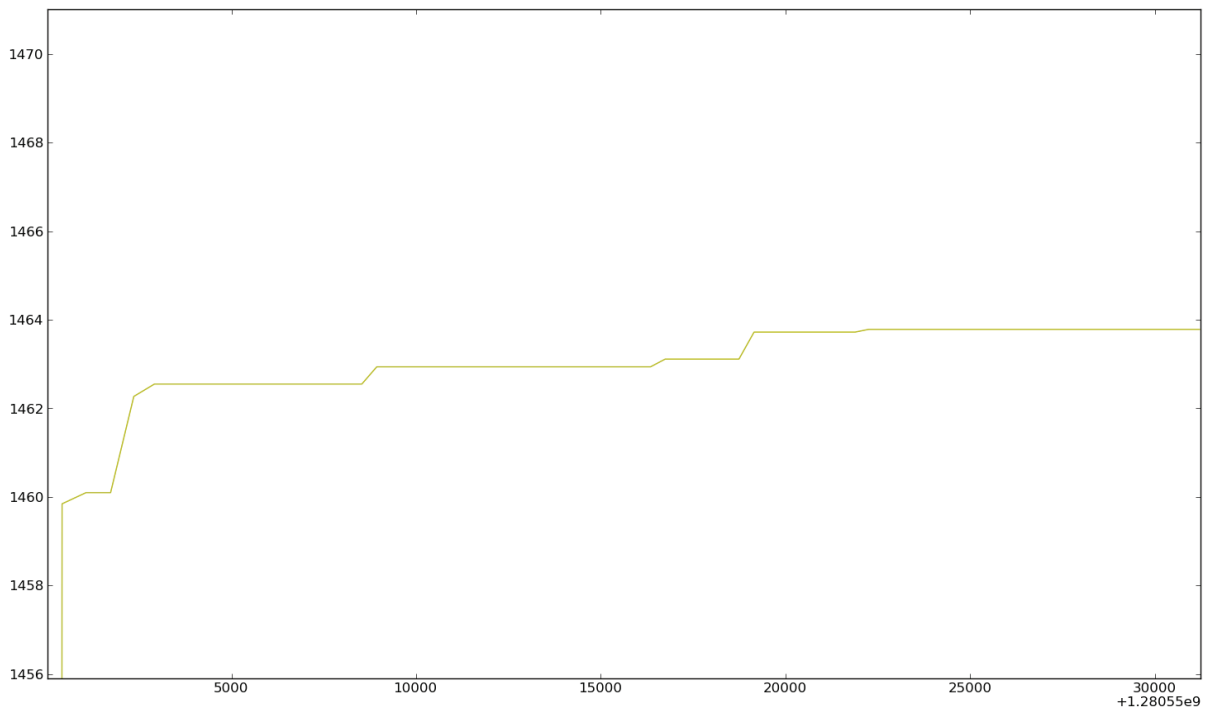


Figure 7: The fitness value of the meta-level optimization task (upper part of the fitness curve)

Average results of the optimization process are given in Table 1.

Table 1: Average values of parameters as a result of meta-level optimization

| No | Pop | Tournament | Generations | Crossover | Mutation | Swap | Inversion | Insertion | Displacement |
|----|-------|------------|-------------|-----------|----------|------|-----------|-----------|--------------|
| 13 | 28.86 | 41.43 | 42.86 | 0.82 | 0.7 | 0.58 | 0.19 | 0.15 | 0.15 |
| 17 | 35.57 | 69.14 | 67.43 | 0.85 | 0.89 | 0.63 | 0.14 | 0.16 | 0.12 |
| 21 | 46.57 | 40.71 | 70.71 | 0.8 | 0.88 | 0.53 | 0.1 | 0.13 | 0.12 |
| 25 | 43.43 | 31.86 | 95.86 | 0.85 | 0.77 | 0.61 | 0.08 | 0.13 | 0.15 |
| 29 | 48.86 | 65.29 | 92.43 | 0.8 | 0.89 | 0.74 | 0.07 | 0.1 | 0.16 |
| 33 | 61.43 | 37.71 | 96.43 | 0.91 | 0.74 | 0.72 | 0.13 | 0.06 | 0.13 |

As we calculated correlation coefficients (Table 2) we could see that there is strong linear correlation between the number of security activity areas (the size of task) and the number of individuals in a population ($r=0,95$) and the number of generations ($r=0,92$). There is also positive correlation between the size of task and crossover probability (0.45). With the most other probability values the correlation is negative.

Table 2: Correlation coefficients of all 35 selected results

| | No | Pop. | Tourn. | Gen. | Crossover | Mutate | Swap | Inversion | Insertion | Displace. |
|--------------|-------|-------|--------|-------|-----------|--------|-------|-----------|-----------|-----------|
| No | 1 | 0.95 | -0.13 | 0.92 | 0.45 | 0.06 | 0.73 | -0.64 | -0.92 | 0.16 |
| Population | 0.95 | 1 | -0.21 | 0.82 | 0.48 | 0.08 | 0.57 | -0.53 | -0.93 | -0.12 |
| Tournament | -0.13 | -0.21 | 1 | -0.1 | -0.29 | 0.7 | 0.37 | -0.06 | 0.24 | -0.04 |
| Generations | 0.92 | 0.82 | -0.1 | 1 | 0.4 | 0.18 | 0.62 | -0.8 | -0.71 | 0.16 |
| Crossover | 0.45 | 0.48 | -0.29 | 0.4 | 1 | -0.47 | 0.4 | 0.2 | -0.51 | -0.28 |
| Mutate | 0.06 | 0.08 | 0.7 | 0.18 | -0.47 | 1 | 0.05 | -0.56 | 0.18 | -0.3 |
| Swap | 0.73 | 0.57 | 0.37 | 0.62 | 0.4 | 0.05 | 1 | -0.29 | -0.7 | 0.38 |
| Inversion | -0.64 | -0.53 | -0.06 | -0.8 | 0.2 | -0.56 | -0.29 | 1 | 0.33 | -0.21 |
| Insertion | -0.92 | -0.93 | 0.24 | -0.71 | -0.51 | 0.18 | -0.7 | 0.33 | 1 | -0.12 |
| Displacement | 0.16 | -0.12 | -0.04 | 0.16 | -0.28 | -0.3 | 0.38 | -0.21 | -0.12 | 1 |

In Figure 8 we could see that the probabilistic values of variation operators (Crossover, Mutation and Swap) had quite high values and the others value was rather small and even diminished as the problem grows. Probably their computational cost was relatively high comparing the gain of fitness.

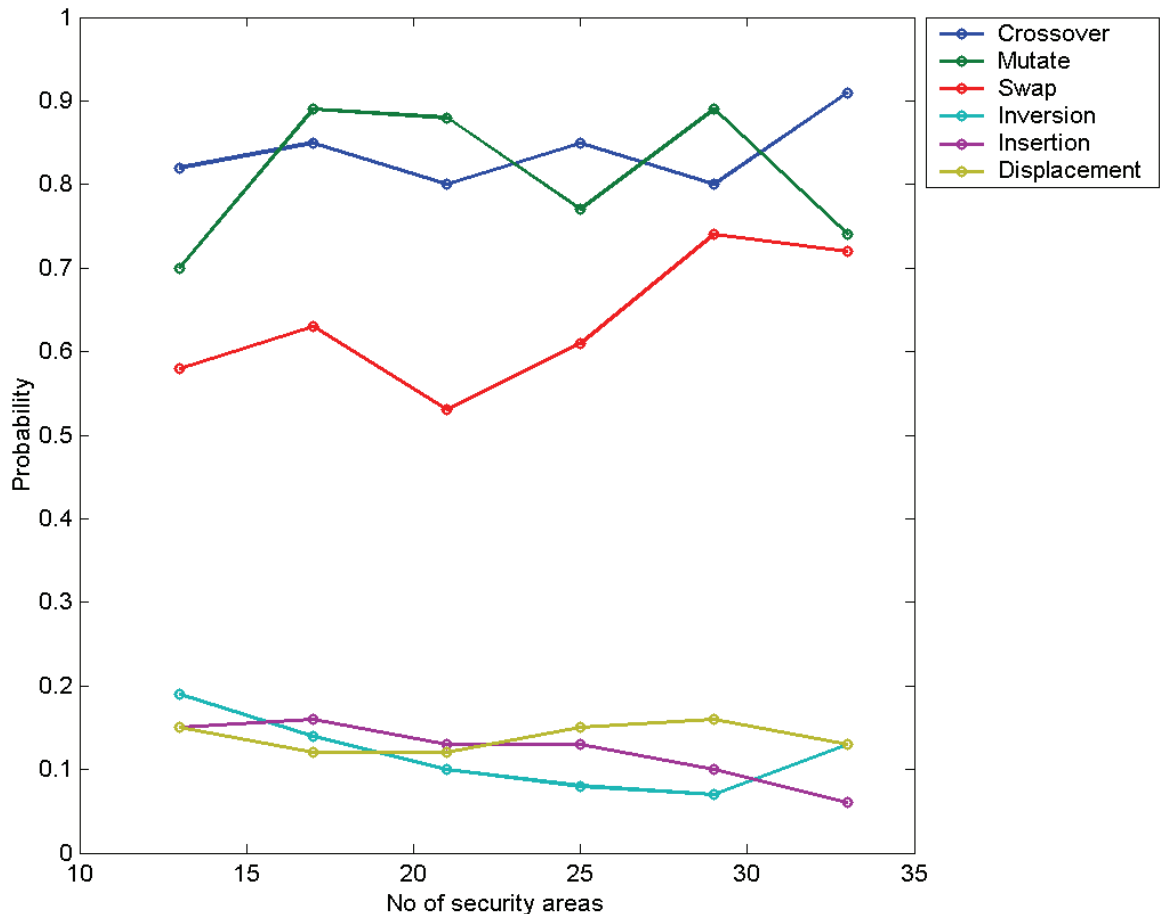


Figure 8: Change of probability of variation operators

In Figure 9 we could see that there is a clear linear relation between the problem size and the population size and the number of generations.

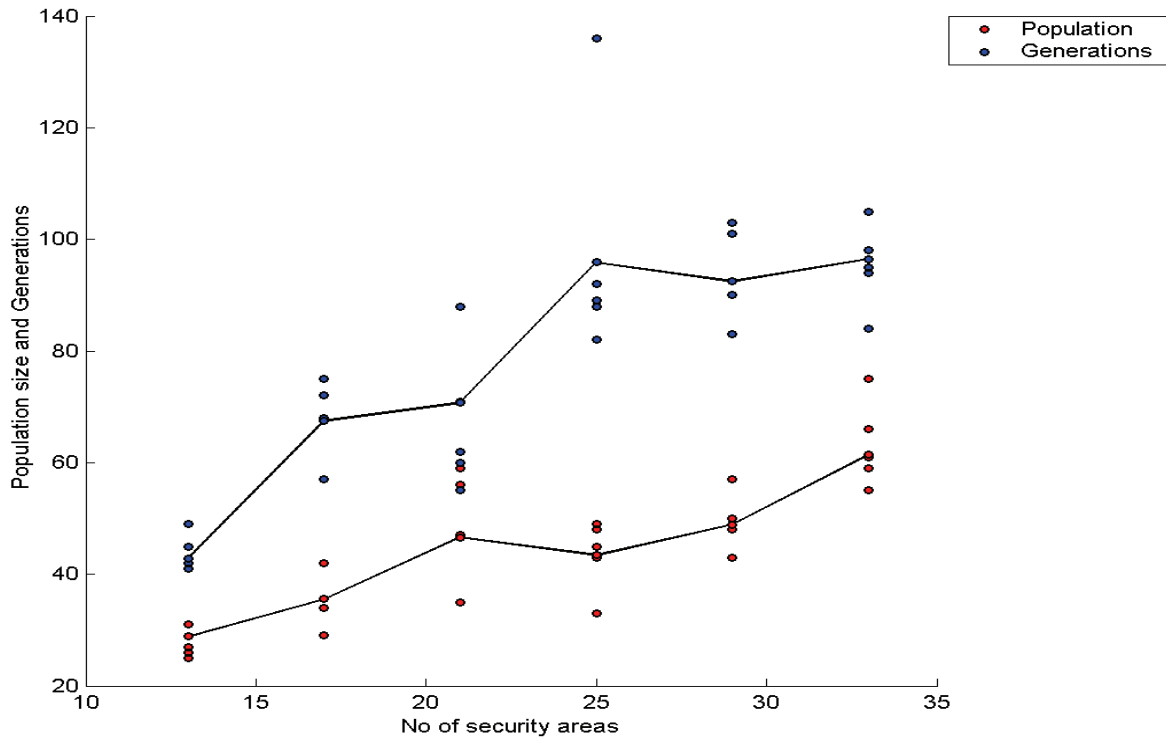


Figure 9: Distribution of population and generation values and their mean value (line)

Based on the measurements we were able to generate formulas to specify the parameters of evolutionary optimizer. As we added to the mean value and the standard deviation $\mu + \sigma$ to get rough estimate for the population related values (e.g., based on the mean value of Generations / Number security activity areas $\mu = 3.429$, standard deviation $\sigma = 0.5688$, we can calculate the coefficient $3.429 + 0.5688 \approx 4$). The results could be as follows:

| | |
|-----------------|---------|
| repeat | 3 |
| population size | $N * 3$ |
| tournament size | 50 |
| generations | $N * 4$ |

where N is the number of security activity areas as the number of security levels is 4.

As there was a tendency to move closer to certain values we decided to use in further optimizations the following parameter set for variation operators:

| | |
|-------------------|------|
| crossover rate | 0.9 |
| mutation rate | 0.8 |
| swap rate | 0.6 |
| inversion rate | 0.1 |
| insertion rate | 0.07 |
| displacement rate | 0.11 |

As we could predict optimal population related parameters and also identified optimal values for probability operator values we could estimate optimization time and to perform optimization tasks much faster.

7. Conclusions

We have performed an analysis to identify linear coefficients for estimating the parameter values of the evolutionary algorithm. As a result we have found a way to calculate the value for population size and the number generations that are based on the problem size and also identified optimal parameter set for variation operators. It makes the use of evolutionary algorithm more efficient and enables us to increase the optimization speed. As there are certain restrictions related to the other optimization techniques the

evolutionary approach also enables us to enhance the IT security methodology and a new graph-based model is proposed.

But wider application of the graph-based model will depend on the availability of expert knowledge or statistics that binds costs and security confidence values with the security measures. This expert data will depend on the type of the infrastructure where information must be protected - different for different countries and economy areas. The only realistic solution is an expert system that can be adjusted by experts to suit concrete situations. Therefore some further work is needed to enhance the model and provide appropriate expert knowledge to turn the model more accurate.

References

- Aine, S., Kumar, R., and Chakrabarti, P.P. (2006) "Adaptive Parameter Control of Evolutionary Algorithms Under Time Constraints", in A., Tiwari, J. Knowles, E. Avineri, K., Dahal, and R., Roy (Eds.), *Applications of Soft Computing*, Berlin, Springer, pp. 373–382.
- Cicirello, V. A., and Smith, S. F. (2000) "Modeling GA performance for control parameter optimization", in D., Whitley, D., Goldberg, E., Cant-Paz, L., Spector, I., Parmee, and H., Beyer (Eds.), *GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference*, Las Vegas, NV, pp. 235–242.
- De Jong, K. (1975) "The analysis of the behavior of a class of genetic adaptive systems", Ph.D. dissertation, Department Computer Science, University of Michigan, Ann Arbor, MI.
- DOE (1999) *Classified Information Systems Security Manual*. Retrieved February 1, 2010, from https://www.directives.doe.gov/directives/archive-directives/471.2-DManual-2/at_download/file.
- Dracopoulos, D. C. (2008) "Evolutionary Learning", in B. Wah (Ed.), *Wiley Encyclopedia of Computer Science and Engineering*. New York, John Wiley and Sons.
- Eiben, A. E., Hinterding, R., and Michalewicz, Z. (1999) "Parameter control in evolutionary algorithms", *IEEE Transactions on Evolutionary Computation*, Vol 3, No. 2, pp. 124–141.
- Eiben, A. E., and Smith, J. E. (2003) *Introduction to Evolutionary Computing*, Berlin, Springer.
- 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.
- ISACA (2009) "An Introduction to the Business Model for Information Security," ISACA.
- Kirt, T., and Kivimaa, J. (2010) "Optimizing IT security costs by evolutionary algorithms", in C. Czosseck, and K. Podins, (Eds.), *Conference on Cyber Conflict Proceedings 2010*, Tallinn, Estonia, Cooperative Cyber Defence Centre of Excellence Publications, pp. 145–160.
- 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*, Reading, UK, Academic Publishing Limited, pp. 142–153.
- Kivimaa, J. Ojamaa, A. and Tyugu, E. (2009) "Graded security expert system", in *CRITIS 2008: Third International Workshop on Critical Information Infrastructure Security*, Rome, Springer.
- Michalewicz, Z., and Fogel, D. B. (2004) *How To Solve It: Modern Heuristics*, Berlin, Springer.
- NISPOM (2006) "National Industrial Security Program Operating Manual," U.S. Department of Defense..
- Ojamaa, A., Tyugu, E., and Kivimaa, J. (2008) "Pareto-optimal situation analysis for selection of security measures", in *Military Communications Conference MILCOM 2008: Unclassified Proceedings*, Piscataway, NJ, IEEE, pp. 3224–3230.
- Wolpert, D., and Macready, W. G. (1997) "No free lunch theorems for optimization", *IEEE Transactions on Evolutionary Computation*, Vol 1, No. 1, pp. 67–82.