Hybrid Fuzzy-Genetic Algorithm Approach for Crew Grouping

Hongbo Liu, Zhanguo Xu
Department of Computer
Dalian University of Technology, Dalian, China
lhb@dlut.edu.cn

Ajith Abraham
School of Computer Science and Engineering
Chung-Ang University, Seoul, Korea
ajith.abraham@ieee.org

Abstract

Crew grouping is an important problem and formulating a good solution always involves many challenges. For example, grouping soldiers intelligently to tank combat units, we should take into consideration the combined technical proficiency of the soldiers, the amount of military training, the units from which the soldiers come, their service age, personal background, etc. In this paper, we propose a hybrid Fuzzy-Genetic Algorithm (FGA) approach to solve the crew grouping problem. Fuzzy logic based controllers are applied to fine-tune dynamically the crossover and mutation probability in the genetic algorithms, in an attempt to improve the algorithm performance. The FGA approach is compared with the Standard Genetic Algorithm (SGA). Empirical results clearly demonstrates that while the SGA approach gives satisfactory solutions for the problem, the FGA method usually performs significantly better.

1. Introduction

The solution to crew grouping always involves many factors. For grouping soldiers efficiently to tank combat units, we should take into consideration the combined technical proficiency of the soldiers, the amount of military training, the units from which the soldiers come, their service age, personal background, etc. Genetic algorithms (GAs) are particularly suited for obtaining approximate solutions for multi-variable optimization problems where mathematical analysis are difficult [1, 2]. They are inherently parallel, since the search for the best solution is performed over data structures that can represent a number of possible solutions. In this paper, we investigate the use of a fuzzy logic controller to dynamically fine-tune the crossover and mutation probability of a standard genetic algorithm for the crew grouping problem. The structure of this paper is as follows. We describe briefly the problem of crew grouping soldiers into tank combat units in Section 2. We design our algorithms for the problem in Section 3. Experiment results and discussions are provided in Section 4. Finally, we conclude our work in the paper.

2. Problem Statement and Modeling

The armored cavalry consists partly of a number of tanks, each a basic, self-contained combat unit. The tank commander, the gunner, the driver and the loader, all indispensable, make up the tank's crew. For maximum battle effectiveness, we should consider the following factors:
1. The combined technical proficiency of the soldiers: The assignment of soldiers to tank units must take into consideration the different professional achievement levels of individual members, and strive for the combined technical proficiency of the crew as a whole.
2. Amount of military training: The more military training a solider has, the more experienced he becomes. This enhances his ability to respond to real battlefield situations. Such soldiers should be assigned evenly among tank crews so that their experience benefits more.
3. Unit: There is a period of learning to work together when soldiers from different units carry out a battle task. They must adapt and cooperate. Therefore, it is better to assign soldiers from the same unit to the same tank crew.
4. Service age: There is more service age spread in the armored cavalry than that in others. We assign soldiers with different service ages into the same crew if possible, so that the new soldiers can learn from the old.
5. Personal background: Soldiers with similar personal background communicate with each other better. They should be assigned to the same tank crew if possible.

Based on the above analysis, the problem can be modeled as follows:

$$G_i = (C_i + D_i + G_i + L_i) \times p_{i1} \times p_{i2} \times p_{i3} \times p_{i4} \times p_{i5} \quad (1)$$

where G_i, C_i, D_i, G_i, L_i are the battle effectiveness of the ith crew, the ith tank commander, the ith driver, the ith gunner, the ith loader, respectively. $p_{i1}, p_{i2}, p_{i3}, p_{i4}, p_{i5}$ are the ith unit exchanging coefficients, which is determined by the combined technical proficiency of the soldiers, amount of military training, unit, service age, personal background, respectively. So system theory states that the whole is larger than the sum of the parts, there

$$p_{i1} \times p_{i2} \times p_{i3} \times p_{i4} \times p_{i5} \geq 1 \tag{2}$$

For the problem of grouping soldiers to tank combat units, we have to solve the objective function:

$$f(x) = \max \sum_{i=1}^{n} G_i \tag{3}$$

If we assign n commanders, n drivers, n gunners, n loaders for n tanks, the number of permutation and combination will be $(n!)^3$.

3. Algorithm Design

In this section, we design genetic algorithms to solve the problem outlined above. The standard genetic algorithm (SGA) is a useful optimization algorithm in a wide variety of circumstances [3]. However, one of its features is a tendency for all of the population to converge to a single solution which is suboptimal. If all the members of the population are very similar, the crossover operator has little function and mutation turns out to be the primary operator. This effect is known as premature convergence [4]. Adaptive genetic algorithms, which dynamically adapt selected control parameters or genetic operators during the evolution, have been built to avoid the premature convergence problem and improve GA behaviour. One of the adaptive approaches is the parameter setting techniques based on the use of fuzzy logic controllers (FLCs), the fuzzy genetic algorithm (FGA) [5].

3.1. Mapping the Problem Space

1. Encoding scheme: GAs work with a population of chromosomes, each of which can be decoded into a solution of the problem. Encoding scheme in genetic algorithm is the basis of its development, which directly affects the construction of genetic operators and performance of genetic algorithm. Real-coded scheme is used in our model. A matrix describes the structure of the model. The change of its node implies different soldier is assigned. As an example, for tank commander $No.1$, the information to be coded in the GAs is 10000001. Similarly, the driver $No.1$ is 20000001, the gunner is 30000001, and the loader is 40000001. For example, a solution for six tanks is

$$\begin{bmatrix} 10000006 & 20000005 & 30000002 & 40000001 \\ 10000005 & 20000001 & 30000003 & 40000005 \\ 10000004 & 20000004 & 30000001 & 40000003 \\ 10000003 & 20000002 & 30000007 & 40000006 \\ 10000001 & 20000006 & 30000004 & 40000008 \\ 10000002 & 20000007 & 30000008 & 40000004 \end{bmatrix}$$

2. Initial population: An initial population is created from a random selection of solutions. Note that the number of troops within the four kinds of soldiers usually is not equal. For example, there may be 6 tank commanders, 7 gunners, 8 drivers, and 9 loaders, to be assigned to 6 tank crews. Every soldier within a kind has the same chance to join any crew.
3. Fitness function: During the search procedure, each individual is evaluated using the fitness function. Our objective is to maximize battle effectiveness. So in our algorithm the fitness function is defined as Equations (1) and (3).
4. Selection: For selection, two individuals are randomly chosen from the population and they form a couple for crossover. Selection can be based on different probability distributions, such as uniform distribution or a random selection from a population where each individual is assigned a weight dependent on its fitness, so that the best individual has the greatest probability to be chosen. We order the individuals of the population according to their fitness values. The selection probability for the individuals is a linear distribution.
5. Crossover: The crossover operator generates new chromosomes. Crossover is usually applied to selected pairs of parents with a probability equal to a given crossover rate. In our algorithms, the idea behind a crossover operation is as follows: it takes as input 2 individuals, selects a random point, and exchanges the column behind this point as a whole. It avoids effectively unfeasible solution during crossover. It means any soldier couldn't be assigned into different crew at the same time. To illustrate this idea, we consider two individuals A and B. Then, a crossover between them at point 2 may result in offsprings C and D as illustrated below.

$$\mathbf{A} = \begin{bmatrix} 10000006 & 20000005 & 30000002 & 40000001 \\ 10000005 & 20000001 & 30000003 & 40000005 \\ 10000004 & 20000004 & 30000001 & 40000003 \\ 10000003 & 20000002 & 30000007 & 40000006 \\ 10000001 & 20000006 & 30000004 & 40000008 \\ 10000002 & 20000007 & 30000008 & 40000004 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 10000005 & 20000005 & 30000004 & 40000007 \\ 10000003 & 20000007 & 30000002 & 40000003 \\ 10000006 & 20000004 & 30000007 & 40000002 \\ 10000004 & 20000006 & 30000008 & 40000008 \\ 10000001 & 20000003 & 30000006 & 40000005 \\ 10000002 & 20000002 & 30000003 & 40000004 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 10000006 & 20000005 & 30000002 & 40000001 \\ 10000005 & 20000007 & 30000003 & 40000005 \\ 10000004 & 20000004 & 30000001 & 40000003 \\ 10000003 & 20000006 & 30000007 & 40000006 \\ 10000001 & 20000003 & 30000004 & 40000008 \\ 10000002 & 20000002 & 30000008 & 40000004 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} 10000005 & 20000005 & 30000004 & 40000007 \\ 10000003 & 20000001 & 30000002 & 40000003 \\ 10000006 & 20000004 & 30000007 & 40000002 \\ 10000004 & 20000002 & 30000008 & 40000008 \\ 10000001 & 20000006 & 30000006 & 40000005 \\ 10000002 & 20000007 & 30000003 & 40000004 \end{bmatrix}$$

6. Mutation: The mutation operation modifies an individual. In defining the mutation operator, we take into account the domain type of an attribute. We consider two important situations: The individual must be a feasible solution. Those soldiers who didn't take part in the assignment process in the last generation are as much as possible exchanged into the group of new individuals. For example, new soldiers of the same kind are added into the end of the list as gene A before mutation. If the mutation points 3 and 7 are selected, the result of the mutation operation results in gene B as depicted below.

$$\begin{array}{ccc} 30000002 & & 30000002 \\ 30000003 & & 30000003 \\ 30000001 & & 30000005 \\ 30000008 & & 30000008 \\ 30000004 & \longrightarrow & 30000004 \\ 30000007 & & 30000007 \\ 30000005 & & 30000001 \\ 30000006 & & 30000006 \\ gene-A & & gene-B \end{array}$$

7. End criterion If the termination criterion (statistical or temporal) is not satisfied, then a return is made to the third step; otherwise, the algorithm is terminated. The criterion is usually a sufficiently good fitness, or a maximum number of iterations, or the global best fitness is steady-going within determinated iterations.

3.2. Genetic Algorithm Parameter Control Based on Fuzzy Logic

The performance of the genetic algorithm is correlated to directly with its careful selection of parameters. It is possible to destroy an high fitness individual when a large crossover probability is set. The performance of the algorithm would fluctuate significantly. For a low crossover probability, sometimes it is hard to obtain better individuals and does not guarantee faster convergence. High mutation introduces too much diversity and takes longer time to get the optimal solution. Low mutation tends to miss some near-optimal points. The use of fuzzy logic controllers to adapt genetic algorithm parameters is useful to improve the genetic algorithm performance [6]. An FLC is composed by a knowledge base, that includes the information given by the expert in the form of linguistic control rules, a fuzzification interface, which has the effect of transforming crisp data into fuzzy sets, an inference system, that uses them together with the knowledge base to make inference by means of a reasoning method, and a defuzzification interface, that translates the fuzzy control action thus obtained to a real control action using a defuzzification method. The generic structure of an FLC [7] is shown in Figure 1.

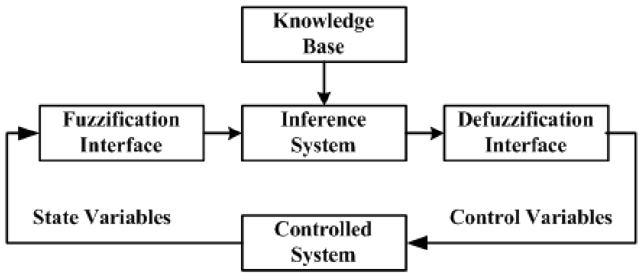

Figure 1. Generic structure of an FLC.

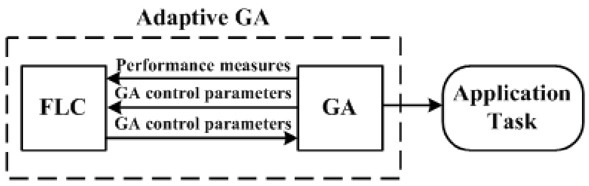

Figure 2. Structure of a fuzzy GA based on FLCs.

Applications of FLCs for parameter control of GAs are to be found in [8]. The main idea is to use an FLC whose inputs are any combination of GA performance measures or current control parameters and whose outputs are GA control parameters. Current performance measures of the GA are sent to the FLCs, which computers new control parameters values that will be used by the GA. In our FGA approach, the crossover probability and the mutation probability are defined on specific individuals of the population using several FLCs that take into account fitness values of individuals and its distances. The next subsections present the design of the FLC that adapts the crossover probability Pc and the mutation probability Pm.

Our strategy for updating the crossover and mutation

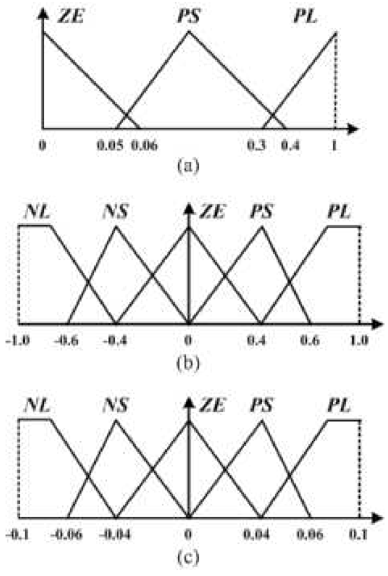

Figure 3. Membership functions. (a) for e_1, (b) e_2, (c) for $\Delta Pc(t)$.

Table 1. Fuzzy rules for crossover operation ($\Delta Pc(t)$).

	e_2				
e_1	NL	NS	ZE	PS	PL
PL	NS	ZE	NS	PS	PL
PS	ZE	ZE	NL	ZE	ZE
ZE	NS	NL	NL	NL	NL

Table 2. Fuzzy rules for mutation operation ($\Delta Pm(t)$).

	e_2				
e_1	NL	NS	ZE	PS	PL
PL	PS	ZE	PS	NS	NL
PS	ZE	ZE	PL	ZE	NS
ZE	PS	PL	PL	PL	PS

probabilities is to consider the changes of the maximum fitness and average fitness in the GA population of two continuous generations. The occurrence probabilities would be increased if it consistently produces a better offspring during the recombination process; however, Pc would be decreased and Pm increased when $f_{ave}(t)$ approaches to $f_{max}(t)$ or $f_{ave}(t-1)$ approaches to $f_{ave}(t)$. This scheme is based on the fact that it encourages the well-performing operators to produce more offspring, while also reducing the chance for poorly performing operators to destroy the potential individuals during the recombination process. The FLC proposed has two inputs: A two-dimension FLC system is used in our GA, in which there are two parameters e_1 and e_2:

$$e_1(t) = \frac{f_{max}(t) - f_{ave}(t)}{f_{max}(t)} \tag{4}$$

$$e_2(t) = \frac{f_{ave}(t) - f_{ave}(t-1)}{f_{max}(t)} \tag{5}$$

where

t is timestep,

$f_{max}(t)$ is the best fitness at Iteration t,

$f_{ave}(t)$ is the average fitness at Iteration t,

$f_{ave}(t)$ is the best fitness at Iteration t,

$f_{ave}(t-1)$ is the average fitness at Iteration (t-1).

The membership functions are shown in Figure 3, where NL is Negative large, NS is Negative small, ZE is Zero, PS is Positive small, PL is Positive large. The inputs of the mutation FLC are the same as those of the crossover FLC. But the membership function for $\Delta Pm(t)$ was scaled by 10%. Fuzzy rules describe the relation between the inputs and output. Tables 1 and 2 show the Rule-Base used by the FLCs presented. For the parameter control in our GAs, the outputs $\Delta Pc(t)$ and $\Delta Pm(t)$ of fuzzy logic controllers are translated the fuzzy control action thus obtained to a real control action. Center of gravity [9] is used as our defuzzification method. Then we use the crisp value to modify the parameters Pc and Pm as follows:

$$Pc(t) = Pc(t-1) + \Delta Pc(t) \tag{6}$$

$$Pm(t) = Pm(t-1) + \Delta Pm(t) \tag{7}$$

As summary, the general structure of our algorithm is as follows:

```
Begin FGA
    $t = 0$ Iteration counter
    Initialize population $P(t)$
    Evaluate population $P(t)${i.e., compute fitness values }
    while (the end criterion is not met) do
        $t = t + 1$
        Select $P(t)$ from $P(t-1)$
        Crossover $P(t)$
        Mutate $P(t)$
        Evaluate $P(t)$
        Regulating GA parameters
        {
            Call fuzzy logic contrller($e_1$,$e_2$)
            Update according to equations (6) and (7)
        }
    End while
End FGA
```

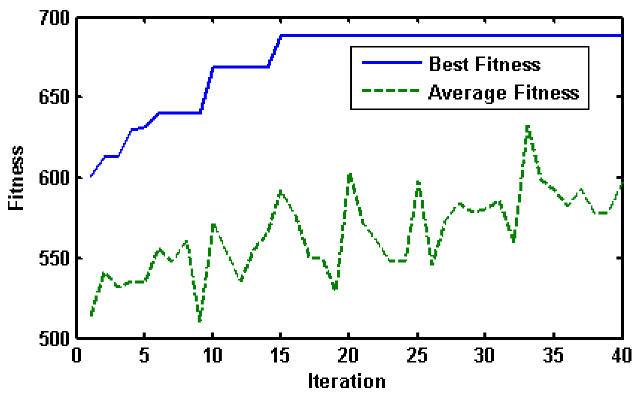

Figure 4. The performance curves of SGA

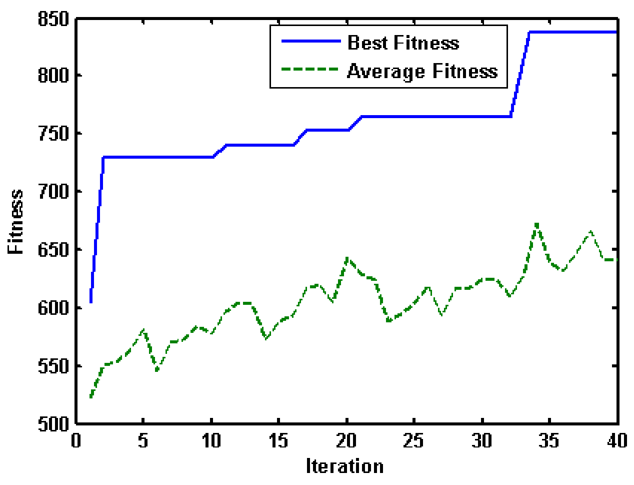

Figure 5. The performance curves of FGA

4. Experiment Setup and Results

Based on our algorithms, we have built a re-targetable prototype of a positioning tool, which means that the tool can be coupled to different databases. For example, there are 6 tank commanders, 7 gunners, 8 drivers, and 9 loaders in the unit. They would be grouped into 6 tank crew. Every soldier within a kind has the same chance to join any crew. Considering the problem of finding the whole best assignment, we search for the best solutions using SGA and FGA, and compare their results. The initial population size is set to 20. For SGA, the crossover probability Pc is set to 0.8 and the mutation probability Pm, 0.01, respectively. We set the initial parameters in FGA as same as ones in SGA. The performance curves (best and average fitness values) of SGA and FGA are illustrated in Figures 4 and 5. SGA converges quickly but there is a larger probability to get trapped in local optima, while FGA spends more time to explore better solutions with a larger probability to arrive at the global optima. Other statistical results are showed in Table 3, in which each algorithm was run for 10 times. Although SGA also searches for satisfactory solutions, empirical results clearly indicates that FGA usually performs significantly better.

In Figure 4, its maximization of battle effectiveness (best fitness) obtained is 688.0021, and the grouping result is depicted below:

10000005	20000007	30000006	40000008
10000003	20000001	30000001	40000005
10000001	20000004	30000005	40000002
10000004	20000003	30000004	40000003
10000002	20000005	30000007	40000009
10000006	20000002	30000002	40000001

In Figure 5, its maximization of battle effectiveness (best fitness) obtained is 837.3537, and the grouping result is depicted below:

10000003	20000007	30000008	40000009
10000006	20000003	30000004	40000004
10000005	20000006	30000002	40000003
10000002	20000002	30000007	40000002
10000001	20000004	30000001	40000007
10000004	20000005	30000005	40000006

Table 3. The best fiteness (10 times) of SGA and FGA.

Sequence Number	SGA	FGA
1	750	780
2	693	790
3	740	820
4	720	837
5	735	831
6	760	804
7	732	820
8	770	829
9	688	794
10	710	812

5. Conclusions

In this paper, genetic algorithms were introduced for the crew grouping problem of allocating soldiers intelligently for different tank combat units. We used fuzzy logic based controllers to adapt the parameters of genetic algorithms, thereby improving its performance. The standard genetic algorithm converges quickly with a larger probability to get trapped in local optima, while the fuzzy genetic algorithm spends more time to explore more feasible solutions with a larger probability to find global optimal solutions. Empirical results reveal that the proposed FGA method for the crew grouping problem is efficient when compared to standard GA approach.

Acknowledgments

The authors wish to thank to Qingsheng Zhang, Xiangding Su and Bo Li, for all the discussions, feedback and technical insights. We are indebted to the anonymous reviewers for their helpful comments on the manuscript.

References

[1] M. Srinivas and L.M. Patnaik, ”Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms”, *IEEE Transaction on System, Man and Cybernetics,* IEEE, 24, 1994, pp. 656-667.

[2] A.A. Jamshidifar and C.Lucas, ”Genetic Algorithm Based Fuzzy Controller for Nonlinear Systems”, *Proceedings of Second IEEE International Conference on Intelligent Systems,* IEEE, 3, 2004, pp. 43-47.

[3] Goldberg, D.E., *The Design of Competent Genetic Algorithms: Steps Toward a Computational Theory of Innovation,* Kluwer Academic Publishers, Dordrecht, 2002.

[4] L. M. Schmitt, ”Theory of Genetic Algorithms II: Models for Genetic Operators Over the String-tensor Representation of Populations and Convergence to Global ptima for Arbitrary Fitness Function Under Scaling”, *Theoretical Computer Science,* Elsevier, 310, 2004, pp. 181-231.

[5] Y. Yun, ”Performance Analysis of Adaptive Genetic Algorithms with Fuzzy Logic and Heuristics”, *Fuzzy Optimization and Decision Making,* Kluwer Academic Publishers, 2, 2003, pp. 161-175.

[6] L. Mark and E. Shay, ”A fuzzy-based lifetime extension of genetic algorithms”, *Fuzzy Sets and Systems,* Elsevier, 149, 2005, pp. 131-147.

[7] F. Herrera and M. Lozano. ”Fuzzy adaptive genetic algorithms: design, taxonomy, and future directions” *Soft Computing,* Springer-Verlag, 7, 2003, pp. 545-562.

[8] O. Cordon, F. Gomide, F. Herrera, F. Hoffmann and L. Magdalena, ”Ten years of genetic fuzzy systems: current framework and new trends”, *Fuzzy Sets and Systems,* Elsevier, 141, 2004, pp. 5-31.

[9] A. E. Eiben, R. Hinterding, and Z. Michalewicz, ”Parameter Control in Evolutionary Algorithms”, *IEEE Transations on Evolutionary Computation,* IEEE, 3(2), 1999, pp. 124-141.