Social Learning Strategies Tournament

Rules for entry

Kevin Laland and Luke Rendell, University of St Andrews
4th January 2008

We are holding an open, computer-based, tournament to determine the most effective social learning strategy or strategies, as part of the EU 'Cultaptation' project. The author(s) of the winning entry will receive a cash prize of €10,000. This document contains background information on the rationale for the tournament, a description of how the tournament will be run, and technical details of how to enter. It is designed to provide all the necessary information needed by anyone wishing to enter the tournament. The primary contact for all issues regarding entries and any queries not covered here is Luke Rendell (ler4@st-andrews.ac.uk).

"This tournament is a wonderful opportunity to advance our understanding of the evolution of social learning, and I was glad to have been able to give advice about the rules. It has my wholehearted support and I hope that as many people as possible will have a go."

Robert Axelrod, University of Michigan

THE TOURNAMENT: BACKGROUND AND OBJECTIVES

Introduction

In recent years there has been growing interest (spanning several research fields, but especially economics, anthropology and biology), in the problem of how best to acquire valuable information from others. Mathematical and computational solutions to this problem are starting to emerge, often using game-theoretical approaches. We judge that the time is now right for a tournament, inspired by Robert Axelrod's famous Prisoner's Dilemma tournament on the evolution of cooperation (Axelrod 1984), but with the objective of establishing the most effective strategies for learning from others. We have received funding to organize such a tournament from the European Commission as part of the EU-NEST 'Cultaptation' project (www.intercult.su.se/cultaptation/). We hope that the competition will increase understanding of, and stimulate research on, social learning strategies, as Axelrod's tournament did for research on the evolution of cooperation.

Background

It is commonly assumed that social learning is inherently worthwhile. Individuals are deemed to benefit by copying because they take a short cut to acquiring adaptive information, saving themselves the costs of asocial (e.g. trial-and-error) learning. Copying, it is assumed, has the advantage that individuals do not need to re-invent technology, devise novel solutions, or evaluate environments for themselves. Intuitive though this argument may be, it is flawed (Boyd & Richerson 1985; Boyd & Richerson 1995; Rogers 1988; Giraldeau et al. 2003). Copying others *per se* is not a recipe for success. This is easy to understand if social learning is regarded as a form of parasitism on information (Giraldeau et al. 2003): asocial learners are information *producers*, while social learners are information *scroungers*. Game-theoretical models of producer-scrounger interactions reveal that scroungers do better than producers

only when fellow scroungers are rare, while at equilibrium their payoffs are equal (Barnard & Sibly 1981). Similarly, theoretical analyses of the evolution of social learning in a changing environment (e.g. Boyd & Richerson 1985; Boyd & Richerson 1995; Rogers 1988; Feldman et al. 1996) reveal that social learners have higher fitness than asocial learners when copying is rare, because most 'demonstrators' are asocial learners who will have sampled accurate information about the environment at some cost. As the frequency of social learning increases, the value of copying declines, because the proportion of asocial learners producing reliable information appropriate to the observer is decreasing. An equilibrium is reached with a mixture of social and asocial learning (Enquist et al. 2007). These mathematical analyses, together with more conceptual theory (e.g. Galef 1995), imply that copying others indiscriminately is not adaptive; rather, individuals must use social learning *selectively*, and learn asocially some of the time. Natural selection in animals capable of social learning ought to have fashioned specific adaptive *social learning strategies* that dictate the circumstances under which individuals will exploit information provided by others (Boyd & Richerson 1985; Henrich & McElreath 2003; Laland 2004; Schlag 1998). At present, it is not clear which social learning strategy, if any, is best. The tournament has been set up to address this question.

Objective for entrants

To enter the tournament, you need to devise a *strategy* – a set of rules that specify when an individual organism should perform an established behaviour from its repertoire (EXPLOIT), when it should engage in trial-and-error learning (INNOVATE) and when it should learn from other individuals (OBSERVE) in deciding how to behave in a spatially and/or temporally variable environment. Performing the right behaviour is important, as fitness depends on how well behaviour is matched to the current environment. However, learning is not free, and fitness costs are imposed each time an individual learns for itself, or samples the behaviour of other individuals in its environment. For the purposes of the tournament, organisms will be assumed to know their own individual histories of behaviour and the fitness payoffs they received.

Strategies will be tested in a computational simulation framework. The specification of the simulations, details on how to enter, and detailed tournament rules are given in the technical details section below. Entrants should ensure they are familiar with this material, as the details given are crucial in ensuring that your strategy will be considered in the tournament.

Strategy evaluation

Strategies will take part in a two-stage competition; this is summarised here and full details are provided in section 2 below.

Stage 1: Strategies will take part in round-robin contests between all pairs of entered strategies. A contest, say between strategies A and B, involves exploring whether strategy A can invade a population containing only strategy B, and vice-versa. Each contest will involve several repeated simulations, with each strategy as the invader 50% of the time. In each simulation, after a fixed number of iterations, the frequency of each strategy (that is, the proportion of the population with that strategy) will be recorded, and the average frequency across repetitions will be the score of that strategy in that contest.

Stage 2: At least the first ten highest scoring strategies will then be entered into a *melee* in which all strategies compete simultaneously in a range of simulation conditions. After a fixed number of rounds, the frequency of each strategy will be the score for that strategy. The procedure will be repeated and the strategy with the highest average score deemed the winner. Participants should therefore try and construct their strategies so that they are likely to work well under most conditions.

Committee

The tournament will be organised and run by Kevin Laland and Luke Rendell, both of the University of St Andrews. A committee has been formed to oversee the running of the tournament, and formally adjudicate when necessary, to ensure that the contest is run transparently. The committee is composed of the organizers plus the following persons, all of whom have expertise relevant to the tournament:

Robert Boyd, University of California, Los Angeles
Magnus Enquist, University of Stockholm
Kimmo Eriksson, Mälardalen University
Marcus Feldman, Stanford University

This committee has been extensively involved in designing of the tournament; we are also very grateful to Robert Axelrod of the University of Michigan for providing important advice and support with regard to the tournament design.

TECHNICAL DETAILS

1. Simulation specifications

Each simulation will contain a population of 100 individuals, and run for up to 10,000 rounds[1]. A single round will consist of the following computational steps:

(i) Individuals are selected sequentially to choose a move (see below) until all individuals have played.
(ii) Individuals reproduce with probabilities proportional to their average lifetime payoffs.
(iii) The environment changes.

1.1 Environment and behaviour

1.1.1 The environment will be represented as a 'multi-arm bandit' wherein actors select from a range of possible behavioural acts and receive a payoff associated with that act. There will be 100 possible acts, and the payoff for each act will be chosen at the start of each simulation from a distribution with many relatively small payoffs and some relatively large ones. Therefore the environment can be represented a table with two rows associating behavioural acts with payoffs, for example:

Act:	1	2	3	4	5	…	100
Payoff:	4	0	17	1	7	…	3

[1] If it is found that results are identical for shorter simulation runs then we may reduce this number for computational convenience.

1.1.2 The environment is not constant, and the payoff associated with a given behavioural act will change between each round of the simulation with a fixed probability, p_c. There is no association between payoffs for acts before and after the environment changes - the new payoff will be chosen at random (from the same distribution used in 1.1.1). The payoff for each act will change independently of the others, so that p_c also represents the average proportion of payoffs that change in each round. In the round-robin stage of the tournament, p_c will initially be fixed to a single value drawn from the range between 0.001 and 0.4, but we may test multiple levels if computational constraints permit. In the *melee* stage, we will run simulations with varying levels of p_c, drawn from the range [0.001-0.4].

1.1.3 The simulations will contain a single population of 100 individuals, representing a focal deme embedded in a meta-population. Each individual will have a behavioural repertoire, containing a subset of the acts from the table specified above. Individuals are born naïve; they have an empty repertoire. Each individual's repertoire can subsequently contain only those acts, and knowledge about their payoffs, that are acquired through some form of learning (see below). Note that environmental change means that the payoff recorded for a given act relates to when the act was learned, and if the payoff for that act has subsequently changed (see 1.1.2 above), then the payoff level that the individual has recorded in its repertoire will be wrong.

1.2 Moves

1.2.1 Participants must specify a set of rules, henceforth a 'strategy', detailing when individuals should perform each of three possible moves. The options are:

1. INNOVATE (individual selects a new act at random from those outside its current repertoire, and learns that act and its payoff)
2. OBSERVE (individual selects another agent(s) at random, learn its (or their) act(s) and acquire an estimate of the relevant payoff or payoffs)
3. EXPLOIT (individual performs a specified act from its repertoire and reaps the payoff)

1.2.2 INNOVATE is equivalent to trial-and-error learning, and does not guarantee an improvement in available payoffs. INNOVATE selects a new act at random, from those acts not currently present in the individual's repertoire, and adds that act and its exact payoff to the behavioural repertoire of the individual. If an individual already has the 100 possible acts in its repertoire, it gains no new act from playing INNOVATE.

1.2.3 OBSERVE is equivalent to social learning. OBSERVE selects one or more other individuals at random, and observes the act(s) they performed in the last round, and an estimate of the payoff(s) they received. This knowledge is then added to the observing individual's repertoire. The number of other individuals sampled when playing OBSERVE is a parameter of the simulation, termed $n_{observe}$. In the pair-wise tournament phase $n_{observe}=1$; in the second phase we will run further conditions with $n_{observe}>1$. Note that individuals playing OBSERVE sample agents solely from among the subset that played EXPLOIT in the last round. If no individual played EXPLOIT in the last round then nothing is learned (see 3.5 below). Note also that it is possible for an individual to OBSERVE an act already in its repertoire, in which case only the payoff recorded for that act is updated.

1.2.4 Social learning is error prone with regard to both act and payoff. With a probability fixed to a single value between 0 and 0.5, the behavioural act returned by OBSERVE will not be that performed by the observed individual, but rather an act selected at random. Furthermore, the returned payoff estimate will be $\mu + \varepsilon$, where μ is the actual payoff of the observed individual and ε is a normally distributed random variable rounded to the nearest integer, with a mean of 0 and the standard deviation fixed to a single value between 0 and 10 (if $\varepsilon<0$ and $|\varepsilon|>\mu$ then the payoff estimate will be set to 0). These errors could represent the observation of migrants performing acts that are inappropriate in the current environment and/or mistakes in observational learning.

1.2.5 Individuals remember their own history of moves and payoffs, so strategies can access this information. Strategies can also, if desired, use this knowledge to update the payoffs stored in individual's repertoires over and above the updating described in 1.2.2-4.

1.2.6 EXPLOIT is the only move that results in a direct payoff to the acting individual (EXPLOIT here does not mean that another individual is taken advantage of, only that an individual is exploiting its knowledge). An individual can only EXPLOIT acts it has previously learned. When an individual chooses to EXPLOIT an act, the payoff it receives is used to update the payoff recorded in its repertoire (that is, we assume an individual can, by performing an act, update its knowledge, stored in its behavioural repertoire, of how profitable that act is).

1.3 Evolutionary dynamics: Lifespan, fitness and reproduction

1.3.1 Evolutionary change will occur through a death-birth process. Individuals die at random, with probability of 0.02 per simulation round giving an expected lifespan of 50 rounds, and are replaced by the offspring of individuals selected to reproduce with probability proportional to their mean lifetime payoffs. For individual *z*,

$$Pr(reproduction) = \frac{P_z}{\sum_i P_i}$$

where P_z is the mean lifetime payoff of individual *z* (that is, the sum of its payoffs from playing EXPLOIT divided by the number of rounds *z* has been alive) and the denominator is the summed mean lifetime payoff of the population in that round.

1.3.2 Offspring are behaviourally naïve: they have no behavioural acts in their repertoire and no knowledge of payoffs. Unless mutation occurs, offspring inherit the strategy of their parents. Mutation will occur with probability 1/50, and when it does, the offspring will have a strategy randomly selected from the others in that simulation. These mutations are how other strategies will first arise in a population initially containing only a single strategy. Mutation will not occur in the last quarter of each *melee* simulation (see 2.2 below).

2. Running the simulations

Details of how the simulations will run and how scores will be recorded in each evaluation stage are as follows:

2.1 Stage 1 (Pairwise contests)

Strategies will take part in round-robin contests against all other strategies. A contest involves each strategy invading a population of the other strategy. In a given simulation, a population of the dominant strategy will be introduced, and run for 100 rounds to establish behavioural repertoires. At this point, mutation will be introduced, providing the second strategy the opportunity to invade. Simulations will then run for up to a further 10,000 rounds. Each pairwise contest will be repeated 10 times with strategy A as the invader and 10 times with strategy B as the invader. The mean frequencies of each strategy in the last quarter of each run (i.e. the last 2,500 rounds in a 10,000 round run) will be averaged over the 20 repetitions. This average will then be recorded as the score of that strategy in that contest. Strategies will be assessed on their total score once every strategy has been tested against every other strategy.

2.2 Stage 2 (*melee*)

Simulations will start with an initial population consisting of individuals with a simple asocial learning strategy (INNOVATE once and then EXPLOIT on every subsequent move). Every time an individual reproduces, it has a 1/50 probability of mutating to a strategy chosen at random from the pool of 10 winners from stage 1. However, there will be no mutation in the last quarter of the simulation so that mutation does not unduly influence results when strategies have similar fitnesses. After 10,000 rounds, the mean frequency of each strategy in the last quarter of the simulation will be recorded as the score for that strategy. In addition to manipulating p_c, we will also vary the error rates associated with OBSERVE (the probability of learning an incorrect act will be drawn from the range [0 -0.5], and the standard deviation of ε, the error distribution of payoff observations, will be drawn from the range [1-10]), and the number of individuals observed for each OBSERVE move ($n_{observe}$ will be drawn from the range [1-6]). Simulations will be repeated 100 times for each of the conditions, and the strategy with the highest average score will be deemed the winner. The exact number of conditions we test will depend on computational constraints.

3. How to enter

3.1 Strategies will take the form of computer code functions that take the data specified below as arguments and return a decision on which move to play. An example strategy is given below. Strategies can be submitted as a Matlab function (using only those commands available in the base installation, excluding toolboxes), and/or 'pseudocode' (that is, linguistic instructions breaking down how decisions are made into a series of mathematical and logical operations that can each be directly translated into a single line of computer code[2]). If submitted as Matlab code, a pseudocode version should also be provided, to facilitate debugging. In all cases the code should be straightforward to translate between the formats. We provide in section 3.9 below an example strategy in both Matlab and pseudocode form and refer to that strategy by line number in the following descriptions.

[2] For an example, see Mangel & Clark (2000) *Dynamic state variable models in ecology*. Oxford University Press, e.g. p55.

3.2 The strategy function should return an integer number representing the individual's move in this round (here termed `move`), and a 2-row array representing their behavioural repertoire (here termed `myRepertoire`).

3.3 To play INNOVATE, `move` should be returned as -1; to play OBSERVE, it should be set to 0. Any positive integer greater than 0 will be interpreted as playing EXPLOIT, and the value of `move` will specify which behavioural act to perform (i.e. an integer value equal to one of the acts in the individual's behavioural repertoire). This act must be present in the individual's repertoire. If any individual tries to EXPLOIT an act not in its repertoire then it gets nothing for that round – no payoff and no addition to the behavioural repertoire. On the assumption that such attempts are mistakes in strategy algorithms, we will, for strategies submitted sufficiently before the deadline (see rules 8 and 11 below), attempt to contact the entrant(s) and invite them to revise their strategy, provided they do so before the entry deadline expires.

3.4 Each individual has access to its own behavioural repertoire. Strategies will be provided with this information in the form of a 2 by *n* array, where *n* is the number of acts in the repertoire, the first row of the array represents the acts themselves and the second row their payoffs. We assume that an individual can remember what it did over its lifetime, and how long it has been alive. Thus strategies will be provided with information on age, moves, acts exploited or learned, and the associated payoffs.

3.5 Strategies will receive the above knowledge in the form of three variables: `roundsAlive`, `myRepertoire` and `myHistory`. An individual that has survived 5 model rounds might receive the following data:

```
roundsAlive = 5
```

Number of previous rounds this individual has survived.

```
myRepertoire = [19 2  64
                3  7  6 ]
```

The individual's behavioural repertoire, containing three acts: 19, 2, and 64 (first row) with, according to the individual's current knowledge, payoffs of 3,7 and 6 respectively (second row).

```
myHistory = [ 1  2  3  4  5
              0 -1  0  2  2
              2  19 64 2  2
              8  3  6  7  7]
```

Previous moves and their results – the first number in each column is the round to which that column pertains, the second is the move played (-1 for INNOVATE, 0 for OBSERVE, >0 for EXPLOIT), the third is the act learned (if OBSERVE or INNOVATE were played in that round) or exploited (if EXPLOIT was played), and the fourth is the payoff learned (OBSERVE or INNOVATE) or collected (EXPLOIT).

In this example case, $n_{observe}$ = 1. In its first model round, this individual played OBSERVE. As a result, it added act 2 to its repertoire and learned that this act returned a payoff of 8. In the second round it played INNOVATE, and added the act 19 with payoff 3 to its repertoire. In the third round it played OBSERVE and learned the act 64 with observed payoff 6. In rounds four and five, this individual played EXPLOIT, performed act 2, and received a payoff of 7. Note that its actual payoff received for act 2 was not exactly equal to the payoff learned for act 2 (when playing OBSERVE on the first round), because of the error in social learning (see 1.2.4).

Note also that in the case of new individuals, there will be no data – all the values shown above will be empty (i.e. of zero length) and if your strategy uses these inputs it should specify what to do in that case (and not crash!). The example strategy given below is robust to this as it specifically checks if `roundsAlive>1` (see section 3.9, line 2 of the MATLAB code).

3.6 Note that in above case, $n_{observe}$ = 1. If $n_{observe}$ = 3, for example, then the `myHistory` variable might look like this:

```
myHistory = [ 1  1  1  2  3  3  3  4  5
              0  0  0 -1  0  0  0  2  2
              2 86 10 19 64  2  0  2  2
              8  6  1  3  6  7  0  7  7]
```

Here, each OBSERVE move is represented by $n_{observe}$ (=3) columns in the `myHistory` variable, highlighted in bold. On the first OBSERVE move (round 1), the individual observed acts 2, 86 and 10 with payoffs of 8,6 and 1 respectively. On the second OBSERVE move (round 3) it observed two acts – 64, and 2 again. Note that there are two estimates here for the payoff associated with act 2 (8 in round 1, 7 in round 3) – these differences are due to the error in observing payoffs associated with OBSERVE (see 1.2.4 above). In the case that fewer than $n_{observe}$ individuals play EXPLOIT in the previous round, then some information returned by OBSERVE will be set to zero, as is shown for the underlined data from round 3 (zeros will also be returned if an individual that already has 100 acts in its repertoire plays INNOVATE). The behavioural repertoire for this individual in this case would contain acts 2, 86, 10 and 64.

3.7 Strategies can choose to use their own rules to update their current knowledge of payoffs in the `myRepertoire` variable. However, strategies that do this must only change the second row of the matrix; the simulation engine will check that the repertoire of behaviours has not been altered. Strategies that do not update payoffs should return the `myRepertoire` array unchanged (this will happen automatically if the syntax used in the first line of the example strategy below is used). Payoff updates resulting from observing a behaviour already in the repertoire, or from exploiting a behaviour for which the payoff has changed, will be carried out automatically by the simulation program.

3.8 There are some rules that relate directly to the form of strategies; these are given below but also appear in the general tournament rules at the end of this document.

(1) There is no limit to the length of the function, but it cannot, on average, take more than 25 times as long as the example strategy, given in section 3.9, to reach a decision. If, on completion of the pair-wise tournament, this is found to be the case for your strategy, then it will not be eligible to win the tournament. However, if it proves to be an effective strategy, we may still discuss it in our reports of the tournament.

(2) Your strategy cannot access the disk or memory storage of the computer in any way beyond the information provided as input.

(3) Strategies playing EXPLOIT must specify which act to use from their repertoire. This act must be present in the individual's repertoire. If any strategy returns acts not in the repertoire then on the assumption that such attempts are mistakes in strategy algorithms, provided the strategy submitted sufficiently before the deadline we will attempt to contact the entrant(s) and invite them to revise their strategy, provided they do so before the entry deadline expires.

(4) Strategies modifying their own behavioural repertoires to update the stored payoffs can alter only those payoffs and not the list of acts stored. If any strategy attempts to do this, the same rules as in (3) will apply.

(5) We reserve the right to edit code for computational efficiency, but we will notify entrants if this occurs and they will be given the opportunity to check that the operation of their strategies has not been compromised.

(6) Strategies *must* be accompanied by both brief prose description of how they are intended to function and, if submitted as computer code, a 'pseudocode' version.

3.9 We provide below an example strategy to illustrate what is expected of entrants. This strategy is called 'copy when payoff decreases' (here given the function name 'cwpd'). It starts by playing INNOVATE, and then EXPLOIT with the behaviour it learns. In subsequent rounds, it calculates the mean payoff the individual has received during its life, and if the last EXPLOIT returned less than that average, it plays OBSERVE and then EXPLOIT with the highest-payoff behaviour in its repertoire. This example illustrates how strategies must be prepared to handle the start of an individual's life, when it has no acts in its repertoire, and how strategies can change as individuals survive over several rounds.

Matlab version (text in green are comments to aid interpretation):

```
function [move, myRepertoire] = cwpd(roundsAlive, myRepertoire, myHistory)
if roundsAlive>1; %if this isn't my first or second round…
  myMeanPayoff = mean(myHistory(4,(myHistory(2,:)>0))); %mean payoff from EXPLOIT
  lastExploit = find((myHistory(2,:)>0),1,'last'); %find last EXPLOIT
  lastPayoff = myHistory(4,lastExploit); %get the payoff from last EXPLOIT
  lastMove = myHistory(2,find((myHistory(1,:)==roundsAlive-1),1,'first')); %find
  last move
  if (lastMove==0) || (lastPayoff>=myMeanPayoff) %if lastMove was observe or
  lastPayoff at least as good as myMeanPayoff then EXPLOIT
    rankedR_Matrix = sortrows([myRepertoire'],-2); %rank acts by payoffs
    move = rankedR_Matrix(1,1); %perform the act with best payoff
  else %otherwise
    move = 0; %OBSERVE
  end
elseif roundsAlive>0; %if this is my second round…
    move = myRepertoire(1,1); %only have one behaviour from INNOVATE, so use that
else
    move = -1; %if this is my first round, then INNOVATE
end
```

Pseudocode version:

copy_when_payoffs_decrease:

1. **If** *roundsAlive*=0 **then** INNOVATE (move=-1)
2. **If** *roundsAlive*=1 **then** EXPLOIT with behaviour learned in first round (move = first value in myR)
3. **If** *roundsAlive*>1 **then:**
    4. Calculate my average myP value when myM>0, i.e. my average payoff when EXPLOITing, call it *myMeanPayoff*
    5. Find out when I last EXPLOITed, and store the payoff from that EXPLOIT as *lastPayoff*
    6. Find what my last move was, store as *lastMove*
    7. **If** *lastPayoff*<*myMeanPayoff* **and** *lastMove*<>0 (not OBSERVE) **then** OBSERVE (move=0) **otherwise** EXPLOIT by ranking acts by payoffs and choosing the highest ranking act (move=rankedActs(1))

4. Tournament Rules

1. Entry into the tournament must be accompanied by explicit acceptance of these rules.

2. The decisions of the committee shall in all cases be final and binding.

3. Anyone may enter the tournament, with the exception of current members of Kevin Laland's research group, and members of the committee. Students of committee members are permitted to enter, but the committee will not be informed of entrant's identities if they are asked to adjudicate specific issues.

4. Entrants may be single individuals, or collaborative groups. In the latter case, groups must select a corresponding entrant who will be the sole point of contact for the tournament organisers and the only person with whom the organisers will discuss that entry, and also provide a list of the group

members. Entrants may submit only one strategy, and individuals may only participate in one group entry.

5. Only entries received by 1700 GMT on the closing date, 30th June 2008, will be accepted, and no further modification of entries is permitted after this date. Entrants are *strongly* advised to submit their strategies well before this time so that the organisers can check the code and inform entrants of any problems before the final closing date (see 8 and 11).

6. All entrants agree to the *content* of their submission being made public as part of the communication of this research exercise, although entrants can choose not to have their *name* associated with their entry.

7. There is no limit to the length of the function, but it cannot, on average, take more than 25 times as long as the example strategy, given in section 3.9, to reach a decision. If, on completion of the pair-wise tournament, this is found to be the case for your strategy, then it will not be eligible to win the tournament. However, if it proves to be an effective strategy, we may still discuss it in our reports of the tournament.

8. Your strategy cannot access the disk or memory storage of the computer in any way beyond the information provided as input. The organizers reserve the right to disqualify strategies that are deemed not in the spirit of the contest.

9. Strategies playing EXPLOIT must specify which act to use from their repertoire. This act must be present in the individual's repertoire. If any strategy returns acts not in the repertoire then on the assumption that such attempts are mistakes in strategy algorithms, provided the strategy submitted sufficiently before the deadline (see 11 below) we will attempt to contact the entrant(s) and invite them to revise their strategy, provided they do so before the entry deadline expires.

10. Strategies modifying their own behavioural repertoires to update the stored payoffs can alter only those payoffs and not the list of acts stored. If any strategy attempts to do this, the same rules as in (3) will apply.

11. We reserve the right to edit code for computational efficiency, but we will notify entrants if this occurs and they will be given the opportunity to check that the operation of their strategies has not been compromised.

12. Strategies *must* be accompanied by both brief prose description of how they are intended to function and, if submitted as computer code, a 'pseudocode' version.

13. Entrants must be prepared to enter into a reasonable dialogue with the organisers to remove ambiguities from the entered strategy for the purposes of coding the simulations and improving computation efficiency. We will endeavour to inform entrants if their strategies do not function correctly. If a strategy is deemed inadmissible prior to the closing date, entrants will be informed forthwith and given the opportunity to revise their submission.

Strategies deemed inadmissible after the closing date will be disqualified. We guarantee to test any strategies submitting up to one month before the closing date; we can give no such guarantee for strategies submitted after this time, although we will endeavour to do so.

14. If a strategy is submitted that, in the opinion of the organisers, is so similar to one already submitted as to be reasonably considered identical, then the first submission will take precedence and the submitter of the identical strategy will be informed that their entry is ineligible. The submitter will however be eligible to revise and resubmit their entry, provided that they do so prior to the closing date.

15. Any strategy that, in the opinion of the organisers, has been designed so as in any way to recognise and specifically help other entered strategies at their own expense will be disqualified and the authors of the strategy will be given no further opportunity to enter a modified strategy. This rule is essential to preserve the evolutionary validity of the tournament.

16. In the event of a tie, the tied strategies will be submitted to further tests under varied simulation conditions as deemed appropriate by the organising committee. If the committee judges that the tied strategies do indeed have equal merit, then they may decide at their discretion to share the prize between the tied entrants.

17. In the event that the number of submitted strategies renders a complete set of pairwise contests computationally unfeasible, we reserve the right to use a different system to select which strategies move forward to the *melee* stage, for example by splitting the strategies into randomly assigned groups from which winners will be selected to go forward to the *melee* stage.

18. The winning entrant will receive a cash prize of 10,000 Euros to be presented at a conference organised around the tournament at the University of St Andrews in 2009. The winning entrant will also be invited to co-author a paper reporting on the contest, although the organizers reserves the right to produce the paper without the entrant's participation and to judge authorship merits at their discretion.

References

Axelrod, R. 1984. The Evolution of Cooperation. New York: Basic Books.

Barnard, C. J. & Sibly, R. M. 1981. Producers and scroungers – a general-model and its application to captive flocks of house sparrows. Animal Behaviour, 29, 543–550.

Boyd, R. & Richerson, P. J. 1985. Culture and the Evolutionary Process. Chicago: Chicago University Press.

Boyd, R. & Richerson, P. J. 1995. Why does culture increase human adaptability? Ethology and Sociobiology, 16, 123-143.

Enquist, M., Eriksson, K. & Ghirlanda, S. 2007. Critical social learning: a solution to Rogers' paradox of non-adaptive culture. American Anthropologist, in press.

Feldman, M. W., Aoki, K. & Kumm, J. 1996. Individual Versus Social Learning: Evolutionary Analysis in a Fluctuating Environment. Anthropological Science, 104, 209-231.

Galef Jr., B. G. 1995. Why behaviour patterns that animals learn socially are locally adaptive. Animal Behaviour, 49, 1325-1334.

Giraldeau, L.-A., Valone, T. J. & Templeton, J. J. 2003. Potential disadvantages of using socially acquired information. Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences, 357, 1559-1566.

Henrich, J. & McElreath, R. 2003. The evolution of cultural evolution. Evolutionary Anthropology, 12, 123-135.

Laland, K. N. 2004. Social learning strategies. Learning and Behaviour, 32, 4-14.

Rogers, A. 1988. Does biology constrain culture? American Anthropologist, 90, 819-813.

Schlag, K. H. 1998. Why imitate, and if so, how? Journal of Economic Theory, 78, 130-156.