CERIAS Tech Report 2005-119

MATCHING INFORMATION SECURITY VULNERABILITIES TO ORGANIZATIONAL SECURITY PROFILES: A GENETIC ALGORITHM APPROACH

by Mukul Gupta, Jackie Rees, Alok Chaturvedi, Jie Chi

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Decision Support Systems 41 (2006) 592–603

Decision Support Systems

www.elsevier.com/locate/dsw

Matching information security vulnerabilities to organizational security profiles: a genetic algorithm approach☆

Mukul Gupta[a,*], Jackie Rees[b,1], Alok Chaturvedi[b,2], Jie Chi[c,3]

[a]*Department of Operations and Information Management, School of Business, University of Connecticut, 2100 Hillside Road, Storrs, CT 06269-1041, United States*
[b]*Krannert Graduate School of Management, Purdue University, 403 West State Street, West Lafayette, IN 47907-2056, United States*
[c]*Purdue e-Business Research Center, Purdue University, 403 West State Street, West Lafayette, IN 47907-2056, United States*

Available online 10 August 2004

Abstract

Organizations are making substantial investments in information security to reduce the risk presented by vulnerabilities in their information technology (IT) infrastructure. However, each security technology only addresses specific vulnerabilities and potentially creates additional vulnerabilities. The objective of this research is to present and evaluate a Genetic Algorithm (GA)-based approach enabling organizations to choose the minimal-cost security profile providing the maximal vulnerability coverage. This approach is compared to an enumerative approach for a given test set. The GA-based approach provides favorable results, eventually leading to improved tools for supporting information security investment decisions.

Keywords: Information security; Genetic algorithms

1. Introduction

Information technology (IT) infrastructure has evolved into a critical strategy-enabling and supporting resource for organizations. The IT infrastructure and information systems, which are so powerful in creating and disseminating knowledge throughout organizations, also possess weaknesses or vulnerabilities. These vulnerabilities range from allowing unauthorized access to the data and information stored within such systems to the full-scale destruction of an organization's infrastructure. Entities such as hackers, terrorists, disgruntled employees and business competitors are on the lookout for any vulnerability in the information systems of organizations and may seek to exploit found weaknesses for psychological, political or economic advantage. These entities pose a serious threat to organizational information systems and the

☆ This research is partially funded by National Science Foundation grants, numbered EIA-0075506 and DMI-01222.

* Corresponding author. Tel.: +1 860 486 6178; fax: +1 860 486 4839.

E-mail addresses: mukul.gupta@business.uconn.edu (M. Gupta), jrees@mgmt.purdue.edu (J. Rees), alok@mgmt.purdue.edu (A. Chaturvedi), chij@purdue.edu (J. Chi).

[1] Tel.: +1 765 494 0320; fax: +1 765 494 1526.

[2] Tel.: +1 765 494 9048; fax: +1 765 494 1526.

[3] Tel.: +1 765 494 2667; fax: +1 765 494 1526.

IT infrastructure upon which the systems reside and often result in negative financial and other repercussions for the affected organizations.

Organizations attempt to prevent unauthorized access and other harm to their systems by using security technologies that address known vulnerabilities in their systems. We refer to the organization's collection or portfolio of security technologies used to protect against vulnerabilities as the security profile. However, any given security technology addresses only specific vulnerabilities and could possibly create additional vulnerabilities. It is worth mentioning that the security profile can only reduce the risk of a particular vulnerability from being exploited. However, for the purposes of this research we assume that if a known vulnerability is covered by a particular security technology, the risk of that vulnerability being exploited is close to zero. Additionally, organizations also must take into account the cost of using each security technology. Therefore, it is usually a difficult decision for organizations to create and manage a security profile that addresses as much vulnerability as possible while minimizing the total cost of the profile.

This paper provides an innovative decision-making approach to selecting the appropriate security technologies to address the vulnerabilities in the system. The approach provides insights to management for managing security risks. This paper demonstrates a technique that could be incorporated into decision support software tools that provide recommendations for choosing the best combination of security solutions that would minimize the risk for the organization.

To facilitate the above decision, we present and evaluate a Genetic Algorithm (GA)-based approach that improves the organization's ability to choose a minimal cost security profile providing maximal vulnerability coverage. We then compare the GA approach to an enumerative approach of evaluating all possible security profiles for a given set of test problems. We perform these evaluations for several combinations and sizes of matrices of vulnerabilities and security profiles.

The remainder of the paper is organized as follows: Section 2 describes relevant background on GAs and how they have been used to address similar problems in the past. Section 3 presents our research definitions of vulnerabilities and security technologies and the method used to match vulnerabilities to specific security technologies. Section 4 contains our model of the problem and Section 5 details the experimental design. Results are reported in Section 6 and conclusions and future directions are given in Section 7.

2. Genetic algorithms for multi-objective optimization problems

The problem of matching vulnerabilities to security technologies can be characterized as multi-objective since we are trying to create security profiles to achieve two objectives: minimize the exposed vulnerabilities of an organization and minimize the cost of the security technologies used to address the vulnerabilities. As will be discussed later in this paper, such multi-objective problems are typically difficult to solve without the use of heuristics. As an example of such a heuristic, genetic algorithm has successfully been used in search, single-objective optimization and machine learning. A general description of the genetic algorithm and how it can be designed and implemented for solving single-criterion optimization problems appears in [5]. Researchers have since developed techniques for applying GAs to multi-objective problems. A survey of various GA-based techniques for solving multi-objective optimization problems is presented in [3]. The first approach involved combining multiple objectives into one objective using addition, multiplication or any other combination of arithmetical operations. It was argued that if the combination of objectives is possible, this is not only one of the simplest approaches but also one of the most efficient ones [3].

A weighted combination of fitness functions to add or subtract values during the schedule evaluation of a resource scheduler, depending on whether or not the constraints were violated was used in [12]. [7,14] also use weighted combination of objectives to address multi-objective problems. Goal programming was developed in [2,6] where decision makers have to assign targets or goals that they wish to achieve for each objective. [10,13] have used goal programming in conjunction with genetic algorithms to solve multi-objective problems. These values are incorporated into the problem as additional constraints and the objective function tries to minimize the absolute deviations from the targets to the objectives. An ε-constraint approach that optimizes one objective function while considering other objective functions as constraints bound by some allowable levels of ε_i was described in [9] and used in [8].

In this paper, we utilize the weighted sum of objectives technique, used in [12] to combine the conflicting objectives of minimizing the security technology costs of addressing vulnerabilities and minimizing the number of uncovered vulnerabilities after implementation of security technologies. The problem with this approach is that it requires some information regarding the range of weights [3]. However, in our problem, a good estimate of the weights of the objectives can be made based on the type and preferences of the organization. Organizations such as financial institutions, for which covering vulnerabilities is critical, will put more emphasis on maximizing the number of vulnerabilities covered, while organizations whose security requirements are not as stringent might wish to minimize the cost of security while still maintaining an adequate security profile.

3. Definition of vulnerabilities and securities

Organizational information systems and their underlying IT infrastructure contain vulnerabilities that can be exploited by various entities. Organizations utilize security technologies to reduce the risk of damage presented by these vulnerabilities. A RAND report [1] identifies a set of generic vulnerabilities potentially present in most organizational information systems and their supporting infrastructures. The report provides a *generic* set of vulnerabilities rather than specific bugs or weaknesses in information system. By utilizing a generic set of vulnerabilities, the problem of novel and previously unheard-of threats can be addressed. For example, a new type of Internet-based attack can be handled using the generic framework, whereas it would be outside a very specific list of known vulnerabilities. However, the generic approach can be easily extended to a more specific set of vulnerabilities and security technologies. These vulnerabilities are classified into seven categories. Organizations can then map their information systems and infrastructure to these vulnerabilities to provide an initial assessment of the state of risk to their systems. This list of vulnerabilities and the variables we used to represent them in our model is presented in Table 1.

Organizations address the vulnerabilities described above in several ways, including changing business processes, employee awareness program and investing in security technologies. Security technologies reduce the vulnerabilities, identify attacks and breaches and react to these attacks and breaches. Each security technology addresses certain vulnerabilities directly by design; they reduce certain other vulnerabilities indirectly as a second order effect. However, security technologies can also directly or indirectly create certain other vulnerabilities in the system. We present the generic security technologies described in [1] and their variable representation in our model in Table 2. These generic security technologies are implemented in the context of the organization's systems using actual security technologies. Similar classifications can be found in a recent NIST report [11].

Table 1
Technology vulnerabilities

Category	Vulnerability	Representation
Inherent Design/ Architecture	Uniqueness	v_1
	Singularity	v_2
	Centralization	v_3
	Separability	v_4
	Homogeneity	v_5
Behavioral Complexity	Sensitivity	v_6
	Predictability	v_7
Adaptation and Manipulation	Rigidity	v_8
	Malleability	v_9
	Gullibility	v_{10}
Operation/ Configuration	Capacity Limits	v_{11}
	Lack of Recoverability	v_{12}
	Lack of Self-awareness	v_{13}
	Difficulty of Management	v_{14}
	Complacency/ Co-optability	v_{15}
Indirect/Nonphysical Exposure	Electronic Accessibility	v_{16}
	Transparency	v_{17}
Direct/Physical Exposure	Physical Accessibility	v_{18}
	Electromagnetic Susceptibility	v_{19}
Supporting Facilities/ Infrastructures	Dependency	v_{20}

Table 2
Generic security technologies

Security	Representation
Heterogeneity	s_1
Static Resource Allocation	s_2
Dynamic Resource Allocation	s_3
Redundancy	s_4
Resilience and Robustness	s_5
Rapid recovery and Reconstitution	s_6
Deception	s_7
Segmentation, Decentralization and Quarantine	s_8
Immunologic Identification	s_9
Self-organization and Collective Behavior	s_{10}
Personnel Management	s_{11}
Centralized Management of Information Resources	s_{12}
Threat/Warning Response Structure	s_{13}

Each of these security technologies addresses some of these vulnerabilities either fully or partially and creates some vulnerability either directly or indirectly. We make a simplifying assumption that if a technology partially covers a vulnerability or indirectly creates a vulnerability, the scope is uniform. In other words, we do not distinguish how severe or mild the value of the variable is. The mapping of these security technologies to vulnerabilities is presented in Table 3 [1]. The organization looking to cover its vulnerabilities by investing in security technologies has two major goals:

(1) Minimize the number of uncovered vulnerabilities after implementation of security technologies.
(2) Minimize the cost of security to cover vulnerabilities.

Achieving these two objectives is not easy since the organizations have to constantly make trade-offs between security costs and allowing some vulnerabilities to be uncovered. Malicious agents can exploit these uncovered vulnerabilities resulting in damages to the organization. In the next section, we present the model used to address this problem.

4. Research model

We now demonstrate that the objective of covering a given set of vulnerabilities while minimizing security costs and new vulnerabilities introduced as a result of

Table 3
Matching security technologies to vulnerabilities

		Security												
		s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}
Vulnerability	v_1			1		1	1			1	−1			
	v_2		1	1	1	0.5	1	1	1		1			
	v_3		−0.5			0.5	1	1	1		1		−1	
	v_4		−1	1	1		1		−1		1			
	v_5	1			−1	0.5			1				−0.5	
	v_6	−1		−1		0.5	1	−1	1	−1	−1			
	v_7	1		1				1	−1	1			−0.5	
	v_8		−1						−1	1	1			0.5
	v_9	1						1		1		0.5		0.5
	v_{10}			−1						1				0.5
	v_{11}			−1										0.5
	v_{12}			1	1	0.5	1		1		1		0.5	0.5
	v_{13}									1		0.5	0.5	
	v_{14}	−1		−1				−1	1	−1	1	1	1	−0.5
	v_{15}		0.5			−0.5		−1	1	1	−0.5	1		1
	v_{16}		1	1				1	1	1		0.5		1
	v_{17}							1						1
	v_{18}		1	1	1	1	1	1	1			0.5		1
	v_{19}	1	1		1	1	1	1	1					0.5
	v_{20}	1		1	1	1			1		1		−0.5	0.5

1, Security directly addresses vulnerability; 0.5, security indirectly addresses vulnerability; −0.5, security indirectly creates vulnerability; −1, security directly creates vulnerability.

incorporating a security technology is a generalization of the well-known set-covering problem. Set-covering is defined in [4] as a finite set X and a family F of subsets of X, such that every element of X belongs to at least one subset in F. From an optimization standpoint, the objective of a set-covering problem is to find the minimal subset in F such that the selected subset contains all the elements of X. We can also map the set-covering problem into graphs. Imagine that each element of the set X is represented by edges in the graph and each of the vertices in the graph represents a subset that contains some edges in the graph; i.e., the edges are connected to the vertex. The objective of the set-covering problem is now to find the minimum number of vertices that cover all the edges in the graph. The union of the subsets represented by each vertex gives us the minimal subset that contains all the elements of set X. It has been shown that the set-covering problem is NP-complete [4].

The objective of the vulnerability-covering problem is to maximize the number of vulnerabilities covered while minimizing the cost of security technologies. However, the implementation of a security technology might result in new vulnerabilities. We use the term "residual" vulnerabilities to describe these newly introduced vulnerabilities. The problem now can be defined such that

If there exists a set of vulnerabilities V and a set of security technologies S whose subsets cover some elements in set V, but also result in creating some subset of residual vulnerability set R, then the problem is to find the minimal subset of S that covers all elements in V while having the smallest resulting subset of R.

Representing the above problem through graphs, let each unique vulnerability in set V be represented by a colored edge. Each color represents a unique vulnerability. Let each vertex in the graph represent distinct security technologies from the set S. Each security technology that covers a vulnerability will have an edge with that color on the vertex. If a vulnerability is addressed only by one security technology, it is represented by a self-edge. The residual set is represented by unique vulnerability edges that are created after the covering set is determined.

We now need to ensure that all vulnerabilities are covered by finding a minimal set of vertices that cover all unique-color edges. If we assume that any selection of a subset of S results in the same subset of set R, our problem is reduced to a set-covering problem. But if the subset of S results in different subsets of set R, our problem is more difficult to solve than the set-covering problem. Therefore, we can assume that our problem is NP-hard and cannot be solved in polynomial time. From the algorithm perspective, the time to find the solution increases exponentially with the problem size, i.e., if the number of security technologies increases, the time to find the solution increases exponentially in the number of security technologies. A heuristic technique, such as a GA, must be used to search for a good solution in such situations.

A GA was chosen as the heuristic solution to this problem for several reasons. First, GAs have been shown to perform well in similar situations. Second, and perhaps more importantly, the genetic algorithm structure is particularly well-suited to the features of the problem being solved and therefore provides a very natural representation of the problem.

We map the vulnerabilities, the security technologies and the residual vulnerabilities into genes or strings of a GA. Let each vulnerability in the vulnerability set be represented by a single bit in the vulnerability gene. Thus, the vulnerability profile of a firm is represented as,

$$V = \{v_i\} \forall i = 1 \ldots m \tag{1}$$

where, v_i (i=1...m) represents individual vulnerability and can take two values, 0 or 1. The value 0 signifies that a particular vulnerability is not present in the organization's information systems and the value 1 indicates the presence of the vulnerability. In this research, m is set to 20, as there are 20 generic vulnerabilities. For example, a vulnerability profile '00001011011100010101' indicates that the vulnerabilities: homogeneity, predictability, rigidity, gullibility, capacity limits, lack of recoverability, electronic accessibility, physical accessibility and dependency are present.

Every vulnerability has a different significance for each organization. Some vulnerabilities are more critical than others and need to be addressed immediately while others are not that critical. It may be cost effective for the organization to address a vulnerability indirectly instead of controlling it directly. We assign each vulnerability a weight to indicate varying significance to different vulnerabilities. The weight for vulnerability v_i is represented by a_i $\forall i$=1…m. The weights are assigned such that,

$$\sum_{i=1}^{m} a_i = 1 \qquad (2)$$

To cover the vulnerabilities, organizations choose a set of security technologies. Let each security technology be represented by a single bit in the security gene or string. Thus, the security of the firm is represented as

$$S = \{s_j\}\ \forall j = 1 \ldots n \qquad (3)$$

Each individual security technology can take two values, 1 or 0. Similar to the vulnerability string, a 1 represents the presence of the security technology represented by that bit and a 0 represents the absence of that security technology. In this research, n is set to 13, as there are 13 generic security technologies. For example, the security profile '1000000001001' indicates the implementation of the following security technologies: heterogeneity, self-organization and collective behavior and threat/warning response structure. Each security technology has an associated cost. We represent these costs as relative to each other. The cost of any security s_i is represented by c_i $\forall i$=1…n. The costs are assigned such that,

$$\sum_{i=1}^{n} c_i = 1 \qquad (4)$$

Once a security technology is chosen, each technology bit directly reduces one or more vulnerabilities, indirectly reduces vulnerabilities, indirectly creates some vulnerability or directly creates some vulnerability. For 20 vulnerabilities and 13 security technologies, the matching of the security technologies and vulnerabilities is presented in Table 3 [1]. Therefore, after a security technology is implemented, we must reassess the vulnerabilities that the organization's information systems still possess. We call this the residual vulnerability portfolio. This residual vulnerability exists either because of some uncovered vulnerability, some partially covered vulnerability or can be directly or indirectly created by implementation of security technologies.

We represent the residual vulnerability portfolio by a string with each bit representing the same vulnerability described by the corresponding bit in the vulnerability string. Each bit in residual vulnerability string also has the same weight as the corresponding bit in the vulnerability. The residual vulnerability of the firm is represented as

$$R = \{r_i\}\ \forall i = 1 \ldots m \qquad (5)$$

Each bit in the residual vulnerability string can have three values: 1, 0.5 and 0. The value 1 represents the presence of the particular vulnerability and the value 0 represents the absence of that vulnerability. The value 0.5 represents the presence of a partially covered vulnerability or an indirectly created vulnerability. By partially covered vulnerability, we refer to the fact that the technology was able to reduce the spread of vulnerability and limited its damage, however, the technology failed to address the underlying cause of vulnerability. Thus, the technology managed to reduce the risk but did not eliminate the risk. The coverage of vulnerabilities by security technologies can be described by a matrix t_{ij} $\forall i$=1,…,m and j=1,…,n. The sample matrix of t_{ij} is described in Table 3. The residual vulnerability is determined based on the following rules (These rules are exhaustive and are not dependent on the order of rule application.):

(1) If v_i=0 or 1 and any t_{ij}=1 for $\forall j/s_j$=1, then r_i=0.
(2) If v_i=0 and all t_{ij}=0 for $\forall j/s_j$=1, then r_i=0.
(3) If v_i=1 and all t_{ij}=0 for $\forall j/s_j$=1, then r_i=1.
(4) If v_i=0 and no t_{ij}=1 for $\forall j/s_j$=1 and any t_{ij}=−1 for $\forall j/s_j$=1, then r_i=1.
(5) If v_i=0 and no t_{ij}=1 for $\forall j/s_j$=1 and any t_{ij}=−0.5 for $\forall j/s_j$=1, then r_i=0.5.
(6) If v_i=1 and no t_{ij}=1 for $\forall j/s_j$=1 and any t_{ij}=0.5 for $\forall j/s_j$=1, then r_i=0.5.

5. Experimental design

This section describes the specific implementation of the GA used to search for a good solution to the model presented above. We also present the details of the experiments run to demonstrate the approach. The same problem was then attempted using an enumerative approach to solve test problems of small size for comparison purposes.

5.1. Specific GA Implementation

Organizations have two major objectives for managing their information security investments:

(1) Choose a security combination that maximizes the coverage of vulnerabilities or in other words minimizes the weighted residual vulnerability. This objective can be represented as

$$\operatorname*{Min}_{s} \sum_{i=1}^{m} a_i r_i \tag{6}$$

(2) Choose a security combination that minimizes the costs to the firm. This objective can be represented as

$$\operatorname*{Min}_{s} \sum_{j=1}^{n} c_j s_j \tag{7}$$

The fitness function for learning the security profiles combines both objective functions into one. The fitness function for any security combination S^k is described as

$$F = \alpha \sum_{i=1}^{m} a_i r_i + \beta \sum_{j=1}^{n} c_j s_j \tag{8}$$

where

$\alpha + \beta = 1$ and $\alpha, \beta \leq 1$

α and β are non-negative and represent the preferences of the organizations. Some organizations wish to cover as many of the vulnerabilities as possible. For these organizations, addressing these vulnerabilities is more important than paying an extra price for security technologies. This preference can be implemented by choosing $\alpha > \beta$. For other organizations, the security budget is quite limited so they try to minimize the cost of security while covering most of the vulnerabilities. This case can be represented by choosing $\alpha < \beta$.

We use a simple GA, similar to [5], including the operations of selection, crossover and mutation. We start with a random population of security profiles in the initial generation. An elitist selection strategy is used to seed the next generation. The strings are sorted in descending order of their fitness. A certain percentage of the best security profiles are automatically copied (inherited) into the next generation. The remaining strings in the previous generation are selected for the next generation using stochastic selection with replacement until the full population size is reached. The probability of a security profile getting selected is proportional to the value of its fitness function. This process is undertaken for pairs of strings. Once a pair of strings has been selected, a weighted coin is tossed to determine if they will undergo a single-point crossover operation. If they are chosen to crossover, the pair of strings will exchange bits after a preset crossover position. After the crossover operation is completed, the population undergoes mutation where each bit in all security profiles is toggled according to a pre-defined mutation probability. The above operations are repeated until the best solution does not improve for a certain number of generations.

5.2. Simulation experiment design

A set of 25 random vulnerability scenarios was created for testing the performance of the GA. We compare the GA approach to an enumerative or exhaustive search (which we call “Brute Force” (BF)) approach to compare the efficiency and efficacy of the GA approach for problems of small size. We also examine different sizes of problems (in terms of numbers of vulnerabilities and security technologies) to examine the scalability of the GA approach. The results of these simulation experiments and a discussion of these results are provided in the next section.

6. Results

For the first set of experiments, we tested 25 random vulnerability combinations and searched for the optimal security portfolio using both the GA approach as well as a BF search. We assigned random weights to each vulnerability and security technology. We assumed that organizations consider covering the vulnerabilities and the cost of security to be equal and assign the same weight to these parameters ($\alpha = \beta$). The results of the first set of experiments are presented in Table 4.

Table 4
GA performance

Vulnerability	GA solution	BF solution	GA fitness	BF fitness	% Error	GA time (ms)	BF time (ms)	GA faster by factor of
00001011011100010101	0000000101000	0000000101000	0.1314	0.1314	0.00	220	420	1.91
00110001000010011000	0000000001011	0000000001011	0.1152	0.1152	0.00	240	360	1.50
01111110011101011110	1000000001001	1000000001001	0.1322	0.1322	0.00	180	360	2.00
11111101000100011110	1000000001001	1000000001001	0.1055	0.1055	0.00	150	370	2.47
11101011111010111110	1000000001011	1000000001011	0.1470	0.1470	0.00	160	360	2.25
11001010101001010001	1000000001000	1000000001000	0.1249	0.1249	0.00	160	370	2.31
10001011111101010101	1000000001001	1000000001001	0.1322	0.1322	0.00	140	360	2.57
11010100110111000110	1000010000010	1000010000010	0.0875	0.0875	0.00	190	370	1.95
10001011010010001000	1000000001011	1000000001011	0.1274	0.1274	0.00	160	360	2.25
00111100010011010111	0000000110000	0000000110000	0.0982	0.0982	0.00	150	350	2.33
10111110101000110101	1000000001001	1000000001001	0.1252	0.1252	0.00	160	360	2.25
01111111011010110001	0000000011000	0000000011000	0.1275	0.1275	0.00	170	350	2.06
10111111101100111001	1000000001001	1000000001001	0.1252	0.1252	0.00	160	360	2.25
01101001111000001101	1000000001001	1000000001001	0.1322	0.1322	0.00	160	350	2.19
10100000110011011111	1000000001011	1000000001011	0.1274	0.1274	0.00	140	370	2.64
10000001110100000110	1000010001000	1000010001000	0.0816	0.0816	0.00	190	360	1.89
01000111101101001000	1000000001001	1000000001001	0.1252	0.1252	0.00	170	360	2.12
00110101010000000101	0000010001000	0000010001000	0.0684	0.0684	0.00	160	370	2.31
11100100101000010110	1000000001001	1000000001001	0.1252	0.1252	0.00	150	370	2.47
00010111011111101101	0000000001011	0000000001011	0.1442	0.1442	0.00	180	350	1.94
01001110100100000110	1000010000010	1000010000010	0.0611	0.0611	0.00	170	350	2.06
01000011110111100100	0000000110000	1000010001010	0.0982	0.0966	1.72	170	370	2.18
10111101011111001101	1000000001011	1000000001011	0.1470	0.1470	0.00	150	360	2.40
10001010011111010000	0000000011000	0000000011000	0.1275	0.1275	0.00	150	370	2.47
10001101101011110000	0000000011000	0000000011000	0.1275	0.1275	0.00	160	370	2.31

GA specifications	
Population size	100
Elitist/inherited	40%
Population	
Crossover rate	0.85
Crossover point	Mid-point (fixed)
Mutation rate	0.01
Stopping criteria	50 stable generations

From the above results, we can easily see that the GA-based approach was able to find the best solution for all but 1 of 25 test cases that we analyzed. Not only did the GA-based approach give us the right solution, the execution time (CPU time for finding the solution) for the algorithm was much faster than the exhaustive BF approach. The GA-based approach performed consistently better than the BF technique. We carried out the analysis for 20-bit vulnerabilities and 13-bit security profiles. These vulnerability and security profiles matched with the classification of vulnerabilities and security technologies provided in [1]. Table 5 presents the best security profiles for actual vulnerabilities for some of the above results.

In the section above, we demonstrated the effectiveness of the GA-based approach of matching security technologies and vulnerabilities for a problem size of 20 vulnerabilities and 13 security technologies. However, it is much more realistic to have other possible combinations of vulnerabilities and security technologies. In this section, we analyze the performance of the GA-approach based on the problem size.

Table 5
Security profiles for selected vulnerabilities

Vulnerability profile	Vulnerabilities	Security profile	Security technologies
00001011011100010101	Homogeneity, Predictability, Rigidity, Gullibility, Capacity Limits, Lack of Recoverability, Electronic Accessibility, Physical Accessibility, Dependency	0000000101000	Segmentation, Decentralization and Quarantine, Self-organization and Collective behavior
01111110011101011110	Singularity, Centralization, Separability, Homogeneity, Sensitivity, Predictability, Gullibility, Capacity Limits, Lack of Recoverability, Difficulty of Management, Electronic Accessibility, Transparency, Physical Accessibility, Electromagnetic Susceptibility	1000000001001	Heterogeneity, Self-organization and Collective behavior, Threat/ Warning Response Structure
10001101101011110000	Uniqueness, Separability, Homogeneity, Predictability, Rigidity, Gullibility, Lack of Recoverability, Lack of Self-awareness, Difficulty of Management, Complacency/ Co-optability	0000000011000	Immunologic Identification, Self-organization and Collective behavior
01001110100100000110	Singularity, Homogeneity, Sensitivity, Predictability, Malleability, Lack of Recoverability, Physical Accessibility, Electromagnetic Susceptibility	1000010000010	Heterogeneity, Rapid Recovery and Reconstitution, Centralized Management of Information Resources

Additionally, the population size used in the GA-based approach also affected the performance of the algorithm. We analyzed different population sizes along with the problem size to demonstrate that relatively small population sizes result in an effective GA performance. Fig. 1 demonstrates the accuracy of the GA-based approach compared to the optimal solution obtained by the BF search of the security technologies and vulnerabilities. The error rate as compared with the optimal solution appears on the y axis and the problem size appears on the x axis.

From Fig. 1, it can be seen that the GA-based approach has error levels of less than 5% with

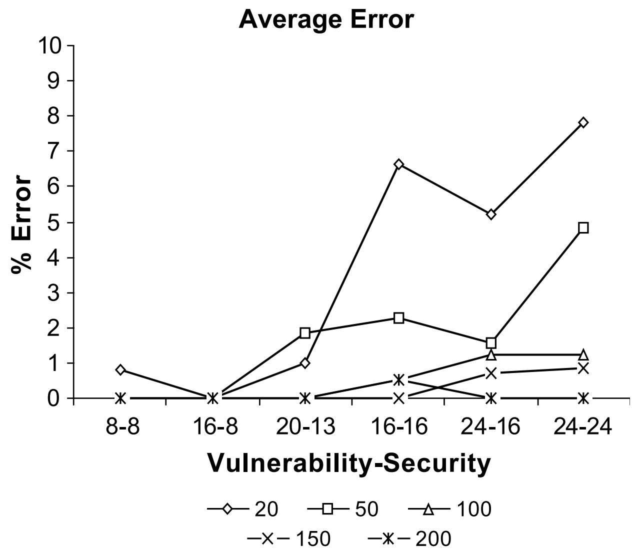

Fig. 1. GA accuracy for varying problem sizes.

optimal population sizes, even with the problem size of 24 vulnerabilities and 24 security technologies. The GA-based approach accuracy suffered for larger problems when, for example, the population size of 20 or 50 security profiles was used for each generation. However, a population of 50 security profiles represents only 0.0002% of all the possible security profiles. We can easily see that GA-based approach is accurate for population sizes above 100 security profiles. In instances when the GA-based technique does not provide the optimal solution, the error is low enough for the solution to be considered to be a viable solution. It has to be noticed that given the complex nature of information security technologies, even the optimal solutions do not guarantee a complete coverage of all vulnerabilities. However, such a solution provides an approach to manage the risk of exposure while keeping the costs under control. From a management perspective, such an approach provides insight into viable security solutions and generates an awareness of the exposed risk that management can further address through disaster recovery measures.

Nevertheless, the question still remains: Is the GA-based approach fast enough for larger problems with a bigger population size? Fig. 2 presents the performance of the GA-based approach as the problem size increases. From Fig. 2, it is evident that the BF approach performs better for smaller problem sizes. However, the execution time of the BF approach increases exponentially with the problem size. For larger problem sizes, the GA-based approach performs several thousand times faster than the BF approach. It is also interesting to see that the execution time for GAs does not increase exponentially with an increase in the population size.

7. Conclusions and future directions

In this research, we presented a GA-based approach to allow organizations to match their security technologies against their vulnerabilities. Matching security technologies and vulnerabilities is a dual-objective problem: maximizing the vulnerabilities covered and minimizing the security costs. The problem was reduced to a set-covering problem, known not to be solvable in polynomial time. Therefore, a heuristic approach using GAs was used. We used a weighted average GA approach to combine the objective functions and find a good security match to the vulnerabilities while trying to minimize the security cost. The GA-based approach was much faster than the BF approach of searching

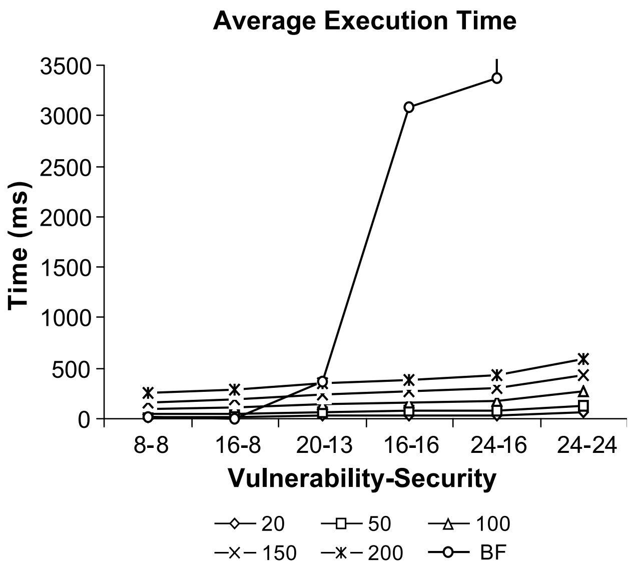

Fig. 2. GA execution time.

for the best security–vulnerability combination. The algorithm scales very well with the problem size and the execution time increases almost linearly with the increase in the problem size (search space), i.e., the increase in the number of vulnerabilities and security technologies. The GA-based approach also showed acceptable levels of accuracy in determining the correct solution.

In future research, this mapping can easily be extended to match securities to vulnerabilities with specific implementations of vulnerabilities and security technologies. Carrying out case studies in organizations and applying the techniques described in this paper would truly demonstrate and hopefully underscore the practicality and usefulness of the approach. The implementation of the GA used here can be refined and extended to improve performance and scalability of the approach, as well as compared against other heuristics to further measure performance. Eventually, we envision this GA-based approach being incorporated into a decision support tool for supporting managers in information security planning and management activities. Finally, much more research in general needs to be done to assist organizations in protecting their informational assets from harm at an acceptable cost.

Appendix A. Pseudo-code for genetic algorithm

```
Procedure reproduce
Input: cur_pop: a vector of solution population
   mid: the position at which crossover take place
   mutate_prob: mutation probability
Output: next_pop: a vector of the same size as the
   input pop.
Begin:
   sort cur_pop according to the solution fitness in
   descending order
   inherit a percentage of best performing solutions
   to the next_pop
   while next_pop.size<cur_pop.size(), do
      copy two solutions, s1, s2, from the cur_pop.
      //The chance for each solution getting picked
      is proportional to its fitness.
      if (crossover_prob)
      call crossover(s1, s2, mid)
      mutate s1, s2, according to mutate_prob
      insert s1, s2 in next_pop
      increment next_pop by 2
   end loop
End
Procedure crossover
Input:s1, s2: parent strings
   mid: the position at which crossover take place
Output: none
Begin:
   exchange the bits from 0 to mid between s1 and
   s2.
End
Procedure GA
Input:
Output:
Begin:
   call Generate_initial_population(pop)
   old_best=0;
   for $i$<−0 to max_generation, do
      best=Compute_fitness(pop);
      if old_best!=best
            old_best=best;
            ctr=0;
      else
            ctr++;
            if ctr>stable
                  return best;
            end if
      end if
      next_pop=reproduce(pop);
      pop=next_pop;
   end loop
End
```

References

[1] R.H. Anderson, P.M. Feldman, S. Gerwehr, B.K. Houghton, R. Mesic, J. Pinder, J. Rothenberg, J.R. Chiesa, Securing the U.S. Defense Information Infrastructure: A Proposed Approach, RAND Document MR-993-OSD/NSA/DARPA (1999).

[2] A. Charnes, W.W. Cooper, Management Models and Industrial Applications of Linear Programming, John Wiley and Sons, NY, 1961.

[3] C.A.C. Coello, An updated survey of GA-based multiobjective optimization techniques, ACM Computing Surveys 32 (2) (2002) 109–143.

[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introductions to Algorithms, The MIT Press, Cambridge, MA, 2001.

[5] D.E. Goldberg, Genetic Algorithms: In Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[6] Y. Ijiri, Management Goals and Accounting for Control, North Holland Publishing, Amsterdam, 1965.

[7] W. Jackob, M. Gorges-Schleuter, C. Blume, Applications of genetic algorithms to task planning and learning, in: R. Männer, B. Manderick (Eds.), Proceedings of the Second Workshop on Parallel Problem Solving from Nature, North Holland Publishing, Amsterdam, The Netherlands, 1992, pp. 291–300.

[8] D. Quagliarella, A. Vicini, Coupling genetic algorithms and gradient based optimization techniques, in: D. Quagliarella, J. Périaux, C. Poloni, G. Winter (Eds.), Genetic Algorithms and Evolution Strategies in Engineering and Computer Science: Recent Advances in Industrial Applications, John Wiley and Sons, Chichester, UK, 1995.

[9] B.J. Ritzel, J.W. Eheart, S. Ranjithan, Using genetic algorithms to solve a multiobjective groundwater pollution containment problem, Water Resources Research 30 (5) (1994) 1589–1603.

[10] E. Sandgren, Multicriteria design optimization by goal programming, in: H. Adeli (Ed.), Advances in Design Optimization, Chapman & Hall, London, UK, 1994, pp. 225–265.

[11] G. Stoneburner, Underlying Technical Models for Information Technology Security, National Institute for Standards and Technology SP 800-33 (2001).

[12] G. Syswerda, J. Palmucci, The Applications of Genetic Algorithms to Resource Scheduling, in: R.K. Belew, L.B. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms, University of California, San Diego, Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[13] P.B. Wienke, C. Lucasius, G. Kateman, Multicriteria target optimization of analytical procedures using a genetic algorithm, Analytica Chimica Acta 265 (2) (1992) 211–225.

[14] P.B. Wilson, M.D. Macleod, Low implementation cost of IIR digital filter design using genetic algorithms, Proceedings of the IEE/IEEE Workshop on Natural Algorithms and Signal Processing, 1993.

Mukul Gupta is an assistant professor in the Department of Operations and Information Management in the School of Business at the University of Connecticut. He earned his doctorate from the Krannert School of Management at Purdue University in 2003. His current research interests are information security management and agent-based simulations.

Jackie Rees is an assistant professor in the Krannert School of Management at Purdue University. She earned her doctorate from the University of Florida in 1998. She has published in *Decision Support Systems*, *INFORMS Journal of Computing*, *Communications of the ACM*, *European Journal of Operational Research* and *Information Technology and Management.* Her current research interests are in the intersection of information security risk management and machine learning.

Alok Chaturvedi is an associate professor at the Krannert School of Management and founder and director of Purdue e-Business Research Center (PERC) at Purdue University. He earned his doctorate from the University of Wisconsin-Milwaukee in 1989. He has published in such journals as *Information Systems Research*, *Communications of the ACM*, *Decision Support Systems*, *International Journal of Information Management*, *Journal of Organizational Computing and Electronic Commerce* and *Database.* His current research interests are information warfare and synthetic environments.

Jie Chi is a doctoral student in Computer Science at Purdue University. He also serves as a graduate research assistant at the Purdue e-Business Research Center.