LARGE TARGET-WEAPON ASSIGNMENT PROBLEMS WITH EVOLUTIONARY PROGRAMMING AS APPLIED TO THE ROYAL THAI ARMED FORCES GROUND-TO-GROUND ROCKET SIMULATION SYSTEM

Settapong Malisuwan[1,2], Teeranan Nandhakwang[2] and Tawiwat Veeraklaew[1]
[1]*Laboratory of Mechanical and Aeronautic Engineerings*
Defence Technology Institute, THAILAND
[2]*Dynamic Spectrum Management Project*
National Broadcasting and Telecommunications Commission, THAILAND
settapomg_m@hotmail.com
teeranan@rtarf.mi.th
Tawiwat.v@dti.or.th

ABSTRACT

An algorithm based on evolutionary programming (EP) is developed and presented for large numbers of target-weapon assignment. An optimal assignment scheduling in one, which allocates target to weapon such that the total expected of target surviving the defense, is minimized. The proposed method improves EP with reordered mutation operator to handle a large-scale assignment problem. The main advantage of this approach is that the computation time can be controlled via tradeoff performance between the computation time and target surviving value.

Keywords: Assignment problem, Large-scale problems, Evolutionary computation, Evolutionary programming, Reordered mutation, Target-weapon assignment problem

1. INTRODUCTION

Considering a battlefield scenario, Figure 1.1 illustrates terrain of the area of operation (AO). Figure 1.2, which maps the 3D terrain into 2D, illustrates location of threats that need to be attacked in AO. Each threat can move to different locations at any time, and thus places different demands weapon allocations. Therefore, to maximize the probability of threat or target surviving value, it is desirable to assign weapons at optimum engagement opportunities.

In military operations, problems in planning and scheduling often require feasible and near to optimal solutions with limited computing resources and within very short time periods. To overcome this time dilemma, fast-execution weapon-target (WT) algorithms are needed. This paper proposes evolutionary programming (EP) based algorithm to solve large scale weapon-target assignment problems with reordered mutation operator. The main advantage of this approach is that computation time can be controlled via tradeoff performance between computation time and target surviving value.

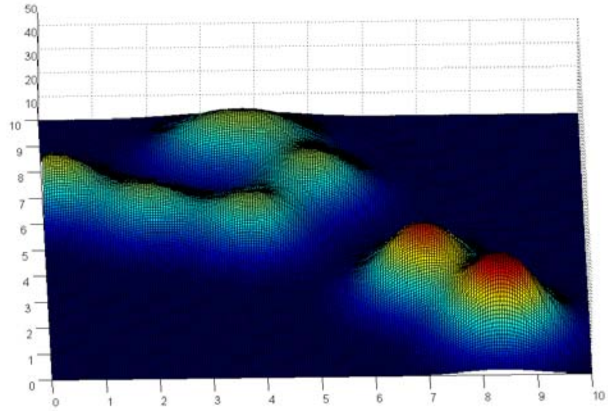

Figure 1.1 An example of terrain of the area of operation (AO).

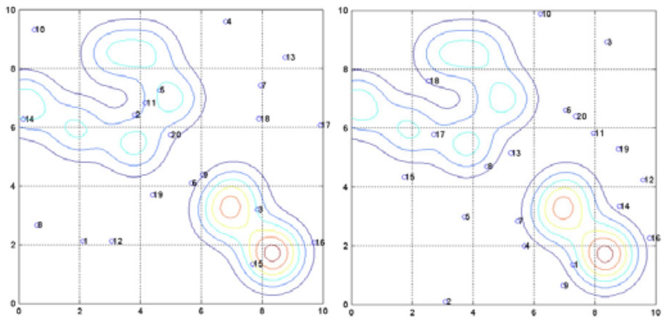

Figure 1.2 Threats location in the area of operation (AO) for 2 time intervals.

The remainder of this paper is organized as follow: Section 2 presents a brief description of WT problems in general; section 3 describes an evolutionary programming overview; section 4 explains the proposed algorithm in greater detail; section 5 provides numerical results; and section 6 summarizes the experiment.

2. The WT Problem

The general weapon-target assignment problem deals with allocation of weapons to offensive targets or threats. One objective is to minimize the surviving value of targeted assets [1, 2]. An asset is an any entity (or collection of entities) of the important military objects, e.g. tactical command posts, bridges, missiles, battleships, etc. Each asset has a value assigned to it. A threat can be anything potentially causing damage to defensive assets. Any threats may be come from different position, speed, distance or all off them. A weapon is an object to eliminate a threat, e.g. missile, artillery, close air support, which is assumed to have an inventory of shots [2]. Difference weapons may require different amounts of time to engage the same threat. The number of threats N_T, the number of weapons N_W, and the number of assets N_A may be dependent upon time t. Any weapon can kill any threat in range with a probability of kill and fire time.

The problem is to minimize the expected threat by determining the best firing schedule over time, given $N_w(t)$, weapons to defend $N_A(t)$ assets against $N_T(t)$ threat, where N_w, N_A, $N_T \in \mathrm{Z}^+$ [2]. The size of this problem may be very large.

To formulate the problem, let a_{ijk} represent the assignment of weapon i to threat j on the time period engagement interval k. The a_{ijk} are the decision variables, and have strictly nonnegative integer value: $a_{ijk} \in \mathrm{Z}^+$. Let $p_{ijk} \in [0,1]$ denote the probability that weapon i fired during the engagement interval k and kills threats j. For scheduling purposes the time between some initial firing time and a time horizon T may be divided into a number, P, of discrete temporal intervals $t_1, t_2, \ldots t_P$ of equal length t, with $T = P.t$. The objective is to find the maximization of the total expected surviving value of the asset under attack, or alternatively the minimization of the total threat [1]:

$$S(A^*) = \max_{ijk} \sum_{k=1}^{P} \sum_{j=1}^{N_T} \sum_{i=1}^{N_W} p_{ijk} a_{ijk} \qquad (2.1)$$

subject to $\sum_{k=1}^{P} \sum_{j=1}^{N_T} \sum_{i=1}^{N_W} a_{ijk} \le M$ upper bound on total number of shorts fired,

$\sum_{i=1}^{N_W} a_{ijk} \le 1$ at most one shot at each threat j in any interval

$\sum_{i=1}^{N_T} a_{ijk} \le 1$ at most one shot by each weapon in any interval,

$a_{ijk} \in \{0,1\}$ a weapon is either active or not during interval, where $S(A^*)$ represents the surviving value, corresponding to firing plan $A = \sum_{k=1}^{P} A_k$, and A^* denote the optimal firing schedule.

3. Evolutionary Programming [3, 4, 5]

Evolutionary programming (EP) is a stochastic optimization strategy, which is similar to genetic algorithms. In 1961, L. J. Fogel proposed applying the concepts of natural evolution, selection, and stochastic mutation to achieve intelligent behavior through simulated evolutions. Originally, EP was developed to evolve finite-state machines. The basic EP algorithms use normally distributed mutations to modify real-valued vectors and emphasize mutation as essential operators for searching in the search space.

EP starts its search with a set of randomly initialized population in a given bound space. Thereafter, each individual of a created population is altered through application with a mutation operator, and produce the new individual. Then, each individual is evaluated to calculate its fitness function. Frequently, the fitness function can also be the same as the objective function. An EP algorithm is formulated as follows [3]:

Algorithm 1 (EP)

$\Phi :=$ ***Problem Formulation***(weapons, targets, costs)
$t := 0$
initialize $\mathbf{P}(0) := \{a_1'(0),\ a_2'(0), \ldots, a_\mu'(0)\}$
evaluate $\mathbf{P}(0) := \{\Phi(a_1'(0)),\ \Phi(a_2'(0)), \ldots,\ \Phi(a_\mu'(0))\}$
while not terminate ***do***
{
  mutate $\mathbf{P}(t) := m\Theta_m(\mathbf{P}(t))$
  evaluate $\mathbf{P}'(t) := \{\Phi(a_1'(t)),\ \Phi(a_2'(t)), \ldots,\ \Phi(a_\mu'(t))\}$
  select $\mathbf{P}(t) := \max\{s\Theta_s(\mathbf{P}'(t) \cup \mathbf{Q})\}$
  $t := t + 1$
}
return $\max\{\mathbf{P}(t)\}$

where a' is an individual member in the population, $\mu \geq 1$ is the size of the parent population, $P(0) := \{a'_1(t), a'_2(t), ..., a'_\mu(t)\}$ is the population at time t, $\Phi : I \to \Re$ is the fitness mapping, $m\Theta_m$ is the mutation operator with parameters Θ_m, $s\Theta_s$ is the selection operator, and $Q \in \{0, P(t)\}$ is a set of individuals additionally accounted for the selection step.

4. Proposed Method

We present an Evolutionary Programming approach to solve target-weapon assignment problems that uses reordered technique to mutate solutions. The use of reordered mutation and is useful for problems that require permutations of the integers and for which traditional mutation presents feasibility problems.

The objective is to find a maximum total expected surviving value or a minimum total threat, which is the summation of the maximum surviving to weapon i to threat j to on the time period engagement interval k. A number of weapon and threat is randomly generated and established in term of Equation (2.1).

4.1. Representation

Although traditional EAs use a bitstring representation, EAs are not restricted to bitstring [3, 4, 5, 6]. Real-valued vectors are also used as a representation. In permutation problems such as the assignment problem, the feasible solutions are usually an order-based number representing a permutation. For example, $\{4,1,3,2,3,4,2,1\}$ is a possible solution, which is represented weapon 4 is assigned to threat 1 at engagement period 1, weapon 1 is assigned to threat 2 at engagement period 1, weapon 3 is assigned to threat 3 at engagement period 1, weapon 2 is assigned to threat 4 at engagement period 1, weapon 3 is assigned to threat 1 at engagement period 2, weapon 4 is assigned to threat 2 at engagement period 2, weapon 2 is assigned to threat 3 at engagement period 2, and weapon 1 is assigned to threat 4 at engagement period $2.$

4.2. Initial Population

The initialization process is created using a population of individuals, where each individual is set of an ordered number from $1,...,n.$ All individuals are generated randomly with Uniformly Distribution.

4.3. Selection Operator

A selection is a major operator used in evolutionary algorithms which emphasizes the better solution in a population. Indeed, after it selects good solutions to be parents, the remainder solutions are deleted form the population. Afterward, the best individuals from the combination pool of parents will create offspring by mutation operator.

4.3. Objective and Fitness Function

A modification of the Equation (2.1) is chosen as the fitness function. The fitness function Ξ is expressed as

$$\Xi(a_m) = \sum_{k}^{P}\sum_{i}^{N_W} x_{k,i,per(i)} \tag{4.1}$$

where N_W is the number of weapons, P is a total time interval, a_m is a individual mth in a population, x_{kij} is a survival value of weapon i assigned to threat j at engagement period k, k and i is a set of ordered integer form $[1, n]$, and $per(i)$ is a set of permutation i. For example, let payoff array represent in Table 4.1 as follow:

Table 4.1 Payoff array for survival values of each weapon engages to each threat

Survival Value	Engagement Period 1				Engagement Period 2			
	Threat 1	Threat 2	Threat 3	Threat 4	Threat 1	Threat 2	Threat 3	Threat 4
Weapon 1	0.35	0.48	0.72	0.66	0.85	0.19	0.56	0.15
Weapon 2	0.82	0.25	0.64	0.44	0.66	0.56	0.42	0.53
Weapon 3	0.35	0.83	0.92	0.16	0.77	0.90	0.35	0.22
Weapon 4	0.88	0.52	0.77	0.34	0.35	0.75	0.84	0.75

For example, individual a_1 has parameters $k = 1$, $i = \{1,2,3,4\}$, $j = \{4,2,1,3\}$, and $k = 2$, $i = \{1,2,3,4\}$, $j = \{2,1,4,3\}$. Therefore, $\Xi(a_1)$ can be evaluated by $x_{114} + x_{122} + x_{131} + x_{143} + x_{212} + x_{221} + x_{234} + x_{243}$ or 0.66 + 0.25 + 0.35 + 0.77 + 0.19 + 0.66 + 0.22 + 0.84, which equals to 3.94.

4.3. Mutation Operator

The assignment problems are naturally represented as permutations. The major problem of permutations is that it cannot be processed using the mutation operator in the traditional way. Reordered mutation has been applied to mutation operator [7, 8]. The primary difference of this operator and traditional mutation operator is the use of permutation technique on the selected values in represent solutions instead of change $0 \to 1$ or $1 \to 0$. In this case, a feasible solution and the reordered mutation operator, denote Θ, as follows:

$$\Theta_{\{8,10,2\}}\{5,8,2,3,12,6,11,1,4,7,9,10\} = \{5,1,2,3,12,6,11,7,4,8,9,10\}.$$

Where a feasible solution is **{5,8,2,3,12,6,11,1,4,7,9,10}** with rate 0.25% (3-value need to be reordered), a selected index value is **{2,8,10}**, and reordered index is **{8,10,2}**, therefore, an offspring from this feasible solution is **{5,1,2,3,12,6,7,1,4,8,9,10}.** In reordered mutation, we generate index and reordered index with uniformly distribution.

5. Numerical Results

In the experiment, the payoff array is uniformly generated in a unit cost from [0, 1). The initial population is created by generating μ feasible solutions. This experiment uses 10 – 40 population sizes. A mutation rate of 0.20% is used in this experiment. The problem size is 10000^{10} variables. This algorithm was implemented in MATLAB and tested with a 1 GHz AMD Athlon processor and 512 MB memory. The algorithm terminates when 100 iterations have been executed, or when 20 consecutive iterations have failed to improve the best-known solution.

The experiment compares the proposed evolutionary programming with the different population size of the evolution in MATLAB. Results form the experiment using 100^{10} decision variables are shown in Figure 5.1. – 5.2.

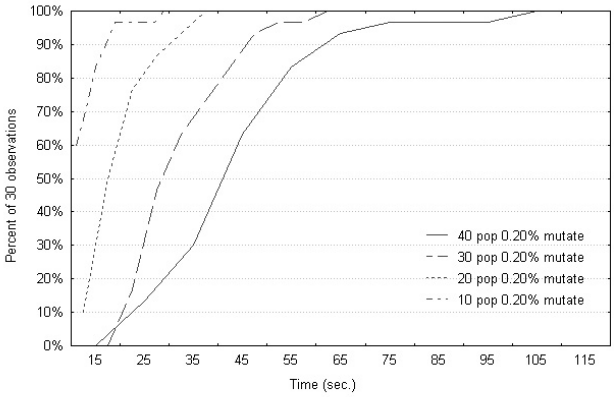

Figure 5.1 Cumulative of computation time with 10000^{10} decision variables in 30 observations using different population sizes.

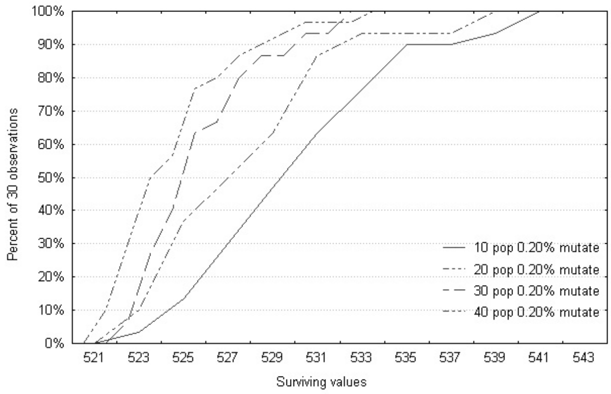

Figure 5.2 Cumulative of survival value with 10000^{10} decision variables in 30 observations using different population sizes.

Table 5.1 shows computation time statistics for this experiment. The average of computation time of EP with 40 as the population size using 0.20% mutation rate is 48.31467 seconds when the problem size is 10000^{10} decision variables. Parameters can be altered to reduce computation time, such as using 20 as the population size with mutation rate 0.20% will reduce the computation time to 21.98577 second. Moreover, computation can be reduced to 11.80433 second when a 10 population size with mutation rate 0.20% is used.

Table 5.2 shows total cost statistics. Average of total surviving value of EP with 10-population size using 0.20% mutation rate is 524.8358. While increase population size, the proposed EP can find the lager average total surviving, e.g. the average total cost of EP with 40 population size using mutate rate 0.20% is 530.7279, which is better than the average total cost of EP with 10 population size using mutate rate 0.20%, as shown in Table 5.2.

Table 5.1 Computation time statistics using 10000^{10} decision variables

Computation Time (sec.)	Mean	Confid. -95.0%	Confid. +95.0%	Minimum	Maximum	Variance	Std.Dev.
EP 10 pop 0.20% mut.	11.80433	9.95462	13.65405	6.59900	28.3310	24.5385	4.95363
EP 20 pop 0.20% mut.	21.98577	19.44620	24.52533	13.25900	39.7370	46.2548	6.80109
EP 30 pop 0.20% mut.	33.61610	29.76347	37.46873	20.30900	62.0190	106.4516	10.31754
EP 40 pop 0.20% mut.	48.31467	42.31826	54.31108	26.30800	103.2790	257.8812	16.05868

Table 5.2 Total survival value statistics using 10000^{10} decision variables

Surviving Values	Mean	Confid. -95.0%	Confid. +95.0%	Minimum	Maximum	Variance	Std.Dev.
EP 10 pop 0.20% mut.	524.8328	523.7413	525.9244	521.2625	533.6613	8.54549	2.923266
EP 20 pop 0.20% mut.	525.9841	524.9654	527.0027	522.7080	532.4460	7.44203	2.728008
EP 30 pop 0.20% mut.	528.5580	527.0562	530.0599	522.9000	538.7854	16.17687	4.022048
EP 40 pop 0.20% mut.	530.7279	529.0559	532.4000	523.5912	540.7522	20.05048	4.477776

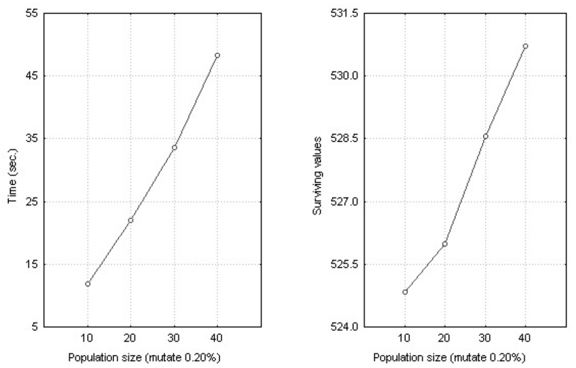

Figure 5.3 Compare and contrast computation time and survival values with 10000^{10} decision variables using mutation rate 0.20%

Figure 5.3 illustrates compares and contrasts the parameters of the proposed EP. It can be seen that the proposed EP can find the better solution; however, computation will increase. Therefore, tradeoff analysis needs to be performed to decide optimal parameters.

6. Conclusion

This paper presents a robust heuristic to solve target-weapon assignment problems. The proposed method is developed by applying evolutionary programming with a reordered mutation initial experiment. Simulation results in this research reveal that this method is very effective. In fact, the evolutionary algorithm (e.g. genetic algorithm, evolutionary strategy, and evolutionary programming) works very well in the combinatorial optimization problems when the number of decision variables is large.

This paper makes no claim that the proposed algorithm is the most efficient for large-scale assignment problems. The contribution here is that evolutionary algorithms can effectively solve large and complex problems. This paper does not compare the proposed approach with other evolutionary algorithms, especially genetic algorithm, because crossover operator will increase the computation time in which we need to reduce. Work is continuing on building an algorithm to have further validation for the obtained results.

Reference

[1] Toet, A., Waard H. (1995). The Weapon-Target Assignment Problem. CALMA Report CALMA.TNO.WP31.AT.95c.

[2] Wacholder, E. (1989). a neural network-based optimization algorithm for the static weapon-target assignment problem. INFORMS Journal on Computing (Vol.1 No. 4), pp. 232-246 .

[3] Back, T. (Editor), Fogel, D. B. (Editor), Michalewicz, Z. (Editor) (1999). Evolutionary Computation 1: Basic Algorithms and Operators: Iop Pub.

[4] Back, T. (Editor), Fogel, D. B. (Editor), Michalewicz, Z. (Editor) (1999). Evolutionary Computation 2: Advance Algorithms and Operators: Iop Pub.

[5] Michalewicz, Z. (1999). Genetic Algorithms + Data Structures = Evolution Programs 3rd: Springer-Verlag New York.

[6] Bean, J. C. (1994). Genetic Algorithms and Random Keys for Sequencing and Optimization. ORSA Journal on Computing (Vol. 6, No. 2), pp. 154-160.

[7] T. Nandhakwang, M. A. Shaikh, An Efficient Algorithm for Large-Scale Assignment Problems, Hawaii International Conference on Statistics and Related Fields, Honolulu, Hawaii; 2003.

[8] T. Nandhakwang, M. A. Shaikh, "Reordered Mutation Operator in Evolutionary Programming: An Assignment Problem Application", 4th International Conference on Dynamic Systems and Applications, Atlanta, Georgia; 2003.

AUTHORS BIOGRAPHY

Colonel Tawiwat Veeraklaew received the Ph.D. degree in mechanical engineering from University of Delaware, Newark, DE, USA in 2000. He is a Platform and Material Senior Researcher at Defence Technology Institute (Public Organization), Bangkok, Thailand and supervise the Platform and Material Laboratory. He has published more than 30 both in conference and journal articles. His current research interests are in the area of controlled mechanical systems, dynamic optimization and special software hardware design.

Colonel Settapong Malisuwan received the Ph.D. degree in Electrical Engineering from Florida Atlantic University, Boca Raton, FL, USA in 2000. He is a member of Spectrum Refarming Committee, National Telecommuni- Cations Commission, Thailand. He has published more than 90 both in conference and journal articles.