CAAI Transactions on Intelligence Technology

Special Issue: Bio-inspired Computing for Learning and Optimisation

Multi-objective evolutionary approach to select security solutions

ISSN 2468-2322
Received on 20th December 2016
Revised on 20th January 2017
Accepted on 20th February 2017
doi: 10.1049/trit.2017.0002
www.ietdl.org

Yunghee Lee[1], Tae Jong Choi[2], Chang Wook Ahn[1] ✉

[1]*School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology (GIST), 123 Cheomdangwagi-ro, Gwangju, Republic of Korea*
[2]*Department of Computer Engineering, Sungkyunkwan University, 2066 Seobu-ro, Suwon, Republic of Korea*
✉ *E-mail: bluecwan@gmail.com*

Abstract: In many companies or organisations, owners want to deploy the most efficient security solutions at a low cost. The authors propose a way of choosing the optimised security method from many security methods by multi-objective genetic algorithm (GA) considering cost and weakness decrease in this study. The proposed system can find the best security methods in various aspects of security issues. This study uses the NSGA-II algorithm, which has been authorised in a variety of fields, to provide a comparison with old GAs. Their scheme has increased the dominant area by more than 30% compared with the previous scheme and can provide a more diverse solution set.

1 Introduction

As information technology systems and the Internet grew, so did the number of malicious threats to information [1]. To prevent information threats such as this, organisations and enterprises study security solutions to secure information separately from their usual work. Security solutions are generally physical and logical countermeasures to prevent information system failure and destruction [2]. However, almost all of companies do not want to spend money on improving security. Investing in security solutions does not seem to be effective in the short time. Moreover, in order to invest in security solutions, companies have to choose how much to invest in what measures, but it is very difficult to make such a choice without knowing the exact threats and the effectiveness of countermeasures. In this paper, we describe the selection of a security method using NSGA-II, a kind of multi-objective genetic algorithm (MOGA). This can help any business or organisation easily choose the best security solution. This paper is organised as follows: in Section 2, we explain about GAs and Pareto-optimisation. In Section 3, we explain a MOGA. We made a creating security method and weakness-decrease point (WDP) for experiment and show the programme code in Section 4. The system we propose is presented in Section 5. Section 6 concludes this paper.

This paper is an extended paper from 'Design of Selecting Security Solution Using Multi-objective Genetic Algorithm' [3]. We conducted the experiments again with increased number of individuals.

2 Related works

In this section, we talk about Pareto-optimality after the description of the GA and knapsack problem as known as non-deterministic polynomial (NP) problem.

2.1 GA and knapsack problem

A GA is a kind of heuristic search based on the phenomenon of nature. It was first designed by John Holland in 1975. This is one of the techniques to solve the optimisation problem by calculation based on the natural evolutionary process. In general, if it is impossible to obtain an optimised solution of a problem through a formal formula, or if it is too complicated, it may be efficient to solve the problem through a GA. However, the GA does not always find a global optimal solution. This only helps to find solutions that are close to the optimal solution in a short time. Therefore, GAs are generally useful for problems classified as NP time problems [4].

The knapsack problem is one of the most suitable problems to solve with a GA. The knapsack problem is a matter of finding out what items we need to fill the bag to make it the most valuable. The size of the items that can be stored in the bag is fixed, and each item has a predetermined value and size. Therefore, if the item can be split, we can easily find the globally optimal solution to this problem with the greedy algorithm. However, if they cannot break apart, this problem is a problem that cannot be solved with a formal formula. Thus, in this case, this problem becomes an NP-completeness problem [5, 6]. If we use a GA to solve the knapsack problem, we can find an efficient solution for a short time. Recently, various studies related to the research we are trying to do have been preceded [7].

2.2 Pareto-optimality

If you use a simple GA to solve the knapsack problem, the sum of the sizes will naturally approach the maximum size. If you have a budget and do not have any problems with using your whole budget, you can solve this problem using the simple GA. However, companies and organisations want to find low-cost, high-efficiency solutions, and deploy it. Therefore, unlike a simple GA that considers only one objective, in the real world, it is necessary to find the optimal solution considering both the cost and the WDP. Sometimes, a problem may have more than just two objectives. If that happens,

Table 1 List of security method sets that are made randomly

Num	100−cost	WDP
R1	82	74
R2	23	79
R3	57	21
R4	7	66
R5	53	92
R6	66	5
R7	66	46
R8	57	17
R9	64	57
R10	86	54

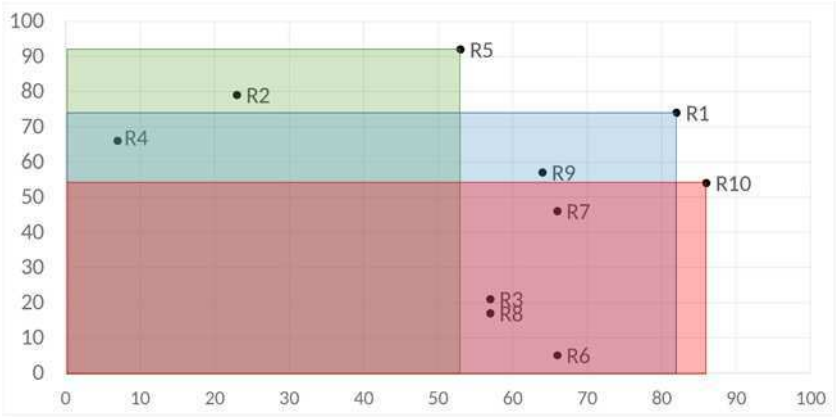

Fig. 1 *Chart to select the best security method from various methods*

the problem will be much more complicated than when considering only one objective. In this paper, we will show a method to clear the problem by considering two objectives: cost and WDP.

In general, we use the 'Pareto-optimality' when there are multiple objectives to find a global optimised solution of the problem. For example, the cost and WDP of security solutions to address security flaws are shown in Table 1. As shown in Fig. 1, the data in Table 1 can be charted. In Fig. 1, the *X*-axis represents (100−cost) and the *y*-axis shows WDP: decrease of dangerous.

In Fig. 1, we can say that the solution in the upper right is more effective and better. The optimal solution is the top-most, right-most solution in the chart. However, in general, higher WDPs result in higher costs, making it difficult to find the ideal solution like that. Instead, we can find a Pareto-optimal that is superior to other solutions [8]. The squares on the chart show Pareto-dominance easily. For example, R2 has a very high WDP, which is very helpful in solving security problems, but solution R2 is not an optimal solution because there is a solution R5 with a lower cost and higher WDP. At this point, Solution R2 is said to be a Pareto-dominated entity. When we create a chart like the one shown in Fig. 1, we call the unconstrained solution Pareto-optimal for any other solution, and call the set a Pareto-optimal set. The line that the Pareto-optimal set forms is called the Pareto-frontier. Ultimately, what we are looking for is a Pareto-optimal set.

3 MOGA: NSGA-II

There are many existing MOGAs to solve the many types of problems: Niched Pareto Genetic Algorithm (NPGA), Non-dominated Sorting Genetic Algorithm (NSGA), Strength Pareto Evolutionary Algorithm (SPEA) etc. All of them are very famous MOGA solutions, and in this paper we used NSGA-II algorithm for clearing the problem. Since NSGA-II is the lightest and fastest method of MOGA known so far. NSGA-II is a new advanced technique compared with NSGA, a conventional MOGA. It can finish the calculation in less time than NSGA and introduces the concept of non-dominant ranking. In addition, NSGA-II introduced a concept called crowding distance.

Therefore, this scheme can distribute resources more efficiently than existing algorithms. Another thing that NSGA-II is different from NSGA is elitism. Elitism is the scheme of keeping the superior population among the population to the next generation. Therefore, solutions with high fitness are not easily lost through generations [9]. The NSGA-II algorithm is not hard to use and can quickly find solutions to high fitness. Moreover, it has a very good performance so this algorithm is very popular [10].

The NSGA-II algorithm is shown in Fig. 2 [11]. Non-dominated rank means the rank that how many other solutions are dominating the solution. In other words, a lower non-dominated rank is a better solution. For example, there is a solution named A. If any solution is dominating a solution A, the non- dominated rank of solution A

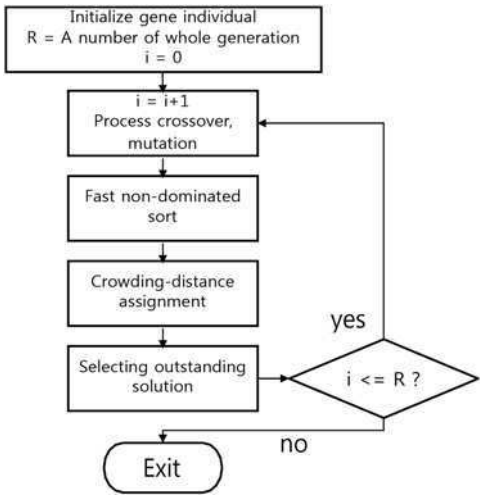

Fig. 2 *Flowchart of NSGA-II*

is 0. Thus, in the same generation, the Pareto-optimal set has the highest priority, and the solution farther from it has an increasingly lower priority. The non-dominated rank alignment process allows solutions to converge on the Pareto-optimal solution. Moreover, crowding distance is a solution to see how many solutions are gathered in a small area when the charts are shown like Fig. 1. This is a value that is calculated to help the solutions with the similar non-dominated rank have diversity. All solutions have a high crowding distance value if it is less similar to the neighbouring solution. This is an element for selecting an object with a different property from the set of genetic entities belonging to the same non-dominated rank [9].

3.1 Performance improvement

We used different mutations and crossover types to improve performance. Mutation and crossover are very important components in the GA. There are a lot of types about mutation and crossover: uniform mutation, parent-centric crossover, bit-flip mutation, half-uniform crossover etc. In this paper, we did with the simulated binary crossover (SBX) for crossover process and polynomial mutation (PM) for mutation process in NSGA-II. SBX is the operator that has search performance similar to that of a single-point binary coded crossover operator [12]. Moreover, the PM is the operator that is much used in evolutionary optimisation algorithms as a variation operator [13]. It attempts to mimic the offspring distribution of binary-encoded bit-flip mutation on real-valued decision variables. In this paper, the type of the value to be calculated is binary, but we used PM because the PM showed higher performance than the bit-flip mutation. PM is similar to SBX, it assists offspring nearer to the parent [14].

```
for (i = 0; i < 500; i++)
{
    arr2[i] = gaussianRand(arr[i], STD);
  // STD is the standard deviation of gaussian random function
  // We setted STD to 50
    if (arr2[i] <= 0)
        arr2[i] = rand() % arr[i] + 1;
}

double gaussianRand(double mean, double stddev)
{   // gaussian random number generater function
    static double n2 = 0.0;
    static int n2_cached = 0;
    if (!n2_cached)
    {
        double x, y, r;
        do
        {
            x = 2.0*rand() / RAND_MAX - 1;
            y = 2.0*rand() / RAND_MAX - 1;
            r = x*x + y*y;
        } while (r == 0.0 || r > 1.0);
        {
            double d = sqrt(-2.0*log(r) / r);
            double n1 = x*d;
            n2 = y*d;
            double result = n1*stddev + mean;
            n2_cached = 1;
            return result;
        }
    }
    else
    {
        n2_cached = 0;
        return n2*stddev + mean;
    }
}
```

Fig. 3 *Creation of a WDP*

Moreover, we made the population size for the GA to 500 and the number of generations to 15,000.

4 Creating security solution and WDP

We need to create a variety of virtual security solutions for the experiment, each with an introduction cost and a WDP. However, WDP is a value that cannot be easily quantified. Therefore, in this paper, we use a reasonable random number as a WDP to create a sample virtual security solution.

First, we need to create 500 random numbers to be used as the cost of introducing a virtual security solution. The total sum of 500 random numbers is 1,000,000. After doing that, we sort 100 random numbers and put them into the array arr [ ]. Then, use the source code below to create a WDP corresponding to each cost, and place it in the array arr2 [ ] (Fig. 3).

So we can make the meaningful random WDP. WDP will be comparable with security solutions cost. However, there can be rarely too big WDP than security methods cost or the opposite case.

5 Proposed scheme

In this section, we suggest techniques for selecting the best security methods using NSGA-II, the MOGA written in the last section. As we mentioned in Section 1, businesses and organisations want security solutions that can get the most out of their business with minimal cost. Park *et al.* [15] has released a method for this problem. They worked to solve this problem using the ordinary GA and used a list of ten virtual security solutions in the experiment. To compare the two schemes, we have coded programmes that perform as well as the simple GAs used in Park *et al.*'s paper [15], and have created more new virtual security solution lists and experimented. We compared the results obtained using our scheme with those obtained using Park *et al.*'s scheme [15]. As a result of using the scheme of Park *et al.* [15], we could find three optimal solutions.

Moreover, also important in the GA is the fitness evaluation function. It is called as fitness function. In the simple GA used by Park *et al.* [15]'s fitness function, it considers only one objective: WDP. In MOGA, however, we can adapt multiple objectives for fitness functions

$$f_1 = \sum_{i=1}^{n} (100{,}000 - vc_i s \times vc_i c) \tag{1}$$

$$f_2 = \sum_{i=1}^{n} vc_i s \times vc_i d \tag{2}$$

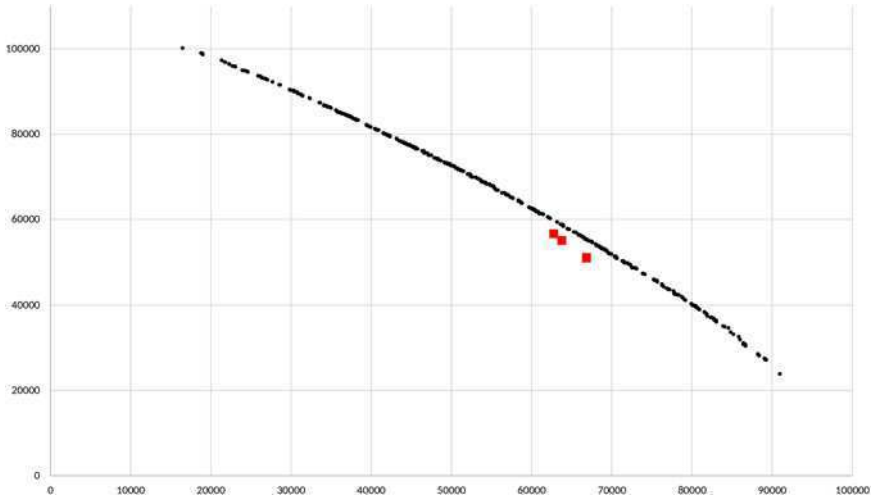

Fig. 4 *Graph of choosing security method using NSGA-II*

CAAI Trans. Intell. Technol., 2017, Vol. 2, Iss. 2, pp. 64–67

In this paper, we used two fitness functions as shown in (1) and (2). Equation (1) uses (100,000 – the total cost of the solution) values for fitness calculations and (2) uses the entire WDP of the solution for fitness calculations. Here, n is the number of whole chromosomes, in other words, n means the number of security methods; vc means each chromosome structure; $vc.d$ includes the decrease point (DP) of security weakness; $vc.c$ includes the cost for choosing that method; and $vc.s$ includes the binary number for check whether each solution was choosed or not choosed. So if $vc.s$'s value is 0 that means the method was not choosed.

Fig. 4 compares the best virtual security solutions selected using the NSGA-II algorithm to the best virtual security solutions selected using the simple GA. The horizontal axis indicates the value of f_1, and the vertical axis indicates the value of f_2. The results of using the existing Park *et al.*'s scheme [15] have reversed the cost value for not hard comparison. Therefore, the cost of the original research is actually (100,000 – original cost). For the sake of clarity, we plotted the results of original research as red squares and the results of our research as black dots. Using the NSGA-II-based security solution selection scheme we have studied, we can confirm that the selected security solution set forms the Pareto-frontier and completely dominates the results of existing papers. The results of this paper provide a variety of choices, from low-cost solution selection to high-cost solution selection.

6 Conclusion

In this paper, we propose a scheme to efficiently select the security solutions required by corporations and organisations using NSGA-II in terms of various objectives: cost and value. The proposed method was able to find optimal solutions considering various objectives and showed superiority in the process and performance of fitness evaluation compared with existing papers using a simple GA. A more detailed study on how to quantify the WDP should be conducted and the stability and performance of NSGA-III developed by NSGA-II should be verified.

7 Acknowledgments

This work was supported by the NRF grant funded by the Korea government (MSIT) (NRF-2018R1D1A1A09084148).

8 References

[1] Chai, S.W.: 'Economic effects of personal information protection', Korea Consumer Agency, 2008, 33, pp. 43–64
[2] Kwon, Y.O., Kim, B.D.: 'The effect of information security breach and security investment announcement on the market value of Korean firms', *Inf. Syst. Rev.*, 2007, **9**, (1), pp. 105–120
[3] Lee, Y., Jung, J., Ahn, C.: 'Design of selecting security solution using multi-objective genetic algorithm'. Bio-inspired Computing – Theories and Applications, Xi'an, China, 2016, vol. 48
[4] Mitchell, M.: 'An introduction to genetic algorithms' (MIT press, Cambridge, MA, 1998)
[5] Kellerer, Hans, knapsack problems
[6] Silvano, M., Paolo, T.: 'Knapsack problems: algorithms and computer implementations' (Wiley-Interscience, New York, 1990)
[7] Chu, P.C., Beasley, J.E.: 'A genetic algorithm for the multidimensional knapsack problem', *J. Heuristics*, 1998, **4**, (1), pp. 63–86
[8] Horn, J., Nafpliotis, N., Goldberg, D.: 'A niched Pareto genetic algorithm for multiobjective optimization'. Proc. First IEEE Conf. Evolutionary Computation, Piscataway, USA, 1994, vol. 1, pp. 82–87
[9] Yoon, J., Lee, J., Kim, D., *et al.*: 'Feature selection in multi-label classification using NSGA-II algorithm', *J. KIISE, Softw. Appl.*, 2013, **40**, (3), pp. 133–140
[10] Deb, K., Pratap, A., Agarwal, S., *et al.*: 'A fast and elitist multi-objective genetic algorithm: NSGA-II', *IEEE Trans. Evol. Comput.*, 2002, **6**, (2), pp. 182–197
[11] Khu, S.T., Madsen, H.: 'Multi-objective calibration with Pareto preference ordering: an application to rainfall-runoff model calibration', *Water Resour. Res.*, 2005, **41**, (3), pp. 1–14
[12] Kalyanmoy, D., Amarendra, K.: 'Real-coded genetic algorithms with simulated binary crossover: studies on multimodel and multiobjective problems', *Complex Syst.*, 1995, **9**, (6), pp. 431–454
[13] Hamdan, M.: 'A dynamic polynomial mutation for evolutionary multi-objective optimization algorithms', *Int. J. Artif. Intell. Tools*, 2011, **20**, (1), pp. 209–219
[14] Deb, K., Deb, D.: 'Analysing mutation schemes for real-parameter genetic algorithms', *Int. J. Artif. Intell. Soft Comput.*, 2014, **4**, (1), pp. 1–28
[15] Park, J., Bang, Y., Lee, G., *et al.*: 'Generation of security measure by using simple genetic algorithm', *Proc. KIISE Conf.*, 2003, **30**, (21), pp. 769–771