Social Engineering Optimization (SEO); A New Single-Solution Meta-heuristic Inspired by Social Engineering

Amir Mohammad Fathollahi Fard[a]*, Mostafa Hajiaghaei-Keshteli[b]

Department of Industrial Engineering and Management System, University of Science and Technology of Mazandaran, Behshahr, Iran

Tel: +98 11 34552000 (263), Fax: +98 11 34552008 [a]amirfard@mazust.ac.ir, [b]mostafahaji@mazust.ac.ir

Abstract

Although in recent years many meta-heuristic methods have been presented, most of them are based on population and bear many steps, several parameters and hard to understand and code. So, the researchers usually use the old algorithms instead of recent ones. Contrary to previous work, this paper aims to develop a simple, intelligent and new single-solution algorithm that has four main steps and just three simple parameters to tune. Social Engineering Optimization (SEO) starts with two initial solutions, divided into attacker and defender. The attacker obtains the rules of social engineering techniques to reach desired its goals. The basis of the algorithm is to how an attacker attacks to defender by four different associated techniques. Finally, the proposed SEO is applied to solve some benchmark functions and an engineering problem and also the results show its superiority in comparison with other well-known and recent meta-heuristic.

Keywords:

Social Engineering Optimization (SEO), Meta-heuristics, Optimization techniques.

1. Introduction

Optimization techniques play a key role in today's researches especially in engineering and operations management problems. These techniques are divided into two groups: mathematical programming and heuristic or meta-heuristic algorithms. A well-known classification in meta-heuristics is single solution instead of population-based searches. Holland introduced Genetic Algorithm (GA) inspired by genetics for solving complex and huge real engineering problems as a population-based technique for the first time [11]. Simulated Annealing (SA), based on annealing process of metals, is introduced by Kirkpatrick *et al.* as a single solution technique [16]. Besides, intensification and diversification are two main search phases in meta-heuristics. GA does these important phases by two operators: crossover and mutation which are blind in search space. In the SA, intensification phase is done by neighboring whereas diversification phase are carried by changing the temperature and accepting rule.

In recent developed algorithms, the author finds an intelligent way to search the potential areas by considering two phases. Imperialist Competitive Algorithm (ICA) carries the diversification by forming different empires and does the exploitation in each empire [1]. In Keshtel Algorithm (KA), the Keshtels search the potential places carefully by an intelligent way and do the exploration phase by moving and landing in the lake [8]. In addition, in Red Deer Algorithm (RDA), the authors find a creative way to trade off between the phases by three simple operators. RDA does local search by roaring males and also performs exploitation phase by fighting between different types of males and finally by mating behavior in three ways to search the potential areas, diversification phase is formed [3]. Besides, in Whale Optimization Algorithm (WOA), the bubble-net hunting strategy of Whales probes neighborhoods the best solutions and allows other search agents to explore the potential good areas [20]. All of recent mentioned methods are a population-based meta-heuristic and also it is clear that a single solution approach is simpler than the population based ones. By another point of view, although it seems that these mentioned methods are very good to make a balance between the phases, they have several steps and many parameters to tune and population-based meta-heuristics. Hence, these difficulties motivated the authors to develop a new simple and efficient single-solution meta-heuristic.

In this paper, Social Engineering (SE) science to develop a new simple and intelligent method is implemented. This single-solution meta-heuristic starts with two initial random solutions. These two solutions are divided into two types: attacker and defender. The better solution is selected as the attacker. During the procedure of SE, the attacker wants to defeat the defender by SE attacks' techniques.

Section 2 defines the Social Engineering strategies. Section 3 investigates the Social Engineering Optimization (SEO) and its steps in detail. In section 4, the experimental studies are explored and performance and efficiency of the proposed method is analyzed by benchmark functions and also an engineering problem. Finally, the results and future works are given in section 5.

2. Social Engineering

Social Engineering (SE) is defined as indirect attacks to obtain the people and reveal their information by using certain techniques. Attackers want to reach their desired goals or objectives by advancement technologies and increase the social interactive media how to do changing in political, economical and social advance toward a particular community which has found differences [18].

Sarah Granger explained the main cycles of Social Engineering in her works [5-6]. In this way, she acknowledged that defense against the Social Engineering attacks has a common pattern such as a mass. In addition, she claimed that the Social Engineering attack procedure has a four step cycle.

In the first phase, the training and retraining from attacker to defender is considered. In this regard, the attacker wants to obtain some special information from defender. This information can be put in different contents. Perhaps, the first questions are about famous special clip video, music, sport and public events that occur in the community and other dimension of personal family systems. Any forwards to start and conduct a SE attack needs a series of such data to identify the person's reactions to design an attack on him/her. The idea of SE experts is explained that all social media has expended its range to all ages for trying your aim and bringing the community in the form of their intended route, although they pretend that they're going to help them [22].

In the second phase, the attacker spots a SE attack. It is obvious that before carrying out an attack, on the point where the probability of success is more than other points, it must be identified. The attacker takes you (defender) to a position that he/she prefers the defender to be placed in the way of his/her goals. The most important thing to prevent an attack on his/her main demand is that defender can understand and think like an attacker. According to past learning to spot a Social Engineering attack the dependent techniques are different. Such as: pretext placement, obtaining, diversion theft, phishing and so on. In each technique, aggressive manner is so different and its profit is variable, too.

The next phase is how to respond this SE attack and react of defender and how much information it wishes to make striker. In SE attacks, the most important phase is how to respond to an attack. The pervious experiences and his/her knowledge about this kind of attack may be helped him/her. At the least, in final step, the attacker try to steal each thing that is may be useful and strike to defender. Or if an attack is not achieved satisfactory results or otherwise offensive repeats or stopped by person and someone else chooses.

3. Social Engineering Optimization (SEO)

Fig. 1 shows the flowchart of proposed method. In this algorithm each solution is counterpart to a person and traits of each person such as his/her ability in contents of mathematical, sport, business and so on are the counterpart of all variables of each solution in search space. To start the algorithm, the two random solutions are initialized and better solution is selected as attacker and the other is named defender. To simulate training and retraining of attacker from defender, some random experiments for each trait of defender are designed. The attacker tries to evaluate the defender by his/her traits. The counterpart of training and retraining in search space is copying a trait from attacker to the same trait in defender and computing the rate of retraining of attacker from defender, simultaneously. In the next step, spotting a SE attack from attacker to defender is the counterpart of changing the position of defender by an intelligent way in feasible space. In the responding a SE attack, fitness of new position of defender is calculated and the old and current positions of defender are compared and best position is selected. If the ability of defender is better than attacker, positions of attacker and defender are changed. At the end, to strike to the defender, it is annihilated and generated by a new random solution in search space.

In the proposed SEO, local search or exploitation phase is done by training and retraining between attacker and defender. Also, spotting a SE attack and responding to it, does the intensification action. Indeed, exploration is performed by annihilating the defender and creating a new one.

#####Please insert **Fig. 1** around here#####

3.1. Initialize the attacker and the defender

The goal of optimization is to find an optimal solution among all feasible solutions. In this way, an array of variable values to be optimized is shaped. For instance in GA, the terminology of array is named "*chromosome*", but here in SEO, this term, "*person*" is used for this array. Therefore, person is counterpart of solution. In addition, in GA, "*gene*" is used for each variable of an array. But in SEO, "*trait*" is defined on person for each variable. This is included in different contents of mathematical, sport, business and so on. In an N_{var}–dimensional optimization problem, a person is a $1\times N_{var}$ array. This array is defined by:

$$person=[X_1, X_2, X_3, \ldots, X_{N_{var}}]. \quad (1)$$

Also the value of Objective Function (OF) is evaluated for two persons as follows:

$$Value=f(person)=f(X_1, X_2, X_3, \ldots, X_{N_{var}}). \quad (2)$$

The algorithm starts with two random solutions. The better solution is selected as the attacker and else is named as the defender.

3.2. Train and retrain

This step aims to show the training and retraining the attacker from the defender. In this way, the attacker tries to test on each trait of the defender to recognize the most efficient trait. In this regard, $\propto$ percent of attacker traits are selected randomly and are replaced in the same traits of defender as follows:

$$N_{Train} = round\{\propto . nVar\} \quad (3)$$

where $\propto$ is the percent of selection traits and $nVar$ is the number of all traits in a person. So, N_{Train} is the number of traits that will be tested on the defender.
To explain more, we display an example to clarify this operator in **Fig. 2**. Four traits are considered for each person. The red box is symbol of attacker and green box is considered as defender. In this instance, all of traits are between zero and one. The rate of $\propto$ is equal to 0.5 in this example. In this example, the defender by retaining rate of 10 is replaced by the current defender.

#####Please insert **Fig. 2** around here#####

3.3. Spot an attack

To spot an attack, this paper considers four different techniques involving: obtaining, phishing, diversion theft and pretext. These mentioned techniques are used in a random way to search the feasible space. In addition, they utilize just only one parameter named β as an input variable of search engine. The red circle is specified the attacker and the green for the defender. New position during procedure of technique is shown by arrows. In all equations, def_{new} is shown the new position of defender during attack and def_{old} and att are the current position of defender and attacker. The algorithm is developed in a way that the user can employ one operator among the four ones.

3.3.1. Obtaining

In this technique, the attacker abuses defender directly as guidance to obtain the desired goals. **Fig. 3** is also demonstrated it as a visual simulation used in the fact. In this regard, one new solution is generated. To create the new position, this equation is proposed:

$$def_{new} = def_{old} \times (1 - \sin\beta \times U(0,1)) + \frac{(def_{old} + att)}{2} \times \sin\beta \times U(0,1) \quad (4)$$

#####Please insert **Fig. 3** around here#####

3.3.2. Phishing

To design this technique, the attacker pretends to approach the defender and then defender moves to a place that attacker wants to be there. This proposed technique generates two new solutions as shown in **Fig. 4**. The equations are displayed as follows:

$$def_{new}^{1} = att \times (1 - \sin\beta \times U(0,1)) + \frac{(def_{old} + att)}{2} \times \sin\beta \times U(0,1) \quad (5)$$

$$def_{new}^{2} = def_{old} \times \left(1 - \sin\left(\frac{\pi}{2} - \beta\right) \times U(0,1)\right) + \frac{(def_{old} + att)}{2} \times \sin\left(\frac{\pi}{2} - \beta\right) \times U(0,1) \quad (6)$$

#####Please insert **Fig. 4** around here#####

3.3.3. Diversion theft

In this technique, at first the attacker guides the defender to a position, that in fact, this is a deception for defender. To depict this process, **Fig. 5** shows this proposed technique. In this process, just one solution is created. The following equation is shown the formulation of this action:

$$def_{new} = def_{old} \times (1 - \sin\beta \times U(0,1)) + \frac{(def_{old} + att \times U(0,1) \times \sin(\frac{\pi}{2} - \beta))}{2} \times \sin\beta \times U(0,1) \quad (7)$$

#####Please insert **Fig. 5** around here#####

3.3.4. Pretext

During this technique, the attacker baits some traits which are favors for a defender and guides the defender. This technique is useful to defeat the defender. **Fig. 6** displays this proposed technique. In this process one solution is generated and created during this process by following equation:

$$def_{new} = \left(def_{old} \times U\left(0,1\right) \times \sin\left(\frac{\pi}{2} - \beta\right)\right) \times (1 - \sin\beta \times U(0,1)) + \frac{\left(\left(def_{old} \times U\left(0,1\right) \times \sin\left(\frac{\pi}{2} - \beta\right)\right) + att\right)}{2} \times \sin\beta \times U(0,1) \tag{8}$$

######Please insert **Fig. 6** around here######

3.4. Respond to SE attack

New position of the defender is evaluated and compared with the old position of the defender. Then, the best position for the defender is selected and if the new position of the defender is better than attacker, the defender and attacker are exchanged as shown in **Fig.** 7.

######Please insert **Fig.** 7 around here######

3.5. Select a new person as defender

The attacker annihilates the defender and selects a new person randomly to perform SE rules.

3.6. Stopping condition

Like other meta-heuristics, the stop condition may be maximum time of simulation or the quality of best solution ever found or other selected condition by user. **Fig. 7** displays the pseudo-code of proposed algorithm.

######Please insert **Fig. 8** around here######

4. Experimental studies

As mentioned earlier, there have been developed several metaheuristics with their applications in engineering problems. For example, Hajiaghaei-Keshteli and Aminnayeri firstly used the Keshtel Algorithm (KA) in an integrated scheduling of production and rail transportation problem [9]. Also, Nazeri-Shirkouhi *et al.* [21] used the Imperialist Competitive Algorithm (ICA) to solve an integrated product mix-outsourcing problem for the first time, as well. On the other hand, it is formal to study the performance of novel algorithm with some benchmark functions. After studying the related and recent papers, we planned to use a well-known problem in various research area to probe the behavior of SEO in different situations and to exhibit the performance and effectiveness of this new optimization algorithm.

4.1. Benchmark functions

In this part, four famous benchmark functions which have been used in pervious researches are utilized [2, 19, 24]. These problems are numbered as P1 to P٦ and all of them are minimization problems. In all standard functions, the optimum value is equal to zero. Each of them has certain characteristics. The details on these functions are show in **Table 1**.

######Please insert **Table 1** around here######

In addition, proposed SEO is divided into four versions relay on four developed techniques in designing a SE attack. We also employ some well-known meta-heuristics such as, GA and SA as well-known old methods, PSO as one of the best technique in swarm-based algorithms [15], Artificial Bee Colony algorithm (ABC) [14] and also some recently developed meta-heuristics like Imperialist Competitive Algorithm (ICA), Firefly Algorithm (FA) [23] and Red Deer Algorithm (RDA) to solve the problems, in order to compare with the four proposed methods. **Table 2** shows the value of parameters in all algorithms. Besides, in all methods the stopping condition is time interval and it is equal to 10 seconds in all problems. In addition, we runs them for thirty times.

######Please insert **Table 2** around here######

The outputs of the methods and the behavior of algorithms are shown in **Table 3** and **Fig. 9**. The results show the superiority of SEO and its behavior to find the optimal solution. Four versions of SEO are presented in this study based on four proposed techniques. All of them are reached optimal solutions and the second technique is most successful in among other four techniques. In addition, it seems user can choice suitable technique by considering the type and dimension of the problems.

######Please insert **Table 3** around here######
######Please insert **Fig. 9** around here######

4.2. Single machine scheduling problem

In this part, we consider a single machine scheduling problem. Single-machine scheduling or single-resource scheduling is the process of assigning a group of tasks to a single machine or resource. The tasks are arranged so that one or many performance measures may be optimized. In many practical scheduling problems, costs arising from both earliness and tardiness of individual jobs must be minimized. For example in production systems in which there are shipping dates for orders, inventory carrying cost are incurred if jobs are finished earlier than the shipping dates, and shortage costs (penalty costs and backorder costs) are incurred if jobs are finished later than the shipping dates. To minimize the total costs, tardiness and earliness of jobs should be minimized. We defined the earliness and tardiness as $\max_i(d_i - C_i, 0)$ and $\max_i(C_i - d_i)$, respectively, where d_i the due date and C_i is completion time of job *i*.

We know this problem as an NP-*hard* problem [4]. In the literature, there have been several meta-heuristics are used in this problem [12-13, 17]. In our encoding scheme, a two-stage technique called Random-Key (*RK*) is utilized to solve our discrete problems. Researchers use this technique repeatedly during two recent decades [7, 10]. This technique helps to solve our problem with different methods and operators. In *RK*, we have two phases. At first, we make a solution by random numbers. Secondly, we convert this solution to a feasible solution. So, in the first phase, an array of variable between zero and one is initialized. Then, we sort the variables of this array and specify the sequence of jobs to compute the objective function. **Fig. 10** displays these two steps in initialization of algorithms. In this instance the rate of β is equal to 0.5 in all techniques. And also the total number of traits is equal to 4.

#####Please insert **Fig. 10** around here#####

To compare the proposed algorithms, we utilized three powerful methods. GA, ICA and RDA are used in this study. In order, the problems are sorted as well. We define 6 different sizes of problems and assign equal time. The results are shown in **Table 4.** First of all, these results show the efficiency and performance of proposed SEO among the algorithms. In addition SEO_2 depicts better performance among other. It should be noted that the results shown in the table, is the best solution found among thirty runs. Also, the interaction of problem size and behavior of algorithms are depicted in **Fig. 11**, in terms of mean RPD in thirty runs.

#####Please insert **Table 3** around here#####
#####Please insert **Fig. 11** around here#####

5. Conclusion and future works

In this work, a new optimization algorithm, inspired by Social Engineering science is developed. This algorithm is so simple and intelligent. Besides, it makes a balance between the phases by creative methods. This algorithm starts with two random solutions. The better solution is selected as the attacker. During the rules of SE strategies, the attacker reaches desired goals by obtaining defender and spots a SE attack. In our work, four proposed techniques are proposed to design attacks. The result of experiments shows the performance and efficiency of proposed method.

For future studies, more comprehensive analyses on the SEO may still required to be explored. Moreover, we can employ some other real techniques in SE attacks to develop the proposed SEO. In addition, some other real scale optimization problems can be utilized to evaluate the performance of the proposed algorithm.

References:

[1] E. Atashpaz-Gargari, C. Lucas, "Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition." in: *IEEE Congress on Evolutionary Computation*, Singapore (2007), 4661–4667.

[2] A. Ebrahimi, E. Khamehchi, "Sperm Whale Algorithm: an Effective Metaheuristic Algorithm for Production Optimization Problems." *Journal of Natural Gas Science & Engineering*, 29, (2016), 211-222.

[3] A. M. Fathollahi Fard, M. Hajiaghaei-Keshteli, "Red Deer Algorithm (RDA); A New Optimization Algorithm Inspired by Red Deers' Mating." *12th International Conference on Industrial Engineering (ICIE 2016)*, Tehran, Iran, (2016), 34-35.

[4] M. Feldman, D. Biskup, "Single machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches." *Computers & Industrial Engineering*, 44, (2003), 307–323.

[5] S. Granger, "Social Engineering Fundamentals, Part I: Hacker Tactics." *Security Focus,* December 18, (2001).

[6] S. Granger, "Social Engineering Fundamentals, Part II: Combat Strategies." *Security Focus,* January 9, (2002).

[7] M. Hajiaghaei-Keshteli, "The allocation of customers to potential distribution centers in supply chain network; GA and AIA approaches." *Appl. Soft Comput.* 11, (2011), 2069–2078.

[8] M. Hajiaghaei-Keshteli, M. Aminnayeri, "Keshtel Algorithm (KA); a new optimization algorithm inspired by Keshtels' feeding." *Proceeding in IEEE Conference on Industrial Engineering and Management Systems* (2013), 2249–2253.

[9] M. Hajiaghaei-Keshteli, M. Aminnayeri, "Solving the integrated scheduling of production rail transportation problem by Keshtel algorithm." *Applied Soft Computing*, 25, (2014), 184–203.

[10] M. Hajiaghaei-Keshteli, S. Molla-Alizadeh-Zavardehi, R. Tavakkoli-Moghaddam, "Addressing a nonlinear fixed-charge transportation problem using a spanning tree-based genetic algorithm." *Computers and Industrial Engineering,* 59, (2010), 259–271.

[11] J. H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, University of Michigan Press, Michigan, Ann Arbor, (1975).

[12] J. A. Hoogeveen, S. L. van de Velde, "Scheduling around small common due date." *European Journal of Operational Research,* 55, (1991), 237-242.

[13] R. J. W. James, "Using tabu search to solve the common due date early-tardy machine scheduling problem." *Computers and Operation Research,* 24, (1997), 199-208.

[14] D. Karaboga, B. Akay, "A comparative study of Artificial Bee Colony algorithm." *Applied Mathematics and Computation*, 214, (2009), 108–132.

[15] J. Kennedy, R. Eberhart, "Particle swarm optimization." *Proc. IEEE Int. Conf. Neural Networks*. Perth, Australia, (1995), 1942-1945.

[16] S. Kirkpatrick, C. D. Gelatto, M. P. Vecchi, "Optimization by simulated annealing." *Science*, 220, (1983), 671-680.

[17] C. Y. Lee, S. J. Kim, "Parallel genetic algorithms for the earliness–tardiness job scheduling problem with general penalty weights." *Computers & Industrial Engineering*, 28, (1995), 231–243.

[18] X. R. Luo, R. Brody, A. Seazzu, S. Burd, "Social Engineering: The Neglected Human Factor for." *Managing Information Resources and Technology: Emerging Applications and Theories,* (2013), 151.

[19] S. Mirjalili, "SCA: A Sine Cosine Algorithm for Solving Optimization Problems." *Knowledge-Based Systems,* 96, (2016), 120-133.

[20] S. Mirjalili, A. Lewis, "The Whale Optimization Algorithm." *Advances in Engineering Software*, 95, (2016), 51-67.

[21] S. Nazeri-Shirkouhi, H. Eivazy, R. Ghodsi, K. Rezaie, E. Atashpaz-Gargari, "Solving the integrated product mix outsourcing problem using the Imperialist Competitive Algorithm." *Expert Systems with Applications*, 37, (2010), 7615–7626.

[22] A. M. Weinberg, "Can technology replace social engineering?" Bulletin of the Atomic Scientist, 22, (10), (1966), 4-8.

[23] X.S. Yang, "Nature-inspired Metaheurisitc Algorithm." *Luniver Press*, (2008).

[24] Yu-Jun Zheng, "Water Wave Optimization: A new nature-inspired metaheuristic." *Computers & Operations Research,* 55, (2015), 1–11.

Figures:

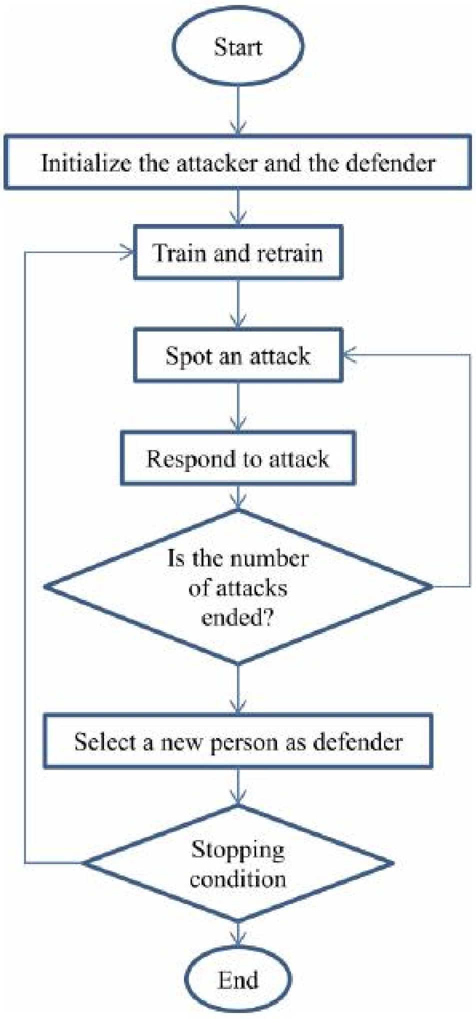

Start
Initialize the attacker and the defender
Train and retrain
Spot an attack
Respond to attack
Is the number of attacks ended?
Select a new person as defender
Stopping condition
End

Fig. 1 The flowchart of SEO

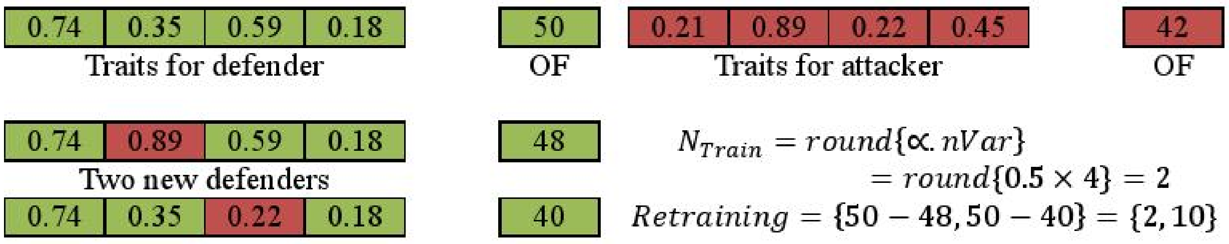


Fig. 2. A simple example of training and retraining process

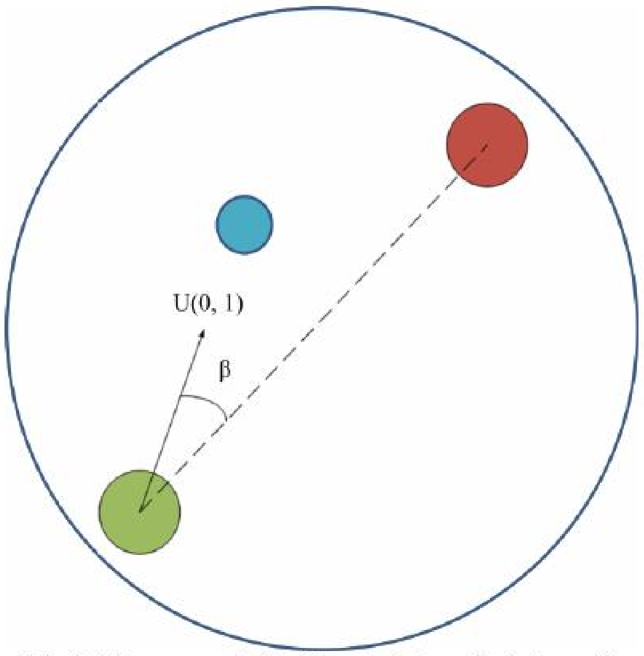


Fig. 3. The proposed obtaining technique (technique 1)

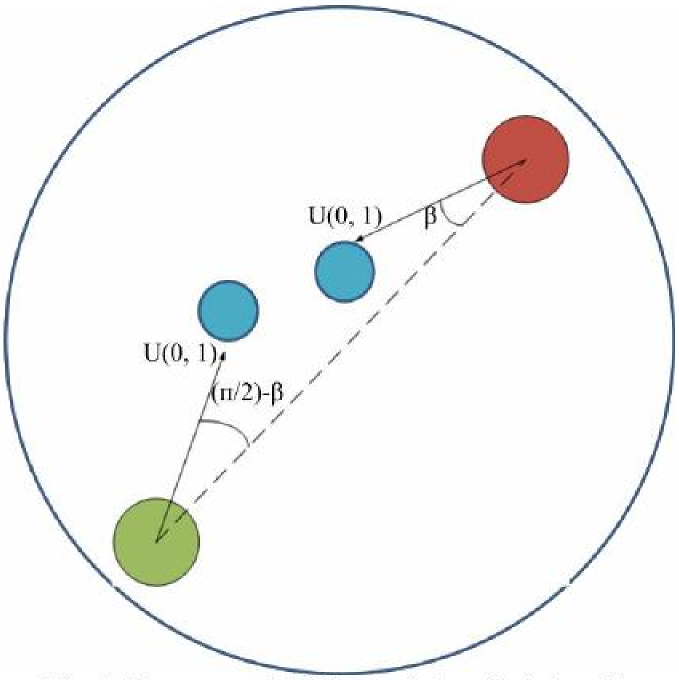


Fig. 4. The proposed phishing technique (technique 2)

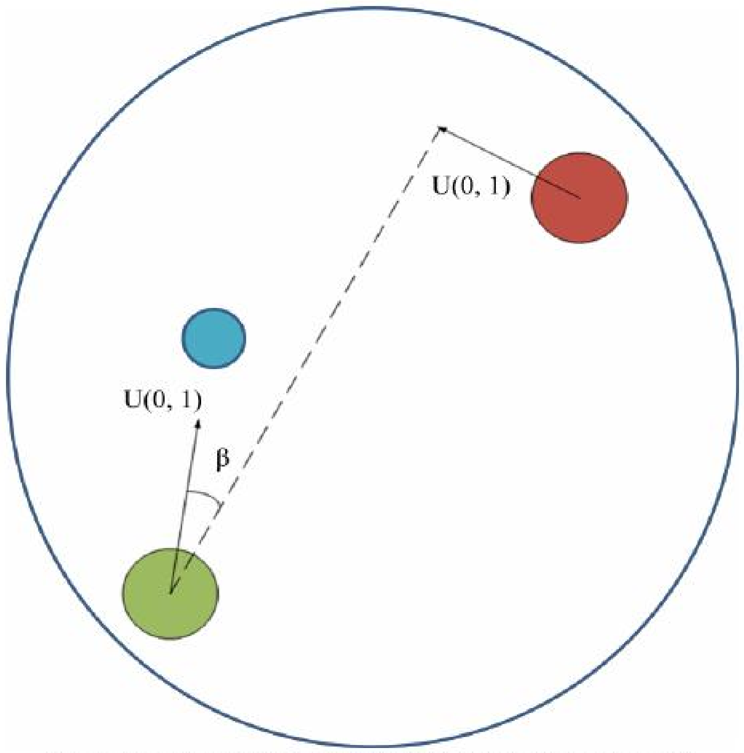


Fig. 5. The proposed diversion theft technique (technique 3)

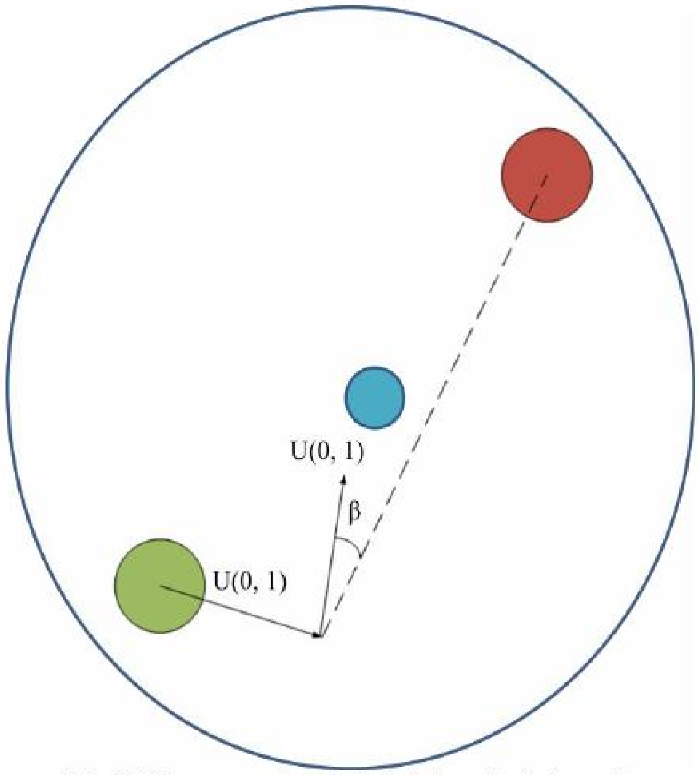


Fig. 6. The proposed pretext technique (technique 4)

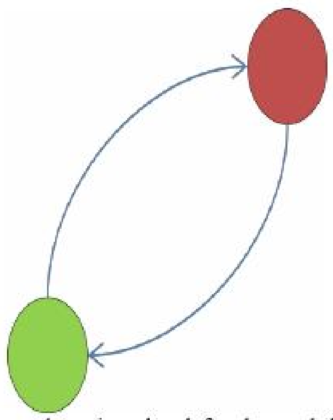
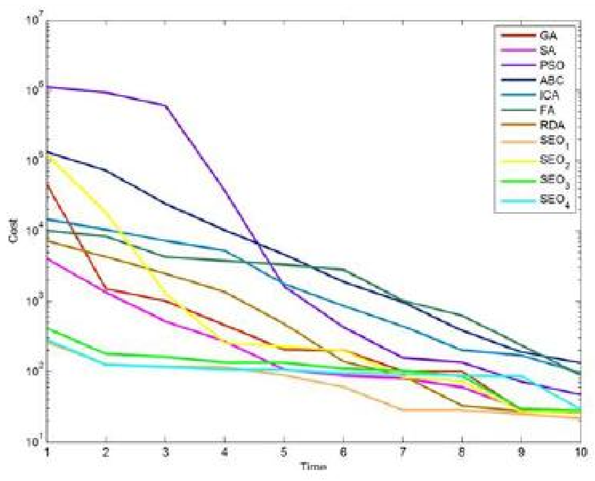


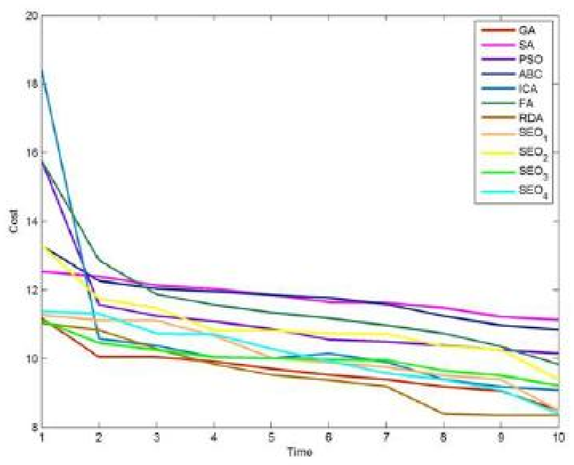
Fig. 7. The exchanging the defender and the attacker

```
T1=clock;
Initialize attacker and defender
It=1;
while solving_time < Max_time
        Do training and retraining;
        Num_attack=1;
        while Num_attack < Max_attack
                Spot an attack;
                Check the boundary;
                Respond to attack;
                if the OF of defender is lower than attacker
                  Exchange the defender and attacker position;
                endif
                Num_attack= Num_attack+1;
        endwhile
        Create a new solution as defender;
        It=It+1;
        T2=clock;
        Solving_time=T2- T1;
endwhile
Return attacker.
```

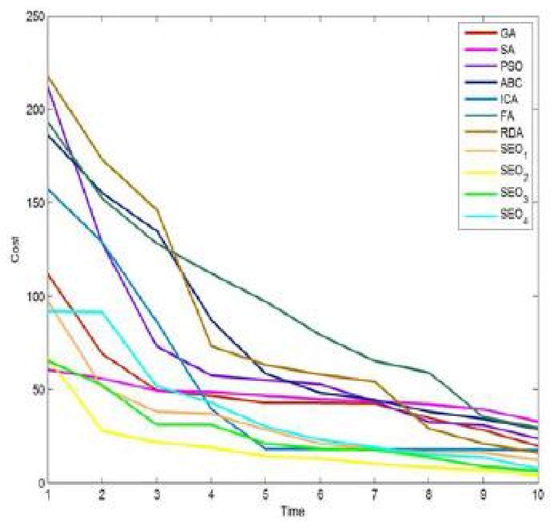
Fig. 8. The pseudo-code of proposed SEO



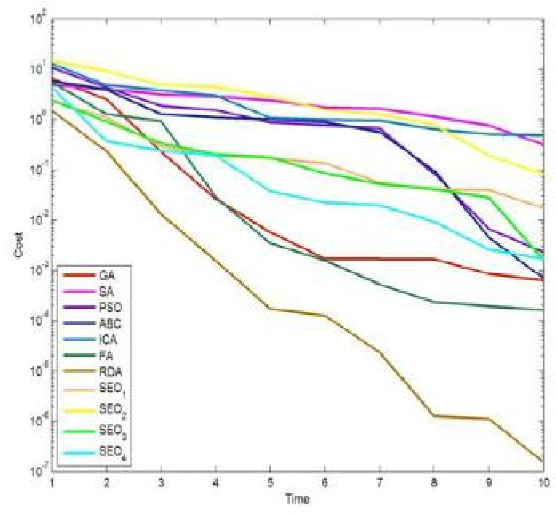
P1: *Rosenbrock* function



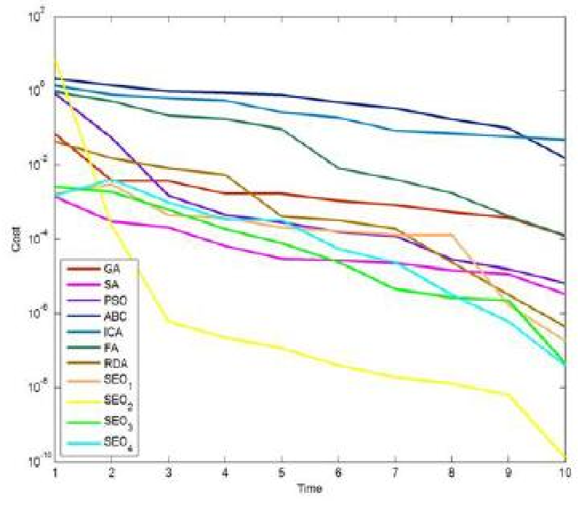
P2: *Quratic* function



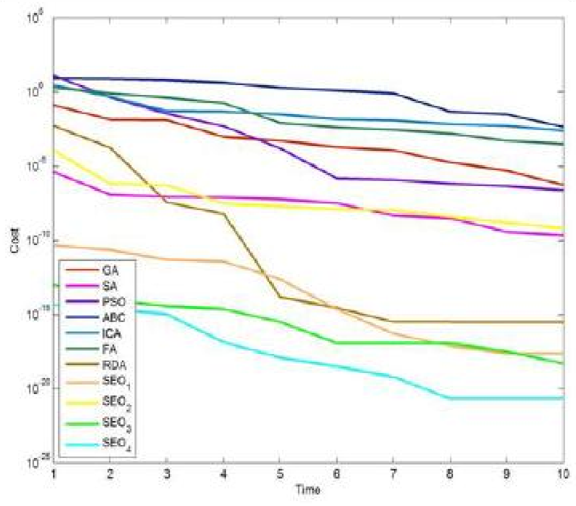
P3: *Rastrigin* function



P4: *Ackley* function



P5: *Greiwank* function



P6: *Schwefel 1.2* function

Fig. 9. The behavior of algorithms in benchmark functions

Defender:

0.67	0.34	0.89	0.26
3	2	4	1

Attacker:

0.08	0.76	0.12	0.73
1	4	2	3

Technique 1:

0.55	0.57	0.64	35

$$0.55 = 0.67 \times (1 - \sin 0.5 \times rand) + \left(\frac{0.67 + 0.08}{2}\right) \times \sin 0.5 \times rand$$

2	3	4	1

$$0.57 = 0.34 \times (1 - \sin 0.5 \times rand) + (\frac{0.34 + 0.76}{2}) \times \sin 0.5 \times rand$$

$$0.64 = 0.89 \times (1 - \sin 0.5 \times rand) + (\frac{0.89 + 0.12}{2}) \times \sin 0.5 \times rand$$

$$0.35 = 0.26 \times (1 - \sin 0.5 \times rand) + (\frac{0.26 + 0.73}{2}) \times \sin 0.5 \times rand$$

Technique 2:

0.22	0.95	0.18	0.48
2	4	1	3

$$0.22 = 0.08 \times (1 - \sin 0.5 \times rand) + \left(\frac{0.67 + 0.08}{2}\right) \times \sin 0.5 \times rand$$

$$0.95 = 0.76 \times (1 - \sin 0.5 \times rand) + (\frac{0.34 + 0.76}{2}) \times \sin 0.5 \times rand$$

$$0.18 = 0.12 \times (1 - \sin 0.5 \times rand) + (\frac{0.89 + 0.12}{2}) \times \sin 0.5 \times rand$$

$$0.48 = 0.73 \times (1 - \sin 0.5 \times rand) + (\frac{0.26 + 0.73}{2}) \times \sin 0.5 \times rand$$

0.72	0.56	0.39	0.47
4	3	1	2

$$0.72 = 0.67 \times (1 - \sin(\left(\frac{3.14}{2}\right) - 0.5) \times rand) + \left(\frac{0.67 + 0.08}{2}\right) \times \sin\ (\left(\frac{3.14}{2}\right) - 0.5) \times rand$$

$$0.56 = 0.34 \times (1 - \sin(\left(\frac{3.14}{2}\right) - 0.5) \times rand) + (\frac{0.34 + 0.76}{2}) \times \sin(\left(\frac{3.14}{2}\right) - 0.5) \times rand$$

$$0.39 = 0.89 \times (1 - \sin(\left(\frac{3.14}{2}\right) - 0.5) \times rand) + \left(\frac{0.89 + 0.12}{2}\right) \times \sin\left(\left(\frac{3.14}{2}\right) - 0.5\right) \times rand$$

$$0.47 = 0.26 \times (1 - \sin(\left(\frac{3.14}{2}\right) - 0.5) \times rand) + (\frac{0.26 + 0.73}{2}) \times \sin(\left(\frac{3.14}{2}\right) - 0.5) \times rand$$

Technique 3:

0.58	0.30	0.64	0.31
3	1	4	2

$$0.58 = 0.67 \times (1 - \sin 0.5 \times rand) + ((0.67 + 0.08 \times \sin(\left(\frac{3.14}{2}\right) - 0.5) \times rand)/2) \times \sin\ 0.5 \times rand$$

$$0.30 = 0.34 \times (1 - \sin 0.5 \times rand) + ((0.34 + 0.76 \times \sin\left(\left(\frac{3.14}{2}\right) - 0.5\right) \times rand)/2) \times \sin\ 0.5 \times rand$$

$$0.64 = 0.89 \times (1 - \sin 0.5 \times rand) + ((0.89 + 0.12 \times \sin\left(\left(\frac{3.14}{2}\right) - 0.5\right) \times rand)/2) \times \sin\ 0.5\ \times rand$$

$$0.31 = 0.26 \times (1 - \sin 0.5 \times rand) + ((0.26 + 0.73 \times \sin\left(\left(\frac{3.14}{2}\right) - 0.5\right) \times rand)/2) \times \sin\ 0.5\ \times rand$$

Technique 4:

0.35	0.29	0.18	0.23
4	3	1	2

$$0.35{=}0.67\times \sin\left(\left(\frac{3.14}{2}\right) - 0.5\right) \times rand \times (1 - \sin 0.5 \times rand) + ((0.67\ \times \sin\left(\left(\frac{3.14}{2}\right) - 0.5\right) \times rand + 0.08)/2)\ \times \sin\ 0.5\ \times rand$$

$$0.29{=}0.34\times \sin\left(\left(\frac{3.14}{2}\right) - 0.5\right) \times rand \times (1 - \sin 0.5 \times rand) + ((0.34\ \times \sin\left(\left(\frac{3.14}{2}\right) - 0.5\right) \times rand + 0.76)/2)\ \times \sin\ 0.5\ \times rand$$

$$0.18{=}0.89\times \sin\left(\left(\frac{3.14}{2}\right) - 0.5\right) \times rand \times (1 - \sin 0.5 \times rand) + ((89\ \times \sin\left(\left(\frac{3.14}{2}\right) - 0.5\right) \times rand + 0.12)/2)\ \times \sin\ 0.5\ \times rand$$

$$0.23{=}0.26 \times \sin\left(\left(\frac{3.14}{2}\right) - 0.5\right) \times rand \times (1 - \sin 0.5 \times rand) + ((0.26\ \times \sin\left(\left(\frac{3.14}{2}\right) - 0.5\right) \times rand + 0.73)/2)\ \times \sin\ 0.5\ \times rand$$

Fig. 11. The encoding scheme

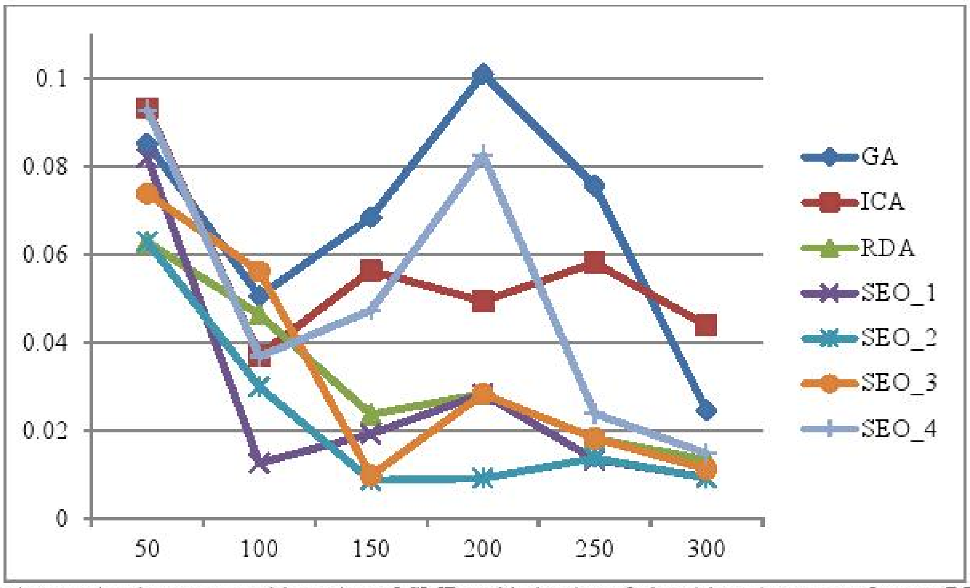


Fig. 12. The interaction between problem size of SMP and behavior of algorithms in terms of mean RPD in thirty time runs

Tables:

Table 1. Benchmark Functions
C=Characteristic, U=Unimodal, M=Multimodal, S=Separable, N=Non-Separable, D=Dimension.

Function		**Formulation**	**C**	**Range**	***D***
P1	Rosenbrock	$f(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	UN	$[-30,30]$	30
P2	Quratic	$f(x) = \sum_{i=1}^{n} ix_i^4 + random[0,1)$	US	$[-1.28,1.28]$	30
P3	Rastrigin	$f(x) = x_i^2 - 10\cos(2\pi x_i) + 10$	MS	$[-5.12,5.12]$	30
P4	Ackley	$f(x) = -20\exp\left(-0.2\sqrt{\sum_{i=1}^{n} x_i^2 \frac{1}{n}}\right) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	MN	[-32, 32]	30
P۵	Greiwank	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	MN	$[-600,600]$	**30**
P6	Schwefel 1.2	$f(x) = \sum_{i=1}^{n} (\sum_{j}^{i} x_j)^2$	UN	$[-100,100]$	30

Table 2
Parameters of the algorithms

Algorithms	Parameters
GA	Initial population=100, Maximum time of simulation=10 (Sec), Mutation rate=0.2, Mutation percentage=0.05, Crossover rate=0.5, Crossover percentage=0.75
SA	Maximum time of simulation=10 (Sec), Number of Sub-iteration=50, Initial Temperature=10, Rate of reduction=0.99, Rate of moving=0.05
PSO	Initial population=100, Maximum time of simulation=10 (Sec), Personal learning coefficient=2, Global learning coefficient =2, Inertia rate damping ratio=0.99, Initial weight=1
ABC	Initial population=100, Maximum time of simulation=10 (Sec), Abandonment limit parameter: L=round(0.5*nVar*nPop), Acceleration coefficient upper bound=0.3

ICA	Initial population=100, Maximum time of simulation=10 (Sec), Number of empires=8, Number of colonies=Initial population-Number of empires, Assimilation coefficient=0.2, Revolution probability=0.1, Rate of revolution=0.05, Colonies mean cost coefficient=0.1
FA	Initial population=10, Maximum time of simulation=10 (Sec), Attraction coefficient base of value=2, Light absorption coefficient=1, Mutation coefficient=0.2, Mutation Coefficient Damping Ratio=0.99
RDA	Initial population=100, Maximum time of simulation=10 (Sec), Number of males=15, Number of hinds=Initial population-Number of males, Alpha=0.9, Betta=0.4, Gamma=0.7
SEO_1	Maximum time of simulation=10 (Sec), Rate of training=0.2, Rate of spotting a Social Engineering attack=0.1, Number of attacks=50
SEO_2	Maximum time of simulation=10 (Sec), Rate of training=0.2, Rate of spotting a Social Engineering attack=0.5, Number of attacks=50
SEO_3	Maximum time of simulation=10 (Sec), Rate of training=0.2, Rate of spotting a Social Engineering attack=0.05, Number of attacks=50
SEO_4	Maximum time of simulation=10 (Sec), Rate of training=0.2, Rate of spotting a Social Engineering attack=0.05, Number of attacks=50

Table 3
The result of Final output of algorithms

P_i	GA	SA	PSO	ABC	ICA	FA	RDA	SEO_1	SEO_2	SEO_3	SEO_4
P1	27.7778	26.515	47.1011	132.2873	9.63E+01	88.49522	26.2837	21.7232	26.6891	28.1382	28.8505
P2	8.4918	11.1288	10.1528	10.8443	9.0834	9.8375	8.3482	8.4883	9.3847	9.2113	8.3181
P3	19.8992	32.9452	23.879	28.9711	17.8435	29.5132	16.2931	12.4404	3.9806	6.6029	7.5429
P4	6.32E-04	0.3201	0.0023	0.0007	0.4816	1.63E-04	1.54E-07	0.0173	0.0825	0.0016	0.0017
P5	1.23E-04	3.32E-06	6.54E-06	1.54E-02	4.83E-02	1.28E-04	4.37E-07	1.84E-07	1.28E-10	4.64E-08	3.95E-08
P6	5.47E-07	2.19E-10	2.53E-07	4.59E-03	2.55E-03	3.26E-04	3.28E-16	2.38E-18	6.53E-10	5.28E-19	2.19E-21

Table 4
The results of Single Machine Problem (SMP)

Number of jobs	Time (Sec)	GA	ICA	RDA	SEO_1	SEO_2	SEO_3	SEO_4
50	5	8019	8912	8358	8641	7934	8249	7831
100	10	111940	118537	114255	112269	109802	118357	110452
150	15	256639	254402	254517	248842	250374	248573	247721
200	20	962543	995432	998197	956733	946702	972881	958941
250	25	1636638	1622396	1609515	1529561	1495936	1671159	1502090
300	30	2294702	2327774	2308051	2204118	2380018	2239834	2358895