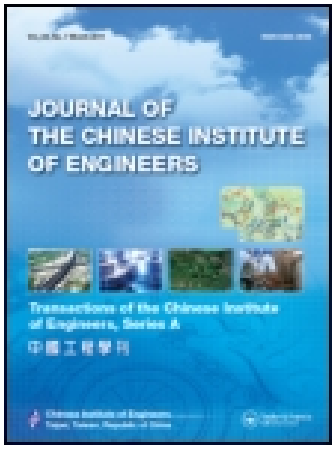

Journal of the Chinese Institute of Engineers
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tcie20

A genetic algorithm with domain knowledge for weapon-target assignment problems

Zne-Jung Lee [a], Shun-Feng Su [a] & Chou-Yuan Lee [a]
[a] Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan 106, R.O.C.
Published online: 03 Mar 2011.

To cite this article: Zne-Jung Lee, Shun-Feng Su & Chou-Yuan Lee (2002) A genetic algorithm with domain knowledge for weapon-target assignment problems, Journal of the Chinese Institute of Engineers, 25:3, 287-295

To link to this article: http://dx.doi.org/10.1080/02533839.2002.9670703

A GENETIC ALGORITHM WITH DOMAIN KNOWLEDGE FOR WEAPON-TARGET ASSIGNMENT PROBLEMS

Zne-Jung Lee, Shun-Feng Su*, and Chou-Yuan Lee
Department of Electrical Engineering
National Taiwan University of Science and Technology
Taipei, Taiwan 106, R.O.C.

Key Words: genetic algorithm, weapon to target assignment, greedy algorithm.

ABSTRACT

In this paper, a novel genetic algorithm, including domain specific knowledge into the crossover operator and the local search mechanism for solving weapon-target assignment (WTA) problems is proposed. The WTA problem is a full assignment of weapons to hostile targets with the objective of minimizing the expected damage value to own-force assets. It is an NP-complete problem. In our study, a greedy reformation and a new crossover operator are proposed to improve the search efficiency. The proposed algorithm outperforms its competitors on all test problems.

I. INTRODUCTION

On modern battlefields, it is an important task for a planner to make a proper weapon-target assignment (WTA) for threat targets. The WTA problem is to find an assignment of weapons to targets with the objective of minimizing the expected damage value to own-force assets. It is easy to see that a WTA problem is an NP-compete problem and is difficult to solve directly (Lloyd and Witsenhausen, 1986; Pepyne *et al*. 1997; William and Fred, 1990). Various methods for solving optimization problems have been reported in the literature (Bjorndal *et al*., 1995; Hammer, 1965; Hammer *et al*., 1984; Ibarraki *et al*., 1988; Sahni and Gonzales, 1976). Those methods are based on graph search approaches and usually result in exponential computational complexities.

Genetic Algorithms (GAs) have been widely used as search algorithms in various applications and have demonstrated good performance. Fundamentally, GAs are based on the mechanism of natural selection to search for a solution that optimizes a given fitness function. Vast GAs have been proposed in the literature (Back *et al*., 1997; Davis, 1991; Goldberg and Lingle, 1985; Lai and Coghill, 1998; Tate and Smith, 1995; Sun and Wang, 1994; Ngo and Li, 1998) and have shown superior performance over other methods. As a consequence, GAs seem to be good approaches for solving WTA problems. However, GAs may cause deterioration in search performance if their operators are not carefully designed (Jiao and Wang, 2000; Merz and Freisleben, 2000). In our previous work (Lee *et al*., 2000), we have employed simple GAs to solve WTA problems. Even though the employed GA could find the best solution in those simulated cases, the search efficiency seemed not good enough. Recently, genetic algorithms with local search have also been considered as good alternatives for solving optimization problems (Goldberg, 1989; Kolen and Pesch, 1994; Lin and Lee, 1995). They have incorporated a local search mechanism into GA search algorithms and have shown good capability in solving complex search problems.

*Correspondence addressee

In this paper, we reported our work on employing GAs for solving WTA problems. In order to have more efficient searches, the used chromosomes and genetic operators for GAs must be carefully designed. The chromosome is used for encoding solutions, and genetic operators, crossover and mutation, can help to efficiently search for feasible solutions. In our study, we first focused on designing suitable chromosomes for the WTA problem. Furthermore, a new crossover operator is proposed to keep good genes and enhance the search performance. In this study, we also added local search mechanisms into GAs to find solutions for WTA problems. Various local search mechanisms have been considered in our research including those used in (Kolen and Pesch, 1994) and Simulated Annealing (SA) (Lin *et al*., 1993). In this paper, a heuristically greedy local search is also proposed to improve the search efficiency for the WTA problem. From our simulations shown later, it can be seen that local search mechanisms indeed can improve the search efficiency of GAs. Several examples are used in our study, and the results have shown the superiority of the proposed approach over other existing algorithms.

The paper is organized as follows. In Section II, a mathematical formulation for WTA problems is introduced. Section III describes pseudo algorithms of general genetic algorithms with local search mechanisms. In Section IV, the simulated annealing algorithm is introduced. It is employed as one local search mechanism in our implementation. In Section V, the proposed algorithm is presented and discussed. Then, the results of employing the proposed algorithm to solve WTA problems are presented in Section VI. Several existing algorithms were also employed for comparison in that section. The performance showed the superiority of the proposed algorithm. Finally, Section VII concludes the paper.

II. THE WTA PROBLEM FORMULATION

How to effectively assign weapons to destroy targets is the main concern of WTA problems. To aid in choosing among various alternatives for WTA problems, a suitable criterion is required to define which assignment is better (Kewley and Embrechts, 1998; United Stated Naval Academy, 1989). In this study, we assumed that there are N targets and N weapons on the battlefield. The following assumptions are made for our WTA problem. The first assumption is that all weapons must be assigned to targets and all targets must be assigned. This assumption seems infeasible. Nevertheless, it is still possible to fulfill the assumption by creating some dummy entities. When there are more targets than weapons, we can create dummy weapons, which do not have any destroying capability for targets and thus, will be assigned to less threatening targets. On the other hand, when weapons outnumber targets, we can also create dummy targets. They can hardly generate damage to our assets and we will assign less useful weapons for them. Secondly, K_{ij}, the individual probability of killing of assigning the j-th weapon to the i-th target, is known for all i and j. This probability defines the effectiveness of the j-th weapon to destroy the i-th target. The overall probability of killing (PK) value for the i-th target to damage the asset can be computed as:

$$PK(i) = \prod_{j=1}^{N} (1 - K_{ij})^{X_{ij}} \tag{1}$$

where X_{ij} is a Boolean value indicating whether the j-th weapon is assigned to the i-th target. X_{ij}=1 indicates that the j-th weapon is assigned to the i-th target. The WTA problem is to minimize the following cost function:

$$C(\pi) = \sum_{i=1}^{N} EDV(i) \times PK(i) \tag{2}$$

subject to the assumption that all weapons must be assigned to targets and all targets must be assigned; that is,

$$\sum_{j=1}^{N} X_{ij} = 1, \quad i=1, 2, ..., N$$

$$\sum_{i=1}^{N} X_{ij} = 1, \quad j=1, 2, ..., N \tag{3}$$

Here $EDV(i)$ is the expected damage value of the i-th target to the asset and π is a feasible assignment list and $\pi(j)=i$ indicates weapon j is assigned to target i.

We have also studied various formulations for WTA problems. Those simulations showed that the above formulation could have good performance in all aspects (Lee *et al*., 2000). This definition of WTA problems is similar to that for the quadratic assignment problem (QAP) (Nissen, 1994; Smith, 1995). They have the same constraints but with different forms of cost functions. In fact, due to the difference in cost functions, those approaches used for QAP must be redesigned to become useful for WTA problems.

III. GENETIC ALGORITHMS WITH LOCAL SEARCH

The WTA problem is difficult to solve directly while the numbers of targets and weapons are large. Various methods such as dynamic programming, separable convex objective functions, graph theory, etc., have been employed to find the best solutions for this kind of problem (Ibarraki and Katoh, 1988;

Bjorndal *et al*., 1995). Recently, GAs have often been employed to solve this type of problem. Fundamentally, GAs or more generally, evolutionary algorithms (Back *et al*., 1997), are search algorithms, which adopt the mechanism of natural selection to search for the best solution from candidates. GAs can handle almost any kind of objective function with linear or nonlinear constraints without any auxiliary mathematical operations, such as derivatives or matrix inverses for the considered objective function. GAs have also been employed in various applications and shown that they are suitable to solve optimization problems (Gen and Cheng, 1997; Goldberg, 1989; Michalewicz, 1994). However, if genetic operations are not carefully designed, the rate of convergence may be very low. Recently, genetic algorithms with local search have also been considered as good alternatives for solving optimization problems. The general structure of the GA with local search is shown in the following where $P(t)$ and $C(t)$ are parents and offspring in generation t (Merz and Freisleben, 2000).

Procedure: GA with local search
Begin
$t \leftarrow 0$;
Initialize $P(t)$;
While (not matched for the termination conditions) do
Recombine $P(t)$ to generate $c_1(t)$;
Apply local search on $c_1(t)$ to yield $c_2(t)$;
Apply mutation on $c_2(t)$ to yield $c_3(t)$;
Apply local search on $c_3(t)$ to yield $c_4(t)$;
$C(t)=\{c_1(t), c_2(t), c_3(t), c_4(t)\}$;
Select $P(t+1)$ from $P(t)$ and $C(t)$ based on the fitness;
$t \leftarrow t+1$;
End
End

It is noted that the GA with local search becomes a general GA if the local search is omitted. In this study, the chromosome uses N numerical genes to represent a set of weapons to targets assignment with an integer number for one weapon to target assignment. Take N=4 as an example. A chromosome can be (2 4 3 1), which represents a feasible solution where target 2 is assigned to weapon 1, target 4 is assigned to weapon 2 and so on. In this algorithm and the following algorithms, the traditional roulette wheel approach (Davis, 1991) is employed in the selection of chromosomes for recombination from the parent population. For such a representation, the partially mapped crossover (PMX) (Goldberg and Lingle, 1985) and the inversion mutation (Gen and Cheng, 1997) operators are traditionally implemented. The idea of the PMX operation is to generate the mapping relations so that the offspring can be repaired accordingly and become feasible. PMX has been showed effective in many applications, and its algorithm is stated as:

Step 1: Select substrings from parents at random.
Step 2: Exchange substrings between two parents to produce proto-offspring.
Step 3: Determine the mapping relationship from the exchanged substrings.
Step 4: Repair proto-offspring with the mapping relationship.

To see the procedure, an example is illustrated. Consider two chromosomes:

A= 1 2 3 4 5 6 7 8 9
B= 4 5 6 9 1 2 7 3 8

First, two positions, e.g., the 3rd and the 6th positions, are selected at random to define a substring in chromosomes and the defined substrings in those two chromosomes are then exchanged to generate the proto-offspring as :

A= 1 2 <u>6 9 1 2</u> 7 8 9
B= 4 5 <u>3 4 5 6</u> 7 3 8

Then the mapping relationship between those two substrings can be established as:

(A's genes) 2↔6↔3 (B's genes)
9↔4
1↔5

Notice that the mapping relations 6↔3 and 2↔6 have been merged into 2↔6↔3. Finally, the proto-offspring are repaired according to the above mapping lists. The resultant feasible offspring are:

A'= 5 3 6 9 1 2 7 8 4
B'= 9 1 3 4 5 6 7 2 8

The algorithm of the inversion mutation operation is stated as:

Step 1: Select two positions within a chromosome at random.
Step 2: Invert the substring between these two positions.

Consider a chromosome:

A=1 2 <u>3</u> 4 5 <u>6</u> 7 8 9.

In a mutation process, the 3rd and the 6th positions are randomly selected. If the mutation operation is

performed, then the offspring becomes:

A'=1 2 6 5 4 3 7 8 9.

The general idea of the local search can be illustrated by the following pseudo code (Merz and Freisleben, 2000):

Procedure: General Local search
Begin
while (local search has not been stopped)
do Generate a neighborhood solution π'
If $C(\pi')<C(\pi)$ **then** $\pi=\pi'$
End
End

General local search starts from the found feasible solution and repeatedly tries to improve the current assignment by local changes. If a better assignment is found, it takes the place of the current assignment. Then, the next search starts from this new assignment again and these steps are repeated until a criterion is satisfied. The used criterion usually is to perform M times local changes, where M ($<N$) is randomly generated. The neighborhood solution is defined as a chromosome obtained by randomly swapping two positions of the current chromosome.

From the above algorithm, it can be seen that local search is performed for the chromosomes obtained by recombination and by mutation, respectively. The offspring set $C(t)$ also includes those original chromosomes before local search to retain the genetic information obtained in the evolutionary process. Such a procedure has been suggested in (Reeves, 1994; Merz and Freisleben, 2000) and has been claimed to have better performance with the inclusion of local search. In our implementation, we also employed this procedure for simulations.

Finally, selection is performed after the offspring have been generated and evaluated. A set of chromosomes as $P(t+1)$ is selected from the pool of parents and offspring to reduce the population to its original size. The used selection strategy is referred to as $(u+\lambda)$-ES (evolution strategy) survival (Merz and Freisleben, 2000), where u is the population size and λ is the number of offspring created. The process simply deletes redundant chromosomes and then retains the best u chromosomes in $P(t+1)$.

IV. SIMULATED ANNEALING AS LOCAL SEARCH

Simulated Annealing (SA) algorithms are based on the analogy to the annealing process of solids, which is the process of heating a solid to a high temperature and then cooling it down gradually. With the capability of escaping from local optimums by incorporating a probability function in accepting or rejecting new solutions, SA is also widely used for solving combinatorial optimization problems (Van Laarhoven and Arts, 1992). In our approach, SA is employed as local search to take advantage of search strategies in which worse neighborhood solution may possibly be accepted in searching for optimal solutions (Lin *et al*., 1993; Van Laarhoven and Arts, 1992). In other words, in addition to better-fitness neighbors always being accepted, worse neighbors may also be accepted according to a probability that is gradually decreased in the cooling process.

The basic idea of SA is that if a modified solution is found to have a better fitness value than its ancestor does, then the modified solution is retained and the previous solution is discarded. If the modified solution is found to have a lower fitness value, the modified solution may be still retained with a probability related to the current temperature. As the process continues and the temperature decreases, it becomes less likely that unsatisfactory solutions are accepted. Under this process, it is possible for the SA algorithm to move out of local minima. The pseudo code of the SA algorithm is described as follows (Metropolis *et al*., 1953).

Procedure: SA algorithm
Begin
Define an initial temperature T_1 and a coefficient γ ($0<\gamma<1$);
$\lambda \leftarrow 1$;
While (SA has not been frozen) **do**
$\sigma \leftarrow 0$; $\phi \leftarrow 0$;
While (The state does not approach equilibrium sufficiently closely) **do**
Generate a new solution from the current solution;
ΔF= fitness of the current solution - fitness of the new solution;
$P_r=\exp(-\Delta F/T_\lambda)$;
If $P_r \geq$ random[0,1] **then**
Accept the new solution;
current solution $\leftarrow$ new solution;
$\phi \leftarrow \phi+1$;
End
$\sigma \leftarrow \sigma+1$;
End
Update the maximum and minimum fitness;
$T_{\lambda+1} \leftarrow T_\lambda * \gamma$;
$\lambda \leftarrow \lambda+1$
End
End

The initial temperature can be set as (Jiao and Wang, 2000):

$$T_1=\ln(F^{elitist}/\lambda+1) \quad (4)$$

where $F^{elitist}$ is the elitist fitness in the beginning of the search. The way of generating new solutions is to invert two randomly selected positions in the current solution. In this algorithm, the newly generated solution is regarded as the next solution only when $\exp(-\Delta F/T_\lambda) \geq \text{random}[0,1]$, where random[0,1] is a random value generated from a uniform distribution in the interval [0,1]. It is easy to see that when the generated solution is better than the current solution, ΔF is negative and $\exp(-\Delta F/T_\lambda)$ is greater than 1. Thus, the solution is always updated. When the new solution is not better than the current solution, the solution may still take the place of its ancestor in a random manner. The process is repeated until the state approaches equilibrium sufficiently close. In our implementation, the following simple equilibrium state, which was also used in (Jang *et al*., 1997; Metropolis *et al*., 1953) is used:

$$(\sigma \geq \Gamma) \text{or } (\phi \geq \Phi) \quad (5)$$

where σ is the number of new solutions generated, ϕ is the number of new solutions accepted, Γ is the maximum number of generations, and Φ is the maximum number of acceptance. In our implementation, $\Gamma=1.5N$ and $\Phi=N$. This algorithm is repeated until it enters a frozen situation, which is:

$$(F^{\max}-F^{\min})/F^{\max} \leq \varepsilon \text{ or } T_\lambda \leq \varepsilon_1 \quad (6)$$

where $F^{\max}$ and $F^{\min}$ are the maximum and minimum fitness, respectively, ε and ε_1 are pre-specified constants and $\varepsilon=0.001$ and $\varepsilon_1=0.005$ in our implementation.

It is easy to verify that the used neighbors generating approach of SA is similar to that of general local search. It is natural to apply such an SA as the local search mechanism in the algorithm presented in the above section. In fact, from our simulation, it can be found that by using SA as the local search mechanism we can have better search efficiency than that of the algorithm presented in the above section.

V. THE PROPOSED ALGORITHM

In the above algorithms, no matter with or without general local search or SA, no domain specific knowledge was used in the algorithms. In this paper, we propose to use domain knowledge to improve the search efficiency. Two domain specific mechanisms are proposed in this paper. The first one is regarding the crossover operator. It is adopted from CX, which is proposed in (Merz and Freisleben, 2000). In CX crossover operators, the genes shared by both parents are all preserved and the remaining genes are randomly swapped to generate offspring. The proposed crossover operator also adopts a similar concept, except it preserves only those genes which are possibly good genes. This operator is called the elite preserving crossover (EX) operator. The other domain specific mechanism is the local search mechanism. In the proposed algorithm, instead of using a random mechanism to find a neighbor, a greedy reformation scheme is employed to find a neighbor with the possibly highest fitness around the current solution. We shall describe them in detail in the following.

The proposed EX operator is to preserve good genes and then to randomly swap other genes to generate offspring. Here, a gene is good if it is an assignment of the j-th weapon to the target with the highest $K_{ij}*EDV(i)$ among all i. The EX operator is described as follows:

Step 1: Find the genes with the same values (targets) in both parents.
Step 2: Inherit good genes from both parents.
Step 3: Randomly select two genes that are not inherited from parents.
Step 4: Exchange the selected genes in both parents to generate offspring.
Step 5: Repair the chromosome to be feasible.
Step 5: Go to *Step 1* until a stop criterion has been satisfied.
Step 6: Return the solution.

An example is illustrated to show the procedure. Consider two chromosomes:

A= 1 5 2 7 8 6 3 4 9
B= 1 4 2 9 8 5 3 6 7

It can be found that the 1st, the 3rd, the 5th, and the 7th genes have the same values in both parents. The values of $K_{ij}*EDV(i)$ among these weapon-target pairs of the 1st, the 3rd, the 5th, and the 7th weapons (j=1, 3, 5, and 7) to all targets (i=1 to N) are evaluated. Assume that these values of $K_{11}*EDV(1)$, $K_{23}*EDV(2)$, and $K_{85}*EDV(8)$ are the highest values among these weapon-target pairs of genes, respectively. Then, they are called good genes and inherited from parents. Thereafter, two positions, e.g., the 4th and the 7th positions, are selected at random in those two chromosomes and then genes are exchanged to generate offspring as:

A'= 1 5 2 3 8 6 9 4 9
B'= 1 4 2 3 8 5 7 6 7

Since there are duplicate values in the new

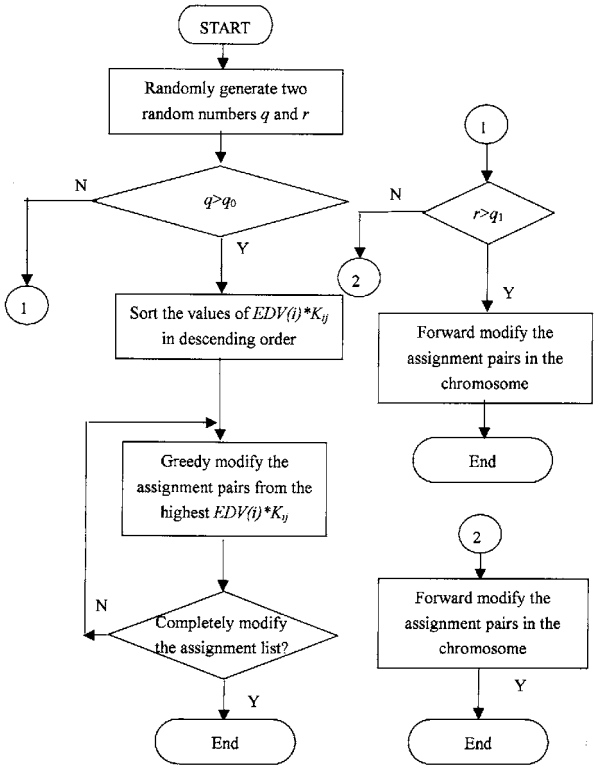

Fig. 1 The flow diagram of greedy reformation

chromosome, the chromosome must be repaired to be a feasible solution. The repairing process similar to PMX, introduced in Section III, is used here. After the repairing process, the offspring become:

A'= 1 5 2 3 8 6 9 4 7
B'= 1 4 2 3 8 5 7 6 9

These exchange steps are repeated until a stop criterion is satisfied. The stop criterion used in our implementation is to perform M times of changes where M <N and is randomly generated. Note that the crossover operation will not be performed if the number of genes that can be exchanged is less than two.

In our implementation, the greedy reformation is performed after genetic operators. The flow diagram of greedy reformation is shown in Fig. 1. In Fig. 1, two numbers q and r are randomly generated first. If $q>q_0$, a pre-defined value, then the values of K_{ij}*$EDV(i)$ are sorted in a descending order. The weapon-target pairs are modified from the highest K_{ij}*$EDV(i)$ until the assignment list is completely modified. On the other hand, another random number r is used to decide to modify forward or backward the assigned target-weapon pairs in the chromosome. In this process, a randomly generated integer (i) between 1 and N is chosen as the starting gene in the chromosome. The weapon-target pairs are forward modified from the starting gene (weapon) to the N-th gene or backward modified to the first gene. The assigned target-weapon pairs are forward modified in the chromosome if $r>q_1$ is satisfied, where q_1 is another pre-defined value. In forward modification, the weapon-target pairs from the first up to the (i-1)-th genes are fixed and the i-th gene's assignment is repeatedly replaced by the unassigned target with the available highest K_{ij}*$EDV(i)$ up to the N-th weapon. In backward modification, the assignments from the (i+1)-th up to the N-th weapons are fixed and the i-th gene's assignment is repeatedly replaced by the unassigned target with the available highest K_{ij}*$EDV(i)$ down to the first gene. Even though greedy algorithms may have a great possibility to be trapped into a local optimum, due to the crossover and mutation operations and the parallel search nature used in the evolutionary process, the search can escape from those local optima easily. On the other hand, the greedy reformation scheme provides that the evolutionary process in each generation has the locally best solution, which in turn, may lead to a more effective search. As a result, the proposed algorithm has a more effective search than other algorithms do. This phenomenon can be seen in later simulations.

VI. SIMULATION AND RESULTS

In the following simulations, we keep the following values as default parameters: the size of the initial population for GAs is set as the same as the maximum number of targets and weapons considered, the parameters used are the crossover probability P_c=0.8 and the mutation probability P_m=0.4, q_0=0.9, q_1=0.5, and γ=0.5 for SA. In this paper, the first case is to test the performance of the proposed crossover operator (EX). The used scenario consists of randomized data for 10 targets and weapons. All algorithms use the same initial population, which are randomly generated. In this study, the effects for the crossover operators, PMX, CX, and EX are studied. The maximum number of generations is set as max_gen=1000, and experiments were run on PCs with Pentium 1GHz processor. For all three cases, the simulations were conducted for 10 trials and the results averaged and reported in Table 1. From Table 1, it can be seen that the EX operator has the best performance among the three operators in both general GA, and GA with greedy reformation. Since the EX operator shows better performance, we used it as the crossover operator in the following simulations.

Several cases are considered to compare the performances of existing algorithms. These algorithms include general GA, GA with general local search, GA with SA as local search, and the proposed

Table 1 The simulation results for N=10. Results are averaged over 10 trials

Algorithm	Operator	Best fitness	Percentage of convergence	Converged generation	Converged CPU time (sec)
General GA	PMX	76.9127	20%	N/A	N/A
	CX	68.3652	50%	N/A	N/A
	EX	**65.7672**	60 %	N/A	N/A
GA with greedy reformation	PMX	64.5367	70%	N/A	N/A
	CX	**58.4774**	100%	393	18.08
	EX	**58.4774**	100%	**283**	**14.15**

N/A-Not Available

Table 2 Compare the best fitness of randomized scenarios obtained by various search algorithms. Results are averaged over 10 trials

Algorithms	N=50	N=80	N=100	N=120
General GA	212.0035	378.3746	424.7454	519.1735
GA with general local search	159.6731	348.4352	412.778	495.564
GA with SA as local search	150.9515	347.1698	409.6243	452.2026
The proposed algorithm	**133.3829**	**252.8929**	**321.2934**	**427.7887**

Table 3 The simulation results for randomized data of N=120

Algorithm	Best fitness	CPU time (minute)
GA with general local search	183.6592	198.2
GA with SA as local search	183.6592	190.7
The proposed algorithm	183.6592	**169.5**

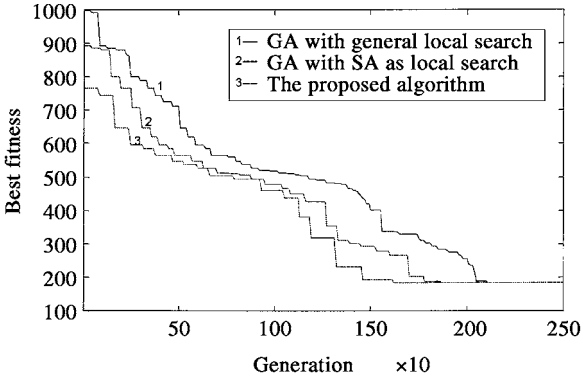

Fig. 2 The fitness curves for N=120

algorithm. Since those algorithms have different computational overhead, it is not easy to stop their search in a fair basis from the algorithm itself. In our comparison, we simply stopped these algorithms after a fixed running time to compare their search efficiencies. Experiments were also run on PCs with Pentium 1GHz processor, and were stopped after two hours of running. The results are listed in Table 2. From Table 2, it is evident that the proposed algorithm provides the best results among all used algorithms.

Furthermore, we want to test convergence performance for algorithms with local search approaches. This scenario consists of randomized data for 120 weapons and targets. In our study, we stopped those algorithms when the best fitness of all algorithms converged to a solution. The results are listed in Table 3, and the fitness curves of algorithms with various local search approaches are shown in Fig. 2. It should be noted that the horizontal axis in Fig. 2 is the number of generations. From the figure, it is clearly evident that our approach can always achieve the lowest values, as expected. For those algorithms, it takes about three hours to converge to the best solution for N=120. This number seems feasible. From Table 3, it can be concluded that the proposed algorithm even though with greedy algorithms, can more quickly find the global optimum than other algorithms do.

VII. CONCLUSIONS

In this paper, we presented a novel genetic algorithm by including domain specific knowledge into the crossover operator and the local search mechanism for solving weapon-target assignment (WTA) problems. From simulations, it can be found that the proposed algorithm can indeed provide the best search efficiency among existing genetic algorithms. Even though the proposed algorithm uses greedy algorithms in the design of search mechanisms, it still can quickly find the global optimum in the search.

ACKNOWLEDGEMENTS

This work was supported in part by the Chung-Shan Institute of Science and Technology of Taiwan under the grant XU89A59P and in part by the National Science Council of Taiwan, R.O.C. under the grant NSC-89-2218-E-011-002.

REFERENCES

1. Back, T., Hammel, U., and Schwefel, H. P., 1997, "Evolutionary Computation: Comments on the History and Current State," *IEEE Trans. On Evolutionary Computation*, Vol. 1, No, 1, pp. 1-17.
2. Bjorndal, A. M. H., Caprara, A., Cowling, P. I., Croce, D., Lourenco, H., Malucelli, F., Orman, A. J., Pisinger, D., Rego, C., and Salazar, J. J., 1995, "Some Thoughts on Combinatorial Optimization," *European Journal of Operational Research*, pp. 253-270.
3. Davis, L., 1991, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold: New York, USA.
4. Gen, M., and Cheng, R., 1997, *Genetic Algorithms and Engineering Design*, John Wiley & Sons, Inc.
5. Goldberg, D. A., 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Adision-Wesley: Reading, MA., USA.
6. Goldberg, D. A., and Lingle, R., 1985, "Alleles, Loci and the Traveling Salesman Problem," *Proceedings of the First International Conference on Genetic Algorithms*, pp. 154-159.
7. Hammer, P. L., 1965, "Some Network flow Problems Solved with Pseudo-boolean Programming," *Operation Research* 13, pp. 388-399.
8. Hammer, P. L., Hansen, P., and Simeone, B., 1984, "Roof Duality, Complementation and Persistency in Quadratic 0-1 Optimization," *Mathematical Programming* 28, pp. 121-155.
9. Ibarraki, T., and Katoh, N., 1988, *Resource allocation Problems*. The MIT Press: Cambridge, Massachusetts, USA.
10. Jang, J.-S. R., Sun, C.-T., and Mizutani, E., 1997, *Neuro-Fuzzy and Soft Computing*. Prentice-Hall Inc.
11. Jiao, L., and Wang, L., 2000, "Novel Genetic Algorithm Based on Immunity," *IEEE Transactions on Systems, Man and Cybernetics*, Part A, Vol. 30, No. 5, pp. 552 -561.
12. Kewley, R. H., and Embrechts, M. J., 1998, "Fuzzy-Genetic Decision Optimization for Positioning of Military Combat Units," *Proc. of IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 4, pp. 3658 -3663.
13. Kolen, A., and Pesch, E., 1994, "Genetic Local Search in Combinatorial Optimization," *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, Vol. 48, pp. 273-284.
14. Lai, W. K., and Coghill, G. G., 1998, "Channel Assignment through Evolutionary Optimization," *IEEE Trans. Vehicular Technology*, Vol. 45, No. 1, pp. 91-96.
15. Lee, Z. J., Lee, C. Y., and Su, S. F., 2000, "A Fuzzy-Genetic Based Decision Aided System for Naval Weapon-target Assignment Problems," *Proc. of R.O.C. Automatic Control Conference*.
16. Lin, C.-T., and Lee C. S. G., 1995, *Neural Fuzzy Systems*, Prentice-Hall Inc.
17. Lin, F. T., Kao, C. Y., and Hsu, C. C., 1993, "Applying the Genetic Approach to Simulated Annealing in Solving Some NP-hard Problems," *IEEE Trans. On Systems, Man and Cybernetics*, Vol. 23, No. 6, pp. 1752-1767.
18. Lloyd, S. P., and Witsenhausen, H. S., 1986, "Weapon Allocation is NP-Complete," *IEEE Summer Simulation Conference*, Reno, Nevada.
19. Metropolis, N., Rosenbluth, A., Rosenbluth, M., and Teller, A., 1953, "Equation of State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, Vol. 21, pp. 1087-1092.
20. Merz, P., and Freisleben, B., 2000, "Fitness Landscape Analysis and Memetic Algorithms for Quadratic Assignment Problem," *IEEE Trans. on Evolutionary Computation*, Vol. 4, No. 4, pp. 337-352.
21. Michalewicz, Z., 1994, *Genetic Algorithms + Data Structure = Evolution Programs*, Springer-Verlag: Berlin.
22. Ngo, C. Y., and Li V. O. K., 1998, "Fixed Channel Assignment in Cellular Radio Networks Using a Modified Genetic Algorithm," *IEEE Trans. Vehicular Technology*, Vol. 47, No. 1, pp. 163-172.
23. Nissen, V., 1994, "Solving the Quadratic Assignment Problem with Clues from Nature," *IEEE Transactions on Neural Networks*, Vol. 5, pp. 66 -72.
24. Pepyne, D. L., Looze, D. P., Logan, D. A.,

Sandell, N. R., and LeBlanc, R., 1997, "A Decision Aid for Theater Missile Defense," *Proceedings of IEEE International Conference on Evolutionary Computation* (ICEC '97).
25. Reeves, C. R., 1994, "Genetic Algorithms and Neighborhood Search," In *Evolutionary Computing: AISB Workshop, leeds, UK, April 1994: Selected Papers, no. 865 in Lecture Notes in Computer Science*, T. C. Forgarty *ed.*, Springer-Verlag, Berlin.
26. Sahni, S., and Gonzales, T., 1976, "P-Complete Approximation Problem," *ACM Journal*, Vol. 23, pp. 556-565.
27. Smith, K., 1995, "Solving the Generalized Quadratic Assignment Problem Using a Self-Organizing Process," *Proc. of IEEE International Conference on Neural Networks*, Vol. 4, pp. 1876-1879.
28. Sun, Y., and Wang, Z., 1994, "The Genetic Algorithm for 0-1 Programming with Linear Constraints," *Proceedings of the First IEEE Conference on Evolutionary Computation*.
29. Tate, D. M., and Smith, A. E., 1995, "A Genetic Approach to the Quadratic Assignment Problem," *Computers Ops. Res*. Vol. 22, No. 1, pp. 73- 83.
30. United Stated Naval Academy, 1989, *Naval Operations Analysis*, 2nd ed., Naval Institute Press, Annapolis, Maryland.
31. Van Laarhoven, P. J. M., and Arts, E. H. L., 1992, *Simulated Annealing: Theory and Applications*, Kluwer Academic Publishers.
32. William, A. M., and Fred, L. P., 1990, *A Suite of Weapon Assignment Algorithms for a SDI Mid-Course Battle Manager*, AT&T Bell Laboratories.

Manuscript Received: Sep. 12, 2001
Revision Received: Oct. 29, 2001
and Accepted: Dec. 24, 2001

使用知識之基因演算法於武器對目標指派問題

李仁鐘　蘇順豐　李秋緣

國立台灣科技大學電機工程學系

摘　要

本文提出使用知識之基因演算法於武器對目標指派問題。武器對目標指派問題乃是NP-Complete類型問題，其目的乃是指派我方所有武器攻擊敵方目標並降低我方的損失。本文提出新的基因配對方法及貪婪式基因演算法則，以提高武器對目標指派問題的求解收斂速度，文中亦將使用知識之基因演算法的模擬結果與其他演算法則相比較，其數據顯示使用知識之基因演算法優於其他演算法則。

關鍵詞 ： 基因演算法則，武器對目標指派問題，貪婪式方法。