DYNAMIC MULTIOBJECTIVE OPTIMIZATION OF WAR RESOURCE ALLOCATION USING ADAPTIVE GENETIC ALGORITHMS

S. Palaniappan, S. Zein-Sabatto and A. Sekmen

Tennessee State University, Nashville, TN, 37209
E-mail: mzein@tnstate.edu

Abstract

Genetic Algorithms (GA) are often well suited for multiobjective optimization problems. The major objective of this research is, to optimize the war resource allocations of sorties, for a given war scenario, using Genetic Algorithms. The war is simulated using THUNDER software. THUNDER software is a stochastic, two-sided, analytical simulation of campaign-level military operations. The simulation is subject to internal unknown noises similar to real war cases. Due to these noises and discreetness in the simulation, as well as in real wars, an adaptive GA approach has been applied to solve this multiobjective optimization problem. Transforming this multiobjective optimization problem to a form suitable for direct implementation of GA was a major accomplishment of this research. A suitable fitness function was chosen after careful research and testing on the GA. Furthermore, the GA parameters were adaptively set to yield smoother and faster fitness convergence. Two fuzzy logic mechanisms were used to adapt the GA parameters. In the first mechanism, the mutation and crossover rates were changed adaptively. In the second mechanism, the fitness function coefficients are changed dynamically in each run. Testing results showed that the adaptive GA outperforms the conventional GA search in this multiobjective optimization problem and was effectively able to allocate forces for war scenarios.

Keywords: Intelligent Systems, Soft Computing, Genetic Algorithms, and Optimization.

Introduction

Planning for a war is a very challenging task often faced with difficult choices and critical decision-makings. The resource allocations for a war scenario, simulated by the THUNDER Software, are currently made by Expert Analysts. They use several analyzing techniques. This task is very time consuming and does not always yield favorable results. The necessity for systematic optimization procedure is because, the claim of obtaining "best results" is subjective, and are based on multiple" figures of merit".

THUNDER software is a very large campaign simulation model, which was built based on Monte-Carlo simulation. This software is a stochastic, two-sided, analytical simulation of military operations developed by System Simulation Solutions Inc. (S3I) for the Air Force Studies and Analyses Agency (AFSAA). This simulation was designed in order to examine issues involving the utility and effectiveness of air and ground power in a theater-level joint warfare context. This software automatically plans military moves and actions in a rule-based manner. The software is capable of supporting campaign analysis involving the integration of effects over time and space.

In most real world problems there is often a situation where multiple objectives need to be simultaneously optimized. Most of such problems end up in several alternative solutions in the search space. These solutions are called pareto-optimal solutions [1]. One of the potential ways to finding solutions to multiobjective optimization problems is the use of genetic algorithms. GAs process set of solutions in parallel exploiting similarities among solutions.

Genetic Algorithms have been the subjects of considerable interest in the recent years [2-4]. The striking feature of these algorithms is that they are based on ideas from the science of biological genetics and the process of natural selection. In most of the optimization problems, it is common to have mixed (continuous and discrete) variables and discontinuities in their search space. If standard non-linear programming techniques were to be used in such cases, then they would be computationally very expensive and inefficient. Genetic algorithms are a suitable solution to such situations. They were introduced in the United States in the early 1970's by Holland [4,5].

Genetic algorithms are search algorithms that come under the range of techniques, collectively known as "evolutionary computing ". The major benefits of these algorithms is that they provide a robust search in complex spaces and are usually less expensive, as far as computation is concerned, when compared to most other optimization solutions. They are also resistant to getting trapped in local optima. This leads to a wide range of applications in large-scale optimization problems of various fields.

Some of the characteristics of the Genetic algorithms compared to normal optimization search procedures are [6]:

1. GAs search from a population of points. Thus, they are less likely to be trapped in a local optimum
2. GAs use values of the objective function, and not their derivatives
3. GAs work with coding. Thus applicable for solving discrete and integer programming problems

0-7803-6748-0/01/$10.00 ©2001 IEEE

4. GAs use randomized parent's selection and crossover from the old generation. Thus they explore new combinations to find a new generation with better fitness values.

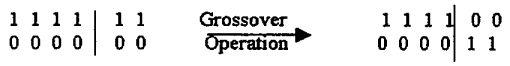
Most of the simple genetic algorithms have three main operators they are: 1) Selection, 2) Crossover and 3) Mutation.

Selection is one of the critical genetic operators. Selection operation may be represented as follows:

$$P_{select}(n) = f(n) / \sum_{k=1}^{pop} f(k) \qquad (1)$$

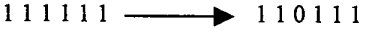
Where, n is the n^{th} string, *pop* is the population size and $f(n)$ is the fitness function. This first population must offer a wide diversity of genetic materials. The gene pool must be as large as possible so that any solution of the search space can be engendered. Generally, the initial population is generated randomly. Some of the most commonly used selection operators are roulette wheel selection, tournament selection, Ranking selection etc.

Grossover is the most important genetic operator, and may be considered as the main engine for exploration in a GA. This operator is responsible for the shuffling and recombination of building blocks. The simplest form of crossover is that, a single point is chosen on two equal length chromosomes and they are crossed at that particular point. It is possible to select two or more points for cross over, to produce more genetic mixing but sometimes while using multipoint crossover it degrades the performance of the GA. Crossover can be shown as follows.



Crossover operator generally consists of forming a new solution by taking some parameters from one solution and exchanging it with another at the very same point. Thus we get new offspring. Some crossover operators use complex geometric methods to generate the offsprings of two parents.

Mutation is another fundamental genetic manipulation operator. It involves, the random alteration of genes during the process of copying a chromosome from one generation to the next. Mutation simply involves the incorrect copying of some parameters, which make up a solution. It may be illustrated as follows.



Mutation is usually used to avoid premature convergence, which is a common problem in GAs, which use fixed length, binary codlings. When proportional selection is used, all the individual chromosomes in the population become very similar before a nearly optimal solution is reached, thus preventing any further progress. In such cases mutation is essential. Mutation acts against this, by constantly generating new chromosomes, this helps in preventing the population from getting trapped in a local maximum in the search space. However, mutation sometimes also result in loss of good individual, thus the need to prevent premature convergence has to be balanced against the loss of efficiency due to the damage of good genetic material. Thus there is a payoff between exploitation and exploration.

Problem Formulation

A multi-objective optimization problem with inequality constraints can be represented as a vector function f, which maps a set of m parameters (decision variables) to a set of n objective [2]. This can be defined as follows

$$Y = f(x) = (f_1(x), f_2(x), f_3(x), \ldots\ldots\ldots f_n(x)) \qquad (2)$$
Subject to $x = (x_1, x_2 \ldots\ldots x_m) \in X$
$y = (y_1, y_2 \ldots\ldots y_n) \in Y$

Where, x *is* a set of the decision vectors and X is the parameter space, while y is the objective vector, and Y is the objective space.

In this research, the underlying problem is that very less information is known about the THUNDER software and it is more like a black box optimization problem than just another conventional optimization problem. The problem considered here could be addressed as follows,

" To optimize war resource allocations for a given war scenario, simulated by the THUNDER software. This multi-objective optimization problem demands effective allocation of resources, such that it minimizes the losses of the friendly side and maximizes the targets killed on the enemy side."

This problem involves both maximization and minimization of nonlinear functions. In this case genetic algorithms may be the best solution, to determine the force allocations in a war simulation.

Research approach

The main purpose of this research is to determine how to effectively allocate force power for a given war using genetic algorithms. The reason why genetic algorithms is used, because they provide robust search procedures for many types of functions including those exhibiting discontinuities, multi-modality, high dimensionality, huge search spaces and noise.

The war allocations are made based on the capabilities of the threat forces, conditions of the war, and capabilities of the friendly forces, all simulated by THUNDER software. The input war allocation file is generally given by the user.

Each mission requires an apportionment. In this research, only four missions were used as inputs and 15 day war scenario was used. The missions were Offensive Counter Air (OCA), Strategic Target Interdiction (STI), Long Range Air Interdiction (INT), and Lethal Direct Air Defense Suppression (DSEAD). OCA, STI and INT are air-to-ground missions. OCA is against airbases and INT is against units moving on the network and in garrison, logistics facilities, transportation network transshipment points, checkpoints, supply convoys, and air defense complexes. STI is against strategic targets. DSEAD is a suppression of enemy air defense missions and it is against air defense sites. According to these definitions, our objectives for these scenarios are

1. Minimize the territory that blue side losses
2. Minimize the blue side aircraft lost
3. Maximize the number of red side strategic targets killed
4. Maximize the number of red side armor killed

This is a typical multiobjective optimization problem because it is desired to optimize all the four objectives simultaneously. A GA based method was used to solve this problem [7]. There are many possible ways to assign fitness values based on the scores provided by THUNDER software. In our case, several methods of determining fitness values of solutions were analyzed and the best one was chosen. This method was based on sum squared fitness assignment. In this method, individual scores are squared and their summation gives the assigned fitness values. This function produces a maximum fitness value of 16 and a minimum value of 4 and can be written as:

$$F = f_1^2 + f_2^2 + f_3^2 + f_4^2 \qquad (3)$$

Where, f_1, f_2, f_3 and f_4 are the individual scores of Blue Territory lost, Red strategic targets killed, Red armor killed, and Blue aircraft lost respectively.

The parameters for the GA were manually and carefully set, by performing a number of trial runs. This increased the efficiency and performance of the GA. The parameters best suited for the GA in our case were, *population size N= 100, crossover rate* $P_c = 0.6$ and *mutation rate* $P_m = 0.02$. At this point, the following fact was established, starting the GA with a relatively higher value for crossover and lower value for mutation rate, and then decreasing the value of the crossover and increasing the mutation rate towards the end of the run would yield better results. To further more enhance and automate this approach two fuzzy mechanisms are used to fine tune the results and improve the performance of the GA. This is discussed in the next section.

Evolutionary Fuzzy system

Fuzzy logic techniques are used for optimization problems [4,7] and are defined as the processes of finding the optimal location of the hyperspace. This section describes, how an adaptive genetic algorithm is implemented to solve a multiobjective problem such as minimizing the territory loses and maximizing enemy air loses. This is accomplished by finding the optimum distribution of aircraft fighting in a war scenario simulated by the THUNDER software. The entire solution to this problem is shown in Figure [1]. This is the system layout of the developed solution. The adaptive genetic algorithm developed in this research includes two mechanisms. Each of these mechanisms has been classified as separate sub-systems as shown in subsystem-2 and subsystem-3.

Adaptive Search with Fixed Fitness Function

This part was dealt with in the subsystem-2 of the entire system. Two of the three parameters, mutation and crossover rates, are adjusted as GA runs. Although they are potentially disruptive, they facilitate an efficient search and guide the search in new directions. Crossover facilitates exploration, while mutation facilitates exploitation of the space. The crossover and mutation rates can be varied during the run. Often starting out by running the GA with a relatively higher value for crossover and lower value for mutation rate. Then tapering off the crossover value and increasing the mutation rate toward the end of the run. In order to change these parameter adaptively, a three-inputs two-outputs fuzzy logic system (FLS) was implemented. The inputs to the FLS are three distribution properties of the fitness values namely, the best fitness value, the average and the variance of the fitness values of the population. Three ranges (LEFT-small, TR-medium, RIGHT- large) and triangular membership functions were used to fuzzify the inputs. A set of rules that describes how to adjust the mutation and crossover rates was constructed. These rules were based upon experience and literature review [8].

The fitness value of the conventional GA is represented as $F_{without\_adap}$ and the fitness value of the adaptive GA is represented as F_{Adap}. After 50 generations were evaluated for each scenario, the best (maximum) fitness function value is obtained for each case and are listed in Table1. The corresponding war allocation values for each of these fitness values are listed in Table 2. Thunder Software was run, using these allocations input values, for 15 day's war scenario. The war results are included in Table 3. The variations in the fitness values over 50 generations are plotted in the Figure 2 for illustration.

The adaptation of the GA parameters allowed to concurrently evolves non-dominated solutions and guide the search in several directions on the multi-dimensional search space. By using automatic settings of these parameters the search did not trap at a local area especially during the start up time. The interesting point to note is that, the adaptation reduced the GA run time. Moreover better fitness value was obtained in fewer generations. Most of the runs that were performed with different fuzzy settings for the adaptive GA settled in about *10-15 generations. In some*

cases it took even less than 10 generations for the GA to converge. These changes allow to explore new solution points and to optimize run time of the GAs. A comparative plot of the conventional GA and the adaptive GA was shown in Figure 2.

Adaptive Search with Dynamic Fitness Function

This part of the paper is depicted in Figure 1 as Subsystem-3. The objectives in combats are functions of time and they are dependent on defense and offense tactics of the enemy side. Hence, the dynamic objective function is needed during optimization process. The adaptive genetic algorithm with added capability of self-adjustment in the objective function enables solving dynamic multiobjective problems. In this subsystem, weights are assigned to the terms of the fitness function. The fitness function weights are changed dynamically in each run. The current fitness values of the objectives are given to the second fuzzy logic system as input and the weights representing the effectiveness of each objective are modified for the next run. It is known that different battlefield tactics ought to be employed in different war stages. These tactics can only be described linguistically. Thus, rule based fitness function was used to calculate the fitness value based on the above objectives. In this method, individual score of each objective is squared and weighted. The assigned fitness value was defined as sum of these terms:

$$F(t) = \sum_{i=1}^{4} \mu_i(t)\, w_i f_i^2 \qquad (4)$$

where μ_i and w_i are the membership function and weight characterizing objective f_i. The adaptive genetic algorithm with dynamic fitness function can be represented as $F_{Adap\_dynamic\_Fit\_fn}$. After 50 generations were evaluated for each scenario, the best (maximum) fitness function value obtained was 93.403. The war results simulated by the THUNDER software for all three algorithms, namely Conventional GA, Adaptive GA with fixed fitness function, Adaptive GA with dynamic fitness function are shown in Table 4.

It is observed that the Adaptive GA system with dynamic fitness function seems to work better on the maximizing objectives and this really could be helpful if a particular objective amongst the multiobjectives have to be emphasized or de-emphasized. It can be concluded that adaptive GA with fixed fitness function has a better performance than dynamic fitness function as it is computationally inexpensive and yields better results if the overall optimization performance is considered. However in a set of multiobjectives, if a particular objective needs to be emphasized or de-emphasized then the adaptive GA with dynamic fitness function could yield better results. Emphasizing and de-emphasizing can be implemented by just changing a few fuzzy settings.

Conclusion

Solutions for war multiobjective optimization problems were proposed and developed using GA. A search procedure using the GA was used to provide an optimal or near optimal solutions for this problem. Several different methods for fitness evaluation were tested and the best one was chosen.

To achieve optimum solutions of four war objectives, two fuzzy logic systems were employed. By judging the overall fitness of the solutions, the rule based GA produced superior performance for all of the objectives. This paper concludes that the adaptive genetic algorithms with fixed fitness function improves the convergence rates considerably and maintains smoother convergence to the best possible solutions than that of the conventional genetic algorithms and adaptive genetic algorithm with dynamic fitness function. However in a set of multiobjectives, if a particular objective needs to be emphasized or de-emphasized then the adaptive GA with dynamic fitness function could yield better results.

Acknowledgement

The authors would like to acknowledge the Boeing Company for their interest and support for this research effort.

References

[1] Brindle, A., "Genetic Algorithms for Function Optimization," Ph.D. Dissertation, University of Alberta, 1981.

[2] D.E Goldberg., "Genetic Algorithms in search, optimization, and machine learning," Addison-Wesley, 1989.

[3] Davis, L,. Handbook of Genetic Algorithms. Van Nostrand Reinhold (New York), 1991.

[4] De Jong, K. A., "Analysis of The Behavior of a Class Of Genetic Adaptive Systems," Ph.D. Dissertation, University of Michigan, Ann Arbor, 1975.

[5] J.H. Holland, "Genetic Algorithms and Classifier Systems: Foundations and Future Directions," Proc. 2nd Int. Conf. On Genetic Algorithms, pp. 82-89, 1987.

[6] J.H. Holland. Adaptation in Natural and Artificial Systems. University of Michigan Press (Ann Arbor), 1975.

[7] Yuhui Shi, Russell Eberhart &Yaboin Chen,"Implementation of Evolutionary Fuzzy Systems", IEEE publication, 1999.

[8] Z. Bingul, A.S.Sekmen, S. Palaniappan And S. Sabatto, An application of multi dimensional optimization problems using genetic algorithms, IASTED, 1999.

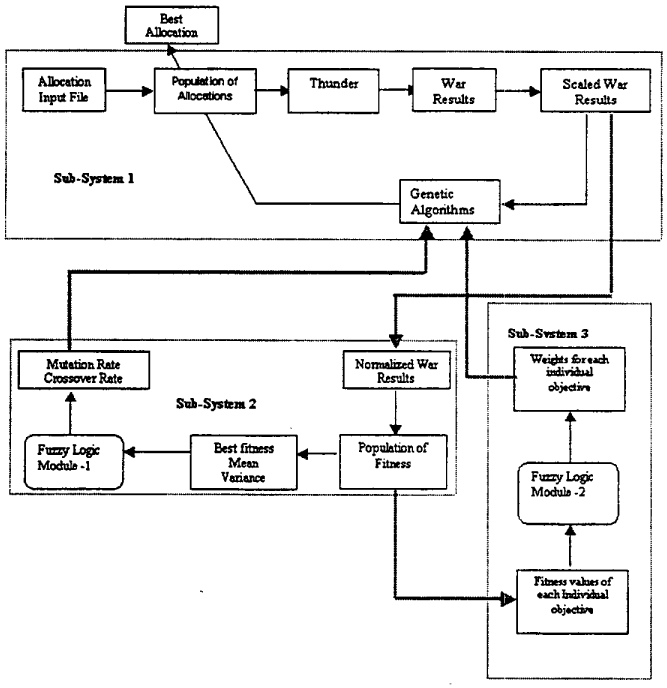


Figure 1. System Layout of the Proposed Solution

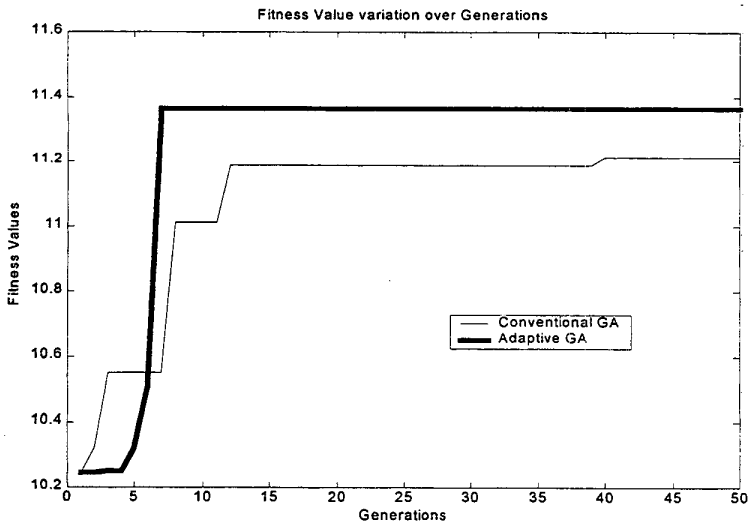


Figure 2.Comparison of Fitness Function Values with and without Adaptation

Table 1.
Maximum Fitness Values with and without Adaptation

Fitness Function	Best fitness values
$F_{without\_adap}$	11.212
F_{Adap}	11.363

Table 2.
Optimum War Allocation Values with and without Adaptation

Method	OCA	INT	DSEAD	STI
$F_{without\_adap}$	27	33	0	40
F_{Adap}	7	20	40	33

Table 3.
Simulation War results with and without Adaptation

Method	Blue Territory Lost	Blue Aircraft Lost	Red Armor Killed	Red Strategic Targets Killed
$F_{without\_adap}$	115	155	1994	552
F_{Adap}	109	146	2064	483

Table 4
Simulation War Results for Conventional GA, Adaptive GA with Fixed and with Dynamic Fitness Function

Method	Blue Territory Lost	Blue Aircraft Lost	Red Armor Killed	Red Strategic Targets Killed
$F_{without\_adap}$	115	155	1994	552
F_{Adap}	109	146	2064	483
$F_{Adap\_dynamic\_Fit\_fn}$	137	168	2201	633