Social Learning Strategies Tournament

Details of Stage II and Final Results

Luke Rendell and Kevin Laland, University of St Andrews
January 5, 2009

This document provides a report on the second and final stage of analysis of submitted entries to the social learning strategies tournament. 104 entries were received in total, of which 10 progressed to Stage II, the 'melee' stage. We congratulate the authors of all the strategies that progressed to Stage II. The large number of entries made the competition very challenging, and progressing past the first stage is a major achievement in its own right. Here we describe the second and final stage of the evaluation process, and present the final results of the tournament. Any comments on these results are welcome; please send them to Luke Rendell <ler4@st-andrews.ac.uk>. Queries are also best directed to the same.

Stage II details

The simulations each contained a single population of 100 individuals. In each simulation, all ten of the strategies selected in Stage I competed simultaneously (for details of Stage I, see www.intercult.su.se/cultaptation/tournament_details.php). Each simulation started with a population consisting purely of a simple strategy that did not use any social learning, called *innovateOnceThenExploit*. This strategy plays INNOVATE on the first round of its life and subsequently plays EXPLOIT continually with the single act that it acquired on the first round[a]. We used this strategy simply to avoid giving any of the ten competing strategies any advantage or disadvantage resulting from being already established in the population. Mutation was introduced from round 1, providing the competing strategies with equal opportunity to invade. Simulations were run for 10,000 rounds. The mean frequencies of each strategy in the last quarter of each run (i.e. the last 2,500 rounds) were recorded as the scores for each strategy in that simulation. Strategies were then ranked on their average score across all simulations. Simulations were run on desktop PCs running Matlab v7.5.

Simulation conditions

We ran two sets of conditions, which we termed *systematic* and *random*. For the systematic condition set, we selected a number of values for each of the four varied parameters, p_c, $n_{observe}$, $p_{copyActWrong}$, and $\sigma_{copyPayoffError}$[b] (Table 1). Fifty simulations were run with each of the 280 possible combinations of these parameter values for a total of 14,000 simulations.

To check that the results of this process were not unduly affected by the specific parameter values we chose, we also ran random conditions, where parameter values

[a] This strategy was entered independently in the tournament as *exploitOneInnovation*. It did not progress past Stage I.

[b] These were parameters unspecified in the tournament rules; p_c is the independent probability of the payoff of an act changing between rounds, $n_{observe}$ is the number of individuals observed when the move OBSERVE is played, $p_{copyActWrong}$ is the probability that an observed payoff is associated with an incorrect act (i.e. an error in social learning) and $\sigma_{copyPayoffError}$ is the standard deviation of the error distribution of observed payoffs.

were chosen at random from statistical distributions weighted in accordance with the values chosen for the systematic conditions (Figure 1). We selected 1,000 unique sets of parameters values in this way and ran a single simulation with each set of values.

Parameter	Values						
p_c	0.001	0.005	**0.01**	0.05	0.1	0.2	0.4
$n_{observe}$	**1**	2	6	12			
$p_{copyActWrong}$	0.01	**0.05**	0.1	0.25	0.5		
$\sigma_{copyPayoffError}$	**1**	10					

Table 1: Details of further conditions run for top 24 strategies; values in bold are those used for the main pairwise contest in Stage I.

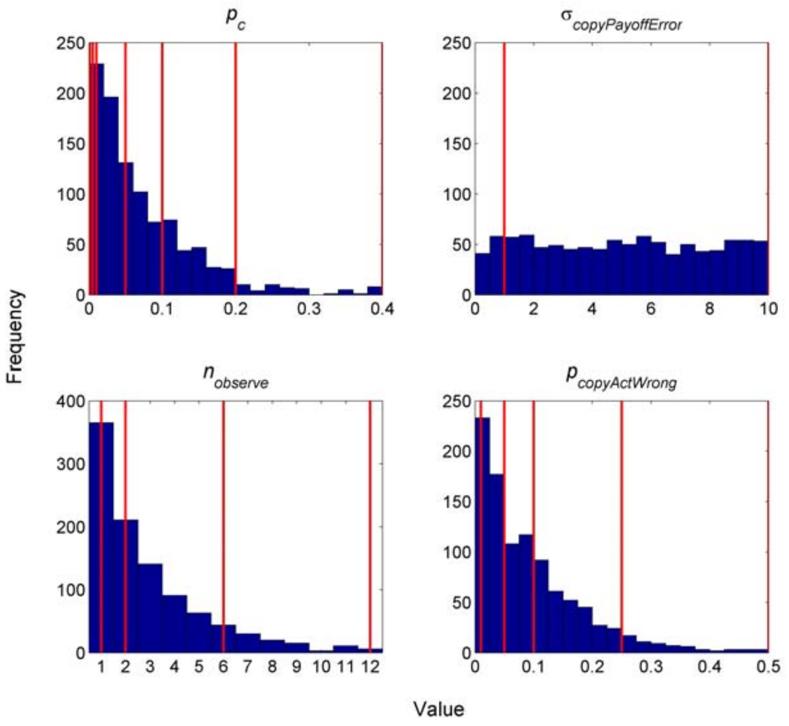

Figure 1: Distributions of parameter values chosen. Blue bars are histograms of values chosen for the random conditions, red lines show values selected for systematic conditions

Results

The overall results for the systematic conditions are given in Table 2, and those for the random conditions in Table 3. The average results over all the melee simulations are plotted in Figure 2. The top three ranked strategies are identical under both sets of conditions, and in both the strategy *discountmachine* is a convincing winner. We therefore name this strategy the overall winner of the tournament, and congratulate its authors, Dan Cownden and Tim Lillicrap of Queen's University, Ontario, Canada, on their success.

The winning strategy was sophisticated and complex, incorporating learning procedures based on neural networks. The strategy plays OBSERVE on its first round, and then plays INNOVATE only if OBSERVE failed (this was actually a rather common opening sequence in the entries). This is the only time the strategy plays INNOVATE, and it subsequently only learns through observation. The strategy also attempts to estimate many things about the environment, including the probability of change in the environment (p_c), the mean of the payoff distribution, the number of individuals observed in any social learning event ($n_{observe}$), and the reliability of social learning as indexed by the correlation between payoffs of the same act when observed or exploited. It uses these estimates to produce an expected payoff for each act in its repertoire if it were exploited from the current round until death, and an expected payoff from playing OBSERVE followed by exploiting an act until death, and selects the move with the highest expected payoff. There are a number of ways in which this strategy stands out and that may have contributed to its success. The first is that it attempts to characterise the environment in a highly comprehensive way, extracting a lot of information from its prior experiences, and is responsive to that information. The second is that this strategy explicitly makes decisions based on expected lifetime outcomes, rather than immediate payoff benefits. Finally, the strategy is noteworthy for the sophistication of the mathematics it uses to calculate expected payoffs. For example, the reliability of social learning is estimated by a regression analysis of the payoffs from observing and exploiting the same act, while the expected value of exploiting an act is calculated from a weighted average where the weights depend on the current estimate of p_c. We are currently analysing the actual moves played by each strategy to try and determine which of these factors, if any, were most critical to its success.

Rank	Strategy	Score
1	*discountmachine*	0.3481
2	*intergeneration*	0.2388
3	*prospero*	0.0919
4	*wePreyClan*	0.0673
5	*valueVariance*	0.0614
6	*dynamicAspirationLevel*	0.0566
7	*copyWhenYoungThenLearnWhenPayoffsDrop*	0.0558
8	*livingdog*	0.0408
9	*rummer*	0.0221
10	*whenTheGoingGetsToughGetScrounging*	0.0167
11	*innovateOnceThenExploit*	0.0005

Table 2: Results for systematic conditions. Scores are given to 4 decimal places.

Rank	Strategy	Score
1	*discountmachine*	0.4023
2	*intergeneration*	0.1224
3	*prospero*	0.0970
4	*dynamicAspirationLevel*	0.0850
5	*copyWhenYoungThenLearnWhenPayoffsDrop*	0.0850
6	*valueVariance*	0.0760
7	*wePreyClan*	0.0746
8	*rummer*	0.0291
9	*livingdog*	0.0167
10	*whenTheGoingGetsToughGetScrounging*	0.0120
11	*innovateOnceThenExploit*	0.0001

Table 3: Results for random conditions. Scores are given to 4 decimal places.

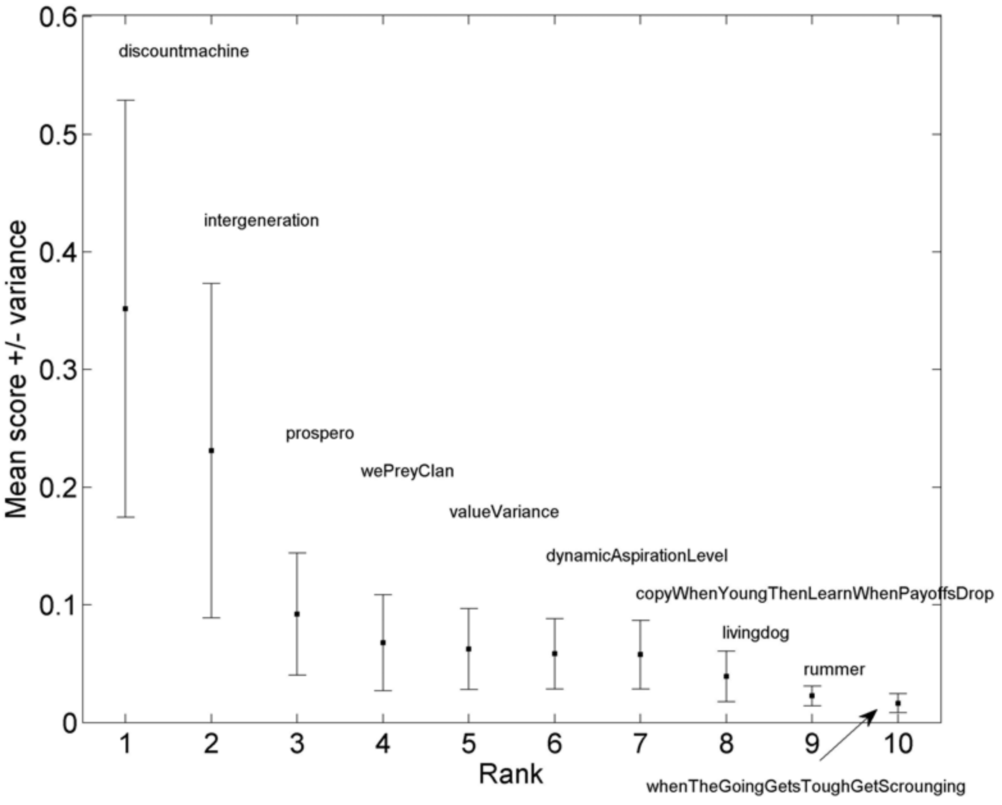

Figure 2: Mean and variance of scores over all melee simulations

School's prize

We would also like to congratulate Ralph Barton and Joshua Brolin, who are both pupils at Westminster School, UK, whose strategy, *whenTheGoingGetsToughGet-Scrounging*, placed 9th in Stage I. It is a tremendous achievement for these young students to have done so well in this competition, in which the vast majority of entrees were by professional academics. In recognition of this achievement, Ralph and Joshua are to be awarded the prize of £1000 given to the best entry from a school.

APPENDIX

Here we list the prose descriptions of the strategies competing in the melee, ranked by final score.

Rank	Strategy	Prose description
1	*discountmachine*	Our creature does three major things: First it estimates/calculates, what we believe to be all the pertinent parameters of the simulation as well as a few other quantities that we believe to be useful. These are P_c, the mean of the payoff distribution, the mean of of the observed values, the correlation between observe and exploit values of the same action, N_observe, and where applicable the number of data points used to make these estimates. Second it uses some of these parameters to estimate the expected payoff for performing each action in its repertoire. Once it has a best exploit chosen from its repertoire it compares the value of Exploiting to the value of Observing using a geometric discounting scheme based on our estimate of P_c and the given probability of death. Lastly a machine learned function, takes into account N_observe and the estimates on the reliability of observing and P_c to adjust the value of Observing accordingly. Our creature then chooses whichever action has the higher perceived value, Observing or Exploiting. As a side note our creature only Innovates when it has an empty repertoire and observe doesn't work, which typically is only on the first turn of a simulation.
2	*intergeneration*	My main idea is (although it seems not be as good as I expected) that an important information for the young is, how much is the "good" payoff (with how much I can be happy). If I have so much or more, I would just EXPLOIT until it changes, otherwise I would 8 times exploit and once observe. The important trial is that the old could "say" something to the young, by "signaling" something to the young. The signal consists of doing an act whose number is divisible by 8. If the fraction is 1,2,3,4 this means that "payoff 8 is very good", 5,6,7,8 means "payoff 20 is very good" and 9,10,11,12 means “payoff 40 is very good”. If the old does not have this in his repertoire, he innovates. If he has more than one of these 4 possible "symbol" acts, he uses that with highest payoff -- because it is a higher chance that this will diffuse. Even the "opponent" can help spread out this signal, without knowing that.

Rank	Strategy	Prose description
3	*prospero*	1. Prospero estimates the mean payoff by taking the mean of all the elements in the 4th row of its history. All elements in this row are included in the mean, no matter whether they are actual payoffs received from exploiting or payoffs learned by innovating or observing. 2. Prospero determines the best act in its current repertoire by finding the highest value in the second row of MyRepertoire. When Prospero exploits, it always exploits the best act in its repertoire. In the case where more than one act in the repertoire has the same best payoff in the second row of MyRepertoire, Prospero exploits one of the acts at random from among those acts whose payoff is equal to the best payoff. 3. Prospero compares its last payoff with the mean payoff. If the last payoff is higher than the mean, then Prospero is satisfied, and it continues to exploit its best act. Furthermore, Prospero never observes or innovates twice in a row, so if the last move was observe or innovate, Prospero always exploits. 4. If the last move was exploit and the last payoff was less than or equal to the mean payoff, Prospero is not satisfied. It then chooses its move randomly with probabilities controlled by two parameters a and b. Observe with probability ab, Innovate with probability a(1-b), Exploit current best act with probability (1-a). I will refer to observe plus innovate together as learning. The parameter a controls the ratio of learn to exploit, and the parameter b controls the ratio of observe to innovate, given that a learning move is played. 5. Prospero estimates the probability of change of the environment, pc, from its history. It chooses to learn with a low probability when pc is low because it is very likely already exploiting a high-payoff act, so it does not want to waste a move on learning. It chooses to learn with a high probability when pc is intermediate because it is important to learn in order to keep up with changes in the environment. It chooses to learn with a low probability when pc is very high because learning has little value if the payoff changes frequently. Prospero counts nsame , the number of times that it exploited the same act as the last round and obtained the same payoff as the last round, and ndiff , the number of times that it exploited the same act as the last round and obtained a different payoff to the last round. Prospero uses the estimate pc = (1+ndiff) / (10+ndiff+nsame). This starts off with a prior of 1/10 when there is no history and changes gradually from this when the history gets longer. Prospero then determines the value of the parameter a using the following rules: if $pc > 0.3$: $a = 0.1$ if $0.2 < pc \leq 0.3$: $a = 0.25$ if $0.15 < pc \leq 0.2$: $a = 0.5$ if $0.05 < pc \leq 0.15$: $a = 1.0$ if $pc < 0.05$: $a = 0.5$ 6. Prospero estimates the diversity of strategies being used in the population from its previous observations. It chooses to observe more frequently (i.e. larger value of b) when its estimate of diversity is high, and to innovate more frequently when its estimate of diversity is low. The rationale is that when the diversity is high it is best to observe, because the acts being performed by others will be much better than random acts, whereas it is not so useful to observe when the diversity is low, because most likely you will observe something that is already in the repertoire. Let nrep = number of acts in the current repertoire, n0 = number of observations made before (= number of 0's in row 2 of history), and n1 = number of innovations made before (= number of -1's in row 2 of history). The number of different strategies that have been observed is nrep – n1. This is less than n0 when the diversity is low, because the same act has been observed more than once. Prospero uses the following rule to set the parameter b: (1+nrep-n1)/(1+n0). In the case where nobserve > 1, n0 = nobserve × number of turns on which observe was played. However, the same formula for b is applicable in either case. 7. Special rules apply at the beginning of Prospero's life when it has no strategies in its repertoire. On the first turn it always observes, on the grounds that an observed act will have a much higher payoff than a randomly innovated act. If no act is observed on the first round (because no individuals exploited), Prospero innovates on the second round. This case will only arise at the beginning of the simulation when all individuals are started at the same time. Usually a new individual will be born into a population of adults, so there will always be something to observe on the first round.

Rank	Strategy	Prose description
4	*wePreyClan*	In the first round (roundsAlive = 0), the strategy observes. In round two, if no one was observed EXPLOITING in round one, the strategy INNOVATES. If individuals were observed, the strategy chooses the best payoff from myRepertoire, and EXPLOITS it. After the second round, the strategy becomes a hybrid of 3 different tactics. If n=1 (OBSERVE one exploiter at a time), the strategy calculates from myHistory the probability that in subsequent rounds the payoff for a given act (either OBSERVED, EXPLOITED, or INNOVATED) has changed. If this probability is >0.6 (and there at least 5 data points to calculate the probability), the individual chooses the act with the highest payoff from myRepertoire and EXPLOITS. If the probability is <= 0.6, then the strategy looks through myHistory to find the greatest recorded payoff (myHistory(4,:)). If the individual has EXPLOITED for the last two rounds (t-1, t-2) and the payoff from the last round (t-1) was less than (0.45 * the maximum payoff value in myHistory), then the individual will OBSERVE, otherwise it will re-sort myRepertoire and then play the top act from there. If n>1, then the strategy calculates from myHistory the mean and median payoff for all acts (either OBSERVED, EXPLOITED, or INNOVATED). Whichever of the two values is greater is set to be MHigh, and the lesser is MLow. The strategy also calculates a correction factor (CF = 1 – 0.2(1 – n/6)). Every 10th roundsAlive, if the strategy EXPLOITED in the previous round and the payoff was < MHigh*CF, the strategy switches to OBSERVE. Otherwise, it EXPLOITS the act from myRepertoire with the highest payoff. For the other 9 rounds the strategy will shift from EXPLOIT to OBSERVE if previous payoff was < MLow*CF. Thus, every 10th round the strategy becomes more likely to OBSERVE.
5	*valueVariance*	The variable declaration section exists just to make sure that I didn't accidentally start using a non-declared variable. Rounds 0 & 1 are obviously Innovate then Observe. The next paragraph checks to make sure that I have at least two values in my repertoire so that I can start calculating the best move. If it doesn't have two values then it either innovates or observes until it get two values. Once we get past that section we do some math to try to determine the distribution of the payoff values. ExploitChance gets set to a high value if the highest payoff amount in the repertoire is significantly higher than the mean value of payoffs. In the end, the strategy assigns a percentage chance from 1-100 for each of the possible actions. If ExploitChance is > 100 then it's definitely going to exploit that highly valued act. When ExploitChance is <100 the left over percentage (aka NonExploitChance) chance is split up between Observe and Innovate. A variable named ObservationMultiplyer is based on the number of observations made when an OBSERVE is performed (constrained to 3). The ObserveChance is calculated and then a function adds a little more based on the ObservationMultiplyer. Now that we know the percent likelihood of Exploiting and Observing, everything else is innovating. So, the last section finds a random number and determines if it is within ExploitChance, greater than ExploitChance but less than ExploitChance plus ObserveChance, or greater.

Rank	Strategy	Prose description
6	*dynamic AspirationLevel*	DynamicAspirationLevel determines an aspiration level that is slightly higher than the mean payoff of all previous EXPLOIT moves. The difference between the maximun payoff in the repertoire and the aspiration level determines the probability with which an OBSERVE move is chosen vs. an EXPLOIT move. Furthermore, "DynamicAspirationLevel" uses a 'decaying repertoire' where old entries and potentially invalid entries are values less than new ones. The strategy in detail: The strategy "DynamicAspirationLevel" consists of two phases, the first of which lasts until the fourth round of the life time of an individual using this strategy. The second phase last during the remainder of the life time of the individual. 1. Phase One: - Play OBSERVE in the first round alive (when 'roundsAlive = 0') - In the second round alive: If the repertoire is still empty (i.e. nothing could be oserved in the first round) play INNOVATE, otherwise chose the best EXPLOIT move from the repertoire. - In the third round alive, play OBSERVE - In the fourth round, chose the best move from the repertoire 2. Phase Two - determine the mean payoff 'm' from all EXPLOIT moves in the history. - determine the maximum payoff 'M' from all EXPLOIT moves in the history. - calculate an aspiration level 'a' according to the magic formula: a := m + (M-m) / 5 - determine the highest payoff 'ace' in the repertoire. - if the highest payoff in the repertoire is greater or equal the aspiration level ('ace >= a') then set the observataion probability 'p' to 0.01. Otherwise, if the aspiration level 'a' is greater zero, let the observation probability 'p' be the maximum of 0.01 and the difference between the aspiration level and the highest payoff in the repertoire, divided by the aspiration level '(a-ace)/a'. Otherwise, if the aspiration level is zero, let 'p' be one. - generate a random number 'r' between 0 and 1. If the random number is less or equal the observation probability 'p' chose an OBSERVATION move with a chance of 99% and an INNOVATE move with a chance of 1%. Always choose INNOVATE if the aspiration level 'a' was not greater zero! Otherwise chose the best EXPLOIT move from the repertoire. - Decay the repertoire by subtracting 1 from each payoff value registered in the repertoire.
7	*copyWhenYoung ThenLearnWhen PayoffsDrop*	At the beginning of their life individuals always observe two times. In this way the individual gets good knowledge about what is worth to exploit. The second observe accounts for the problem that observing is error prone and that the environment maybe just changed. Afterwards individuals always exploit the trait with the highest known pay-off as long as the highest known pay-off is larger then 80 % of the average of all pay-offs that were ever experienced (exploited, observed and innovated). In this way the individuals tend to exploit only traits with high payoffs without getting too choosy. In case the highest known payoff drops too strongly individuals first always innovate, afterwards they always observe once and then randomly decide to either innovate or observe. To always innovate first is done because the environment just changed and in this case observing might provide outdated information. In the next time step this is not the case anymore, therefore the individuals should observe.
8	*livingdog*	Living Dog is a probabilistic strategy based on two parameters: p and q, where p is the probability of playing EXPLOIT (vs. INNOVATE or OBSERVE) and q is the probability of playing OBSERVE (vs. INNOVATE). The parameter values are defined depending on the agent repertoire and history. More specifically, p is computed as a fucntion of the differences among the sorted payoffs included in the agent repertoire. The underlying assumption is that a jump between two payoffs much larger than the median one implies that the agent already knows an act associated with a high payoff. The value of q depends instead on the relative value of the average earnings when the agent exploited an act learned using INNOVATE and of the average earnings when the agent exploited an act learned using OBSERVE. The function uses a two step decision making process: 1) a random extraction based on p determines whether the agent will EXPLOIT its higher act or try to increment its repertoire; 2) if it does not EXPLOIT, a random extraction based on q determines whether it will OBSERVE or INNOVATE (actually, in order to increase the function speed, the value of q is computed only when the 1st step does not return an EXPLOIT decision). Using as imputs roundsAlive, myRepertoire and myHistory as they appear in Section 3.5 of the "Rules for entry" document, livingdog runs about 3.6 times slower than cwpd.

Rank	Strategy	Prose description
9	*rummer*	The main idea behind RUMmer is that successful behavior in an uncertain environment, and in the presence of other "competitors", must be adaptive. A strategy must be able to adapt to different environmental conditions, for example, how hard it is to attain relatively high rewards, how fast is the environment changing. The relative returns on innovating, observing others, and exploiting what you already know depend also on the strategies of others. They might be asocial innovators, but perhaps conformist social learners. Therefore, in addition to the properties of the environment, we conjecture that a successful strategy should also be able to adapt to the types of strategies that its competitors use. RUMmer tries to bring these considerations together by using propensities for possible moves: INNOVATE, OBSERVE, or EXPLOIT. Propensities are calculated from agent's experience of whether and to what extent each of these moves was beneficial. An agent should innovate more often, the higher the benefits are of the acts he learned by innovating. Analogously, an agent should observe more often if benefits of observed acts are high. Another important aspect of RUMmer is that it is probabilistic. Calculated propensities determine the probabilities of playing each of the three moves in every round. We believe that randomizing the move in every round is an advantageous way to behave in a probabilistic (uncertain) environment. Deterministic strategies may run into trouble as in every round some aspect of the environment is likely to change. Thus, beneficial options calculated from past rounds may not be beneficial later on. Finally, RUMmer implements an idea that it can only exploit its knowledge during exploitation rounds. Innovations and observations have an "opportunity cost" equal to the payoff that could have been earned in that round. Thus, the benefits of innovation and observation are reduced with the actual likelihood of exploitation. The basic mechanism is the following. RUMmer INNOVATEs in the first round, EXPLOITs in the second, and OBSERVEs in the third. From the fourth round on, the following quantities are calculated in every round: 1. Highest known payoff based on myRepertoire (hexp) 2. Mean payoff of innovated actions based on myHistory (minn) 3. Mean payoff of observed actions based on myHistory (mobs) 4. Proportion of rounds in which played EXPLOIT (pexp) Additionally, let denom be equal to: exp(pexp*minn) + exp((pexp*mobs) + exp(hexp) Then choose one of the three possible moves with the following probabilities: P(INNOVATE) = exp(pexp*minn) / denom P(OBSERVE) = exp(pexp*mobs) / denom P(EXPLOIT) = exp(hexp) / denom Playing EXPLOIT is choosing the action with the highest utility in myRepertoire.

Rank	Strategy	Prose description
10	*whenTheGoingGetsToughGetScrounging*	In general, after observing twice to learn (at least) 2 acts, the strategy exploits 4 times in order to get some early payoff and decide if Pc is low or high. If it sees an act's payoff change in these 4 moves it decides Pc is high, and observes on move 7 to indicate this decision to itself in future. Otherwise it exploits. From move 8 on it looks at move 7 to remind itself whether it decided Pc was low or high. If it decided Pc was high it exploits the highest payoff act unless this is less than the mean of payoffs it has seen, in which case it exploits the act it hasn't used for longest. If it decided Pc was low it exploits highest unless the last move was an exploit with a payoff lower than its mean lifetime payoff. In this case it observes. It will also always observe on move 8. If its last move was an observe it checks to see if the act that was observed was the same as the act exploited on the round before the observe. If this is the case it innovates, since it is clear that more acts need to get into circulation. There are two exceptions to this mode of behaviour. Firstly, some small changes are made to the opening made if the Observe on move 1 fails due to nobody exploiting in the previous turn. Secondly, if Nobserve is found to be greater than 2, move 7 becomes an exploit even if it decides Pc is high. The strategy will then continue to act in the mode which it does normally if it had in fact decided that Pc were low. "Nobserve greater than 2", both here and in the pseudocode, means strictly greater than 2, not greater than or equal to 2.