Potential AI Strategies to Solve the *Commons Game*: A Position Paper

Petro Verkhogliad[1] and B. John Oommen[2]

[1] School of Computer Science, Carleton University, Ottawa, Canada
pverkhog@scs.carleton.ca
[2] *Chancellor's Professor*; *Fellow: IEEE* and *Fellow: IAPR.*
School of Computer Science, Carleton University, Ottawa, Canada
Also *Adjunct Professor* with the University of Agder in Grimstad, Norway
oommen@scs.carleton.ca

Abstract. In this paper, we propose the use of hill climbing and particle swarm optimization to find strategies in order to play the *Commons* Game (CG). The game, which is a non-trivial N-person non-zero-sum game, presents a simple mechanism to formulate how different parties can use shared resources. If the parties cooperate, the resources are sustainable. However, the resources get depleted if used indiscriminately. We consider the case when a single player has to determine the "optimal" solution, and when the other $N-1$ players play the game by choosing the options with a fixed probability vector.

Keywords: Intelligent Game Playing, Resolving N-person Games, Commons game, AI-based Resource Management.

1 Introduction

Artificial intelligence (AI) techniques have been used for decades to aid in the analysis and solution of games. During this time, most research activity has revolved around popular 2-player games, such as *Chess* and *Checkers*. Considerable effort has also been applied to the study of 2-player social dilemma games [3,6], such as the *Prisoner's Dilemma* [8]. Unfortunately, significantly less attention has been focused on N-player games, partly due to the difficulty of applying previously successful techniques [10] to games of this class. In this work, we consider the *Commons* Game (CG) [9], which is a non-trivial N-person non-zero-sum game. In this position paper, we suggest methods by which we can apply two AI techniques, namely, hill climbing (HC) and particle swarm optimization (PSO), in order to find a solution to this complex N-player social dilemma type game.

2 *Commons Game* Description

The *Commons game* designed by Powers *et al.* can be played by groups of 6 to 50 players. At every turn, every player selects one of five available actions:

A. Farzindar and V. Keselj (Eds.): Canadian AI 2010, LNAI 6085, pp. 352–356, 2010.

selfish use, cooperative use, abstention, penalty to selfish players, and reward to cooperative players. Each of these actions are mapped onto five playing cards identified by their colors: green, red, yellow, black and orange respectively.

The green card symbolizes selfish behavior and returns the maximum number of points. The same player[1] may receive a score of -20 points if a black card is played during the same turn. The red card represents cooperative behavior and returns a reward of 40% of the value of the green card. Red card players receive an additional 10 points for every orange card played in the same round. The yellow card represents abstention. Each yellow card receives 6 points regardless of the state of the environment or the number of players in the game. The black card is used to penalize selfish players. The returned score is defined by $-N_p/N_b$, where N_b is the number of black cards played in that round, and N_p is the number of participants. The orange card is used to encourage cooperative players by increasing their score for that turn by 10 points, and the player receives $-N_p/N_o$ points, where N_p is as above and N_o is the number of orange cards played in the round.

The state of the environment determines the exact number of points scored. The states range from +8 to -8. At the start of the game the environment is at state 0. Table 1 shows the scoring table for states +8, +4, 0, -4 and -8.

The depletion and replenishment of the environment are modeled using a marker, m, which ranges between $[0, 180]$. At the end of every turn, the marker is updated using Eq. (1) below, where m_{t+1} is the marker value in the next turn, N_g is the number of green cards played in the current turn, S_t is the current state number, $I(S_t)$ is the replenishment value in the given state, and t is the current turn number.

$$m_{t+1} = \begin{cases} m_t - N_g + I(S_t) & \text{if } t \mod 6 = 0 \\ m_t - N_g & \text{if } t \mod 6 \neq 0. \end{cases} \tag{1}$$

The value of the marker is used to determine the state of the environment in the next turn as shown in the Eq. (2) below:

$$S_{t+1} = \begin{cases} 0 & \text{if } 80 \leq m_t \leq 100 \\ \dfrac{m_t - 90}{10} & \text{if } m_t < 80 \text{ or } m_t > 100. \end{cases} \tag{2}$$

In the interest of clarification, consider the following example of score calculation. In this example, the number of players, N_p, is 8, the current turn, t, is 0, the value of the marker, m, is 100, and the current state, S_0, is 0. Five players use the red card ($N_r = 5$), two players use the green card ($N_g = 2$), and one player uses the black card ($N_b = 1$). The rewards of the players are then: $R_r = 44, R_b = -8$ and $R_g = -20$ instead of 106, since a black card was played.

[1] Although we consider specific numeric score values as defined in the original manual [9], the principles presented here work even if one changes the values so as to preserve the main properties of the game.

3 Methods

The focus of this work is on finding a strategy vector which constitutes the respective probabilities for playing a given card, $P = \{p_{green}, p_{red}, p_{yellow}, p_{orange}, p_{black}\}$, such that $\sum_{p_i \in P} p_i = 1$. We propose to do this by the use of hill climbing and particle swarm optimization.

3.1 Hill Climbing

Hill climbing (HC) is one of the earliest and most well known optimization techniques. At the start of the game the scheme is initialized with the following parameters in the solution/search space:

P - a random solution
λ - the learning rate parameter
T - number of turns used for learning
$f(P_t)$ - fitness function, where P_t is the vector of card probabilities at turn t.

At every turn, HC attempts to increase its reward by updating the probability vector, P. We propose that *five* candidate vectors are created by choosing a card, p_i, and increasing its probability while decreasing the probabilities of the other cards. Eq. (3) shows the updating function of the non-target cards, and Eq. (4) shows the updating function of the target card, after the others have been updated. Here $p_i(t)$ is the current probability of selecting card i, $p_i(t+1)$ is the probability that will be used in the next turn, and $i, j \in \{green, red, yellow, orange, black\}$ such that $i \neq j$.

$$p_i(t+1) = \lambda p_i(t) \tag{3}$$

$$p_j(t+1) = p_j(t) + (1 - \sum_{i \in P} p_i(t+1)) \tag{4}$$

Each of the five candidate vectors are then tested by playing T turns of the game. The vector with highest fitness value, i.e., the sum of scores over the T

Table 1. Reward table for an 8-player group

State +8			State +4			State 0			State -4			State -8		
$\mathcal{N}_r$	$\mathcal{R}_r$	$\mathcal{R}_g$	$\mathcal{N}_r$	$\mathcal{R}_r$	$\mathcal{R}_g$	$\mathcal{N}_r$	$\mathcal{R}_r$	$\mathcal{R}_g$	$\mathcal{N}_r$	$\mathcal{R}_r$	$\mathcal{R}_g$	$\mathcal{N}_r$	$\mathcal{R}_r$	$\mathcal{R}_g$
0	-	200	0	-	186	0	-	100	0	-	14	0	-	0
1	90	202	1	83	188	1	40	102	1	-3	16	1	-10	2
2	90	202	2	83	188	2	40	102	2	-3	16	2	-10	2
3	90	202	3	83	188	3	40	102	3	-3	16	3	-10	2
4	92	204	4	85	190	4	42	104	4	-2	18	4	-8	4
5	94	206	6	87	192	5	44	106	5	1	20	5	-6	6
6	96	208	7	89	194	6	46	108	6	3	22	6	-4	8
7	98	210	8	91	196	7	48	110	7	5	24	7	-2	10
8	100	-	8	93	-	8	50	-	8	7	-	8	0	-

turns, is selected as the current best solution. This process is repeated for an epoch of J (say 1,000) turns.

3.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) was originally developed and introduced by Kennedy and Eberhart [5]. The algorithm is based on the flocking behavior of fish/birds, and has previously been successfully applied to many optimization as well as some gaming problems [1,2,4,7].

Similar to HC, each particle uses a set of updating equations to direct itself and the rest of the swarm toward the optimum. The updating rule for $\overrightarrow{x_i}(t)$, the position of particle i at time 't', is shown in Eq. (5), where $\overrightarrow{v_i}(t)$ is the particle velocity at time 't'. The updating rule for the latter, is shown in Eq. (6), where g and $\hat{g}$ are, respectively, the particle's and the swarm's best solutions. Further, c_1 and c_2 are the acceleration coefficients, and ω is the inertia weight.

$$\overrightarrow{x_i}(t+1) = \overrightarrow{v_i}(t) + \overrightarrow{x_i}(t) \tag{5}$$

$$v_{ij}(t+1) = \omega v_{ij}(t) + c_1 r_1 j(t)[g_{ij}(t) - x_{ij}(t)] + c_2 r_2 j(t)[\hat{g}_{ij}(t) - x_{ij}(t)]. \tag{6}$$

In the above equations, x_{ij} and v_{ij} are the j^{th} components of $\overrightarrow{x_i}(t)$ and $\overrightarrow{v_i}(t)$ respectively, and r_1 and $r_2 \in U(0,1)^n$. It has been shown that under the following conditions:

$$\omega > \tfrac{1}{2}(c_1 + c_2) - 1; \quad \texttt{and} \quad 0 < \omega < 1,$$

convergence to a stable equilibrium point is guaranteed.

At the start of the game, each particle in the swarm is initialized with a random strategy. At every turn, the fitness of every particle is evaluated by using the particle's position to play T turns of the training instance of the game. The fitness value is the sum of the rewards received over the T turns. The fitness function does not incorporate any information about the game, except for the score that the player received in each of the T turns. If the new fitness value is higher than any previously seen value, the new strategy becomes the particle's "Best" solution. Each of these individual "Best" solutions are then compared to each other in order to find the global "Best" solution. This process is repeated for an epoch of J (say 1,000) turns.

4 Results and Conclusion

Early results from applying HC and PSO to the *Commons Game* are highly encouraging. For example, when the game is confined to the +8 state, both algorithms converge on the solution of playing red with probability 7/8 and playing green with probability 1/8, which have been suggested by Powers as the cooperative "game end" scenario [9]. More detailed results and comparisons are currently being compiled, but are omitted here as this is only a position paper.

In conclusion, in this work we have suggested the use hill climbing and particle swarm optimization to solve the *Commons game*, which is an N-player, imperfect-information, non-zero-sum game. Early results suggest that general purpose optimization techniques can be used to find good strategies for this complex N-person game.

References

1. Abdelbar, A.M., Ragab, S., Mitri, S.: Applying co-evolutionary particle swam optimization to the egyptian board game seega. In: Proceedings of The First Asian-Pacific Workshop on Genetic Programming, pp. 9–15 (2003)
2. Conradie, J., Engelbrecht, A.P.: Training bao game-playing agents using coevolutionary particle swarm optimization. In: Proceedings of the IEEE 2006 Symposium on Computational Intelligence and Games, pp. 67–74 (2006)
3. Dawes, R.M.: Social dilemmas. Annual Review of Psychology 31(1), 169–193 (1980)
4. Franken, N., Engelbrecht, A.P.: Particle swarm optimization approaches to coevolve strategies for the iterated prisoner's dilemma. IEEE Transactions on Evolutionary Computation 9(6), 562–579 (2005)
5. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
6. Kollock, P.: Social dilemmas: The anatomy of cooperation. Annual Review of Sociology 24(1), 183–214 (1998)
7. Laskari, E.C., Parsopoulos, K.E., Vrahatis, M.N.: Particle swarm optimization for minimax problems. In: Proceedings of the IEEE 2002 Congress on Evolutionary Computation, pp. 1582–1587 (2002)
8. Poundstone, W.: Prisoner's Dilemma. Doubleday (1992)
9. Powers, R.B., Duss, R.E., Norton, R.S.: THE COMMONS GAME Manual (1977)
10. Sturtevant, N.: Current challenges in multi-player game search. In: van den Herik, H.J., Björnsson, Y., Netanyahu, N.S. (eds.) CG 2004. LNCS, vol. 3846, pp. 285–300. Springer, Heidelberg (2006)