Using Artificial Intelligence Techniques for Strategy Generation in the *Commons Game*

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 consider the use of artificial intelligence techniques to aid in discovery of winning strategies for the *Commons* Game (CG).

The game represents a common scenario in which multiple parties share the use of a self-replenishing resource. The resource deteriorates quickly if used indiscriminately. If used responsibly, however, the resource thrives. We consider the scenario one player uses hill climbing or particle swarm optimization to select the course of action, while the remaining $N-1$ players use a fixed probability vector. We show that hill climbing and particle swarm optimization consistently generate winning strategies.

Keywords: Intelligent Game Playing, Commons Game, AI-based Resource Management.

1 Introduction

Artificial and computational intelligence (AI/CI) techniques have been used for decades to aid in the analysis and solution of games. In this work, we consider the *Commons Game* (CG) [17], which is a non-trivial N-person non-zero-sum game. We show that winning strategies (against opponents playing random strategies) for the *Commons Game* can be found through the application of either hill climbing (HC) or particle swarm optimization (PSO).

The *Commons Game* represents a social dilemma where the individually-selfish behavior of the members in the "society" leads to detrimental collective outcome. The simplest and most well known dilemma game is *Prisoner's Dilemma* [4].

The *Commons Game* attempts to realistically model the interaction between a self-replenishing resource and a number of consumers of that resource. The interaction is modeled as an N player competition. Each player's aim is to maximize the number of points they earn.

At every turn of the game, a player can take one of five actions: use the resource responsibly, use the resource indiscriminately, abstain from resource use, punish other players, reward other players.

E. Corchado, M. Kurzyński, M. Woźniak (Eds.): HAIS 2011, Part I, LNAI 6678, pp. 43–50, 2011.

As can be seen from the above description, the *Commons Game* can be used to model almost any resource-sharing scenario. Examples range from networking resource sharing to natural resource extraction via automated agents. This broad applicability is the one of the primary motivators for this study.

Although no results on strategy discovery *via* AI/CI methods have been published, the CG has received some attention from the research community. In [5], Baba presented two games that bear significant similarity to the game studied here. Kirts *et al.* [13] have conducted simulations, in a similar game, in order to increase awareness of the complexity associated with the decision-making and cooperation phases. In [7,6,11] Handa and Baba have shown that CG can be made more interesting for human players by altering the rules of the game to those discovered by a combination of genetic algorithms and neural networks. Brunovsky [8], on the other hand, has characterized the equilibria in the original "Tragedy of the Commons" setting. Furthermore, Faysse [10] has presented another variation of the commons dilemma game, as well as results of experiments conducted in common-pool resource sharing settings.

2 Description of the Commons Game

The *Commons Game* designed by Powers *et al.* [17,16] can be played by groups of 6 to 50 players. All of the actions in the game are mapped onto five "playing cards": green, red, yellow, black and orange. The description of the cards and their respective point allocations[1] are given below.

The green card symbolizes indiscriminate use of the commons. Playing it yields the maximum reward, R_g, unless another player uses the black (punishment) card in the same turn, in which case it yields -20 points.

The red card represents careful or cooperative use of the commons. The reward for red card plays, R_r, depends on the total number of red cards played in the current turn, N_r. Red card players also benefit when others play the orange card. Any player using the red card cannot be penalized during the same turn.

The yellow card represents abstention from the use of the commons. Each yellow card play receives 6 points regardless of the state of the environment or the number of players in the game.

The black card can be used to punish players who abuse the environment by playing green. It yields $-N_p/N_b$, where N_b is the number of black cards played in that round, and N_p is the number number of players.

The orange card can be used to encourage players who use the resource responsibly. It grants +10 points to the red card players. The orange card yields a score of $-N_p/N_o$, where N_o is the number of orange cards played during the round, and N_p is the number of players.

[1] Although we consider specific numeric score values as defined in the original manual [17], the principles presented here work even if one changes the values so as to preserve the main properties of the game.

The state of the environment also contributes to determining the reward received by green and red card players. The states range from -8 to +8. At the start of the game, the environment is at state 0.

The depletion and replenishment of the environment are modeled using a marker, m, such that $m \in [0, 180]$. At the end of every turn, the marker is updated using Eq. (1), 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 Eq. (2):

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

3 Methods

The focus of our work is to find winning strategy vectors consisting of the 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 using hill climbing (HC) and particle swarm optimization (PSO).

3.1 Hill Climbing

Hill climbing (HC) is one of the earliest and most well-known optimization techniques. In general, HC starts with either a man-made or a random solution and attempts to find the optimum by sampling the area around it, and choosing the "best" value as determined by a heuristic function. Simple HC selects the first solution that is better than the current one. Steepest ascent/descent HC compares all neighbouring solutions and selects the best one. Finally, stochastic HC selects a random neighbour, provided its heuristic value is greater than the preset threshold.

Due to the complexity of the CG, we have opted to use a variation of the classical HC, inspired by approaches in reinforcement learning [18] and learning automata [15]. Learning automata algorithms attempt to find the best action (from a predefined set of actions) in a stochastic environment by using an action probability vector. The optimal action is found by updating the action probability vector using a well defined updating scheme[2].

[2] There is a variety of action probability updating schemes. For lack of space they are not discussed here.

There are some key differences, however, between the version of HC used in this work and learning automata algorithms. One, at every time step learning automata select and use the action with the highest probability of reward. Two, only some of the learning automata employ a testing step and only as part of the initialization routine. The HC algorithm used here does not select the action which currently has the highest probability of reward. Rather, by generating candidate solutions it attempts to find an action with highest future reward. As such this version of the HC algorithm is closer to the algorithms described in [2,3] then classical HC.

At the start of the game the HC is initialized with the following parameters:

$\mathcal{P}$: A random solution
λ : The learning rate parameter
T : The number of turns used for learning
$f(P_t)$: A fitness function, where P_t is the vector of card probabilities at turn t.

Once initialized, the HC module is added to a game with $N-1$ other players which employ a random action-selection strategy. In our case $N = 8$. At each turn, the HC player updates it's probability vector, P, in an attempt to increase it's overall reward. Five candidate vectors are created (one for each color) by choosing a card, p_i, and increasing its probability while decreasing the probabilities of the other cards. While Eq. (3) shows the updating function of the non-target cards, Eq. (4) specifies the updating function of the target card, *after* the others have been updated. In these equations, $p_i(t)$ is the current probability of selecting a card i, $p_i(t+1)$ is the probability value 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 which returns the highest fitness value, i.e., the sum of scores over the T turns, is selected as the current best solution. The process is then repeated for K iterations.

The strategies discovered by this HC scheme are discussed in the following sections.

3.2 Particle Swarm Optimization

Particle swarm optimization (PSO) was originally developed by Kennedy and Eberhart [12]. The algorithm is based upon the flocking behavior of fish/birds, and has been successfully applied to many optimization and gaming problems [1,9,14]. PSO relies on a set of n-dimensional particles traveling through the solution space. Each particle's position represents a possible solution. Particles track their own best solution, velocity, and the global best solution.

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 equation for a particle's position, $\overrightarrow{x_i}$, is shown in Eq. (5), where $\overrightarrow{v_i}$ is the particle's velocity. The updating equation for the latter, is shown in Eq. (6), where g and $\hat{g}$ are, respectively, the particle's and the swarm's "best" solutions.

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

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

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. Further, c_1 and c_2 are the acceleration coefficients, and ω is the inertia weight. Finally, r_1 and $r_2 \in U(0,1)^n$. It has been shown that under the following conditions:

$$\omega > \frac{1}{2}(c_1 + c_2) - 1; \quad \text{and} \quad 0 < \omega < 1 \ . \tag{7}$$

convergence to a stable equilibrium point is guaranteed.

The structure of the particle neighbourhood used in our work is commonly referred to as *gbest*. This configuration was intentionally selected as this is the first time it has been used in relation to CG. More complex approaches, lattice or Von Neumann, are reserved for future work.

Each particle in the swarm is initialized with a random position vector which represents a game strategy vector. The PSO player is then added to a game with $N-1$ other players which employ a random action-selection strategy. In our case $N = 8$. At every game turn, the fitness of every particle is evaluated by using it's position to play T turns of a training instance of the game. The fitness value is the sum of the rewards received over the T turns. Similar to the HC approach, the fitness function used does not incorporate any information about the game, except for player's score over the T turns. If the new fitness value is higher than any previously seen value, the new strategy becomes the particle's "best" solution. The individual "best" solutions are then compared to determine the global "best" solution. This continues for the rest of the game.

4 Results and Discussion

In order to explore the set of possible strategies for the *Commons Game*, experiments using both, HC and PSO, were performed in the following list of state ranges, $\mathcal{S} = \{[+8,-8],[+8,+8],[+8,+4],[+4,0],[0,-4],[-4,-8]\}$. 100 games were played for every state range S_i, with games of 1,000 turns. As previously mentioned, only one "intelligent" player was used in each game. The remaining $N-1$ players used a random action-selection strategy. Tables 1[3], 3 and 2 show the scores and strategies generated by the HC and PSO players respectively[4].

[3] In this table the subscripts *as* and *sd* are "average score" and "standard deviation", respectively.

[4] Average scores shown in the tables were calculated as the sum of scores for 100 games divided by 100.

Table 1. Average score per player type

State Range	$\text{Random}_{\text{as}}$	$\text{Random}_{\text{sd}}$	HC_{as}	HC_{sd}	PSO_{sd}	PSO_{sd}
+8, +8	25,555	2,043.9	59,566	1,033.36	78,950	1,614.67
+8, +4	23,350	1,679.4	66,809	1,192.01	69,460	778.4
+4, 0	11,570	854.1	35,793	492.4	37,287	563.4
0, -4	-1,382	439.1	6,103	38.2	5,937	36.6
-4, -8	-3,129	193.6	5,993	2.26	5,951	32.4

Table 2. Summary of Hill Climbing Strategies

State Range	Red	Green	Yellow	Orange	Black
+8, -8	0.0000	0.0000	0.9998	0.0001	0.0000
+8, +8	0.6528	0.2089	0.0529	0.0325	0.0526
+8, +4	0.9982	0.0004	0.0004	0.0004	0.0004
+4, 0	0.9029	0.0083	0.0031	0.0120	0.0734
0, -4	0.5059	0.0000	0.4940	0.0000	0.0000
-4, -8	0.0001	0.0000	0.9994	0.0000	0.0000

Table 3. Summary of PSO Strategies

State Range	Red	Green	Yellow	Orange	Black
+8, -8	0.0017	0.0014	0.9958	0.0008	0.0001
+8, +8	0.8179	0.1604	0.0100	0.0062	0.0053
+8, +4	0.9891	0.0071	0.0015	0.0009	0.0012
+4, 0	0.9314	0.0300	0.0303	0.0004	0.007
0, -4	0.2026	0.0060	0.7903	0.0008	0.0000
-4, -8	0.0018	0.0016	0.9955	0.0008	0.0001

[+8, +8] State Range

In this state, red and green players reap the maximum possible rewards. Both HC and PSO allocate most of the probability into the red and green cards. HC is more aggressive. However, this resource overuse results in frequent punishment. PSO prefers red card plays, which can not be punished. This accounts for the difference in scores. The difference is statistically significant with $p < 0.05$.

It should be noted that the strategy discovered by the PSO player is very similar to the one considered by Powers [17] as the cooperative game solution.

[+8, +4] State Range

The "intelligent" players develop equivalent strategies. Neither player allocates any probability weight into playing green cards. Instead, almost all of the probability weight is placed into red card plays. The difference of payoffs

for this range in comparison to the [+8, +8] range accounts for the different probability allocation.

[+4, 0] State Range
This range, as well as the one immediately below, are of particular interest since they include the starting state, 0. Red plays are, once again, the most dominant for both, HC and PSO, schemes. This can be attributed to the fact that the red play yields a positive reward while being nonpunishable.

Unlike in the [+8, +4] state, the difference in the scores between the HC and PSO strategies is statistically significant with $p < 0.05$.

[0, -4] State Range
Here the strategies chosen by both of the players are quite different. PSO primarily abstains from resource use. HC, on the other hand, alternates between responsible use and abstention. This difference accounts for the better performance of the HC player. The difference in scores is statistically significant with $p < 0.05$.

[-4, -8] State Range
This state range represents a degraded commons. A previously unseen strategy of complete abstention arises here. It should also be noted that the strategy developed by the HC performs better with statistical significance, $p < 0.05$.

4.1 Discussion

Overall conclusions of the HC and PSO strategies are discussed below.

One way to interpret the discovered strategies is that they are self-reliant. Both algorithms play nonpunishable cards with positive rewards. The developed strategies, in fact, can be viewed those that recognize the volatility of the opponents' actions.

Also interesting is the fact that a single component of the strategy vector is preferred (when compared to all of the other possibilities) in the majority of the strategies. This suggests that in each state there is a single best response to any action by the random player. This also hints at the "existence" of possible equilibrium strategies.

Finally, we observe that the choice of the fitness function and the strategy representation play an important role in the above. Neither the HC nor the PSO players take into account the state of the game or the previous moves of their opponents. Indeed, in this configuration the players seem to discover those strategies that maximize the score independent of the behavior of the other players.

5 Conclusion

In this work we have investigated the use of artificial/computational intelligence techniques in order to generate winning strategies for an N-player, imperfect-information non-zero-sum game. More specifically, we have shown that general

purpose optimization techniques (such as hill climbing (HC) and particle swarm optimization (PSO)) can be used successfully in the confines of a complex N-person game such as the *Commons Game*.

References

1. Abdelbar, A.M., Ragab, S., Mitri, S.: Applying co-evolutionary particle swam optimization to the egyptian board game seega. In: ASPGP 2003. pp. 9–15 (2003)
2. Abraham, A., Corchado, E., Corchado, J.M.: Hybrid learning machines. Neurocomputing 72(13-15), 2729–2730 (2009)
3. Abraham, A., Köppen, M., Franke, K. (eds.): Design and application of hybrid intelligent systems. Amsterdam, IOS Press (2003)
4. Axelrod, R., Hamilton, W.: The evolution of cooperation. Science 211(4489), 1390 (1981)
5. Baba, N.: The commons game by microcomputer. Simulation & Gaming 15, 487–492 (1984)
6. Baba, N.: Utilization of genetic algorithms in order to make game playing much more exciting. In: KES 1999, pp. 473–476 (1999)
7. Baba, N., Nagasawa, K., Handa, H.: Utilization of Soft Computing Techniques for Making Environmental Games More Exciting –Toward an Effective Utilization of the COMMONS GAME. In: Lovrek, I., Howlett, R.J., Jain, L.C. (eds.) KES 2008, Part II. LNCS (LNAI), vol. 5178, pp. 411–417. Springer, Heidelberg (2008)
8. Brunovsḱý, P.: The commons game. Ekonomicky Casopis 55(8), 811–814 (2007)
9. Conradie, J., Engelbrecht, A.P.: Training bao game-playing agents using coevolutionary particle swarm optimization. In: CIG 2006, pp. 67–74 (2006)
10. Faysse, N.: Coping with the tragedy of the commons: Game structure and design of rules. Journal of Economic Surveys 19(2), 239–261 (2005)
11. Handa, H., Baba, N.: Evolutionary computations for designing game rules of the commons game. In: CIG 2007, pp. 334–339 (2007)
12. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: ICNN 1995, vol. 4, pp. 1942–1948 (1995)
13. Kirts, C.A., Tumeo, M.A., Sinz, J.M.: The commons game: Its instructional value when used in a natural resources management context. Simulation & Gaming 22(1), 5–18 (1991)
14. Laskari, E.C., Parsopoulos, K.E., Vrahatis, M.N.: Particle swarm optimization for minimax problems. In: CEC 2002, pp. 1582–1587 (2002)
15. Narendra, K.S., Thathachar, M.A.L.: Learning Automata: An Introduction. Prentice-Hall, Inc., Upper Saddle River (1989)
16. Powers, R.B., Boyle, W.: Generalization from a commons-dilemma game: The effects of a fine option, information, and communication on cooperation and defection. Simulation & Gaming 14(3), 253–274 (1983)
17. Powers, R.B., Duss, R.E., Norton, R.S.: THE COMMONS GAME Manual (1977)
18. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2004)