Decentralized interaction and co-adaptation in the repeated prisoner's dilemma

Tomas B. Klos[♣]

SOM Theme B: Interfirm coordination and change

Abstract

This paper investigates the behavioral patterns that emerge from the interactions among boundedly rational adaptive agents. They interact in repeated prisoner's dilemma's (RPD's) and adapt their behavior after each RPD tournament. Results are compared across virtual experiments with different regimes of interaction and adaptation. Specifically, round-robin interactions and centralized evolution of the population is compared to locally interacting and co-adaptating agents on a torus. Furthermore, the effects of imposing an additional bound on the agents' perception are explored. The results in the different setups show that centralized evolution may lead to somewhat better performance, but at the cost of a large increase in required computations. Also, the decentralized population endogenously learns a more efficient scheme for adaptation. Finally, placing bounds on the agents' perceptive capabilities, which essentially removes reputation as a source of information is shown to have a substantial negative effect on the population.

[♣] Thanks to John Miller, Roger McCain, Fieke van der Lecq, Bart Nooteboom and Gábor Péli for helpful comments and to the Dutch Organization for Scientific Research (NWO) for the grant (SIR 12-2815) that enabled me to present this paper at the Society for Computational Economics' Third International Conference on Computing in Economics and Finance. Faculty of Management and Organization, University of Groningen, P.O. Box 800, 9700 AV Groningen, The Netherlands; tel: +31 50 3637085, fax: +31 50 3632174; e-mail: t.b.klos@bdk.rug.nl; http://www.bdk.rug.nl/mo/io/tomas.

1 Introduction

This paper investigates the emergence of behavioral patterns in the repeated prisoner's dilemma. Several earlier studies have investigated evolution of behaviors in the RPD, by letting a genetic algorithm (Goldberg 1989, Holland 1992) evolve a population of repeated game strategies (see Axelrod 1987, Darwen & Yao 1995, Ho 1996, Marks 1992, Miller 1988, Stanley, Ashlock & Tesfatsion 1994). The genetic algorithm (GA) is a search algorithm based on natural selection and genetics that evolves a population of solutions to a certain problem, over the course of a number of generations. In each of those generations, first each solution obtains a fitness-score that expresses its performance in solving the problem. In RPD experiments, solutions are RPD strategies and the problem is that of finding successful RPD behavior in some environment. In some of the experiments, the environment consisted of some fixed set of strategies and in others, it consisted of the other strategies—that also evolved under the GA, or in other words *co*-evolved. Second, after scores have been thus assigned, the X fittest strategies proceed to the next generation unchanged (reproduction) while the N - X least successful strategies are replaced by new ones, created through recombination of relatively successful old strategies and possibly random *mutation* of the strategies resulting after recombination. A genetic algorithm thus *exploits* characteristics of successful solutions by reproducing and recombining them and it *explores* new possibilities by introducing some random variation.

Notwithstanding this normative instrument's value in finding solutions to problems in complex social domains, *i.e*. problems characterized by large, non-linear search spaces of social behaviors, it may not be the appropriate model for studying human adaptive social behavior in those domains, to the extent that the assumptions underlying the GA are problematic from the point of view of representing human adaptive social behavior. The aim of this paper is to establish the difference (if any) between what (the GA says) people *should* do if they want to be successful in RPD-type situations and what people *would* do—and whether or not this even makes them successful in the first place. To make this distinction clear, imagine the members of the population that the GA regime operates on to be not simply RPD strategies, but rather (artificial) agents *using* those strategies. This would require some heavy assumptions to underlie the model of such an agent. A more appropriate social science application of learning to play the repeated prisoner's dilemma would assume a population of (artificial) agents that autonomously *use* and *adapt* their own strategy under constraints of appropriately bounded rationality. Furthermore, with respect to adaptation, the GA uses complete, global information to determine which strategies are reproduced and which are replaced, whereas the agents would only use their limited, locally available information to autonomously decide whether they even want to change their strategy and if so, how.

What makes the GA's adaptive mechanism problematic from the point of view of simulating human adaptive social behavior—when instead of modeling each RPD strategy as a separate entity, it is modeled in an appropriate (object-oriented) way, as one of the methods of an agent that uses it—is the agent-model this mechanism would require. Under the GA regime, each agent would play against itself and against every other agent in the population (the so-called round-robin tournament setup), whereas typically, they would more appropriately be modeled as interacting only with those agents in the population that are physically, culturally or emotionally close to them as expressed, for instance, in terms of similarities in age, income, color, religion and the like (*cf*. Schelling 1971). Furthermore, in their decisions about adapting their strategy on the basis of relative performance, each agent would be able to perceive *all* agents' scores in *all* of their games (most of which they did not partake in themselves). Finally, the actual decision about whose strategies are adapted and whose are not would not only be conditional upon the (from the point of view of the individual agent very abstract) GA-parameters, but it would also be made on the basis of individual agents' performance relative to all other agents, whereas—in relation to the first assumption above—comparison with only the limited number of each agent's actual opponents would be more appropriate.

A number of recent studies have investigated localized agent-based adaptation (Axelrod 1984, Hoffmann & Waring 1996, Kirchkamp 1995, Nowak & May 1992, Oliphant 1994, Routledge 1993), but the current study will explicitly compare the GA setup with a decentralized one by starting with a replication of Miller's (1988) experiments and subsequently modifying his model in a number of ways. The first modification, an obvious and often-used way of limiting the agents' rationality, is to spatially disperse the population over a ring or torus (a 1- or 2-dimensional lattice with left-right and top-bottom boundaries respectively, folded onto one another) and let the agents interact with their 8 immediate neighbors only. In such a setup, straightforward application of the GA is problematic (see below). As an appropriate alternative, adaptation will be decentralized along with interaction, so that on the one hand, the agents can only compare their own performance to their immediate neighbors' performance and on the other hand, the agents can only imitate behavior that they perceive among those immediate neighbors. Then, a final model will be presented in which an agent can not perceive its opponents' performance in all of their games but only in their games against the agent itself.

2 Common building blocks

Before presenting in more detail the models mentioned above, some building blocks common to all the models will be discussed: the repeated prisoner's dilemma in which the agents interact, the finite state machine (FSM) that models each individual boundedly rational agent's RPD strategy and the genetic operators *crossover* and *mutation* that implement adaptation.

The interaction between each pair of players takes the form of a repeated prisoner's dilemma (see Table 1 where the payoffs are to the row- and the column-player, respectively).

	C	**D**
C	(R, R)	(S, T)
D	(T, S)	(P, P)

Table 1: Prisoner's dilemma if $T > R > P > S$ and $2 \cdot R > T + S$.

The first set of inequalities above result in mutual defection being the Nash equilibrium in the one-shot game; the second set of inequalities apply only to the repeated PD and prevent alternating unilateral defection and cooperation from being more successful than mutual cooperation. In the experiments reported in this paper (as in many others), the payoffs $T = 5$, $R = 3$, $P = 1$ and $S = 0$ are used.

Following a number of earlier studies (*e.g.* Binmore & Samuelson 1992, Kirchkamp 1995, Linster 1992, Miller 1988, Rubinstein 1986), a Moore machine, *i.e.* a type of *finite state machine*, will be used for modeling a *boundedly rational* agent's strategic decision-making behavior in the RPD. A Moore machine (such as the 2-state machine in Figure 1, representing Rapoport's strategy tit-for-tat, made famous in Axelrod's (1984) computer tournaments) consists of (1) a set of *internal states* (the circles in Figure 1), one of which is designated to be (2) the *starting state* (labeled 'S'), (3) an *output function* that maps each internal state onto an associated choice between the alternative actions *C* and *D* (the uppercase characters inside the circles) and (4) a *transition function* that maps each internal state onto a next state (designated by arrows in Figure 1) for each of the opponent's possible actions (the lowercase arrow labels). In the current experiments, unless specified otherwise, 16 state FSM's are used, which can implement more sophisticated meta-game strategies with larger memory than smaller machines such as in Figure 1.

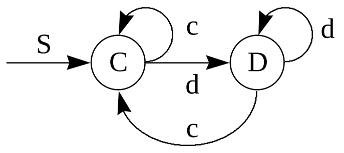
$$\begin{tikzcd} {} \arrow[r, "S"] & C \arrow[loop above, "c"] \arrow[r, "d"'] & D \arrow[loop above, "d"] \arrow[l, bend left=60, "c"] \end{tikzcd}$$

Figure 1: Tit-for-tat as a Moore machine.

Applying genetic operators to strategies requires coding them in an appropriate form. In the current model, each strategy is coded as a string of bits (0's and 1's), which is an often-used representation for members of the population of solutions that a genetic algorithm operates on. The crossover operator then implements recombination by combining (features of) two strategies and the mutation operator introduces random changes to strategies. In the current experiments, 2 point crossover and bitwise mutation operate on 16 state FSM's coded by strings of 148 bits as follows (Miller 1988). Because 4 bits can represent 16 different states, it requires 4 bits to specify which of the 16 internal states will be the starting state.

Additionaly, for each of the internal states, 1 bit is needed to specify the value of the output function, *i.e.* the move to make in that state (0 for *C* and 1 for *D*) and 8 bits are needed to specify the 2 values of the transition function—next states to transit to in response to each of the opponent's possible moves. All this means that it takes $4 + 9 \cdot 16 = 148$ bits to code each 16 state FSM.

Crossover (*i.c.* 2 point crossover) is applied to 2 bitstrings by exchanging between the 2 strings the substring that starts at a randomly chosen point and has a randomly chosen length. For example, applying 2 point crossover to the two bitstrings 0000000 and 1111111 with crossover point 5 and length 4, results in offspring strings 1000111 and 0111000. Applying mutation to a bitstring means that each bit in the string, conditional upon a certain (usually very low) probability, *i.c.* 0.5%, is flipped (from 0 to 1 or vice versa).

3 The models

3.1 A genetic algorithm benchmark

The next section (4) will present results from simulations using three models, which will be discussed in the current section. The starting point, serving as a benchmark, will be a replication of Miller's (1988) GA experiments. Representing each RPD strategy as a 16-state finite state machine and coding FSM's as strings of bits as described above, he evolved a population of 30 randomly initialized bitstrings over the course of 50 generations. After each round robin RPD tournament, the top 20 strategies were allowed to proceed to the next generation right away and 10 new strategies were created through (1) selecting pairs of parents from the old population with selection probabilities proportionate to fitness, (2) creating offspring strategies by applying 2-point crossover to each pair of parent strategies and (3) mutating each of the offspring-strategies' individual bits with 0.5% probability.

The first (technical yet non-trivial) step towards understanding what happens when the global genetic adaptive plan is replaced with decentralized adaptation under individual autonomous agents' discretion, is the assignment of each strategy to an agent, so that individual agents can use and adapt their own strategies. Of course, this change leads to no differences in the results, but helps to better understand the aim of the experiments to follow, in which each strategy is identified in an object-oriented manner as one of a certain agent's attributes. Other attributes (data and methods) can then be encapsulated within the agent, enabling continuous identification of individual agents' identities through time; even though they change the values of their own state variables and methods (*e.g.* their RPD strategy), they remain the same agent from one generation to the next (*cf.* Epstein & Axtell's (1996) Sugarscape implementation, McFadzean & Tesfatsion's (1996) Trade Network Game (TNG) implementation and Minar, Burkhart, Langton & Askenazi's (1996) description of the Swarm implementation). This first step is the implementation of a crucial feature of the current model: the unit of analysis can not be a strategy, seperated from an agent that uses it and adapts it if it so desires on the basis of its perception of its own relative performance.

3.2 Spatial dispersion

The second change from the canonical GA model is the imposition of a spatial structure on the population of agents (*cf.* Axelrod 1984, Hoffmann & Waring 1996, Kirchkamp 1995, Nowak & May 1992, Oliphant 1994, Routledge 1993). The agents are spatially dispersed over a *torus* and each agent only interacts with its 8 Moore neighbors (*i.e.* its neighbors to the north, north-east, east, south-east, south, south-west, west and north-west). Nowak & May's (1992) and Oliphant's (1994) models are not easily comparable to the present model, because they do not allow repeated game strategies nor different moves against different neighbors. Furthermore, there is a problem with Nowak & May's results related to the synchronization of agent interactions. In each round, each player first decides on its next move and then all players' moves are implemented simultaneously. Huberman & Glance (1993) have shown that these results depend critically on this synchronization of updating and that very different outcomes result if updating is asynchronous.

In general, in a spatial setting of agents playing RPD's against their neighbors, straightforward application of the GA methodology is problematic. Routledge

(1993), for instance, used a population on a torus where each agent interacts with its 4 Von Neumann neighbors (the neighbors to the north, east, south and west). After each RPD tournament, the *R* least successful agents' strategies were replaced with strategies, newly created by applying crossover and mutation to pairs of strategies selected from the *entire* population proportionate to fitness. However, although those *R* agents may have performed poorly from a global perspective, they may have been relatively successful locally—*i.e.* more successful than any of their neighbors. Therefore, changing their strategy because they are unsuccessful relative to *all* other strategies in the population may not be what they would have decided themselves, when given the option to choose on the basis of their perception of the success of *only* strategies in their own neighborhood. Furthermore, their new strategies would likely be completely different from their previous strategies, being constructed from very successful strategies that may belong to agents located far away from them; far beyond their span of perception. Finally, there is no reason to assume that the parent strategies' success would be transferable to the *R* agents' own neighborhoods. The current model lets the agents adapt their strategy locally, allowing them to only select from among their immediate neighbors' strategies.

Hoffmann & Waring (1996) used a somewhat more plausible agent-model, applied in a population dispersed over a ring (the one-dimensional equivalent of a torus). After each RPD tournament a strategy update was allowed for a *randomly chosen* half of the population. An agent (*x*) that is chosen to receive a strategy update is paired up with another agent (*y*), fitness-proportionately selected from *x*'s 'learning neighborhood', which consists of *l* agents on the ring to the left and right of *x*. Conditional upon a 60% crossover probability, *x* and *y*'s strategies are crossed over yielding two offspring strategies, one of which is chosen *at random* to replace *x*'s previous strategy. The problem with this particular model from our perspective is that each randomly chosen agent is forced to change its strategy (in 60% of the cases), which it may not have wanted to do (as explained above). Furthermore, the agent is not allowed to determine which of the two offspring strategies it wants to use, but is rather forced to use a randomly chosen one.

Axelrod (1984) used a spatial structure but limited the possibilities for adapting strategies to the ones entered into the second round of his tournament (*cf.* Linster

1992, who also circumvents this limitation by using FSM’s). By using FSM’s, one can make sure that the whole space of possible RPD strategies that can be implemented by 16 state FSM’s is potentially searched, with “no predisposition, either intentional or unintentional, toward a given outcome” (Linster 1992: 881).

In Kirchkamp’s (1995) model, the players interact and learn asynchronously: at a given moment in time, not all the players interact with all their neighbors and at the moment that a given agent changes its strategy, its neighbors may not. We will be using a simpler model in which all agents interact with all their neighbors at the same time (they can make different moves against different neighbors) and in which all agents change their repeated game strategy at the same time (*i.e.* after each RPD tournament), becauce this makes explicit comparison possible with the results from GA evolution.

The background for our model of adaptive social behavior is that in general, for an agent in an environment to be adaptive, (1) its actions need to be assigned a value and (2) it needs to behave in such a way as to improve this value over time (Holland & Miller 1991). In the current model, the value of the agent’s FSM is determined as the agent’s average payoff per move across all moves in its games against all of its neighbors. An agent’s behavior to improve its FSM’s value is guided by a comparison with its neighbors’ APM’s and thus implements imitative, social learning (*cf.* Macy 1996). The individual agents are explicitly given the discretion to autonomously decide if and when which operators are used. Specifically, after each RPD tournament, in which the agents interact with their 8 Moore neighbors, each agent decides whether it wants to adapt its RPD strategy. It compares the average payoff it received across all its moves in the previous 8 RPD’s (*i.e.* its APM = average payoff per move) to its immediate neighbors’ APM’s. If the most successful among those neighbors (or one drawn randomly from multiple most successful neighbors) was more successful than the agent itself, the agent will want to imitate that most successful neighbor. It does this by applying 2-point crossover to its own and the appropriate neighbor’s strategy, *i.e.* by copying a randomly chosen substring from the appropriate neighbor’s bitstring onto its own bitstring. If the agent’s most successful neighbor was equally successful as the agent, it will mutate its own strategy, in an attempt to locally optimize it. If the agent itself performed better than its most successful neighbor, it does nothing. After each agent has thus adapted its strategy if it wanted to, they start playing the next generation's RPD tournament.

3.3 Limited perception

This model would then still incorporate some problematic features. A GA, operating on a population, lets the top *X* performers proceed to the next generation. If the members of the population are thought of not as strategies, but rather as agents *using* strategies, as explained in section 3.1, then these agents are required to know every agent's APM as well as their position in the ranking of agents on the basis of these APM's. This assumption can be dropped if the population is spatially dispersed, as discussed in section 3.2, in which case it is replaced by the assumption that agents can only perceive their neighbors' APM's, representing those neighbors' performance against *their* neighbors. This is still a fairly strong assumption, so instead, the agents could be assumed to know only their neighbors' average score against themselves. Consider Figure 2, for example, where agent 6 plays—among others—against agent 11, whose APM is the average payoff it obtained in the moves it made in RPD's against agents 6, 7, 8, 10, 12, 14, 15 and 16. Agent 6 could be assumed not to be able to know agent 11's APM, but only agent 11's average payoff *against player 6*.[1]

16	13	14	15	16	13
4	**1**	**2**	**3**	**4**	1
8	**5**	**6**	**7**	**8**	5
12	**9**	**10**	**11**	**12**	9
16	**13**	**14**	**15**	**16**	13
4	1	2	3	4	1

Figure 2: A population of 16 agents on a 4 x 4 torus.

[1] A model of intermediate complexity would permit agent 6 to know agent 11's average payoff not only from playing against agent 6, but—additionaly—from playing against agents 7 and 10, because those are agents that agent 6 also interacts with.

4 Experimental design and results

Various results will be compared across three experimental setups. The first setup (see section 3.1) concerns the global evolution of a 36 member population consisting of 16 state FSM's (à la Miller 1988).[2] The second setup (see section 3.2) spatially disperses 36 agents over a 6x6 torus and the third (see section 3.3) lets each agent perceive not its neighbors' APM's, but only their scores against the agent itself. Of each experiment, 40 runs were conducted.[3] Results were typically collected as averages across the entire population and then averaged across the 40 runs.

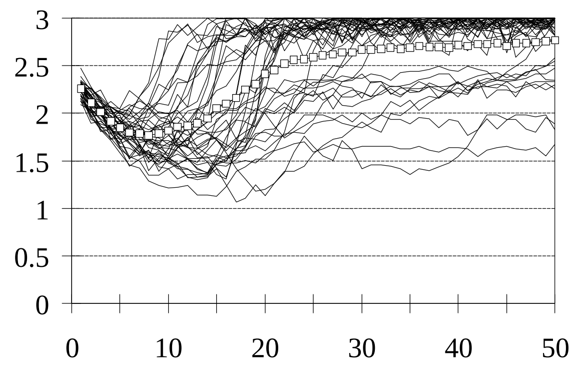

Figure 3: Evolution of population average APM under GA evolution.

Figure 3 shows the evolution of the average APM of the 36 strategy population when it is evolved by a GA. Each of the individual runs is shown, as well as the

[2] Miller used 30 members, but our replication uses 36 members because we compare the results with decentralized interaction and adaptation in a 36 agent population on a 6x6 torus. In each generation, the top 24 strategies are then reproduced from the old population and 12 new strategies are created with crossover and mutation.

[3] The simulations were run on a 90 MHz Pentium PC. All computer code was written by the author in SIMULA, the first object-oriented programming language (Birtwistle, Dahl, Myhrhaug & Nygaard 1973) and is available from the author upon request.

average over all 40 runs. The evolution of the population average APM starts at approximately 2.25, representing the random initialization as the average of the four payoffs 0, 1, 3 and 5. In the initial generations, strategies that often defect do well by preying on cooperative strategies. Immediately, the nasty strategies start running out of prey as the prey start reciprocating their defection (which selection favors to cooperating when opponents defect), which increases the proportion of mutual defection in the population (see Figure 4).

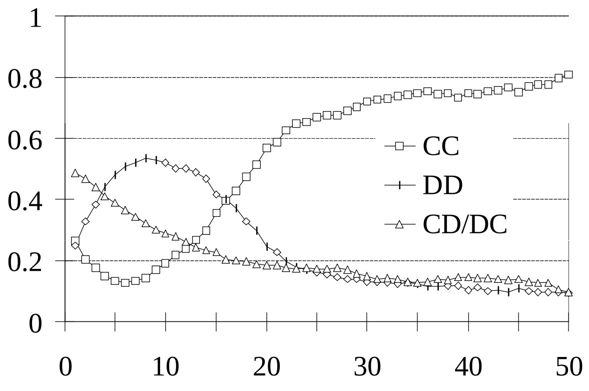

Figure 4: Average distribution of outcomes under GA evolution.

This increase goes at the expense of mutual cooperation and of course unilateral cooperation, until strategies that on the one hand cooperate and reciprocate cooperation themselves and that on the other hand recognize cooperating and cooperation reciprocating others get the opportunity to do well, which increases the proportion of mutual cooperation. Since two such strategies that mutually cooperate do better than two strategies of which one preys on the other (because $2 \cdot R > T + S$) and also better than two strategies that mutually defect (because $R > P$), this mutual cooperation is viable in the long run. After a few more generations, even the mutually defecting strategies start to recognize the success of mutual cooperation so they start imitating that, increasing even further the proportion of mutual cooperation. Eventually, most runs reach a state of mutual cooperation throughout most of the population (Figure 3).

The explanation above also mentions reciprocation, which refers to the way an opponent's move is responded to in the next round. The presentation of reciprocating characteristics in the population takes into account the following observation. Consider first of all that all agents are equipped with a 16 state FSM. However, not all those states can be used, or in other words, are *accessible*. For instance, if an FSM's transition from the initial state in case of an opponent's cooperation as well as defection leads back to the initial state, then only that 1 state is ever accessible and no matter what an opponent does, the agent will always make the initial state's move. Furthermore, not all the accessible states in an agent's FSM are always *used* by the agent. For instance, if the agent plays 'tit for tat' against an opponent who also plays 'tit for tat', then both agents will only use one of their FSM's accessible states. Figure 5 shows the average number of accessible and used states.

Apparently, less states are actually needed to reach the same results. This triggers the question of whether the same results could be achieved using smaller machines. In fact, initial results from using machines with a maximum of 8, 4 and 2 internal states shows that this is indeed the case, although when using 2 state machines, the discrepancy between the number of distinct strings (2^7 = 128) and the number of qualitatively distinct FSM's they represent (26 machines, namely both 1-state machines (always cooperate and always defect) and 24 distinct 2-state machines) becomes disturbingly large, but this is of course a mere technicality.

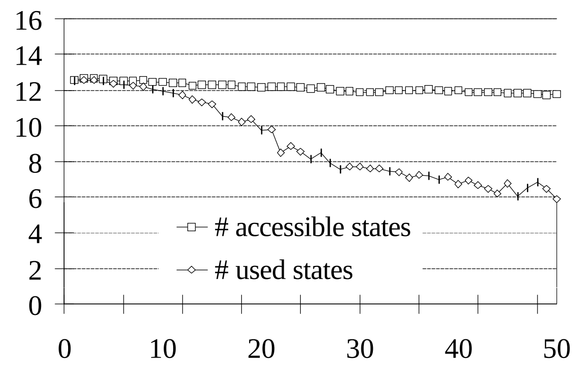

Figure 5: Average number of accessible and used states under GA evolution.

Going back to reciprocation, from Figure 5 it is obvious that there is a difference between states that *can* be used and states that *are* used. Figure 6 shows the development of cooperation and defection reciprocity in accessible states and in actual game play (where ‘cr’ and ‘dr’ refer to cooperation and defection reciprocity, respectively and where the addition of ‘as’ means that accessible states are concerned, rather than actual behavior).

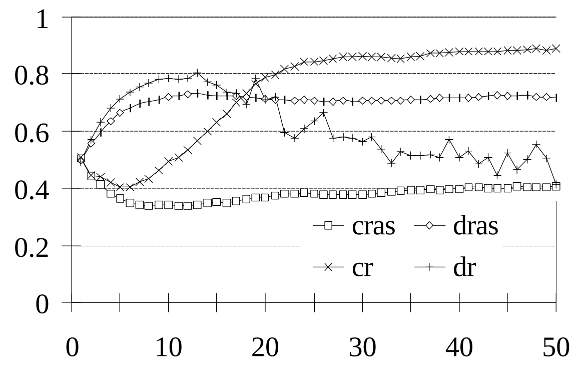

Figure 6: Evolution of cooperation and defection reciprocity under GA evolution.

Figure 6 shows that most of the accessible states that develop in the automata do not reciprocate cooperation, but do reciprocate defection. Eventually, however, states that reciprocate defection are not used frequently while states that reciprocate cooperation are. These results support the explanation for the development of average APM shown in Figure 3.

The strange development of actual mutual defection is illustrated for the 40 individual runs in Figure 7. Initially, more and more defections are reciprocated. Then, a lot of the runs show very freakish developments. Basically, most defection is reciprocated, but in some generations there was no defection that could be reciprocated, so the value of defection reciprocity is then measured as 0. In a typical next generation, there is usually some defection, most of which is again reciprocated, leading to a high value in the graph.

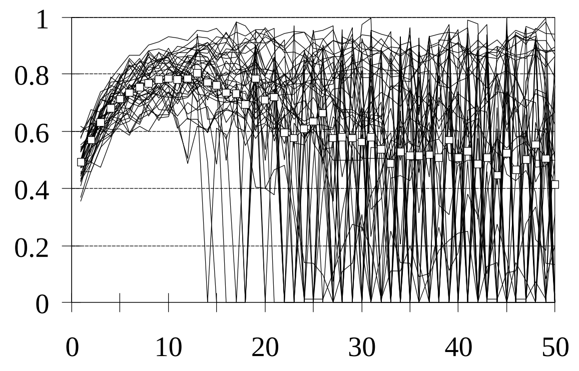

Figure 7: Evolution of actual defection reciprocity under GA evolution.

The next experimental setup concerns a population of the same size (36) that is spatially dispersed over a torus with localized interaction and adaptation as described in section 3.2. The difference between the development of the population average APM in this case with the development under GA evolution (see Figure 3) is shown in Figure 8.

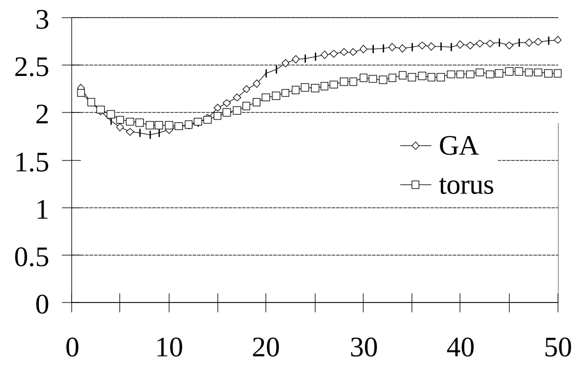

Figure 8: Evolution of population average APM on the torus and under GA evolution.

Two important aspects of the comparison should be noted. The first is the difference between the two setups: the GA is of course better at increasing overall performance, which is what it is meant to do. The second is the similarity between the two setups: even though the spatially dispersed population is not designed to aim for optimization the way the GA is, it produces results that closely resemble the GA's. In particular when looking at the results after the second change to the model (discussed in section 3.3) it becomes apparent how easy it is for a spatial model to produce results dissimilar to the GA's.

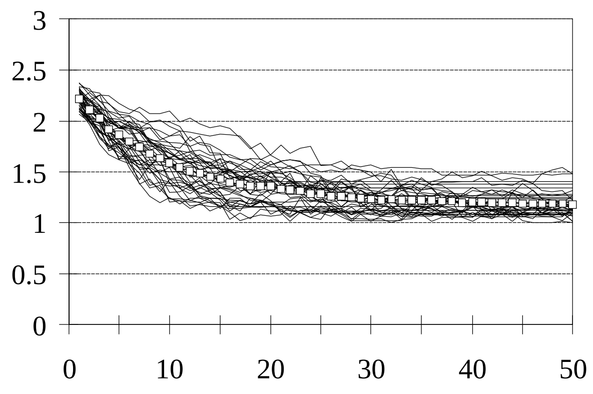

Figure 9: Evolution of population average APM on the torus with limited perception.

Figure 9 shows the development of population average APM when each agent can only perceive its neighbors' scores against itself instead of against all of those neighbors' neighbors. The decrease results from the fact that the agents never perceive their neighbors to be successful by cooperating; they only perceive their neighbors to be successful against themselves by defecting, so this becomes the standard throughout the population. Also note that many of the runs produce stability throughout the population (horizontal lines), which results when each agent thinks it is doing better than all of its neighbors, while in fact, it is only doing better than its neighbors are doing against the agent itself. The fact that the more informed population on the torus (described in section 3.2) does better may be regarded as a positive welfare effect of a *reputation* mechanism. Whatever the specifics of this mechanism, agents knowing what others do against third parties, helps the population.

Also, the similarities between the results on the torus and under the GA are interesting from the point of view of what it takes to reach them. The GA is an abstract entity that performs many computations at a level above that of the agents in the population, while apparently, the same or similar results can be obtained by distributed processing in a spatially dispersed population with much more boundedly rational agents performing much fewer computations in parallel. Remember that under the GA, each agent plays 150 rounds against each of the other agents and against itself, totaling (36 + 35 + … + 2 + 1 =) 666 games per generation, whereas on the torus, each agent plays only against its 8 neighbors, totaling only 144 games per generation. The discrepancy between the number of games played under the GA and on the torus is displayed in Figure 10 for different lattice sizes.

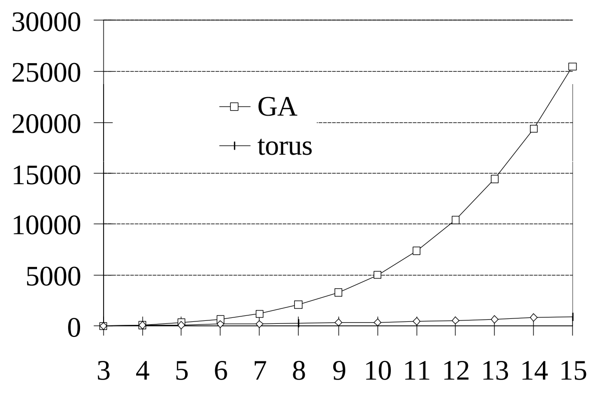

Figure 10: Number of games under the GA and on the torus for different torus sizes.

The two graphs show the number of games that would be played in a population that has a size equal to the value on the *x*-axis squared. The GA graph shows the number of games per generation when this population plays a round-robin tournament and the torus-graph shows the number of games when each agent plays against its 8 Moore neighbors only.

On the other hand, in the adaptive step, the GA produces only 12 new strategies through crossover and mutation in each generation, whereas the development of the number of agents changing their strategy using crossover and mutation on the torus is depicted in Figure 11. The figure on the left corresponds to the model as described in section 3.2 and the one on the right shows the results when using the model described in section 3.3.

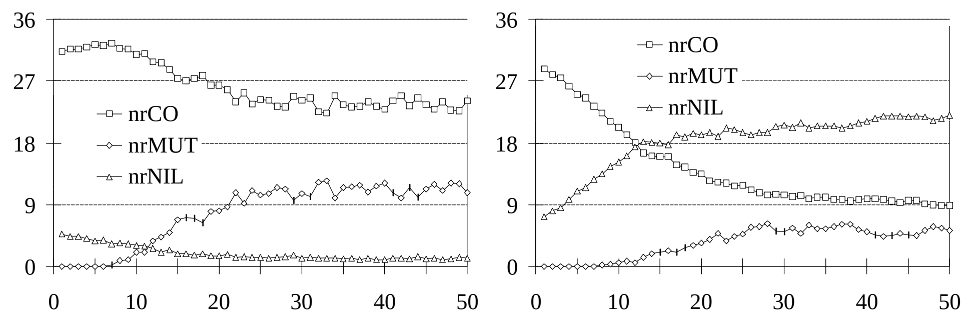

Figure 11: Number of agents using crossover and mutation or doing nothing on the torus.

The figure on the left shows that the decentralized model endogenously learns that the number of crossovers can be reduced once the major adaptations have been made and that the number of mutations can then be increased to locally search for perfections. In genetic algorithms, this knowledge is often used to exogenously decrease the crossover probability and increase the mutation probability over time (*cf*. Davis 1991). Overall, the number of agents not changing their strategy decreases. In the figure on the right, on the other hand, the number of agents doing nothing increases over time, because more and more agents reach situations where they think they are doing better than all of their neighbors, causing them to be complacent and lay back. Accordingly, more and more agents perceive no highly successful neighbors that they might want to imitate by means of crossover, so the number of those decreases.

5 Conclusions

The experiments described in this paper show important differences as well as similarities between a genetic algorithm’s centralized evolution of a population *vs*. decentralized interactions and co-adaptation by the members of the population. Operating on a population of size 36, the genetic algorithm is able to produce better performance than a spatially dispersed population with locally interacting and co-adapting members. There is a difference, therefore, between what the GA says people should do and what people would actually do. However, the results are not that different, especially when seen from the point of view of the volume of computations required in both cases; for larger population sizes, the discrepancy between the number of games played under the GA and on the torus becomes even larger. With respect to the computations during the adaptive step, the GA is more efficient there, but the population in the spatial model endogenously learns to decrease the number of crossovers when the major adaptations have been made and increase the number of mutations to explore the space of behaviors for minor improvements.

The results *are* very different when an agent's perception of its neighbors' scores is limited to those neighbors' interactions with the agent itself. In that case, the agents only imitate neighbors that often defect, since those attain higher scores than neighbors that cooperate, given the agent's own strategy. Also, in this case, the population easily gets stuck in such a state of mutual defection, when all the agents think they perform better than all of their neighbors, while they are actually only doing better than their neighbors are doing against them. The difference between the two agent-models can be interpreted as a positive welfare effect of reputation, that allows agents to perceive what other agents do against third parties.

As usual, more questions are raised than answered. The very different results in the last two models raise the question of which model is better, and in general, what—in this kind of exercise—it means for one model to be better than the next. Further experiments might include smaller machines (those have been performed to some extent) and different lattice sizes. However, it is easier to come up with 'interesting' new experiments than it is to code and run them and to interpret the results.

References

Axelrod, R. (1984) *The evolution of cooperation*. New York: Basic Books.

Axelrod, R. (1987) The evolution of strategies in the iterated prisoner’s dilemma. In: Davis, L. (ed.) *Genetic algorithms and simulated annealing*. London: Pitman. 32 – 41.

Binmore, K.G. & L. Samuelson (1992) Evolutionary stability in repeated games played by finite automata. *Journal of economic theory*, 57 (2): 278 – 305.

Birtwistle, G.M., O.-J. Dahl, B. Myhrhaug & K. Nygaard (1973) *SIMULA Begin*. Lund, Sweden: Studentlitteratur.

Darwen, P.J. & X. Yao (1995) On evolving robust strategies for iterated prisoner’s dilemma. In: Yao, X. (ed.) *Progress in evolutionary computation. LNAI 956*. Berlin: Springer-Verlag. 276 – 292.

Davis, L. (1991) Chapter 3: further evolution of the genetic algorithm. In: Davis, L. (ed.) *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold. 43 – 53.

Epstein, J.M. & R. Axtell (1996) *Growing artificial societies: social science from the bottom up*. Washington, DC: Brookings Institution Press and Cambridge, MA: The MIT Press.

Goldberg, D.E. (1989) *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley.

Ho, T.-H. (1996) Finite automata play repeated prisoner’s dilemma with information processing costs. *Journal of economic dynamics and control*, 20 (1 – 3): 173 – 207.

Hoffmann, R. & N. Waring (1996) The localisation of interaction and learning in the repeated prisoner’s dilemma. *Santa Fe Institute working paper 96-08-064*.

Holland, J.H. (1992, 1975) *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence*. Cambridge, MA: The MIT Press.

Holland, J.H. & J.H. Miller (1991) Artificial adaptive agents in economic theory. *American economic review*, 81 (2): 365 – 370.

Huberman, B.A. & N.S. Glance (1993) Evolutionary games and computer simulations. *Proceedings of the national academy of sciences of the United States of America*, 90: 7716 – 7718.

Kirchkamp, O. (1995) Spatial evolution of automata in the prisoner’s dilemma. *University of Bonn SFB 303 discussion paper B-330*.

Linster, B.G. (1992) Evolutionary stability in the infinitely repeated prisoner's dilemma played by two-state Moore machines. *Southern economic journal*, 58 (4): 880 – 903.

Macy, M. (1996) Natural selection and social learning in the repeated prisoner's dilemma. In: Liebrand, W.B.G. & D.M. Messick (eds.) *Frontiers in social dilemmas research*. Berlin: Springer-Verlag. 235 – 265.

Marks, R.E. (1992) Breeding hybrid strategies: optimal behavior for oligopolists. *Evolutionary economics*, 2 (1): 17 – 38.

McFadzean, D. & L. Tesfatsion (1996) A C++ platform for the evolution of trade networks. *Iowa State University economic report, nr. 39*.

Miller, J.H. (1988) The evolution of automata in the repeated prisoner's dilemma. In: *Two essays on the economics of imperfect information*. PhD dissertation, University of Michigan and (1996) *Journal of economic behavior and organization*, 29 (1): 87 – 112.

Minar, N., R. Burkhart, C. Langton & M. Askenazi (1996) The swarm simulation system: a toolkit for building multi-agent simulations. Santa Fe, NM: Santa Fe Institute.

Nowak, M.A. & R.M. May (1992) Evolutionary games and spatial chaos. *Nature*, 359: 826 – 829.

Oliphant, M. (1994) Evolving cooperation in the non-iterated prisoner's dilemma: the importance of spatial organization. In: Brooks, R & P. Maes (eds.) *Proceedings of the fourth artificial life workshop*. Cambridge, MA: The MIT Press. 349 – 352.

Routledge, B.R. (1993) Co-evolution and spatial interaction. *University of British Columbia working paper*.

Schelling, T.C. (1971) Dynamic models of segregation. *Journal of mathematical sociology*, 1 (2): 143 – 186.

Stanley, E.A., D. Ashlock & L. Tesfatsion (1994) Iterated prisoner's dilemma with choice and refusal of partners. In: Langton, C.G. (ed.) *Artificial life III, SFI studies in the sciences of complexity, proceedings volume XVII*. Reading, MA: Addison Wesley. 131 – 175.