Journal of Intelligent Learning Systems and Applications, **2012, 4, 176-187**
http://dx.doi.org/10.4236/jilsa.2012.43018 Published Online August 2012 (http://www.SciRP.org/journal/jilsa)

A Competitive Markov Approach to the Optimal Combat Strategies of On-Line Action Role-Playing Game Using Evolutionary Algorithms

Haoyang Chen*, **Yasukuni Mori, Ikuo Matsuba**

Graduate School of Advanced Integration Science, Chiba University, Chiba, Japan.
Email: *chenhaoyang@graduate.chiba-u.jp

Received March 1st, 2012; revised June 25th, 2012; accepted July 3rd, 2012

ABSTRACT

In the case of on-line action role-playing game, the combat strategies can be divided into three distinct classes, Strategy of Motion(SM), Strategy of Attacking Occasion (SAO) and Strategy of Using Skill (SUS). In this paper, we analyze such strategies of a basic game model in which the combat is modeled by the discrete competitive Markov decision process. By introducing the chase model and the combat assistant technology, we identify the optimal SM and the optimal SAO, successfully. Also, we propose an evolutionary framework, including integration with competitive coevolution and cooperative coevolution, to search the optimal SUS pair which is regarded as the Nash equilibrium point of the strategy space. Moreover, some experiments are made to demonstrate that the proposed framework has the ability to find the optimal SUS pair. Furthermore, from the results, it is shown that using cooperative coevolutionary algorithm is much more efficient than using simple evolutionary algorithm.

Keywords: Game Design; Game Balance; Competitive Markov Decision Process; Cooperative Coevolutionary Algorithm; Competitive Coevolution

1. Introduction

In recent years, on-line Action Role-Playing Games (ARPGs) become more and more popular all over the world. An on-line ARPG is a virtual world that consists of several distinct races. Player first creates a character of any race, then plays the game by both exploring the virtual world and fighting with others. During gaming, player has direct control over the created character. Usually, the on-line ARPG is regarded as an extension of the off-line one by the reason that it allows players to fight with each other besides AI opponent. Such new feature results in more complexity of the game balance.

What is the game balance? In the off-line ARPG, game balance means the difficulty control of the game and can be reached simply by adjusting the power of AI opponent. On the other hand, however, in the on-line ARPG, game balance mainly refers to power balancing among the races. Currently, on-line ARPGs are balanced by hand tuning, but this approach presents several problems. Human players are expensive in both time and resources, and even human players can not explore all strategies to find out whether a dominate one exists [1]. Moreover, since strengthening one race will definitely weaken the others, the result of tuning operations may, somehow, become worse. It is hard to control. Hence, a more theoretical and efficient design method is needed.

Chen, *et al.* [2,3] have proposed an evolutionary design method for both turn-based and action-based on-line role-playing games. In such approaches, they, successfully, constructed an automated testing framework to verify whether the game world is well-balanced. However, they just investigated the case in which each race has only one skill. The situation of having multi-skills was ignored because they failed to retrieve the optimal Strategy of Using Skill (SUS), which plays a crucial role in the automated testing framework.

In this paper, we propose an evolutionary framework, including integration with competitive coevolution and cooperative coevolution, to search the optimal SUS pair of a basic action-based game model, where the combat is modeled by the Discrete Competitive Markov Decision Process (DCMDP) [4]. Also, by introducing the Chase Model and the Combat Assistant Technology (CAT), we analyze the optimal Strategy of Motion (SM) and the optimal Strategy of Attacking Occasion (SAO) of the game model.

The paper is organized as follows. In Section 2, a brief

*Corresponding author.

description of DCMDP and a theorem are given. The details of Cooperative Coevolutionary Algorithm (CCEA) technology are explained in Section 3. Section 4 presents a basic action-based game model and its optimal SM as well as SAO. Section 5 describes the proposed competetive framework. Experimental results are reported in Section 6. The conclusions and possible future research directions are given in Section 7.

2. Discrete Competitive Markov Decision Process

A DCMDP is a multi-player dynamic system that evolves along discrete time points. At each time point t, the state of the system, denoted by S_t, is a random variable that can take on values from the finite set $S = 1, 2, \cdots, N$. At these discrete time points, called stages, both players have the possibility to influence the course of the system. In this paper, we only consider the case of two players, and we associate two finite action sets $A_1(s) = \{1, 2, \cdots, m_1(s)\}$ for player 1 and $A_2(s) = \{1, 2, \cdots, m_2(s)\}$ for player 2, then at any stage, the system is one of the states and both players are allowed to choose an action out of their respective action sets independently of one another. If in a state s, at some decision moment, player 1 chooses $a^1 \in A_1(s)$ and player 2 chooses $a^2 \in A_2(s)$, then two things happen:

1) Player 1 earns the immediate reward $r^1(s, a^1, a^2)$ and player 2 earns $r^2(s, a^1, a^2)$.

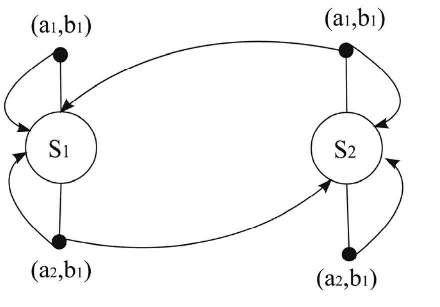
2) The dynamic of the system is influenced. The state at the next decision moment is determined in a stochastic sense by a transition vector which is denoted as:

$$p\left(s,a^1,a^2\right)=\left(p\left(1\middle|s,a^1,a^2\right),\cdots,p\left(N\middle|s,a^1,a^2\right)\right)$$

a simple example of the DCMDP is shown in **Figure 1**.

2.1. Classes of Strategy

Strategies for players are rules that tell them what action to choose in any situation. The choice at a certain decision moment may depend upon the history of the play up to that moment. Furthermore, as is usual in game theory,



A1(S1)={a1, a2}
A1(S2)={a1, a2}
A2(S1)={b1}
A2(S2)={b1}

Figure 1. A discrete competitive markov decision process with two states.

the choice of an action may occur in randomized way, that is, the players can specify a probability vector over their action spaces and next the action is the result of a chance experiment according to this probability vector.

There are three classes of strategy exist, namely, the behavior strategy, the Markov strategy, and the stationary strategy.

Behavior strategy, denoted by F_B, being the most general type of strategy, can be represented by a sequence $\pi = (f(0), f(1), f(2), \cdots)$, where, for each $t = 0, 1, 2, \cdots$, the decision rule $f_t = (f_t(1), f_t(2), \cdots, f_t(N))$ specifies for each state s a probability vector $f_t(s)$ on $A(s)$ as a function of history of the game up to decision moment t.

A Markov strategy, denoted by F_M, is a behavior strategy where, for every $t = 0, 1, 2, \cdots$, the decision rule f_t is completely determined by the decision moment t and the current state s_t at moment t.

A stationary strategy is a Markov strategy where, for every $t = 0, 1, 2, \cdots$, the decision rule f_t is completely determined by the current state s_t at moment t. Thus, a stationary strategy can be represented by a sequence $\pi = (f, f, f, \cdots)$, where $f = (f(1), f(2), \cdots, f(N))$ specifies for each state $s \in S$ a probability vector $f(s)$ on $A(s)$. We will denote such stationary strategy by F_S. If the players use $\pi^1 \in F_S^1$, $\pi^2 \in F_S^2$ as their strategy respectively, there exist stationary transition probabilities:

$$\begin{aligned} p\left(s'\middle|s,\pi^1,\pi^2\right) &= P\left\{S_{t+1}=s'\middle|S_t=s,\pi^1,\pi^2\right\} \\ &= \sum_{a_1}^{m_1(s)}\sum_{a_2}^{m_2(s)} p\left(s'\middle|s,a_1,a_2\right)\pi^1\left(s,a_1\right)\pi^2\left(s,a_2\right) \end{aligned} \quad (1)$$

for all $t = 0, 1, 2, \cdots$, and Formula 1 is called Stationary Markov Transition Property.

2.2. *β*-Discounted Competitive Markov Decision Model

The infinite stream of rewards that results during a particular implementation of a strategy pair $\left(\pi^1,\pi^2\right)\in F_S^1\times F_S^2$ need to be evaluated in some manner. So, the β-Discounted Competitive Markov Decision Model, denoted by Γ_β, is introduced.

We have R_t^k denoting the reward at time t to player k, as well as

$$r^k\left(\pi^1\middle|\pi^2\right)=\left(r^k\left(1,\pi^1,\pi^2\right),\cdots,r^k\left(N,\pi^1,\pi^2\right)\right)^T$$

denoting the immediate excepted reward vector to player k corresponding to a strategy pair (π^1, π^2) mentioned above, where $k = 1, 2$, and $r^k(s, \pi^1, \pi^2)$, for each $s \in S$, can be calculated by the following formula:

$$r^k\left(s,\pi^1,\pi^2\right)=\sum_{a_1}^{m_1(s)}\sum_{a_2}^{m_2(s)} r^k\left(s,a_1,a_2\right)\pi^1\left(s,a_1\right)\pi^2\left(s,a_2\right)$$

The expected reward at stage t to player k resulting from (π^1, π^2) and an initial state s is denoted by:

$$E_{s\pi^1\pi^2}\left(R_t^k\right)=\left[P^t\left(\pi^1,\pi^2\right),r^k\left(\pi^1,\pi^2\right)\right]_s$$

where $[u]_s$ denotes the sth entry of a vector u, $P^t(\pi^1, \pi^2)$ is the t-step Transition Probability Matrix (TPM). Consequently, the sth entry of the overall discounted value vector of a strategy pair (π^1, π^2) to player k will be given by:

$$v_\beta^k\left(s,\pi^1,\pi^2\right)=\sum_{t=0}^{\infty}\beta^t E_{s\pi^1\pi^2}\left(R_t^k\right) \qquad (2)$$

where $\beta\in[0,1)$.

Theorem 1

Let $\left(\pi^{1*},\pi^{2*}\right)\in F_S^1F_S^1$ be a optimal strategy pair of Γ_β, then (π^{1*}, π^{2*}) is optimal in the entire class of behavior strategies. That is:

$$v_\beta^1\left(\pi^1,\pi^{2*}\right)\le v_\beta^1\left(\pi^{1*},\pi^{2*}\right)$$

and

$$v_\beta^2\left(\pi^{1*},\pi^2\right)\le v_\beta^2\left(\pi^{1*},\pi^{2*}\right)$$

for all $\pi^1\in F_B^1,\pi^2\in F_B^2$.

Proof:

Let either π^{1*} or π^{2*} be fixed, then the Γ_β will reduce to the discounted Markov decision model which has only one player. Further, for the discounted Markov decision model, it is well-known that the optimal stationary strategy is optimal in the entire class of behavior strategies [4]. This completes the proof.

In the game theory, we also call that $\left(\pi^{1*},\pi^2\right)\in F_S^1\times F_S^2$ is a Nash Equilibrium Point (EP) of the space $F_B^1\times F_B^2$. This theorem is vary important because it suggests that we can retrieve the EP of $v_\beta^k\left(s,\pi^1,\pi^2\right)$ with $\left(\pi^1,\pi^2\right)\in F_B^1\times F_B^2$ by just searching the space of $F_S^1\times F_S^2$.

2.3. Zero-Sum Γ_β

A Γ_β will be called sum-zero if

$$r^1\left(s,a_1,a_2\right)+r^2\left(s,a_1,a_2\right)=0 \qquad (3)$$

for all $s\in S$, $a_1\in A_1(s)$, $a_2\in A_2(s)$. Thus we may drop the superscript k by defining:

$$r\left(s,a_1,a_2\right)=r^1\left(s,a_1,a_2\right)=-r^2\left(s,a_1,a_2\right)$$

so, a extension of this definition lead to the following:

$$v_\beta\left(\pi^1,\pi^2\right)=v_\beta^1\left(\pi^1,\pi^2\right)=-v_\beta^2\left(\pi^1,\pi^2\right)$$

for all $\left(\pi^1,\pi^2\right)\in F_B^1\times F_B^2$.

Hence, in the case of zero-sum Γ_β, the two sets of inequalities defining an EP reduce to the single set of saddle-point inequality as follow:

$$v_\beta\left(\pi^1,\pi^{2*}\right)\le v_\beta\left(\pi^{1*},\pi^{2*}\right)\le v_\beta\left(\pi^{1*},\pi^2\right) \qquad (4)$$

Formula 4 leads to an important property, that is, if $\left(\theta^{1*},\theta^{2*}\right)\in F_B^1\times F_B^2$ is another pair of optimal strategies, then we have the following equation:

$$v_\beta\left(\theta^{1*},\theta^{2*}\right)=v_\beta\left(\pi^{1*},\pi^{2*}\right) \qquad (5)$$

this simply means that in the case of zero-sum Γ_β, the overall discounted value vectors of all optimal strategy pair coincide and can be denoted by:

$$v_\beta=\left(v_\beta(1),v_\beta(2),\cdots,v_\beta(N)\right)^T$$

Thus, we can retrieve v_β by searching the space $F_S^1\times F_S^2$ instead of $F_B^1\times F_B^2$, and the result follows from Theorem 1 and Formula 5.

3. Cooperative Coevolutionary Algorithm

CCEAs [5,6] have been applied to solve large and complex problems, such as multiagent systems [7-9], rule learning [10,11], fuzzy modeling [12], and neural network training [13]. It models an ecosystem which consists of two or more species. Mating restrictions are enforced simply by evolving the species in separate populations which interact with one another within a shared domain model and have a cooperative relationship. The original architecture of the CCEA for optimization can be summarized as follows:

1) Problem Decomposition: Decompose the target problem into smaller subcomponents and assign each of the subcomponents to a population;

2) Subcomponents Interaction: Combine the individual of a particular population with representatives selected from others to form a complete solution, then evaluate the solution and attribute the score to the individual for fitness;

3) Subcomponent Optimization: Evolve each population separately by using a different evolutionary algorithm, in turn.

The empirical analyses have shown that the power of CCEAs depends on the decomposition work as well as separate evolving of these populations resulting in significant speedups over Simple Evolutionary Algorithm (SEA) [14-16]. Here, we give the theoretical evidence of such results with the following two assumptions.

1) The elitists of CCEA populations are chosen as the representatives;

2) There are no variational operators in both the SEA and CCEA.

Let's begin with some definitions.

Definition 1. Given schemata: H_1, H_2, $\cdots$, H_n where H_i denotes a schema of the ith CCEA population, the n-expanded schema, denoted by H_1, is the sequential concatenation of the n schemata. For example, let H_1 = [1*0*], H_1 = [*1*1], then $H_1^2=[1*0**1*1]$.

Definition 2. Let there be n populations in the CCEA. A complete genotype is the sequential concatenation of n individuals selected from n different populations. If all the individuals are representatives, then the complete genotype is the best one.

Definition 3. Given an individual I of the ith CCEA population, the expanded genotype of I is the best complete genotype in which the ith representative is replaced by I.

Definition 4. Given a target problem J, the two algorithms, SEA and CCEA, are comparable if the population of the SEA consists of all the expanded genotypes in the CCEA.

Theorem 2

Let a target problem f be decomposed into n subcomponents, r_i be the increasing rate of the individuals matching H_i in the ith CCEA population, r_{ccEA} be the increasing rate of the complete genotypes matching H_1^n in the CCEA, and r_{SEA} be the increasing rate of the individuals matching H_1^n in the SEA. If the two algorithms are comparable, then,

1) $r_{SEA} < \sum_{i=1}^{n} r_i$

2) $r_{CCEA} = k \cdot \left(\min\{r_1, r_2, \cdots, r_n\}\right)^n, k \geq 1$

Proof:

Since the two algorithms are comparable, in the SEA, the number of individuals matching H_1^n at generation t can be calculated by:

$$m\left(H_1^n, t\right) = \sum_{i=1}^{n} M\left(H_i, t\right)$$

where $M(H_i, t)$ denotes the number of individuals matching H_i at generation t, in the ith CCEA population. Then according to the schema theorem (refer to Thm. 1), we have,

$$E\left[m\left(H_1^n, t+1\right)\right] \leq m\left(H_1^n, t\right) \cdot \frac{\sum_{i=1}^{n} \overline{f\left(H_i, t\right)}}{\sum_{i=1}^{n} \overline{F\left(i, t\right)}}$$
$$< m\left(H_1^n, t\right) \cdot \sum_{i=1}^{n} r_i$$

where in the ith CCEA population, $\overline{f\left(H_i, t\right)}$ and $\overline{F\left(i, t\right)}$ denote the mean fitness of individuals matching H_i and the mean fitness of all individuals, at generation t, respectively.

In the case of CCEA, because H_1^n is the conjunction of H_i, the number of the complete genotypes matching H_1^n at generation t is given by:

$$M\left(H_1^n, t\right) = \prod_{i=1}^{n} M\left(H_i, t\right)$$

Again, according to the schema theorem, we obtain the following equation:

$$E\left[M\left(H_i, t+1\right)\right] = M\left(H_i, t\right) \cdot r_i$$

Then,

$$E\left[M\left(H_1^n, t+1\right)\right] = \prod_{i=1}^{n} E\left[M\left(H_i, t+1\right)\right]$$
$$= \prod_{i=1}^{n} M\left(H_i, t\right) \cdot r_i$$
$$= M\left(H_1^n, t\right) \cdot k \cdot \left(\min\{r_1, \cdots, r_n\}\right)^n$$

where $k = \prod_{i=1}^{n} \frac{r_i}{\min\{r_1, \cdots, r_n\}} \geq 1$.

Hence, obviously, with the increasing of the min $\{r_1, \cdots, r_n\}$, H_1^n will receive a much higher increasing rate in the CCEA.

However, Theorem 2 does not mean that CCEAs are superior to SEAs, which depends on the target problems. Actually, since the representatives are necessary in calculating the fitness of the individual of an arbitrary population, the relationships between populations impose a great influence on the efficiency of CCEAs [17,18]. It has been proved that even with prefect information, infinite population size and no variational operators, CCEAs can be expected to converge to suboptimal solution [18], while SEAs do not suffer from such affliction [19-21]. However, Liviu [22] has emphasized that CCEAs will settle in the globally optimal solution with arbitrarily high probability, when properly set and if given enough resources.

4. Game Model

In this section, we construct a two-dimensional ARPG model in which three distinct races (A, B and C) are designed with maximum level of L. Each race has seven properties, which are Health (*H*), Dodge Rate (*DR*), Skill Damage (*SD*), Skill Critical Hit Rate (*SCHR*), Skill Cool Down Time (*SCDT*), Skill Range (*SR*) and Velocity (*V*), as well as two skills. Except *H* and *SD* which are monotone increasing functions of $l = \{1, 2, \cdots, L-1, L\}$ he other properties are designed to be constant. SCHR denotes the probability that player outputs the double damage. *SCDT*, belonging to the time feature, means that how much time should the character rest after an attack, and *SR*, a kind of space feature, denotes the range in which the skill can be used. Specially, in the game world, Velocity is measured in pixels per frame instead of miles per second.

In order to evaluate the power of the race, a standard square map with width w is introduced, and all battles will be held in it. In this case, we only concern the "1 versus 1" fighting type which contains two players (the defender and the attacker), because it is the core of the combat system of on-line ARPG, and we design the low-velocity race to play the role of defender, because if a high-velocity defender chose the strategy of keeping escaping, the battle will become meaningless and uninteresting. As a principle, after being attacked, the defender should be able to launch at least one attack. And there is no constrain on battle time, that is, players are allowed to fight as long as they like. The basic design contents are as follows:

1) The width of the standard map: $w = 400$;

2) Health: $H_A(l) > H_B(l) > H_C(l)$;

3) Skill Damage:

$$SD_A^{\max}(l) > SD_B^{\max}(l) > SD_C^{\max}(l);$$

4) Dodge Rate: $DR_C > DR_B > DR_A$;

5) *SCDT*: $SCDT_C > SCDT_B > SCDT_A$;

6) Skill Range: $SR_C > SR_B > SR_A$;

7) Velocity: $V_C > V_B > V_A$.

where $SD_k^{\max}(l)$ denotes the maximum damage among two skills of race k with level l.

Based on these ideas, we model the motion of the players by using the Chase Model (refer to **Figure 2**), and by the theory of vector differential, we acquire the following formula:

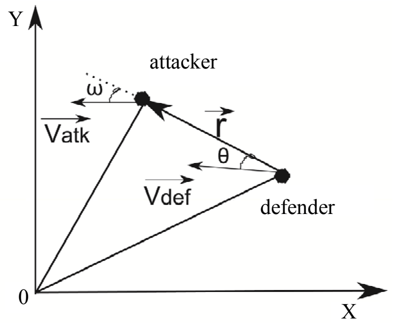


Figure 2. The chase model.

$$\frac{d|\vec{r}|}{dt} = \left|\overrightarrow{v_{atk}}\right|\cos\omega - \left|\overrightarrow{v_{def}}\right|\cos\theta \quad (6)$$

where $\vec{r}$ is the distance vector of the two players, $\overrightarrow{v_{atk}}$ and $\overrightarrow{v_{def}}$ denote the velocity of attacker and defender.

According to Formula 6, the attacker should maximize $d|\vec{r}|$ by maximizing cosω and the defender should minimize $d|\vec{r}|$ by maximizing cosθ. These can be recognized as the optimal SM.

Before analyzing the optimal SAO, we introduce the Combat Assistant Technology (CAT) which has been applied to most of on-line ARPGs (such as JXOnline, World of Warcraft, etc.). It helps players to run towards the opponent automatically and attack if possible. The optimal SAO will be analyzed based on it.

Figure 3 demonstrates a freeze-frame of the attacking process in which each combatant uses the CAT to attack. For the defender, after the attacker launched an attack at point f, the defender will strikes back at a point within (a, b], which enables he(she) to output the maximum damage because $|ab| = \left|\overrightarrow{v_{def}}\right|$. So using the CAT to attack is the optimal SAO of the defender. On the other hand, in fact, the attacker may launch the attack in the range (c, d] or (d, e] or (e, g] with the probability of P_1, P_2 and P_3 respectively. We define (P_1, P_2, P_3) as the Action Probability Vector (APV), which represents the action feature and can be obtained through statistical analyze. Therefore, the CAT with the APV of the expert-level players is the optimal SAO of the attacker.

Unlike the SM and SAO, an optimal SUS completely depends on the competent's SUS, which means that the optimal SUS pair is an EP. Also, each combat group has a limit state space which is defined as follows:

$$S = \left\{(0,1),(0,2)\cdots,\left(T_{def},T_{atk}-1\right),\left(T_{def},T_{atk}\right)\right\}$$

$$T_{def}(l) = \left\lceil \frac{H_{def}(l)}{SD_{atk}^{\min}(l)} \right\rceil \quad T_{atk}(l) = \left\lceil \frac{H_{atk}(l)}{SD_{def}^{\min}(l)} \right\rceil \quad (7)$$

where the (0, 0) state does not exist, T_{def} and T_{atk} are the maximum lifetime of defender and attacker, $H_k(l)$ denotes the health value of k with level l, and $SD_k^{\min}(l)$ is defined as the minimum damage among two skills of k with level l.

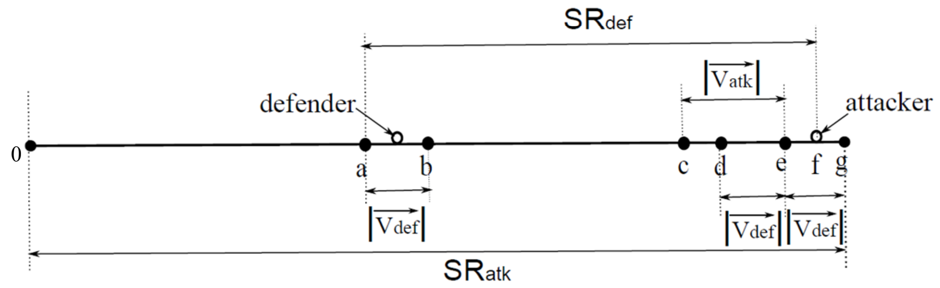


Figure 3. A freeze-frame of the attacking process with the combat assistant technology.

5. Competitive Coevolutionary Framework

In this section, firstly, we model the combat process by using the zero-sum Γ_β process, in this step, the fitness function will be constructed. Next, we will introduce the competitive framework as well as demonstrate how to use it to find the optimal SUS pair.

As the preceding section stated, there is no time constrain during fighting, that is, the combat can be regarded as an infinite process, where players keep changing their state from one to another by their actions. And this infinite process can be evaluated by Γ_β.

It is natural for any player that he(she) wants to maximize his(her) win probability during battles. Thus, we use the increment of win probability from stage t to $t + 1$, as the expectation of immediate reward of the decision rule pair (f_t, g_t). Let $p^1(t)$, $p^2(t)$ be the win probabilities of player 1 and player 2 at stage t respectively, then, by the fact that the larger t becomes, the higher probability of game over will be, we obtain the follows:

$$\lim_{t\to\infty}\left[p^1(t)+p^2(t)\right]=1$$

from such formula, it can be inferred that an increment of $p^1(t)$ will bring a potential decrement to $p^2(t)$.

With these results, in this case, combat can be regarded as the zero-sum game, and can be modeled by zerosum Γ_β process. The immediate reward vector and the overall discounted value vector are defined as follows:

$$r\left(\pi^1,\pi^2\right)=\left(P^1\left(\pi^1,\pi^2\right)-I_N\right)Q$$

$$v_\beta\left(s,\pi^1,\pi^2\right)=\sum_{t=0}^{\infty}\beta^t\left[P^t\left(\pi^1,\pi^2\right)r\left(\pi^1,\pi^2\right)\right]_s \quad (8)$$

where

$$\pi^1=\left(f,f,f,\cdots,f,\cdots\right)$$

$$\pi^2=\left(g,g,g,\cdots,g,\cdots\right)$$

$$Q=\left(q_1,q_1,\cdots,q_N\right)^T$$

$$q_i=\begin{cases}1 & s_i\in S^T\\ 0 & s_i\notin S^T\end{cases} \quad (9)$$

Here, π^1, π^2 are the SUS of the defender and the attacker, N is the size of the state space, I_N is the identity matrix of size N, S^T denotes the absorbing state set in which the lifetime of attacker is 0. Particularly, in this case, the $v_\beta(s, \pi^1, \pi^2)$ will always converge into a certain value, even the discount factor, β, is set to 1.

Since the combat always starts from the state (T_{def}, T_{atk}), denoted by s_N, it is natural that we shall use $v_1(s_N, \pi^1, \pi^2)$, denoting the win probability of defender, as the fitness function of our evolutionary algorithm.

The competitive framework consists of two symmetrical blocks (refer to **Figure 4**), the left block denotes the evolutionary process of π^1 while the right block represents π^2. Both blocks use CCEA to search the solution space, that is, each decision rule, f or g, is divided into N groups which will be evaluated in turn. Such group, for example, group i, denotes the probability vector on actions of state i, that is, $f(i)$ or $g(i)$. Through the elite pair of current generation, (f^*, g^*), the two blocks of framework interact in a competitive way, and this enables the framework to lead such elite pair to converging in the decision rule pair of an EP.

The pseudo-code of the algorithm is shown as follows:

```
1) Create species $s_f^i$ for $f(i)$ and $s_g^i$ for $g(i)$, where $i = \{1, 2, \cdots, N\}$;
2) Initialize population for every species by using the uniform distribution, where each bit of chromosome has the same probability to be a 1;
3) Select a individual from each $f(i)$ and $g(i)$, randomly, to form the initial elite decision rule pair ($f^*$, $g^*$);
4) Set generation $t = 0$;
5) Set $i = 1$;
6) Evaluate the individuals of species $s_f^i$ with $g^*$ as well as $f^*(j)$, where $j \neq i$ (maximize $v_\beta$);
7) Replace the $f^*(i)$ by the current elite, if necessary;
8) Select the outstanding individuals, and construct the new probability distribution from them. Then new generation can be obtained by sampling this distribution and mutation;
```

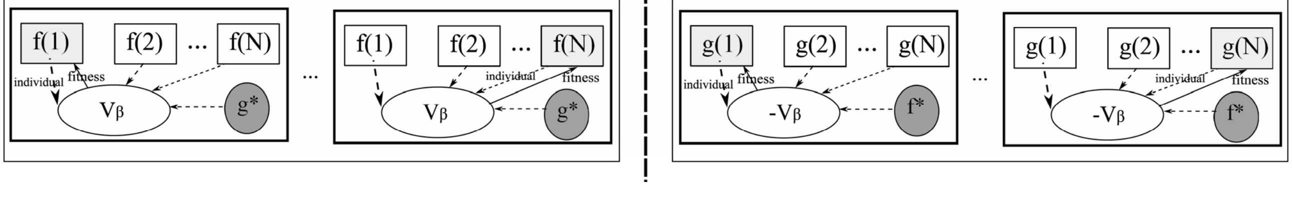

f* = (f*(1), f*(2), ... , f*(N))
Elite
f(1)
f(2)
f(N)
individual
fitness
Vβ
g*
g* = (g*(1), g*(2), ... , g*(N))
g(1)
g(2)
g(N)
-Vβ
f*

Figure 4. The competitive coevolutionary framework.

9) While ($i \le N$) set $i = i + 1$, goto 6;

10) Set $i = 1$;

11) Evaluate the individuals of species s_g^i with f^* as well as $g^*(j)$, where $j \ne i$ (minimize v_β);

12) Replace the $g^*(j)$ by the current elite, if necessary;

13) Select the outstanding individuals, and construct the new probability distribution from them. Then new generation can be obtained by sampling this distribution and mutation;

14) While ($i \le N$) set $i = i + 1$, goto 11;

15) If the termination criteria are not met, set $t = t + 1$, goto 5.

6. Experiments and Results

In this section, we will use the proposed competitive co-evolutionary framework to retrieve the EP of a battle group. A character, from race A with level 3, is chosen as the defender, and with the same level, a character from race C as the attacker. Also, for convenience, we define SK_A^1, SK_A^2 as the two skills of race A, similarly, SK_C^1, SK_C^2 for race C. In accordance with the basic design contents mentioned before, we set the experiment environments as follows:

1) $V_A = 10$, $V_C = 25$, $DR_A = 0.3$, $DR_C = 0.45$, $H_A(3) = 100$, $H_C(3) = 60$;

2) Properties of SK_A^1: {$SD(3) = 30$, $SR = 70$, $SCDT = 1$, $SCHR = 0.1$};

3) Properties of SK_A^2: {$SD(3) = 20$, $SR = 70$, $SCDT = 2$, $SCHR = 0.3$};

4) Properties of SK_C^1: {$SD(3) = 25$, $SR = 190$, $SCDT = 13$, $SCHR = 0.3$};

5) Properties of SK_C^2: {$SD(3) = 25$, $SR = 190$, $SCDT = 14$, $SCHR = 0.6$};

6) The APV of attacker is (0.2, 0.6, 0.2);

7) In CCEA, population size of each species = 100, in SEA, population size = 500;

8) Generation $t = 600$, Mutation Rate = 0.01.

We define the fighting round, a period of time, from attacker launching an attack to the ending of his(her) waiting time. Such definition implies that attacker can attack only one time in a round. Also, based on the experiment environments, we obtain the attacking times of defender, when players chose a certain skill pair and attacker attacked in a certain range. The results are shown in **Table 1**.

According to those results and the optimal SM as well as the optimal SAO, the probability transition matrix defined by skill pair, SK_A^i, SK_C^j where $i, j = \{1, 2\}$, can be calculated (refer to **Table 2**), then according to Formula 1, we can retrieve the probability transition matrix defined by a stationary strategy pair (π^1, π^2). Consequently, such strategy pair can be evaluated by $v_1\left(s_N, \pi^1, \pi^2\right)$.

Besides the proposed algorithm, another one, whose blocks use SEA instead of CCEA, is introduced for the comparison. Considering the stochastic nature of the algorithms, each of the programs is repeated 100 times. The results are shown in **Figures 5** and **6**.

According to the results in **Figure 5**, both two algorithms can retrieve the EPs, $\left(\pi^{1*}, \pi^{2*}\right)$, and we can obtain the exception and the variance of $v_1\left(s_N, \pi^{1*}, \pi^{2*}\right)$, as follows:

$$E\left(v_1\left(s_N, \pi^{1*}, \pi^{2*}\right)\right) = 0.57474595$$
$$\mathrm{Var}\left(v_1\left(s_N, \pi^{1*}, \pi^{2*}\right)\right) = 1.41743336 \times 10^{-8} \tag{10}$$

Table 1. The attacking times of defender in a round.

	Range (c, d]	Range (d, e]	Range (e, g]
$\left(SK_A^1, SK_C^1\right)$	2	1	1
$\left(SK_A^2, SK_C^1\right)$	2	2	1
$\left(SK_A^1, SK_C^2\right)$	2	1	1
$\left(SK_A^2, SK_C^2\right)$	2	2	1

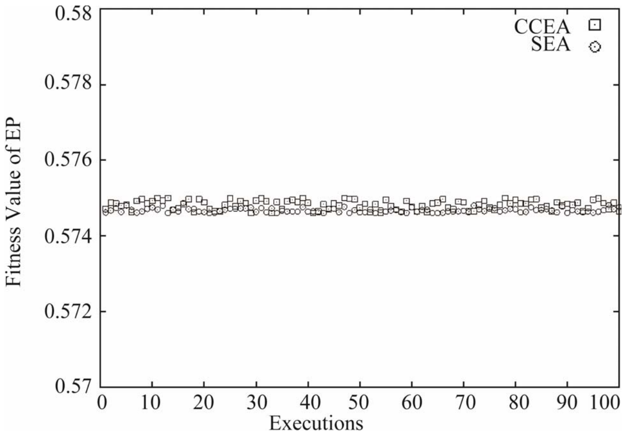


Figure 5. The fitness of the ep retrieved by each execution.

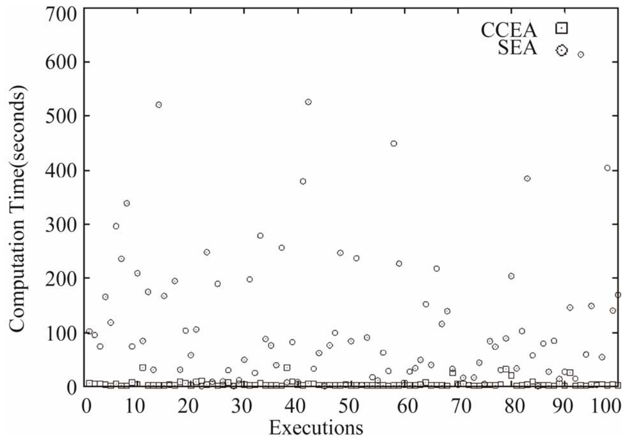


Figure 6. The computation time of retrieving EP in each execution.

Table 2. The probability transition matrix defined by skill pair SK_A^i, SK_C^i.

SK_A^1, SK_C^1	(0, 1)	(0, 2)	(0, 3)	(1, 0)	(2, 0)	(3, 0)	(4, 0)	(1, 1)	(1, 2)	(1, 3)	(2, 1)	(2, 2)	(2, 3)	(3, 1)	(3, 2)	(3, 3)	(4, 1)	(4, 2)	(4, 3)
(0, 1)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(0, 2)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(0, 3)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(1, 0)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(2, 0)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(3, 0)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
(4, 0)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
(1, 1)	0.7	0	0	0.17985	0	0	0	0.12015	0	0	0	0	0	0	0	0	0	0	0
(1, 2)	0	0.7	0	0.0326865	0	0	0	0.132165	0.1351485	0	0	0	0	0	0	0	0	0	0
(1, 3)	0	0	0.7	0.0312015	0	0	0	0	0.14553	0.1232685	0	0	0	0	0	0	0	0	0
(2, 1)	0.21	0	0	0.293755	0.17985	0	0	0.196245	0	0	0.12015	0	0	0	0	0	0	0	0
(2, 2)	0	0.21	0	0.0509625	0.0312015	0	0	0.237699	0.196245	0	0.14553	0.1283621	0	0	0	0	0	0	0
(2, 3)	0	0	0.21	0.02695	0.0165	0	0	0	0.237699	0.196245	0	0.14553	0.167076	0	0	0	0	0	0
(3, 1)	0	0	0	0.125895	0.293755	0.17985	0	0.084105	0	0	0.196245	0	0	0.12015	0	0	0	0	0
(3, 2)	0	0	0	0.0218411	0.0509625	0.0312015	0	0.101871	0.084105	0	0.237699	0.196245	0	0.14553	0.130545	0	0	0	0
(3, 3)	0	0	0	0.0218411	0.0509625	0.0312015	0	0	0.101871	0.084105	0	0.237699	0.196245	0	0.14553	0.130545	0	0	0
(4, 1)	0	0	0	0	0.125895	0.293755	0.17985	0	0	0	0.084105	0	0	0.196245	0	0	0.12015	0	0
(4, 2)	0	0	0	0	0.0218411	0.0509625	0.0312015	0	0	0	0.101871	0.084105	0	0.237699	0.196245	0	0.14553	0.130545	0
(4, 3)	0	0	0	0	0.0218411	0.0509625	0.0312015	0	0	0	0	0.101871	0.084105	0	0.237699	0.196245	0	0.14553	0.13054
SK_A^1, SK_C^2	(0, 1)	(0, 2)	(0, 3)	(1, 0)	(2, 0)	(3, 0)	(4, 0)	(1, 1)	(1, 2)	(1, 3)	(2, 1)	(2, 2)	(2, 3)	(3, 1)	(3, 2)	(3, 3)	(4, 1)	(4, 2)	(4, 3)
(0, 1)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(0, 2)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(0, 3)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(1, 0)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(2, 0)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(3, 0)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
(4, 0)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
(1, 1)	0.7	0	0	0.17985	0	0	0	0.12015	0	0	0	0	0	0	0	0	0	0	0

Continued

(1, 2)	0	0.7	0	0.0326865	0	0	0	0.132165	0.1351485	0	0	0	0	0	0	0	0	0	0
(1, 3)	0	0	0.7	0.0312015	0	0	0	0	0.14553	0.1232685	0	0	0	0	0	0	0	0	0
(2, 1)	0.42	0	0	0.16786	0.17985	0	0	0.11214	0	0	0.12015	0	0	0	0	0	0	0	0
(2, 2)	0	0.42	0	0.029121	0.031202	0	0	0.135828	0.11214	0	0.14553	0.126179	0	0	0	0	0	0	0
(2, 3)	0	0	0.42	0.0154	0.0165	0	0	0	0.135828	0.11214	0	0.14553	0.154602	0	0	0	0	0	0
(3, 1)	0	0	0	0.25179	0.16786	0.17985	0	0.16821	0	0	0.11214	0	0	0.12015	0	0	0	0	0
(3, 2)	0	0	0	0.043682	0.029121	0.031202	0	0.203742	0.16821	0	0.135828	0.11214	0	0.14553	0.130545	0	0	0	0
(3, 3)	0	0	0	0.043682	0.029121	0.031202	0	0	0.203742	0.16821	0	0.135828	0.11214	0	0.14553	0.130545	0	0	0
(4, 1)	0	0	0	0	0.25179	0.16786	0.17985	0	0	0	0.16821	0	0	0.11214	0	0	0.12015	0	0
(4, 2)	0	0	0	0	0.043682	0.029121	0.031202	0	0	0	0.203742	0.16821	0	0.135828	0.11214	0	0.14553	0.130545	0
(4, 3)	0	0	0	0	0.043682	0.029121	0.031202	0	0	0	0	0.203742	0.16821	0	0.135828	0.11214	0	0.14553	0.130545

SK_A^2, SK_C^1	(0, 1)	(0, 2)	(0, 3)	(1, 0)	(2, 0)	(3, 0)	(4, 0)	(1, 1)	(1, 2)	(1, 3)	(2, 1)	(2, 2)	(2, 3)	(3, 1)	(3, 2)	(3, 3)	(4, 1)	(4, 2)	(4, 3)
(0, 1)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(0, 2)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(0, 3)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(1, 0)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(2, 0)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(3, 0)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
(4, 0)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
(1, 1)	0.7	0	0	0.2244	0	0	0	0.0756	0	0	0	0	0	0	0	0	0	0	0
(1, 2)	0	0.7	0	0.10428	0	0	0	0.06468	0.13104	0	0	0	0	0	0	0	0	0	0
(1, 3)	0	0	0.7	0.04356	0	0	0	0.08646	0.10626	0.06372	0	0	0	0	0	0	0	0	0
(2, 1)	0.21	0	0	0.35035	0.2244	0	0	0.12348	0	0	0.09177	0	0	0	0	0	0	0	0
(2, 2)	0	0.21	0	0.141218	0.08646	0	0	0.173558	0.12348	0	0.10626	0.159024	0	0	0	0	0	0	0
(2, 3)	0	0	0.21	0.035574	0.02178	0	0	0.11858	0.173558	0.12348	0.0825	0.10626	0.128268	0	0	0	0	0	0
(3, 1)	0	0	0	0.15708	0.36652	0.2244	0	0.05292	0	0	0.12348	0	0	0.0756	0	0	0	0	0
(3, 2)	0	0	0	0.059552	0.138954	0.085074	0	0.079002	0.05292	0	0.173558	0.12348	0	0.10626	0.1812	0	0	0	0
(3, 3)	0	0	0	0.015246	0.035574	0.02178	0	0.04985	0.074382	0.05292	0.132486	0.173558	0.12348	0.081114	0.10626	0.13335	0	0	0
(4, 1)	0	0	0	0	0.15708	0.35882	0.2244	0	0	0	0.05292	0	0	0.12348	0	0	0.0833	0	0

Continued

(4, 2)	0	0	0	0	0.0595518	0.1389542	0.085074	0	0	0	0.074382	0.05292	0	0.173558	0.12348	0	0.10626	0.18582	0
(4, 3)	0	0	0	0	0.015246	0.035574	0.02178	0	0	0	0.05678	0.074382	0.05292	0.132486	0.173558	0.12348	0.081114	0.10626	0.12642

SK_A^2, SK_C^2	(0, 1)	(0, 2)	(0, 3)	(1, 0)	(2, 0)	(3, 0)	(4, 0)	(1, 1)	(1, 2)	(1, 3)	(2, 1)	(2, 2)	(2, 3)	(3, 1)	(3, 2)	(3, 3)	(4, 1)	(4, 2)	(4, 3)
(0, 1)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(0, 2)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(0, 3)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(1, 0)	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(2, 0)	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(3, 0)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
(4, 0)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
(1, 1)	0.7	0	0	0.2244	0	0	0	0.0756	0	0	0	0	0	0	0	0	0	0	0
(1, 2)	0	0.7	0	0.10428	0	0	0	0.06468	0.13104	0	0	0	0	0	0	0	0	0	0
(1, 3)	0	0	0.7	0.04356	0	0	0	0.08646	0.10626	0.06372	0	0	0	0	0	0	0	0	0
(2, 1)	0.42	0	0	0.2002	0.2244	0	0	0.07056	0	0	0.08484	0	0	0	0	0	0	0	0
(2, 2)	0	0.378	0	0.080696	0.08646	0	0	0.099176	0.07056	0	0.10626	0.178848	0	0	0	0	0	0	0
(2, 3)	0	0	0.378	0.020328	0.02178	0	0	0.06776	0.099176	0.07056	0.0825	0.10626	0.153636	0	0	0	0	0	0
(3, 1)	0	0	0	0.31416	0.20944	0.2244	0	0.10584	0	0	0.07056	0	0	0.0756	0	0	0	0	0
(3, 2)	0	0	0	0.119104	0.079402	0.085074	0	0.158004	0.10584	0	0.099176	0.07056	0	0.10626	0.17658	0	0	0	0
(3, 3)	0	0	0	0.030492	0.020328	0.02178	0	0.0996996	0.148764	0.10584	0.0757064	0.099176	0.07056	0.081114	0.10626	0.14028	0	0	0
(4, 1)	0	0	0	0	0.31416	0.20944	0.2244	0	0	0	0.10584	0	0	0.07056	0	0	0.0756	0	0
(4, 2)	0	0	0	0	0.119104	0.079402	0.085074	0	0	0	0.148764	0.10584	0	0.099176	0.07056	0	0.10626	0.18582	0
(4, 3)	0	0	0	0	0.030492	0.020328	0.02178	0	0	0	0.11356	0.148764	0.10584	0.075706	0.099176	0.07056	0.081114	0.10626	0.12642

Such data are very important because we can judge whether the designed contents are well-balanced based on it. Also, it can be seen from **Figure 6** that the CCEA is much faster than SEA, hence a better placement solution than using SEA.

7. Conclusion and Future Work

In this paper, we constructed a basic ARPG system and modeled the combat by using the zero-sum Γ_β process. We noted that in the case of zero-sum Γ_β, an EP of the entire behavior strategy space can be retrieved just by searching the stationary strategy space, and all the overall discounted value vectors of the EPs are coincide. Also, an evolutionary framework, which includes integration with competitive coevolution and CCEA, has been proposed to search the SUS pairs which are the EPs of the strategy space. Based on the framework, we made some experiments with certain environments. The results showed that the EPs can be obtained, and by using CCEA, the efficiency and capability of the framework have been improved significantly.

In the future work, we will investigate another combat type in which the battle time is constrained, and in such type the strategies of players will no longer be stationary.

REFERENCES

[1] R. Leigh, J. Schonfeld and S. Louis, "Using Coevolution to Understand and Validate Game Balance in Continuous Games," *Proceedings of the* 10*th Annual Conference on Genetic and Evolutionary Computation*, Atlanta, 12-16 July 2008, pp. 1563-1570. doi:10.1145/1389095.1389394

[2] H. Y. Chen, Y. Mori and I. Matsuba, "Design Method for Game Balance with Evolutionary Algorithms Using Stochastic Model," *Proceedings of International Conference on Computer Science and Engineering*, Shanghai, 28-31 October 2011, Vol. 7, pp. 1-4.

[3] H. Y. Chen, Y. Mori and I. Matsuba, "Evolutionary Approach to the Balance Problem of On-Line Action Role-Playing Game," *Proceedings of the* 3*rd International Conference on Computational Intelligence and Software Engineering*, Wuhan, 9-11 November 2011, pp. 1039-1042.

[4] J. Filar and K. Vrieze, "Competitive Markov Decision Processes," Springer-Verlag, New York, 1996. doi:10.1007/978-1-4612-4054-9

[5] P. Husbands and F. Mill, "Simulated Coevolution as the Mechanism for Emergent Planning and Scheduling," *Proceedings of the* 4*th International Conference on Genetic Algorithms*, San Diego, July 1991, pp. 264-270.

[6] M. Potter, "The Design and Analysis of a Computational Model of Cooperative Coevolution," Ph.D. Dissertation, George Mason University, Fairfax, 1997.

[7] L. Bull, T. C. Fogarty and M. Snaith, "Evolution in Multi-Agent Systems: Evolving Communicating Classifier Systems for Gait in a Quadrupedal Robot," *Proceedings of the* 6*th International Conference on Genetic Algorithms* (*ICGA*), Pittsburgh, 15-19 July 1995, pp. 382-388.

[8] M. Potter, L. Meeden and A. Schultz, "Heterogeneity in the Coevolved Behaviors of Mobile Robots: The Emergence of Specialists," *Proceedings of the* 17*th International Conference on Artificial Intelligence*, Seattle, 4-10 August 2001, pp. 1337-1343.

[9] K. S. Hwang, J. L. Lin and H. L. Huang, "Dynamic Patrol Planning in a Cooperative Multi-Robot System," *Communications in Computer and Information Science*, Vol. 212, 2011, pp. 116-123. doi:10.1007/978-3-642-23147-6_14

[10] M. Potter and K. D. Jong, "The Coevolution of Antibodies for Concept Learning," *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, Amsterdam, 27-30 September 1998, pp. 530-539. doi:10.1007/BFb0056895

[11] Y. Wen and H. Xu, "A Cooperative Coevolution-Based Pittsburgh Learning Classifier System Embedded with Memetic Feature Selection," *Proceedings of IEEE Congress on Evolutionary Computation*, New Orleans, 5-8 June 2011, pp. 2415-2422.

[12] A. Carlos and S. Moshe, "Fuzzy CoCo: A Cooperative-Coevolutionary Approach to Fuzzy Modeling," *IEEE Transactions on Fuzzy Systems*, Vol. 9, No. 5, 2001, pp. 727-737. doi:10.1109/91.963759

[13] R. Chandra and M. Zhang, "Cooperative Coevolution of Elman Recurrent Neural Networks for Chaotic Time Series Prediction," *Neurocomputing*, Vol. 86, 2012, pp. 116-123. doi:10.1016/j.neucom.2012.01.014

[14] M. Potter and K. D. Jong, "A Cooperative Coevolutionary Approach to Function Optimization," *Proceedings of the* 3rd *International Conference on Parallel Problem Solving from Nature*, Jerusalem, 9-14 October 1994, Vol. 866, pp. 249-257. doi:10.1007/3-540-58484-6_269

[15] M. Potter and K. D. Jong, "Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents," *Evolutionary Computation*, Vol. 8, No. 1, 2000, pp. 1-29. doi:10.1162/106365600568086

[16] R. P. Wiegand, W. Liles and K. D. Jong, "An Empirical Analysis of Collaboration Methods in Cooperative Coevolutionary Algorithms," *Proceedings of the Genetic and Evolutionary Computation Conference*, San Francisco, 7-11 July 2001, pp. 1235-1242.

[17] T. Jansen and R. P. Wiegand, "Exploring the Explorative Advantage of the CC (1+1) EA," *Proceedings of the Genetic and Evolutionary Computation Conference*, Chicago, 12-16 July 2003, pp. 310-321.

[18] R. P. Wiegand, "An Analysis of Cooperative Coevolutionary Algorithms," Ph.D. Dissertation, George Mason University, Fairfax, 2004.

[19] C. Reeves and J. Rowe, "Genetic Algorithms Principles and Perspectives: A Guide to GA Theory," Kluwer Academic Publishers, Norwell, 2002.

[20] L. Schmitt, "Theory of Genetic Algorithms," *Theoretical Computer Science*, Vol. 259, No. 1-2, 2001, pp. 1-61. doi:10.1016/S0304-3975(00)00406-0

[21] M. Vose, “The Simple Genetic Algorithm,” MIT Press, Cambridge, 1999.

[22] P. Liviu, “Theoretical Convergence Guarantees for Cooperative Coevolutionary Algorithms,” Evolution Computation, Vol. 18, No. 4, 2010, pp. 581-615. doi:10.1162/EVCO_a_00004