A Learning Behavioral Model of CGF

Xianquan Meng[1] Yingnan Zhao[2] Liguo Wang[1] Qing Xue[1]

[1] Simulation Center, Department of Equipment Command & Management, Academy of Armored Force Engineering, Beijing 100072, P. R. China

[2] College of Physics Science & Information Engineering, Jishou University, Jishou 416000, P. R. China

Abstract

The Computer Generated Force (CGF) has decision ability by self-learning mechanism, which is an important research field in applying machine learning technology to military simulation. On the basis of modeling architecture of Agent and Learning Classifier Systems (LCSs) technologies, a learning behavioral model framework based on Genetic Algorithms (GA) of CGF is proposed. It discusses elaborately the learning process of this model, in which memory function is first introduced to accelerate. Also, a visible validation system is designed. The simulated results indicate that the learning model is available and feasible.

Keywords: Genetic algorithms, CGF, Agent, LCSs, Learning behavioral modeling

1. Introduction

Computer generated forces (CGF) are automated or semi-automated entities in a battlefield simulation that are generated and controlled by a computer system, perhaps assisted by a human operator, rather than by human participants in a simulator[1]. Recently, interest in agent-based CGF modeling and simulation has been on increase. According to the definition of CGF and Agent, a simulation entity should be autonomous and intelligent, which can adapt itself to the time-varying state of the environment. How does the simulation entity has such an ability yet? It should be able to learn in a CGF model. However, this kind of research hasn't been studied extensively regardless of its importance for two reasons. One is that the learning process is too complex to be modeled effectively. The other is that some researchers have objections to simulation entities possessing learning ability [1].

But, most of researchers consider that learning ability of intelligent entities is a prerequisite for implementing a behavior model with high fidelity. In an effort to find the best learning model, they pay their attentions to machine learning field. Approaches to it can be classified mainly into four categories: rule-based, paradigm-based, networks-based and GAs-based. Among the different categories of machine learning algorithms, GA-based CLSs method has a great deal of potential in CGF models.

The LCSs formalism was introduced by John Holland in 1976 and the first implementation of LCSs was presented by John Holland and Judith Reitman in 1978. Robert E. Smith employed the LCSs to acquire rules for novel fighter combat maneuvers through simulation demonstrating good success [2]. In this paper, we apply the GA-based LCSs technique to the CGF learning behavior model, where memory function is first brought forward to accelerate.

The paper is organized as follows. Section II presents an overview of the associated concepts such as agent-based modeling, GAs and CLSs. An architecture of CGF based on agent is built in Section III, where we give a detailed specification of learning behavioral model on the basis of GAs. Section IV proposes a visible experimental framework and some simulation results obtained. Finally, Section V concludes the paper with some recommendations for future work.

2. Key ideas and definitions

2.1. Agent-Based Modeling

Agents are model constructs designed to operate independently or semi-independently. A key element of agent design is the ability of agents to interact in a cooperative or competitive fashion. There are however many definitions and many different types of agents. In a word, agent is an intelligent entity that has certain independent ability. Structures of agent-based model range within three fundamental categories [3]: deliberative agent, reactive agent and hybrid agent. In this paper, we place considerable attention on the second type and apply the agent-based behavior modeling technique to CGF behavior modeling.

A representative reactive agent contains four components: a set of detectors, a set of effectors, a set of stimulus-response rules, and a performance system. The agent perceives its environment through its detectors. An event in the environment causes the detectors to generate a message that are routed to all the rules. The rules in turn may or may not generate a

message. The rule generated messages constitute instructions to the effectors to take a specific action in response to the event in the environment.

2.2. Genetic Algorithms

Genetic algorithms (GAs) are stochastic adaptive algorithms whose search method is based on simulation of natural genetic inheritance and Darwinian striving for survival. Mathematical arguments show that GAs bring substantial computational leverage to search problems, without requiring the mathematical characteristics often necessary for traditional optimization schemes. It has a great deal of potential in scientific and engineering optimization or search problems [4].

A simple genetic algorithm (SGA) can be defined by

$$GA ::= \langle C, E, P_0, M, \Phi, \Gamma, \Psi, \Pi, T \rangle$$

where C is chromosome encoding, E is fitness evaluation, P_0 is initial population, M is population size, Φ is selection operator, Γ is crossover operator, Ψ is mutation operator, Π is replacement operator and T is termination conditions.

An important issue in designing a SGA is the procedure used to control the genetic operations. The generational procedure is shown below:

```
SGA( )
{ initialize population P0 ;
  P = P0 ;
  repeat
{  Φ(two parents p1 and p2 from P) ;
    offspring = Γ( p1, p2) ;
    Ψ(offspring) ;
    Π(P,offspring) ;
  } until ( T );
}
```

By giving more chances for the better elements to have offspring in the next generation, the GA facilitates an evolutionary process in which elements in a population progressively improve over time.

2.3. Learning classifier systems

GAs-based learning classifier systems (LCSs) belong to machine learning techniques. In fact, they are rule-found systems, which learn and acquire new rules by interacting with an environment. The rules are usually in the traditional production system form of "IF state THEN action". And one classifier is equivalent to a rule.

LCSs are composed of four components[5]: (1) a classifier list, which is a condition-action rule base that represents the current knowledge of the system; (2) the performance component, which governs the interaction with the environment; (3) the reinforcement component (also called credit assignment component), which distributes the reward received from the environment to the classifiers accountable for the rewards obtained; (4) the discovery component, which is responsible for discovering better rules and improving existing ones.

LCSs learn by trying to maximize the amount of reward which is a feedback from an environment. This learning process is shown as Fig. 1. The performance component of LCSs interacts with the environment. Depending on the consequences of interaction, the reinforcement components update the classifier's fitness. In the end of each learning, the discovery component selects batter classifiers from the classifiers list with probability prosectional to their fitnesses, and produce new classifiers by using these classifiers to replace worse classifiers. At present, LCSs have been quite successfully applied in a wide variety of domains [6] (e.g., autonomous robotics, knowledge discovery, and computational economics).

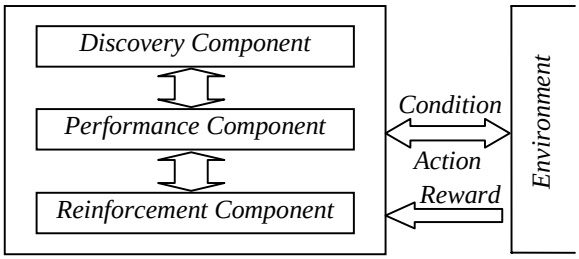


Fig. 1: Learning Classifier Systems.

3. Learning behavior model of CGF

3.1. Framework of agent-based CGF

According to reactive agent model framework, we build an agent-based CGF model shown as Fig. 2. The model is simple because it is mainly used to research CGF learning ability.

CGF accomplish the interactive with the environment through its detectors and effectors to change the state of the environment. Let $A \otimes B$ denotes update operation of B based on A. The transformation process of the environment, form t state to $t+1$ state, can be described as the following equation:

$$\varepsilon(t+1) = \beta(t) \otimes \varepsilon(t) \qquad (1)$$

where $\varepsilon(t)$ is the state of the environment perceived by CGF though detectors at t, $\beta(t)$ is the action executed by CGF though effectors at t. So reward α is defined by:

$$\alpha(t+1) = f(\varepsilon(t+1) - \varepsilon(t)) \qquad (2)$$

where $\varepsilon(t+1) - \varepsilon(t)$ is the transformation of the environment. According to the learning algorithm and the reward α, the learning model updates rules in the rule-base, acquire new better rules and delete worse rules.

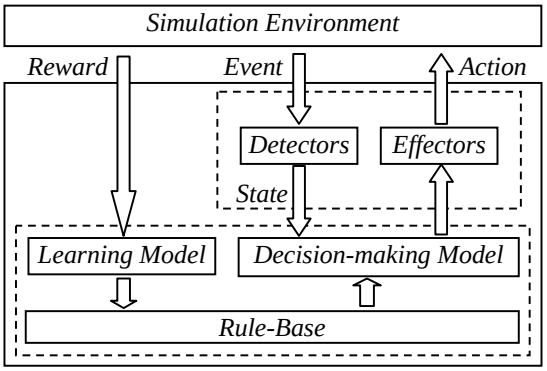


Fig. 2: Agent-based Framework of CGF.

3.2. Decision-making Model

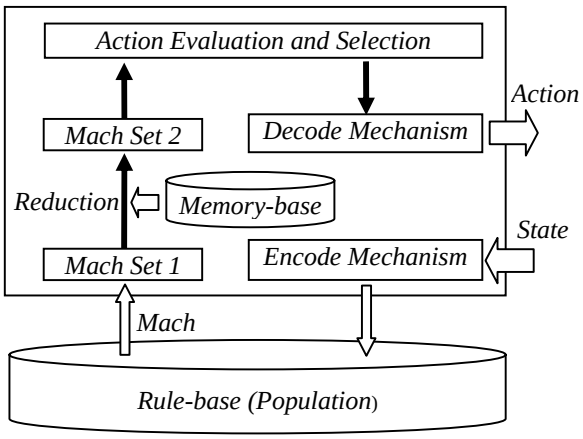


Fig. 3: Decision-making Model.

Fig. 3 describes the decision-making model based on rules. The decision-making model employs the production rule form which is composed of two sections: condition section and action section. The rule condition and action are strings of characters from the ternary alphabet $\{0,1,\#\}$. The # alphabet acts as a wildcard allowing generalization. The string #10#011 is a rule whose front four bits are condition section and back three bits is action section. In other words, the string #10#011 is a rule that says "IF in state 0100 OR state 0101 OR state 1100 OR state 1101, THEN take action 011". Decision-making model represents the condition as a finite-length string by using encode mechanism and transforms string into countermeasures by using decode mechanism.

Memory is the base of intelligence, and an intelligent man does not make the same mistake for several times. Acting as intelligent simulation entities, CGF should have memory ability while learning. Therefore, we add memory-base to Decision-making model. CGF place wrong rules into memory-base and don't adopt them in the latter learning process. This memory function is also accelerate GAs evolution because the wrong rules haven't the chance to be activated.

3.3. Learning Model

Learning model adopt GAs-based reinforcement, whose structure is shown in Fig.4.

The rule base, which is equivalent to the population of GAs, consists of N condition-action rules. Accordingly, one rule is an individual, and N is the population size. A fitness to indicate the "usefulness" of the rule can be considered as an individual fitness. Based on the reward, individual fitness function in Learning model is used to adjust the fitness of rules in the rule-base. Learning model then selects randomly parent rules from the rule-base with probability prosectional to their fitnesses. Offspring are produced via mutation and crossover from the parents in the usual way and replace the rule whose fitness is less in rule-base.

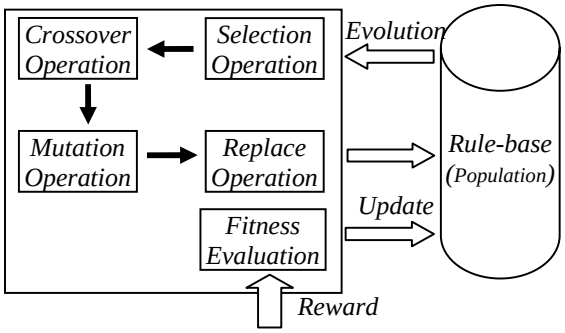


Fig. 4: Learning Model.

3.4. Learning Process

The general operational cycle for CGF learning is described as follows:

Step 1: GAs initialize the rule-base;

Step 2: Initializations of the simulation system, including simulation environment, CGF state and simulation result;

Step 3: The detectors perceive and send the current state of the environment to the decision-making model;

Step 4: Decision-making model determines the rules whose condition sections are adequately matched by the current environment state, and builds a match set containing the rules that don't appear in the memory-base;

Step 5: Decision-making model evaluates the utility of the action section of the rule in the match set. According to the evaluation results, the system selects a rule and sent its action section to the effectors to be performed. And this rule is saved at the same time;

Step 6: The effectors perform the indicated action, usually altering the state of the environment;

Step 7: If termination condition of simulation is satisfied, it turns to Step 8. On the contrary, it turns to Step 3;

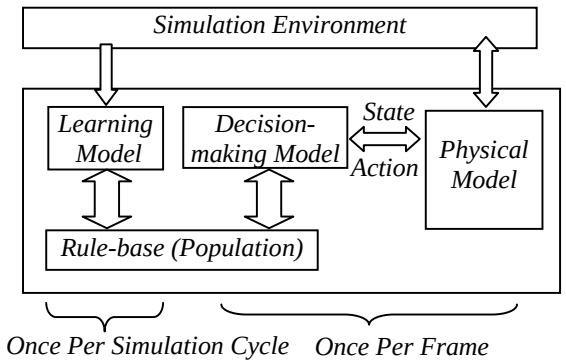
Step 8: According to simulation results, reward α is formed. Learning model uses individual fitness function and reward α to update the fitness of each rule in the rule-base;

Step 9: The GAs employed selection, crossover and mutation operators to generate a new rule-base;

Step 10: Learning model checks the terminate rule of learning process. If it is satisfied, it stops. On the contrary, it turns to Step 2.

On the basis of the above learning cycle, it's obvious that CGF entity perceives the environment state and changes the environment though the action according to the tactic provided by the decision-making model at each step in the simulation. However, Learning model runs only once at the end of each simulation iteration cycle as show in Fig. 5. Here, Physical model, which perceives the environment state, affects and changes the environment, includes dynamics and kinematics model, sensor model and weapon model of CGF.

Fig. 5: Learning Process.



4. Experimental framework and simulation studies

In order to validate the validity of the model built by us, we design a simple visible validation system. The system simulates engagement between two tanks. The tank that employs the rules produced by the learning model is student tank, and the tank that employs the rules prescribed beforehand is teacher tank. At the end of simulation, the results are used by the learning model to produce the next generation of rules. The learning process repeats until student tank win continuously for three times.

4.1. Experimental framework

The settings of the system are described as follows:

1) Rule: The rule population for each generation is generated by a simple GA. Each rule has a fitness assigned by the individual fitness function. It is 9 characters long, with 5 left-hand side and 4 right-hand side. The alphabet consists of $\{1,0\}$. The encoding rules of simulation environment state and action are shown respectively in Table 1 and Table 2, where L is the distance between two tanks, $R_{\max}$ is the maximum cannon-shot and $R_{\min}$ is the minimum cannon-shot. Let l represent the line connecting centroids of two tanks. α in Table 1 is the angle between direction of advance and line l, and β is the angle between direction of muzzle and line l.

code	0	1
1st digit	$L < R_{\max}$	$L > R_{\max}$
2nd digit	$L < R_{\min}$	$L > R_{\min}$
3rd digit	$\alpha \le 0$	$\alpha > 0$
4th digit	$\beta < 0$	$\beta > 0$
5th digit	$\beta \ne 0$	$\beta = 0$

Table 1: Encoding of simulation environment state.

code	0	1
1st digit	stop	advance
2nd digit	no fire	fire
3rd digit	muzzle turn right	muzzle turn left
4th digit	tank turn right	tank turn left

Table 2: Encoding of action.

2) GAs: The GA in this system employs prosectional selection, single-point crossover, and simple mutation. Population size $M = 100$, crossover rate $P_c = 0.9$, mutation rate $P_m = 0.05$. Based on the fitness, new rules are arranged in descending order, and rules of the parent population are also arranged in

descending order. The latter n rules in the parent population are replaced with the first n new rules, where $n = M \times P_u$, P_u is replacement ratio and $P_u = 0.5$.

3) Environment Simulation: The visualized simulation learning environment is realized by VC++6.0. A one-one tank engagement is generated by the environment to evaluate the rules in the rule-base.

4) Decision-making Model: At each time step in simulation, the rules whose condition sections are completely matched with the current environment state are triggered. What's more, these rules could not appear in the memory-base. If more than one rule is triggered at the same time, the rule with the highest fitness will be activated. But if there is no triggered rules, student tank employs the default action. The trigger/activation process continues until the engagement is terminated, and all activated rules will be saved in an activation list.

5) Individual Fitness Function: The function uses the engagement score and the activation list to assign fitness for each individual rule, which insures that the information from previous generations can be inherited to the next generation.

The function was composed of two steps:

Step 1: All activated rules are directly assigned the score of the latest completed engagement by using the following equations:

$$F_i = F_{i-1} + n, \quad \text{if the student tank wins}$$

$$F_i = F_{i-1} - n/2, \quad \text{if the student tank loses}$$

where F_i is individual fitness at the end of simulation i , F_{i-1} is individual fitness at the end of simulation $i-1$, n is the number that individual is activated in the simulation i .

Step 2: In the selection operation, the fitness of the offspring can be evaluated via the following fitness function:

$$F_o = 1/3F_p$$

where F_o is the fitness of the offspring, F_p is the fitness of the parent.

4.2. Experimental Results

In order to reduce variation, each experiment is repeated at least 10 times and the results are the average one. Assume the following notations: N is the number of experiment, M is the population size, $i \in (1,2,\cdots,N)$, $j \in (1,2,\cdots,M)$, $\mathrm{P_i}$ is the population at end of the experiment i , $\mathrm{X_{ij}}$ is the individual in $\mathrm{P_i}$, $\mathrm{F_{ij}}$ is the fitness of $\mathrm{X_{ij}}$.

Step 1: The same individuals in the each population $\mathrm{P_i}$ are merged and sum up their fitness. After the merge, the population size is possibly different, but M is still the population size to simplify the description.

Step 2: The fitness of the same individual among populations is not same for engagement number of each experiment is possibly different. In order to eliminate this abnormity, individual relative fitness $\mathrm{F_{R_{ij}}}$ is calculated as follows:

$$\mathrm{F_{R_{ij}}} = \mathrm{F_{ij}} \Big/ \sum_{\mathrm{j=1}}^{M} \mathrm{F_{ij}} \tag{3}$$

Step 3: The same individuals in the N populations are merged and sum up their fitness. After the merge, N populations become one population whose size is H . Based on the fitness, all individuals are arranged in descending order.

Step 4: The individuals whose relative fitness is small are not be applied in the environment. To avoid influence the analysis of experimental results, the first K $(\mathrm{K} < H)$ individuals are employed.

Assume the relative fitness of individual $\mathrm{X_p}$ is F_{R_p} and environment code is $(x_{p_1}, x_{p_2}, \cdots, x_{p_l})$. If action section of this individual include W , where W is fire or advance, f_{i1} and f_{i2} is calculated as follow:

If $\mathrm{x}_{p_i} = 1$ $(i = 1,2,\cdots,l)$, $f_{i1} = f_{i1} + F_{R_p}$

If $\mathrm{x}_{p_i} = 0$ $(i = 1,2,\cdots,l)$, $f_{i2} = f_{i2} + F_{R_p}$

After processing all K individuals, matrix $\begin{pmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \\ \vdots & \vdots \\ f_{l1} & f_{l2} \end{pmatrix}$ is acquired. If $f_{i1} > 2 f_{i2}$, $x_i = 1$. If $f_{i2} > 2 f_{i1}$, $x_i = 0$. If $f_{i1} \le 2 f_{i2}$ and $f_{i2} \le 2 f_{i1}$, x_i =# .Where $(x_1, x_2, \cdots, x_l)$ is corresponding environment code of W .

In order to simplify the experiment, we only study two actions "advance" and "fire". The move direction of tank and the turn direction of muzzle are controlled by module, so tank will auto turn to the opponent. The simulation experiment runs for ten times. The experiment results are analyzed by using method presented in the paper. Table 3 and 4 are analysis results. Obviously, we get the two rules "010010100" and "110001000" that are just employed by teacher tank. So, student tank has learned two actions "advance" and "fire".

Environment code	relative fitness		Adoption value
	f_{i1}	f_{i2}	
1st digit	0.325319	0.731702	0
2nd digit	0.826195	0.230825	1
3rd digit	0	1.05702	0
4th digit	0	1.05702	0
5th digit	0.731702	0.325319	1

Table 3: Environment code of “fire”.

Environment code	relative fitness		Adoption value
	f_{i1}	f_{i2}	
1st digit	1.15	0.173111	1
2nd digit	1.15	0.173111	1
3rd digit	0	1.32311	0
4th digit	0	1.32311	0
5th digit	0.173111	1.15	0

Table 4: Environment code of “advance”.

5. Conclusions

A learning behavioral model framework based on GAs of CGF is proposed in this paper. The simulation results obtained from a visible validation system reveals that the model is available and feasible. Our contributions can be summarized as follows: GAs-based CLSs technology is first applied to agent-based learning behavioral modeling. This will contribute to the performance improvements of learning ability of intelligent entities in CGF. Another advantage offered by the model is the memory function, which is proven to be useful in the acceleration of GAs convergence.

In the future study, it would be worthwhile improving the coding form to describe the CGF environment condition better and speeding up the convergence rate of GAs.

Acknowledgement

This work is partially supported by National Nature Science Foundation of China (Grant No. 70471089).

References

[1] Mikel D. Petty, Do we really want computer generated forces that learn?. *Proceedings of the Tenth Conference on Computer Generated Force*, pp.125-130, 2001.

[2] R.E. Smith, B.A. Dike, R.K. Mehra, Classifier systems in combat: two-sided learning of maneuvers for advanced fighter aircraft. *Comput. Methods Appl. Mech. Engrg*, 186: 421-437, 2000.

[3] Wooldridge M, *An Introduction to MultiAgent Systems*, John Willey & Sons, New York, 2002.

[4] M. Zhou, *Genetic Algorithms: Theory and Applications*, National Defence Industry Press, Beijing, 1999. (in Chinese)

[5] John H. Holmes and Pier Luca Lanzi, Learning classifier systems: New models, successful applications. *Information Processing Letters*, 82: 23-30, 2002.

[6] L. Bull, *Applications of Learning Classifier Systems*, Springer, 2004.