Partner Selection: Finding the Right Combination of Players

Pedro Mariano[1] and Luís Correia[2]

[1] Transverse Activity on Intelligent Robotics – IEETA – DETI, Universidade de Aveiro, Portugal
[2]LabMAg – Dep. de Informática, Faculdade de Ciências, Universidade de Lisboa, Portugal
plsm@ua.pt Luis.Correia@di.fc.ul.pt

Abstract

In games that model cooperative dilemmas, if players are able to choose with whom they will play, they will seek out cooperative partners while escaping free riders. In this paper we recast the problem of selecting with whom to play as a problem of finding the right combination of players. With this approach, we present a model suitable to any n-player game. The model is adaptive and we present three update policies. If a player has enough cooperative partners, then with our model a player is able to only select them. We show informal proofs of our claim and illustrate our model under different scenarios.

Introduction

Cooperative dilemmas have been modelled by several games, for instance Iterated Prisoner's Dilemma (IPD), Ultimatum, Investment, Centipede, and Public Good Provision (PGP) (Gintis, 2000b; Fudenberg and Tirole, 1991; Axelrod, 1997). Theoretical analysis of these games predicts the prevalence of free-riders, exploiters, and other types of non-prosocial behaviour (Gintis, 2000b). Despite this, experiments involving people show significant pro-social behaviour. Several theories, trust management, reputation, norms, punishments, have been put forward to explain these results under different forms. However these theories are usually attached to particular games.

In this paper we focus on partner selection. It has been reported in human experiments (Coricelli et al., 2004; Ehrhart and Keser, 1999) that if players are able to select their partners they will seek cooperative partners while escaping free riders. We present a model of partner selection tailored for any n-player game that allows a player to select the most favourable combination of partners. In contrast with previous results, our model relies solely on private information.

The model we present should be used by a player during its life cycle when it has to play a game. The player uses private information gathered from previous games to select partners to play a game. Although with our model a player can in some conditions only select cooperative partners, we do not prevent it from being selected by uncooperative players.

The goal of our model is to allow cooperative players to tentatively select cooperative partners. We assume that a selected player cannot refuse to play and therefore it can be selected by uncooperative players. This situation is not unlike neighbourhood choice, for instance. Someone chooses a neighbourhood for its general reputation but she may not refuse to have any new neighbour no matter how the newcomer is uncooperative.

Related Work

Volunteering is a form of partner selection where a player can choose to participate in a game or not, Aktipis (2004); Hauert et al. (2002); Orbell and Dawes (1993). For each interaction, it introduces the possibility of not playing. However the payoff for not playing lays between the maximum and minimum payoffs obtainable in the game. This relation alters the equilibria in the original game and thus creates new ones. This is the case in Orbell and Dawes (1993) where the payoff for not playing is zero (in their game there are positive and negative payoffs). They justify their choice of this value because people can evaluate and compare game actions that lead to positive or to negative payoffs. The same happens in Hauert et al. (2002). They focused on the PGP game. Players that do not play get a payoff that is higher than the payoff obtained by a defector in a group of defectors but lower than the payoff obtained by a cooperator in a group of cooperators. They found out that their system exhibits a rock-scissors-paper dynamics where players with the option of participating cyclally appear and disappear from the population. In both works players do not have memory of past encounters nor can identify other players. In Price (2006) the author refers that in experiments involving human subjects, people usually cooperate when they can choose their interaction partners, and they cooperate when they perceive altruistic behaviour.

Model Description

In a n-player game, a player has to select $n-1$ partners to play a game from a population of m candidates. Its problem is to find those combinations that yield the highest utilities.

We assume that the player has access to those m candidates, but our model can easily be adapted to a scenario where candidates may enter or leave the population. We assume that the population may contain candidates that behave stochastically, namely, they sometimes cooperate but they also free ride.

For large m and n it may not be feasible for a player to process all the possible combinations. Therefore, a player maintains a pool $\mathbf{c}$ of l combinations that is updated as it plays games. Each combination has a probability to be selected. These probabilities are stored in a vector $\mathbf{w}$. Finally, the player has a utility threshold u_T. Representing a strategy by s, a player is then characterised by a 4-tuple:

$$\alpha = (s, \mathbf{c}, \mathbf{w}, u_T) \quad . \tag{1}$$

When a player has to play a game, it selects a combination from vector $\mathbf{c}$ using the probability vector $\mathbf{w}$. It compares the utility obtained with u_T and decides if it should update the two vectors. If it is lower, then other combination should be favoured.

In the following discussion, we will assume that k is the slot index of the selected combination. We will now discuss some vector update policies.

Drastic Update – Policy A

If the selected combination yields a utility lower than u_T, its probability is multiplied by a factor, δ, lower than 1.

$$w_k^{t+1} = \begin{cases} \delta w_k^t & \text{if} \quad u < u_T \\ w_k^t & \text{if} \quad u \geq u_T \end{cases} \quad . \tag{2}$$

The probabilities of other combinations are updated as follows:

$$w_i^{t+1} = \begin{cases} w_i^t + \dfrac{(1-\delta)w_k^t}{l-1} & \text{if} \quad u < u_T \\ w_i^t & \text{if} \quad u \geq u_T \end{cases} \quad , \tag{3}$$

where $i \neq k$, in order to maintain sum to unit.

In slot k of vector $\mathbf{c}$ a randomly drawn combination replaces the selected combination in case it yielded a lower utility:

$$c_k^{t+1} = \begin{cases} \mathrm{rnd}(\mathcal{C} \setminus \{c_i^t : 1 \leq i \leq l\}) & \text{if} \quad u < u_T \\ c_k^t & \text{if} \quad u \geq u_T \end{cases} \quad , \tag{4}$$

where $\mathcal{C}$ is the set of all combinations of $n-1$ elements out of m candidates, and rnd is a function that given a set returns a random element.

The initial probability vector, $\mathbf{w}^0$, may have random values or constant value l^{-1}. It has been shown that the choice of $\mathbf{w}^0$ does not change game dynamics (Mariano et al., 2009a). In order to give a fair chance to all initial combinations, we prefer the uniform distribution.

The rationale for the drastic update is that combinations that contain free riders, exploiters, etc., are removed from the pool. It explores new combinations because it is always replacing lower ones. Although the replacing combination has initially a lower probability to be selected, it may absorb the probabilities of other lower combinations. An important aspect is that combinations with only cooperators never leave the pool and absorb the probabilities of lower combinations. This means that in the long run, the probability mass of combinations with cooperators approaches 1.

If there are no good combinations, then the pool will never stabilise, with combinations constantly entering. Their time in the pool will be proportional to their cooperation level.

Smooth Update – Policy B

This update policy has a parameter $\epsilon < 1$ that determines when the combination vector is updated. Whenever a combination yields a utility lower than u_T, its probability decreases as it is multiplied by a factor δ lower than 1. If the probability reaches value ϵ we consider that the corresponding combination should leave the pool. It will be replaced by a new randomly generated combination. In order be fair, the new combination is assigned probability l^{-1}. This means that we have to decrease the other combinations' probabilities. We opt for a decrease proportional to their value. Formalising, the probability to select combination c_k is updated as:

$$w_k^{t+1} = \begin{cases} l^{-1} & \text{if} \quad u < u_T \wedge w_k^t \leq \epsilon \\ \delta w_k^t & \text{if} \quad u < u_T \wedge w_k^t > \epsilon \\ w_k^t & \text{if} \quad u \geq u_T \end{cases} \quad , \tag{5}$$

and the probability to select the other combinations is:

$$w_i^{t+1} = \begin{cases} w_i^t \dfrac{1-l^{-1}}{\sum\limits_{j \neq k} w_j^t} & \text{if } u < u_T \wedge w_k^t \leq \epsilon \\ w_i^t + \dfrac{(1-\delta)w_k^t}{l-1} & \text{if } u < u_T \wedge w_k^t > \epsilon \\ w_i^t & \text{if } u \geq u_T \end{cases} \quad . \tag{6}$$

The combination vector is updated as follows:

$$c_k^{t+1} = \begin{cases} \mathrm{rnd}(\mathcal{C} \setminus \{c_i^t : 1 \leq i \leq l\}) & \text{if } u < u_T \wedge w_k^t \leq \epsilon \\ c_k^t & \text{otherwise} \end{cases} \tag{7}$$

The first probability vector, $\mathbf{w}^0$ is initialised with constant value l^{-1}, in order to give a fair chance to all initial combinations.

As long as the pool size is smaller than the number of good combinations, in the long run, the pool will only contain those combinations. Again, a good combination is never replaced. If the pool size is higher, then bad combinations will always have in the long run a probability of being selected ranging from ϵ to l^{-1}.

Drastic Proportional Update – Policy C

The probability of a good combination is only indirectly increased by the update policies we have described. A better solution is a probability proportional to the utility obtained with the corresponding combination. Even among good combinations there can be differences due to different types of cooperators in the population. For instance, some candidates may behave stochastically in terms of their cooperativeness.

In this policy, vector $\mathbf{w}$ is best described as a weight vector. Whenever a combination is selected, if the utility obtained, u, is higher than threshold u_T its weight is updated in order to approach the true combination utility. If the utility obtained is lower than u_T, a random combination is selected and the weight reset to some value.

Like in previous policies, we opt for having an initial weight vector with identical values, $w_k^0 = u_T - \underline{u}$. The decision threshold u_T is used when a new combination enters the pool. Parameter $\delta < 1$ is used to gradually approximate the true utility of a combination. Formalising, the update policy is:

$$w_k^{t+1} = \begin{cases} \delta w_k^t + (1-\delta)(u - \underline{u}) & \text{if} \quad u \geq u_T \\ u_T - \underline{u} & \text{if} \quad u < u_T \end{cases}, \quad (8)$$

where $\underline{u}$ is the lowest utility obtained by a player. The combination vector is updated using the policy described by Equation (4).

This policy is general enough to encompass games with negative utilities. To guarantee this, weights assigned to new combinations are shifted by $\underline{u}$.

As in the previous vector update policies, if the pool size is smaller than the number of good combinations, in the long run the pool will only contain those combinations. Again, a good combination is never replaced. If the pool size is higher, then bad combinations will always have, in the long run, a non-zero probability of being selected, which is less than l^{-1} and higher than:

$$\frac{u_T - \underline{u}}{u_T - \underline{u} + (l-1)(\overline{u} - \underline{u})} \quad (9)$$

which corresponds to the limit probabilities of a pool with $l-1$ perfect combinations. Although this value is inversely proportional to l, if we increase l but other parameters remain constant (in particular number of good combinations), the probability mass of good combinations decreases.

Adaptive Utility Threshold

As the goal of this model is for cooperative players to only select their kin, the ideal value for threshold u_T is the utility obtained by a strategy profile composed of only cooperative strategies. We will use parameter u_P to represent this value. It may happen that a player does not have enough pure cooperative partners. Therefore, no single partner combination will remain forever in vector $\mathbf{c}$. In this case, the player could lower threshold u_T in order to reach a stable regime.

The player should raise the threshold if vector $\mathbf{c}$ is stable. But we must take care in order to guarantee that the threshold does not oscillate too much. We opt for a regime similar to the thermal one used in a Simulated Annealing algorithm (Kirkpatrick et al., 1983).

The rule to update the utility threshold is based on the number of changes that occurred in the combination vector in the last h games. The rationale being that a high number of changes, larger than h_T, means that there are not enough cooperative candidates and the threshold should decrease. On the other hand, no changes means that the threshold can increase in order to select better cooperators. The model has additional parameters that control the change in utility threshold, β and γ. The utility threshold update policy is:

$$u_T^{t+1} = \begin{cases} (1-\beta e^{-\gamma t}) u_T^t + \beta e^{-\gamma t} u_P & \text{if} \quad \#c = 0 \\ (1-\beta e^{-\gamma t}) u_T^t + \beta e^{-\gamma t} \underline{u} & \text{if} \quad \#c > h_T \\ u_T^t & \text{otherwise} \end{cases} \quad (10)$$

where $\#c$ represents the number of changes in the combination vector in the last h games. Parameter $\beta \in [0,1]$ controls the magnitude of change in u_T. For $\beta = 0$ there is no change. The value of $\gamma \in [0,1]$ controls the decay of u_T with the number of games. For $\gamma = 0$ there is no decay and for other values we may consider that the threshold stabilises after $10/\gamma$ games.

The initial utility threshold is set to the Pareto utility, $u_T^0 = u_P$. The threshold can never go bellow the lowest utility obtained by a player, $\underline{u}$.

Discussion

We have presented three policies of partner selection suitable for any n-player game with stochastic players. We stress the fact that in the three models a player selects partners based only on private information. This information consists on the utilities obtained in each game. The utility is not necessarily equal to the payoff a game ascribes to a player. It may depend on the payoff of all players, as in the utility of *homo equalitarium* (Gintis, 2000a).

Update policy A is identical to the policy presented in Mariano et al. (2009a). However here we extend that model to select partner combinations instead of a single partner. Moreover we can handle stochastic strategies. Update policy A only requires one combination of good partners while update policies B and C require l combinations of good partners. If there are fewer, then with update policies B and C there will be bad combinations in the pool with non-zero probability. While this is a drawback, update policy B does not promptly remove bad combinations from the pool, but only removes them when their probabilities are lower than threshold ϵ. This allows combinations with stochastic players to remain longer in the pool. As for policy C, the probability of a partner combination is proportional to their utility, thus the best combinations are favoured over bad ones.

All models aim at keeping the combinations that yield the highest utility in the long run. Despite the computational effort needed by the policies, it is rational for a player to follow one of them instead of randomly selecting partners.

The vector update policy is performed by the player that selects partners, but it can also be performed by players that are selected. In particular, if the combination, from the viewpoint of the partner exists in his pool, then he can apply one of the three update policies. This is an improvement over previous work (Mariano et al., 2009b) as the partner selection model was only used by the player that selected partners. In 2-player games, if the player has enough computational resources its pool can cover the entire population of candidates.

This paper also introduces an adaptive process to modify the utility threshold used in all the policies. The goal of this adaptation is to stabilise the contents of the combination vector while maintaining a higher probability to select the best possible combinations. For instance, if the number of pure cooperators is scarce, a player should accept, as good, combinations with stochastic cooperators, which provide sub-optimal utilities (less than u_P). Also, the adaptive process may recover from a situation where the threshold is low and new good candidates appear.

Experimental Analysis

We have performed simulations using the PGP game (Boyd et al., 2003; Hauert et al., 2002). This game is commonly studied to analyse cooperative dilemmas. Moreover, it is a n-player game. We analysed the games played by a particular player paying special attention to the evolution of vectors $\mathbf{w}$ and $\mathbf{c}$ and the number of games played with every candidate.

Simulation Description

In the PGP game, a player that contributes to the good, incurs in a cost c. The good is worth g for each player. Let x be the proportion of players that provide the good. The payoff of a player that provides the good is $gx - c$ while players that defect get gx. The game has a single iteration. The strategy used by players is probabilistic and is defined by parameter p which is the probability to provide the good. We assume that the utility of a player is equal to its payoff. In the simulations we set $g = 10$ and $c = 4$. The number of players in a game varied between three and five.

Partner candidate population composition was chosen in order to illustrate interesting behaviour of update policies: with update policy A the population has fewer than $n - 1$ cooperative partners; with update policies B and C the number of combinations with only cooperative partners is less

		Players		
		3	4	5
Candidates	10	45	120	210
	30	435	4060	27405
	50	1225	19600	230300
	100	4950	161700	3921225

Table 1: Number of available partner combinations for different number of candidates and players in the PGP game.

id	strategies			
P_1	2	$(p = 1)$	8	$(p = 0.5)$
P_2	3	$(p = 1)$	7	$(p = 0.5)$
P_3	4	$(p = 1)$	6	$(p = 0.5)$
P_4	2	$(p = 1)$	18	$(p = 0.5)$
P_5	3	$(p = 1)$	17	$(p = 0.5)$
P_6	4	$(p = 1)$	16	$(p = 0.5)$

Table 2: Candidate populations used in the simulations.

than l. Table 1 lists the number of available partner combinations per population size and players.

Different hand-tailored partner candidate populations were used. They varied in the number of cooperative strategies and population size. Table 2 presents the candidate populations used. The number of cooperative partners varied between two and four. The rest of the population was filled with mixed strategies that cooperated with probability 0.5. Population size was either ten or twenty. The size of the population of candidates was chosen to reflect the size of small communities (Price, 2006).

Pool size, represented by parameter l, was selected from set $\{10, 20, 30\}$. A higher value means more combinations may be analysed, but there will be more bad combinations in the pool.

The player that was used to analyse the partner selection algorithm used a pure cooperative strategy $(p = 1)$. The player ran the algorithm during $R = 1000$ games. After each game, we measured vectors $\mathbf{w}$ and $\mathbf{c}$, the selected combination, utility threshold u_T and the player payoff.

All probability vector update policies used $\delta = 0.5$. Regarding update policy B extra parameter, ϵ, instead of using an absolute value, in the simulations we used $\epsilon = l^{-1}\epsilon'$, with $\epsilon' \in \{0.2, 1\}$.

Regarding the adaptive utility threshold policy, for update policies A and C a history size of 20 was used. Since update policy B only updates the probability vector when the probability is lower than parameter ϵ different history sizes and values for parameter ϵ were tested in order to observe any relevant behaviour. History size was taken from set $\{20, 40, 60, 80, 100\}$. As for the remaining parameters, we set $\beta = 0.1$, $\gamma = 0.002$ and $h_T = 8$.

To obtain statistically significant results, 30 simulations were performed for each parameter combination. The appendix describes the implementation of the vector update policies and other relevant details.

Result Analysis

Figure 1 shows the average and standard deviation per each parameter combination of the following values: average payoff, number of changes in combination vector and last utility threshold u_T^R. The key is shown separately in Figures 1a and 1b.

Average payoff is higher with policy A mainly due to bad combinations having a low probability value. Recall that in this policy the probability of good combinations never decrease. This causes bad combinations to have a probability approaching zero. In contrast, policies B and C decrease the probability of combinations (good ones included) when a new combination enters the pool. Therefore, in these two policies, bad combinations will always have a non-zero probability of being selected. Average payoff increases with the number of cooperators in the candidate population while in most parameter combinations it decreases with pool size. The bigger is the number of cooperators the higher is the number of available partner combinations. The bigger is the pool size the higher is the probability to select bad combinations. Average payoff is inversely proportional to candidate population size. The reason being the higher number of uncooperative partners.

As for changes in the probability vector, update policy A has lower values compared with the other update policies. A higher number means that a player takes longer to find a suitable combination of partners. There is not a clear trend on the number of changes versus other parameters: in some settings the number of changes is proportional to pool size. In update policy A in particular, when the number of cooperators is equal to or higher than n, the number of players in a game, there are few changes. There are simulations with candidate population size equal to 20 (results not shown) where the number of changes in $\mathbf{c}$, the combination vector, is higher then the corresponding parameter combination but with size equal to 10. The reason being the higher number of uncooperative partners.

The plots of u_T^R, the last utility threshold, show that update policy A has slightly larger values than policy C. In simulations where the number of cooperative partners is equal to $n-1$, the best payoff a cooperative player can get is $g(n-1)/n-c$. This is a reasonable value for u_T as it guarantees a combination of partners where all but one are cooperative. For other values of the number of cooperative partners and number of players, Table 3 presents the best payoff a cooperator can obtain.

The simulations where the number of cooperators in candidate population is equal or higher than $n-1$ are a special case for update policy A. This policy is able to find a combination of only cooperative partners, thus the threshold is

		Players		
		3	4	5
Cooperators	2	$\frac{2g}{3}-c$	$\frac{2g}{4}-c$	$\frac{2g}{5}-c$
	3	$g-c$	$\frac{3g}{4}-c$	$\frac{3g}{5}-c$
	4	$g-c$	$g-c$	$\frac{4g}{5}-c$

Table 3: Best payoff obtained by a cooperative player per number of players and number of cooperators in candidate population.

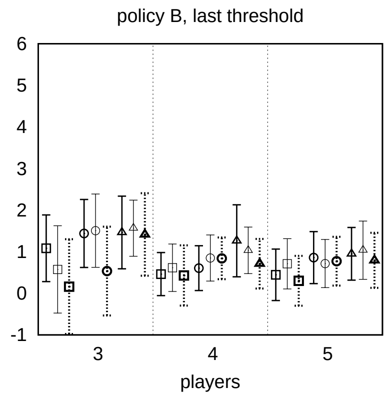

Figure 2: Plot of average and standard deviation of u_T^R from simulations where negative values where observed. Results from simulations with update policy B, $\epsilon' = 1$, population size is 20 and history size is 60.

nearer $g-c=6$.

We comment the results of update policy B separately because of its rule to update the combination vector. Since an update is only triggered when the probability is lower than ϵ, if the probability is very low, then the corresponding combination is selected infrequently. Thus changes in the probability vector are rare. In particular, when history size is 20 and $\epsilon' = 0.2$, no changes occur. Despite this, average payoffs are similar to those obtained by a player that uses update policy C. When we increase history size and use $\epsilon' = 1$ then there are simulations were changes occur, but in a lower quantity when compared to the other policies. As for utility threshold, we observed simulations with negative values (see Figure 2). This is due to a large history size. Let h_s be history size. If there are h_T consecutive rounds with changes in $\mathbf{c}$, then in the following $h_s - h_T$ rounds u_T will be decreased towards the minimum utility obtained in a game. Recall that the minimum utility in PGP is $g/n - c \approx -1$ (all players do not cooperate except one). When changes are scarce, the utility threshold remained at u_P.

The plots in Figure 1 only show an inversely relation be-

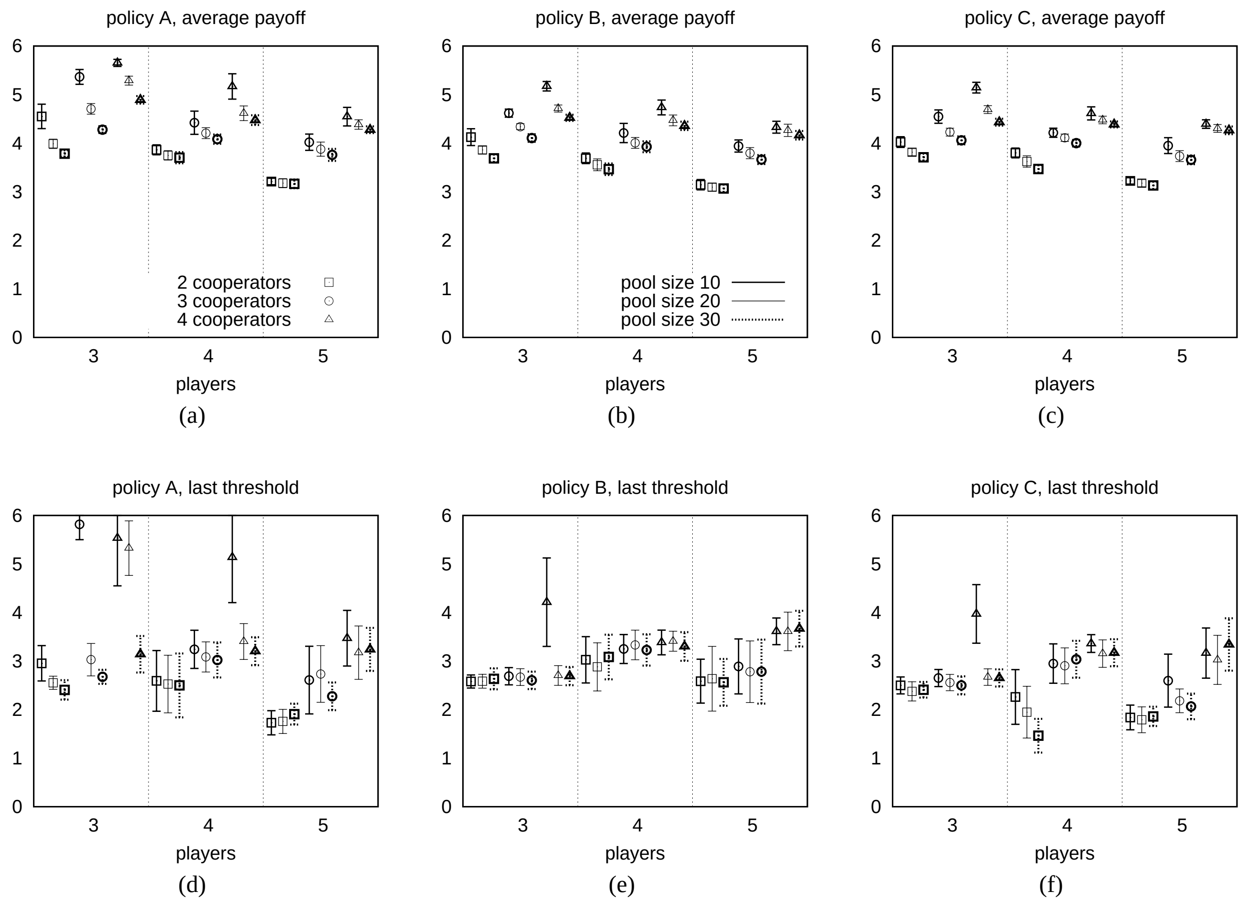

Figure 1: Results from the simulations with population size equal to 10 and history size equal to 20. Plots on the left column are from update policy A, the middle column has plots with update policy B with $\epsilon' = 1$ while the rightmost refers to update policy C. Error lines show the average and standard deviation of, from top to bottom, average utility, number of changes in combination vector, **c**, and last utility threshold, u_T^{1000}. Due to layout reasons, the key is displayed in Figures 1a and 1b. Line style represents pool size, from left to right: bold solid $l = 10$, mild solid $l = 20$, dotted $l = 30$. Point style represents number of cooperators in candidate population, from left to right: square $\#(p = 1) = 2$, circle 3, triangle 4.

tween average payoff and pool size. In order to search for other relations between parameters, we performed significance tests for the product-moment correlation coefficient at 0.5% level between parameters and measured values. We have found that average payoff is directly proportional to the number of cooperators (in the partner candidate population) and inversely proportional to the number of players in a game (Tables 4a and 4b). As for the number of changes of the combination vector and the last value of the utility threshold, u_T^R, we did not find a clear correlation. However, analysing in more detail, we could see that, for policies A and C, there is an inversely proportional correlation between the average payoff and the pool size. Also, for policies A and C, u_T^R is correlated with the number of cooperators and the number of players. It is directly proportional to the number of cooperators and inversely proportional to the number of players. For most of policy B cases there is no correlation. This can be explained by its use of parameter ϵ. For instance, when ϵ' is 0.2 the chance of a combination being replaced is so low that the utility threshold mostly remains unchanged.

In Table 4d we see the results obtained while maintaining all parameters and varying only the update policy. There is a clear correlation between the policy and changes, average payoff and u_T^R. It indicates that policy B has the worst results and that policy A is the best. Nevertheless we made a deeper comparison between policies A and C (in Tables 4e and 4f). The result observed in Table 4d while still favouring policy A is not so clear. Policy C in a few cases obtains better results and in some more is comparable to A.

Conclusions

We have recast partner selection in n-player games, with stochastic strategies, as a problem of selecting the right combination of players. To support this approach, each player maintains a pool of partner combinations and a probability it associates to each combination. We have presented three policies to update probabilities and to replace player combinations. We have given informal proofs of how a player will only select combinations with cooperative players. One of these policies, A, is able to increasingly select a single good combination, if there is only one. We have also presented a model that updates a threshold for policy replacement used by the three policies. This update aims at adapting a player to situations were there are not enough cooperative partners.

The experimental part focused on the interesting behaviour of a player, which is the situation of not having sufficient cooperative partners. Results show that with the threshold update policy a player was able to select combinations mostly with good cooperators. Results also showed that the threshold converged to a reasonable value.

A drastic update policy, A, is able to obtain better results in most cases. This confirms that the capacity of policy A to increase the probability of selecting a good combination, even if it is the only one in the pool, is a significant advantage for partner selection in n-player games.

As for future work, we aim at improving the selection of partner combination. Instead of randomly picking partners to the new combination, a proportional selection should be done. We plan to assign to each partner a probability of entering a combination.

We are currently investigating the conditions that favour the evolution of partner selection.

As we have said, our model does not prevent a player from begin selected by uncooperative. We also plan to investigate the possibility of refusal. However, this raises the question of the refusal payoff. As we have mentioned some authors chose a payoff higher than the minimum payoff in the original game, thus altering the equilibria in the game.

References

Aktipis, C. A. (2004). Know when to walk away: contingent movement and the evolution of cooperation. *Journal of Theoretical Biology*, 231:249–260.

Axelrod, R., editor (1997). *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton Studies in Complexity. Princeton University Press.

Boyd, R., Gintis, H., Bowles, S., and Richerson, P. J. (2003). The evolution of altruistic punishment. *Proceedings of the National Academy of Sciences*, 100(6):3531–3535.

Coricelli, G., Fehr, D., and Fellner, G. (2004). Partner selection in public goods experiments. *Journal of Conflict Resolution*, 48(3):356–378.

Ehrhart, K.-M. and Keser, C. (1999). Mobility and cooperation: On the run. CIRANO Working Papers 99s-24, CIRANO.

Fudenberg, D. and Tirole, J. (1991). *Game Theory*. MIT Press.

Gintis, H. (2000a). *Game Theory Evolving - A problem-centered introduction to modeling strategic interaction*. Princeton University Press.

Gintis, H. (2000b). Strong reciprocity and human sociality. *Journal of Theoretical Biology*, 206:169–179.

Hauert, C., Monte, S. D., Hofbauer, J., and Sigmund, K. (2002). Volunteering as red queen mechanism for cooperation in public goods games. *Science*, 296:1129–1132.

Kirkpatrick, S., Jr., C. D. G., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Nature*, 220(4598):671–680.

Mariano, P., Correia, L., and Grilo, C. (2009a). How to build the network of contacts. In Lopes, L. S., Lau, N., Mariano, P., and Rocha, L. M., editors, *New Trends in Artificial Intelligence*, pages 65–76. Universidade de Aveiro. 14th Portuguese Conference on Artificial Intelligence, EPIA 2009.

Mariano, P., Correia, L., and Grilo, C. (2009b). Selection of cooperative partners in n-player games. In Kampis, G. and Szathmáry, E., editors, *Advances in Artificial Life*.

Matsumoto, M. and Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30.

	changes	avg payoff	u_T^R
$+$	12	72	46
$-$	24	0	0
$\times$	36	0	26

(a) Correlation with number of cooperators in partner candidate population

	changes	avg payoff	u_T^R
$+$	10	0	4
$-$	9	72	40
$\times$	53	0	28

(b) Correlation with number of players in the game.

	changes	avg payoff	u_T^R
$+$	17	0	1
$-$	10	53	19
$\times$	45	19	52

(c) Correlation with pool size.

	changes	avg payoff	u_T^R
$+$	54	54	0
$-$	0	0	49
$\times$	0	0	5

(d) Correlation with update policies ordered as B with $\epsilon' = 0.2$, B with $\epsilon' = 1$, C and A.

	changes	avg payoff	u_T^R
$+$	4	34	24
$-$	18	0	0
$\times$	32	20	30

(e) Correlation with update policies A and C ordered with C first then A.

	changes	avg payoff	u_T^R
$+$	11	42	31
$-$	23	0	0
$\times$	20	12	23

(f) Correlation with update policies A and C, with significance level 5%.

Table 4: Correlation significance tests at 0.5% level except in 4f. The values represent the number of parameter combinations with $+$ positive, $-$ negative and $\times$ no correlation. All possible parameter combinations were used except for history size fixed at 20.

Orbell, J. M. and Dawes, R. M. (1993). Social welfare, cooperators' advantage, and the option of not playing the game. *American Sociological Review*, 58(6):787–800.

Price, M. (2006). Monitoring, reputation, and "greenbeard" reciprocity in a Shuar work team. *Journal of Organizational Behavior*, 27:201–219.

Appendix

Implementation Details of the Probability Vector Update Policy

The probabilities in vector $\mathbf{w}$ where represented as partial sums of 31 bit integers. The motivation to use integers is due to the fact that floating point division can yield approximate values and thus the sum of the probabilities may not add up to 1. As we used integers, whenever a probability was decreased, the others were incremented by the quotient of the division presented in the policy equations (see for instance Equation (3)). As for the remainder, a random probability was chosen.

The use of partial sums allows a faster algorithm, with time complexity $O(\log l)$, to select a combination to play with. A random integer in the range $[0, 2^{31}]$ was chosen and then a binary search was performed. Although updating the probability vector has time complexity $O(l/2)$, because on average half partial sums must be updated, when the vector converges only selections take place.

As for the pseudo-random number generator, we used an implementation of the Mersenne Twister, a uniform generator with a large period (Matsumoto and Nishimura, 1998).