Shape Formation in Games: a Probability-based Evolutionary Approach

Caitao Zhan, Changhe Li*
School of Computer Science, China University of Geosciences (Wuhan), China
caitaozhan@163.com, changhe.lw@gmail.com

***Abstract*—Shape formation for a group of agents is crucial in many strategy games as it brings collective payoff through cooperations. Finding a good formation for a group of agents in games is considered as a combinatorial optimization problem in this paper. To address this issue, this paper proposes a novel and simple evolutionary algorithm based on a probability learning matrix, which is the foundation of a formation learning and discovery mechanism. Experimental results on several test problems show that the proposed algorithm works well in comparison with another peer algorithm.**

***Keywords*-Shape formation, probability learning, evolutionary computation.**

I. Introduction

Artificial intelligence (AI) are widely used in games, e.g., game of computer go and combat games, to generate human like behavior in non-player control. Also, AI can help players finish some complex tasks such as pathfinding and controlling a group of agents in games. Agents can be different figures in different games: e.g., tanks, fighter planes, or warship in war games and heros in multi-player online battle arena games like Warcraft and League of Legends etc. In cooperative game theory, formations focuses on which coalition to form in order to result in collective payoff. In this paper, a formation of a group of agents is defined as a special shape for agents, either in one region or several separate regions in a game map. A game in this paper is a combat game that involves two teams of agents. A good formation (a group of agents F) should gain collective payoff and follow two principles: 1) the concentration of superior forces of F on attacking weak parts of enemy; 2) the avoidance of superior forces of enemy from attacking weak part of F itself.

In real-world wars, infantry, cavalry, vehicles come into different formations e.g., column, line, square, wedge, and vee etc., in different attack or defence circumstances. Many video games simulate real-world wars and thus game programs also needs to generate formations. Traditionally, formations are formed according to classical tactics and empirical experience. But those rules of thumb in real-world could hardly work well in modern video games where scenes change quickly and players need more attractive and competitive formations. The problem is a combinatorial optimization problem and is considered NP-hard: given a grid map and a formation of enemy agents, the objective is to find a best formation for a group of agents F that maximize the values of a function followed the two principles mentioned above.

Exhaustive search strategies are not acceptable to solve the problem as time consumption of brute forcing explodes. Thus heuristic search strategies are often used. Heuristic search strategies can be classified either as deterministic or non-deterministic. Deterministic search strategies always produce the same output given a same input, such as hill climbing and Newton method. Their major drawback is having a risk of getting trapped in a local optimum. Non-deterministic search strategies have a chance to escape from local optimum by means of randomness. Evolutionary computation(EC) methods are non-deterministic metaheuristic search algorithms, which are usually inspired from natural phenomenon. Typical algorithms are genetic algorithms (GAs)[5], ant colony optimization (ACO)[4], particle swarm optimization(PSO) [6], estimated distribution algorithm (EDA)[1] etc.

EC is the science of designing and applying algorithms based on Darwinian principles of natural selection. It is a universal problem solver that gives a optimal/near-optimal solution to combinatorial problems, function optimizations, and many real world problems. Traditional EC algorithms depend on associated parameters like operators and probabilities of crossover and mutation, size of population etc. In recent years, many new algorithms are introduced in the realm of EC but as stated in[8] some they are unnecessarily complicated to artificially mimic the natural process or to create novelty by mixing up ideas of existing algorithms.

Estimation of distribution algorithms (EDA)[9] are introduced by the motivation of creating an new type of EC algorithm that is easier to predict the movements of the population in the search space as well as to avoiding so many parameters [2]. Sometimes called probabilistic model-building genetic algorithms, EDAs are stochastic optimization algorithms that reproduce a new generation not by traditional crossover nor mutation operators, but by explicitly sampling from a probability distribution estimated from promising candidate individuals. Probabilistic models used to estimate the distribution a population varying from a simple vector to complicated trees and networks. Probabilistic graphic models are widely applied to EDAs[7], because they are powerful in expressing the interrelationships between different variables through the joint probability distribution. Typical models include Bayesian networks, Markov networks, dependency networks, chain graphs etc.

In this paper, we design a novel and simple metaheuristic evolutionary algorithm based on a probability matrix with a formation learning and discovery mechanism. The proposed algorithm is shown effective through experimental studies and is tested against another algorithm. The formation learning/discovery mechanism learns a probability distribution to identify good agents.

The paper is organized as follows. Sect. II describes the proposed algorithm in detail. Sect. IV. shows experimental results and discussions. Finally, conclusions and future work are presented in Sect. V.

II. Probability-based Evolutionary Algorithm

In this section, the probabilistic learning mechanism is implemented to a new formation construction algorithm, namely probability-based evolutionary algorithm(PEA), to solve shape formation problems.

A. Game Map

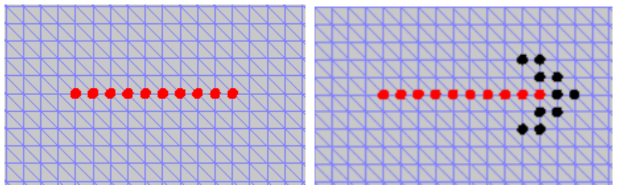

a) line formation b) red VS. black formation

Figure 1. Formation examples

Fig.1 shows a game map, which is discretized into a grid where each cross point in the grid is represented by a two dimensional integer coordinate: (i, j). The left bottom corner is the origin of the map. A colored point stands for an agent and it can only be placed on a cross point. An agent can can be regarded as a tank in tank combat games such as World of Tanks, heros or monsters in multi-player online battle arenas, e.g., League of Legends. A team of agents are homogeneous: all have the same attacking ability distance in all directions and the same color. No two agents can be placed on one cross point in one team. Different teams with different colors all share a same attacking ability distance. A team forms a formation in a game map, e.g., the line shape formation in Fig.1-a.

B. Fitness Evaluation

Based on the two principles for a good formation mentioned in Sect. I, we design a simple fitness evaluation method shown in Algorithm 1. The method returns a fitness value for the first formation in the parameter list. Take Fig.1-b as an example. There are ten black agents that can attack any one of the red agents, but only two red agents that

Algorithm 1 Evaluate(formation1, formation2)

```
1: 1attack2, 2attack1 ← 0
2: for (each agent $a_i$ in formation1) do
3:     if $a_i$ can attack any agent in formation2 then
4:         1attack2 ← 1attack2 + 1
5:     end if
6: end for
7: for (each agent $a_i$ in formation2) do
8:     if $a_i$ can attack any agent in formation1 then
9:         2attack1 ← 2attack1 + 1
10:    end if
11: end for
12: return 1attack2 - 2attack1
```

can attack any one of black agents (the attacking ability distance is two, i.e., two blocks in a grid). The fitness value for a formation of a team is the number of agents that can attack the opponent team minus the number of agents being attacked by its opponent team. For example, in Fig.1-b the fitness value of the black formation is eight (10-2), and fitness value for red formation is -8 (2-10).

C. Formation Learning

In this paper, a population contains a set of individuals, which are different shape formations in this scenario, therefore an individual/formation comprises a set of integer coordinates, which represent the positions of agents. This paper introduces a probability matrix, inspired by the idea from [11], to predict or learn the probability of positions of agents be to located in a game map.

An element PM_{ij} in probability matrix (PM) reflects the quality of a coordinate (i, j). A higher value PM_{ij} simply indicates that more individuals having an agent at coordinate (i, j), vis versa for a lower value of PM_{ij}. Note that at the very beginning, the population is randomly initialized so that every coordinate should have a similar value with each other. A probability matrix PM reflects a probability distribution with each element PM_{ij} indicating the probability of agents selecting an integer coordinate (i, j) of a grid.

PM is also able to reflect the learning process as the evolutionary process goes on. When population evolves, they usually improves. What underlays this improvement are two types of knowledge: 1) The number of good agents increases in the population (good agents in this paper means agents which belong to the global/near-global optima); 2) The frequency of the appearance of a particular good agent increases in the population.

During the search, we simply count the frequency appearance of an agent (coordinate) in a population. Based on the knowledge above, we assume that as the population improves, the higher frequency an agent has, the better an agent is. So when reproducing a next generation, such good agents should have higher probabilities for survival. Moreover, given the knowledge that a formation with a higher fitness value normally contains more good agents than a formation with lower fitness value, agents in good individuals have right to be paid more attention than agents in less good individuals. To implement this, we assign each individual a weight w in a population, and the higher the fitness value, the higher the weight. Weight is calculated from fitness in two steps:

The first step is normalization, i.e. mapping fitness values within [0, 1] as follows:

$$f^{'}(x_i) = \frac{f(x_i) - f(x^{worst}) + 1}{f(x^{best}) - f(x^{worst}) + 1}, \tag{1}$$

where $f(x_i), f(x^{best}), f(x^{worst})$ are the original fitness value of x_i, the best individual, and the worst individual respectively, $1 \leq i \leq PS$. The original fitness value is computed by Algorithm 1.

The second step is calculating weight from normalized fitness through a sigmoid function as below:

$$w_i = \frac{1}{1 + e^{-f^{'}(x_i)}} \tag{2}$$

The higher the fitness an individual has, the higher the weight it has. A set of preliminary experimental results show that the sigmoid function works well.

Finally, we compute each element PM_{ij} of PM as follows:

$$PM_{ij} = \sum (w_k \cdot o_k) + \epsilon, \tag{3}$$

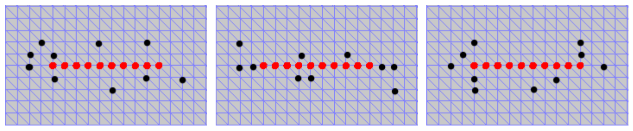
where $o_i = 1$ if exists an agent $\in formation_k$ that locates at coordinate (i, j), else $o_i = 0$, $k \in [1, PS]$, $(i, j) \in Grid$, ϵ is a small number and biases the reproduction toward random formations (it can thus be regard as a form of mutation). We assign ϵ inspired from [10] as follows

$$\epsilon = \frac{popSize \cdot indiSize}{availablePoints} \cdot ratio \tag{4}$$

where $indiSize$ is the individual size (the number of agents), $availablePoints$ is the number of cross points in the grid minus the number of cross points where an agent should not be located, e.g., cross points where is occupied by rival agents. $ratio$ is a constant related to the pressure toward randomness and is set to 0.0002 by default. ϵ is proportional to the expected value of PM_{ij}, and is designed to be general to different problems.

The following example illustrates the calculation of PM_{25} with a population of three individuals on a formation with ten agents in a 18×10 grid as shown in Fig. 1-a. Assume the red formation as enemy formation, and our objective is to evolve to certain formations like the black formation in Fig. 1-b, where black formation is in a superior circumstance (team black sieges the two right agents of team red with the attacking ability distance of two). By observing to the population, position (2,5) appears in all three individuals. According to Eqs.(3) and (4), $PM_{25} = w_1 \cdot 1 + w_2 \cdot 1 + w_3 \cdot 1 + 0 = 1.895$.

This probability matrix is a mixture of the matrix in the original ACO and EDA: the weight part derives from pheromone in ants and the matrix is similar to the distribution in EDA. After randomly initialized, the probability matrix is updated whenever a formation is improved.



f_1=0, w_1=0.622 f_2=-2, w_2=0.542 f_3=3, w_3=0.731

Figure 2. A population with 3 black individuals, attacking distance=2

D. Formation Discovery

In this subsection we introduce a simple formation discovery algorithm based on the probability matrix introduced above, namely probability-based evolutionary algorithm (PEA). The framework of PEA is presented in Algorithm 2.

Algorithm 2 Probability-based Evolutionary Algorithm

```
1: Initialize a fixed target formation $F_{target}$
2: Initialize a population $X = [x_1, x_2, \ldots, x_{ps}]$ with $PS$ individuals
3: Create an archive $A = X$ and initialize the probability matrix $PM$ with $A$
4: while (termination criteria not satisfied) do
5:     for (each individual $x_i$) do
6:         $x_i \leftarrow A_i$
7:         Construct a new formation $x_t$ based on $x_i$, starting with j ← 0
8:         while ($x_t$ is not completed) do
9:             if ($rand() < PL$) then
10:                $x_{tj} \leftarrow x_{ij}$
11:            else    ▷ $get(PM)$ returns an agent based on the probability matrix
12:                $x_{tj} \leftarrow get(PM)$
13:            end if
14:            if ($x_{ij} \neq x_{tj}$) then
15:                $x_{ij} \leftarrow x_{tj}$
16:                if (Evaluate($x_i$, $F_{target}$)>Evaluate($A_i$, $F_{target}$)) then
17:                    $A_i \leftarrow x_i$
18:                else if (Evaluate($x_i$, $F_{target}$)==Evaluate($A_i$, $F_{target}$)) then
19:                    $A_i \leftarrow x_i$ with 50% probability
20:                end if
21:            end if
22:            $j \leftarrow j + 1$
23:        end while
24:    end for
25:    Update $PM$ based on population A
26: end while
```

An archive population(A) is utilized for storing the best solutions found so far for each individual. We regard this as a memory population saving each individual's historical best solution. In the work flow of the algorithm, a new individual x_t is constructed based on the historical best solution of $x_i(A_i)$. PL is a probability learning parameter $\in (0, 1)$, controlling how many percent of agents in x_t is directly derived from x_i, and the rest is selected from PM through some strategy such as roulette. If x_i is updated by the constructed x_t, and x_i has a higher fitness than A_i, then update A_i with x_i.

III. Experimental Studies

In this section, eight set of formation test problems are shown in Fig. 3. The test problems are carefully designed so that they are typical enough to represent a range of real

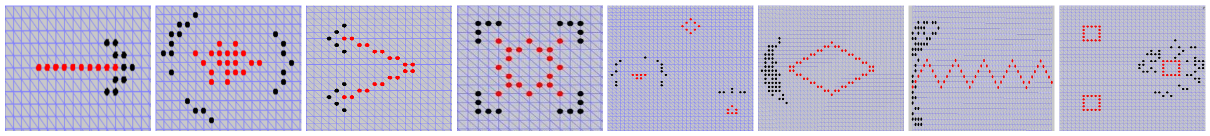


Figure 3. Eight shape formation problems

world situations. An EDA version named Node Histogram Based Sampling Algorithm With Template proposed in [10] is chosen to compare with the performance of PEA due to its good performance in comparison with other peer EDAs [3].

A. Experimental Setups

Usually, an error between the best solution found by the algorithm and the global optimum is used to evaluate an algorithm's performance. However, the eight test problems are designed by ourselves without the knowledge of the global optima. Here we explain how the problems are designed. In Fig. 3, eight test problems are designed: eight red formations represent eight problems in each graph. Assume black formations are the ones the algorithm obtained, which start with a random formation scattering across the game map and gradually evolves into some certain formations shapes by taking advantages of red team in a battle. A rule of naming a problem is as follows, for example, name "Line10-2" consists of three parts: a) "Line" means that the shape of red formation is a line; b)"10" denotes that the size of a formation (individual) is ten, c)"2" indicates that the attacking ability distance of an agent is two. The test problems are designed in those ways based on real-world military backgrounds. In real-world battles, six formations are commonly used for tanks or infantry: column, line, vee, wedge, echelon and square. The formation problems designed in this paper are inspired from these six basic formations.

We set the population size as follows

$$PS = c\sqrt{IndivudialSize \times GridSize} \qquad (5)$$

where c is a coefficient. Generally speaking, the larger the individual size, the more complex the problem is, and thus need a larger population to search in the search space. Also, the similar relationship exists with grid size. Therefore we designed the above equation. Note that grid size could be reduced based by the knowledge of attacking ability distance: search space that are out of reach of target(red) formation could be eliminated. Based on some preliminary tests, in PEA, c is set to two by default, and c is set to five in NHBSA/WT.

In PEA, an important parameter is PL, which controls how much percent of agents of an individual learn form the probability matrix. We set PL to 0.5 by default. In NHBSA/WT, a selection rate is a crucial parameter which determines how many individuals in a parent population will survive. An offspring population is sampled based on the selected individuals. In this paper, $selectRatio$ is set to 0.8 based on some preliminary experiments, i.e., 80% of good individuals in a population are selected in every generation. The result in this paper are averaged over 30 independent runs.

Table I
PERFORMANCE COMPARISON BETWEEN PEA AND NHBSA/WT, WHERE NUMBERS ARE FITNESS VALUE IN THE WORST CASE, IN THE BEST CASE, BY MEAN, AND STD STANDS FOR STANDARD DEVIATION. FOR FITNESS, THE HIGHER THE BETTER.

Problem	PEA				NHBSA/WT			
	Worst	Best	Mean	STD	Worst	Best	Mean	STD
Line10-2	8	8	8	0	8	8	8	0
Random20-4	17	17	17	0	16	17	16.87	0.35
Vee10-2	7	8	7.93	0.25	7	8	7.77	0.43
Symmetry20-2.9	16	16	16	0	15	16	15.97	0.18
Distribute20-5	15	17	16.87	0.43	16	17	16.77	0.43
Diamond50-8	35	41	40.6	1.13	0	41	36.4	7.78
ZigZag50-8	36	37	36.63	0.49	33	41	36.77	2.25
ThreeSquare48-6	32	32	32	0	32	32	32	0

B. Experimental Results and Discussions

Table I shows the result of comparison between our proposed PEA and NHBSA/WT on the eight problems. Figure 4 shows the convergence process of the best solutions of the two algorithms. From the table, PEA have a better performance on most problems. Both algorithms achieve the same mean values on three problems. PEA obtains better mean results than NHBSA/WT on four problems. NHBSA/WT outperforms PEA only on one problem. Figure 4 shows that the convergence speed of PEA is faster than that of NHBSA/WT at the early stage on all test problems. PEA achieves better results with less time than NHBSA/WT on most problems. Another interesting thing is that NHBSA/WT needs a larger population size than PEA. This is because NHBSA/WT looses some diversity every generation when a part of poor individuals are removed, and thus a large population is needed to initialize to compensate for this. On the contrary, in PEA every individual is remained over generations until a better solution is found.

Note that, we have not found a better solution for the first five problems with human observations. However for the last three problems, we have observed better solutions. In problems Diamond50-8, ZigZag50-8, and ThreeSquare48-6, we can get solutions with fitness values of 42, 42, 34 respectively. However the algorithms can only obtain the best results with values of 41, 41, and 32. We think that

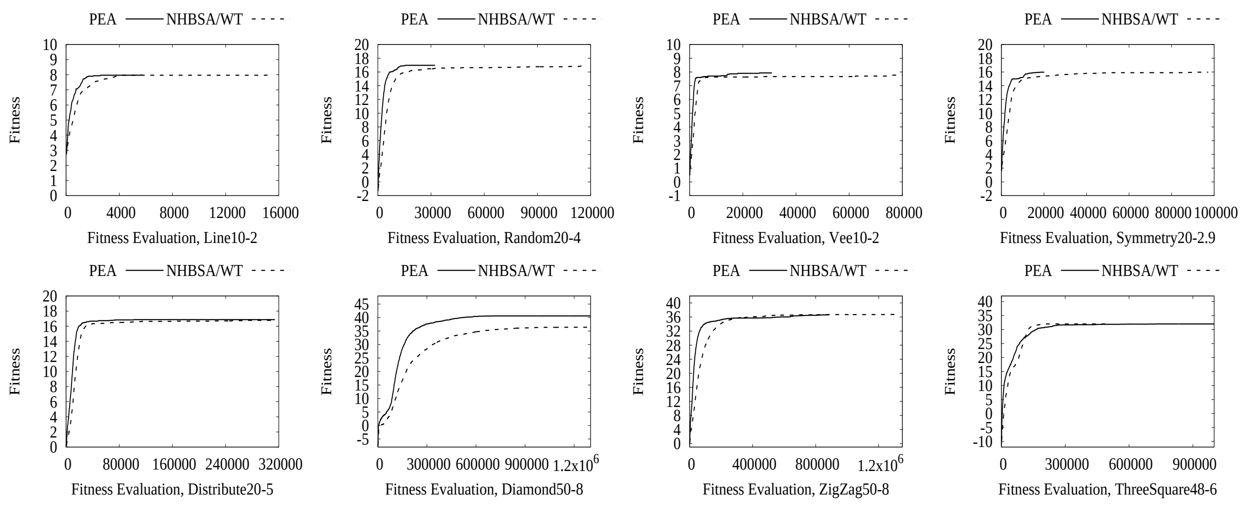


Figure 4. Compare PEA with NHBSA/WT based on average of 30 experiments

the reason for the failure may be the lack of local search techniques. More work should be done like discovering whether exists interdependencies between agents in a formation. Also, multi-population methods could be applied to problems like ThreeSquare48-6, where enemy agents distributes in several separate areas. This is because multi-population techniques may help to find more than one shape in such a case.

IV. CONCLUSION AND FUTURE WORK

This paper proposes a probability-based evolutionary algorithm that learns through a statistical matrix for solving shape formation problems. PEA is inspired by the common characteristics shown during the improvement of a population for almost all population-based algorithm. The knowledge of improvement is transformed into a probability matrix. Experiment results reveals that the proposed algorithm is effective for solving formation problems and performs better than an EDA algorithm.

For future work, some local search strategies need to be designed for large scale problems. New strategies like multi-population and partitioning techniques may be useful to solve sophisticated problems. At present all agents are homogeneous, but in the future they shall be heterogenous. Then new firepower assignment issues are needed to be solved.

ACKNOWLEDGEMENT

This work was supported by the National Natural Science Foundation of China under Grants 61673355.

REFERENCES

[1] S. Baluja, “Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning,” DTIC Document, Tech. Rep., 1994.

[2] E. Bengoetxea, “Inexact graph matching using estimation of distribution algorithms,” Ph.D. dissertation, Ecole Nationale Supérieure des Télécommunications, Paris, France, Dec 2002.

[3] J. Ceberio, E. Irurozki, A. Mendiburu, and J. A. Lozano, “A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems,” *Progress in Artificial Intelligence*, vol. 1, no. 1, pp. 103–117, 2012.

[4] M. Dorigo and L. M. Gambardella, “Ant colony system: a cooperative learning approach to the traveling salesman problem,” *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 53–66, 1997.

[5] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.

[6] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, Nov 1995, pp. 1942–1948 vol.4.

[7] P. Larrañaga, H. Karshenas, C. Bielza, and R. Santana, “A review on probabilistic graphical models in evolutionary computation,” *Journal of Heuristics*, vol. 18, no. 5, pp. 795–819, 2012.

[8] Y. Lin, M. Clauss, and M. Middendorf, “Simple probabilistic population based optimization,” *Evolutionary Computation, IEEE Transactions on*, no. 99, pp. 1–1, 2015.

[9] H. Mhlenbein and G. Paa, “From recombination of genes to the estimation of distributions i. binary parameters.” Springer-Verlag, 1996, pp. 178–187.

[10] S. Tsutsui, S. Tsutsui, M. Pelikan, M. Pelikan, D. E. Goldberg, and D. E. Goldberg, “Node histogram vs. edge histogram: A comparison of pmbgas in permutation domains,” Tech. Rep., 2006.

[11] C. L. Yong Xia, “Memory-based statistical learning for the travelling salesman problem,” in *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 2016, accepted.