THE GENETIC ALGORITHM AND THE PRISONER'S DILEMMA[*]

STUDENT PAPER

Benjamin Hosp
Roanoke College
Salem, VA
bhosp@roanoke.edu

ABSTRACT

The Prisoner's Dilemma is a game theory simulation used by sociologists to study human interactions. This game places two "players" in a situation wherein both of them, as a pair, would be better off if they cooperated with each other, but each of them, individually, is better off if he or she works towards his or her own selfish interests. However, when two players play this game repeatedly, cooperation becomes possible among rational players. In this paper, we will examine a method called the genetic algorithm for using a computer to derive strategies for the IPD. The genetic algorithm is applicable to many problems, but it does have many limitations, and this paper will demonstrate one of them in particular: the genetic algorithm cannot effectively operate on its own crossover and mutation rates.

1. INTRODUCTION

The Prisoner's Dilemma is a game theory simulation used by sociologists to study human interactions. This game places two "players" in a situation wherein both of them, as a pair, would be better off if they cooperated with each other, but each of them, individually, is better off if he or she works towards his or her own selfish interests. Each player can make one of two moves, cooperate or defect. Each player selects his or her move in secret, and then compares it with his or her opponent's move, according to this table (Coveney and Highfield, 223-4):

	Player 1 Cooperates	Player 1 Defects
Player 2 Cooperates	Both players score a 3 point Reward	Player 1 scores a 5-point Temptation; Player 2 scores a 0-point Sucker
Player 2 Defects	Player 2 scores a 5-point Temptation; Player 1 scores a 0-point Sucker	Both players score a 1 point Punishment

Game theory tells us that each player will see that he or she is better off defecting and assume that the other player will come to the same conclusion. Therefore, rational players of the Prisoner's Dilemma will always defect. (Poundstone, 105) However, when two players play this game repeatedly, cooperation becomes possible among rational players. It is still in each player's short-term best interest to defect, but the threat of retaliation from the other player and the promise of increased gains through future cooperation can lead the players to cooperate. (Orkin, 125) This makes the Iterated Prisoner's Dilemma (IPD) a more interesting game.

In this paper, we will examine a method called the genetic algorithm for using a computer to derive strategies for the IPD. The genetic algorithm is applicable to many problems, but this paper will demonstrate that it does have some limitations.

2. OVERVIEW OF LITERATURE

An effective strategy in this game is "TIT FOR TAT" (TFT). According to this strategy, a player cooperates in the first round, and then, in subsequent rounds, simply makes whatever move the opponent made in the previous round. This strategy cannot be exploited in the game more than once, but the player does tend to cooperate a lot, generating many reward payoffs (because he or she starts out cooperating, and will cooperate as long as the opponent does the same). This main feature of TFT, reciprocal cooperation, is generally seen as characteristic of any effective strategy for an IPD. (Coveney and Highfield, 224-6)

Many problems in biology, sociology, psychology, and economics can be abstracted as an IPD (Orkin 124). Essentially, any time an actor is able to recognize other actors, cooperation can evolve. For example, several species of animal seem to use a strategy sort of like TIT FOR TAT, such as the tree swallow and a hermaphroditic fish called the hamlet (for example, the hamlet will only approach a predator or other dangerous situation when others in its school are moving with it). Even animals of different species can evolve this sort of cooperation with each other, and human societies can develop "reciprocal altruism," as demonstrated by Robert Trivers (Coveney, 225-6). Conserving resources is an example of the way human interaction can form a Prisoner's Dilemma: if we all conserve, we all benefit, but if most conserve and a few don't, the few get all the benefits of conservation (resources will last longer and will be more plentiful), but are also able to squander as they wish. Unfortunately, people tend to notice this, and when too many decide to squander rather than conserve, everybody loses, but not as much as the suckers who conserved, hoping that everyone else would do the same, but now receive neither the benefits of mass conservation nor the benefits of individual excess. (Hardin)

Robert Axelrod conducted experiments that demonstrated the supremacy of reciprocal cooperation. Axelrod conducted experiments based on John Holland's work with the genetic algorithm. The genetic algorithm is a method that attempts to emulate the forces of evolution and natural selection to solve a problem. The first thing to do is to devise a way to express any solution to the problem as a string of bits, which we can refer to as a "chromosome." We will also need a function *F(s)* which will give a solution s a score expressing how well it solves the problem. Next, we generate a random set of bit strings of the proper length. Call this set the population. Now:

```
For $n$ generations:
    For each member $p_i$ of the population:
        Calculate and store $F(p_i)$.
    Reproduce the best members of the population, with better solutions earning more
        reproductions than poorer solutions, so that we maintain a constant population
        size. This creates a set of children. Discard the current population, and treat
        the children as the new population.
```

The "reproduction" operation is defined to simulate sexual reproduction. Two parents who have each earned at least one reproduction by having high F-scores (fitness) relative to the other members of the population pair off. Parent A starts copying bits from its chromosome into Child A's chromosome, and Parent B starts copying its bits into Child B. Each time a pair of such copies occurs, there is a chance of crossover, meaning that the parents switch their "destination" children, so that we copy as described above if there have been an even number of crossover events, otherwise Parent A copies into Child B, and Parent B copies into Child A. Also, each time a bit is copied, there is a chance of mutation, meaning that the bit that is written is the opposite of the bit that was read. The premise of the genetic algorithm method is that each subsequent generation is likely to contain better and better solutions, just as each subsequent generation of biological organisms is likely to be better and better adapted to its environment.

Axelrod noted that any IPD strategy that could remember the last three turns could be represented by seventy bits: sixty-four to tell the strategy what to do based on those last three turns (because there are four payoffs and three turns of memory, so the number of permutations are $4^3=64$), and six more to represent the initial assumptions about both players' last three moves before actual play begins. The function *F* in this case was the total number of points a strategy would accumulate after playing an IPD with either a list of human-designed strategies or with each other member of the population.

Each strategy received a reproduction if its fitness was within one standard deviation of the mean fitness for that generation. If the fitness was below this range, the strategy received zero reproductions; if it was above this range, the strategy received one extra reproduction for every full standard deviation its score was above the mean, so that a strategy with a score between one and two standard deviations above the mean would receive two reproductions, a strategy with a score between two and three standard deviations above the mean would receive three reproductions, and so on. If this system awarded fewer reproductions than the population size, the balance would be distributed randomly among the strategies that had already earned a reproduction. Axelrod reports that in both types of simulation, the strategies that evolved generally closely resemble TFT. Axelrod observed that the cooperation rate, which started out at 50% (because the first strategies are randomly-generated), quickly plummets, because only selfish strategies get good scores. Soon, however, some strategy is generated which has figured out reciprocal cooperation. This strategy and its descendants quickly dominate the population, and the cooperation rate rises to almost 100% and stays there until the end of the simulation. Axelrod concludes that the genetic algorithm is extremely effective at searching the large (about 10^{21}) number of IPD strategies that can "remember" three rounds for effective strategies.

In this paper, we will examine several variations on Axelrod's work. Axelrod always used the same rates at which crossover and mutation events occur: crossover would occur about once every reproduction (1/70 chance each copy), and mutation would occur about once every other reproduction (1/140 chance each copy). We will examine what happens when these rates are set to different values. In addition to examining systems with fixed rates of crossover, we will examine what happens when the genetic algorithm is allowed to operate on these rates, causing them to vary from generation to generation, and also from strategy to strategy.

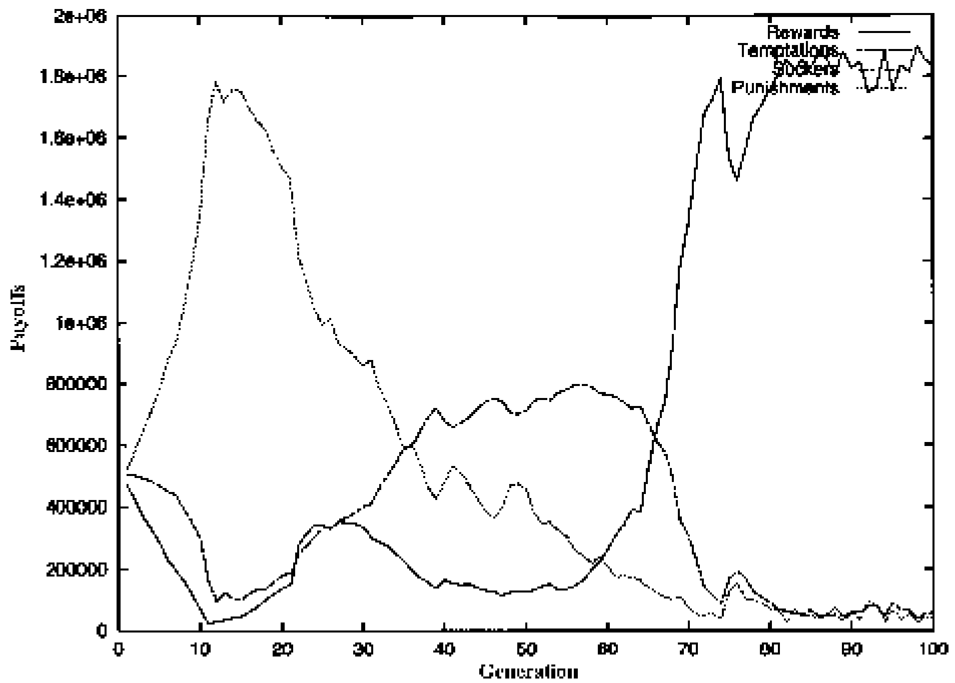

Figure 1: Payoffs graphs for a recreation of Axelrod's simulations. Each data point represents the total number of payoffs acquired by the population in each generation. Note that only three lines are visible because there will always be the same number of Suckers as of Temptations.

3. BASIC PROGRAM

Figure 1 shows the payoffs graphs from a recreation of Axelrod's simulations. Each point on the graph represents the total number of payoffs one generation of the population received. The key features of this graph match Axelrod's report, except for the rise of the "trader" strategies between the defectors and reciprocal cooperators, represented by the plateau in the Temptation and Sucker lines between generations 35 and 65. These strategies will cooperate, then defect, then cooperate, as long as their opponent is doing the same. The idea is that one strategy starts on a cooperation and one starts on a defection, resulting in both getting Temptation payoffs about half the time and Sucker payoffs about half the time. This results in an average payoff of about 2.5/round, as opposed to the 1/round of all Punishments, or the 3/round of all Rewards. This means that the traders do better than the defectors, and are able to replace them as the dominant members of the population, but are replaced when the reciprocators arise[1].

Observation 1: A population of IPD strategies can begin trading Temptation and Sucker payoffs as an intermediate step between simple defection and reciprocal cooperation.

4. EFFECTS OF CHANGING THE CROSSOVER AND MUTATION RATES

When the rates of crossover and mutation are altered, we observe that varying the rates between 1/20 and 1/256 doesn't have any significant effect on the simulation; the dip in cooperations followed by a sharp rise and plateau are still there, although higher rates of crossover and mutation do seem to make all the graphs look more "jagged," meaning that things change more from generation to generation, and seem more generally unstable. However, rates outside of this range are likely to cause significantly different behavior. First of all, when the crossover and mutation rates are both set to 0, the population will obviously contain only copies of the strategies randomly generated at the beginning; with no crossover or mutation, all the children will look exactly like one or the other of their parents. This means that the strategies that come to dominate the population will be exact copies of whatever randomly-generated strategy happened to do the best at the beginning. In the first generation, some strategies are best at finagling cooperations out of their neighbors, earning themselves Rewards or Temptations. These strategies will have the most children in the next generation, so when the next generation starts playing games with each other, the children of the most successful strategies find many copies of themselves. This means that if one of these strategies is particularly good at cooperating with copies of itself, then it will do extremely well, because it will be able to generate more and more copies of itself with each successive generation. This means that

[1]Of course, multiple exact copies of the same organism can't do this because they'll always be making the same moves as each other. This is more of a population-scale strategy, and doesn't actually result in *all* Temptations and Suckers, because it takes each pairing a few turns to work out the pattern. Examining each pairing won't give us much clue about what's going on, but this is a very recognizable state of a population.

simulations with crossover and mutation rates set at 0 will eventually consist of 200 copies of whichever randomly-generated strategy happened to be the best at cooperating with itself. Usually at least one strategy which cooperates with itself all or almost all the time is generated, so these populations end up with quite high rates of cooperation, but not always. They do always level off and become 200 copies of one strategy fairly quickly, usually within 30 generations, often faster.

Observation 2: When crossover and mutation are turned off, the strategies cannot change, so whatever randomly-generated initial strategy is best at cooperating with a copy of itself will take over the entire population.

Next, we can set the mutation and crossover rates to an unreasonably high number, like 1/4 or 1/2. This prevents any order from arising in the population at all. The cooperation rate takes a random walk around 50%, which causes the rate of occurrence of each payoff to take a random walk around 25%. As we decrease the rates, there is no improvement in the situation, until we reach 1/20, at which point we see evolution as normal, although the graphs do contain more noise than we saw before. The lower bound on this range of "reasonable" rates appears to be close to 1/256. The lower bound is not as clear as the upper bound because, while it is clear when a population has begun random walks it will not recover from, it will not be clear whether it will never begin evolving (because the crossover and mutation rates are too low) or whether it is simply evolving very slowly (because the rates are low, but not too low).

Observation 3: With crossover and mutation rates less than or equal to 1/20 (but still greater than some lower bound which is probably approximately 1/256), we see the sort of evolution towards reciprocal cooperation that Axelrod describes. With rates above 1/20, we instead see random walks.

The exact value of 1/20 is probably related in some way to the size of the chromosome. The one exception to the random walk with high rates scenario occurs when crossover and mutation rates are both set to 1.0 (or very close to 1.0), meaning that a crossover and a mutation happen each time a bit is copied. Bizarrely, a pattern is able to emerge where the children will look nothing like their parents. A cycle emerges of 200 identical strategies that do quite well, followed by their 200 children, which are their exact opposites. These don't do very well, but they are only competing with exact copies of themselves, so they are all still able to reproduce, producing an exact copy of the generation before them, and so on.

So far, we have only examined simulations where both crossover and mutation were carried out. What would the effect be if mutation were turned off? Surprisingly, it doesn't matter much what the crossover rate is when mutation is turned off; whatever the crossover rate is (even if it is extremely high), the same thing happens: the population evolves essentially as normal (although it gets slower as the crossover rate gets higher) until it is cooperating all the time. Once this happens, all breeds[2] die off except one. On the other hand, when crossover is

[2]A breed is defined as a set of all strategies in a population who will always make the same moves in any situation reachable through play. This does not always mean that the chromosomes necessarily have to be the same, because sometimes certain locations on the

turned off, but the mutation rate is set high, the population can never evolve beyond making moves essentially at random.

Observation 4: The mutation rate is what is actually causing the behavior in Observation 3. Without mutation, almost any crossover rate will allow evolution to reciprocal cooperation. Without crossover, however, a very small range of mutation rates allow such evolution. However, a small amount of mutation helps crossover move evolution along faster.

There are clear differences between systems with different crossover and mutation rates. We can verify that there is an advantage to having low, reasonable crossover and mutation rates by running a simulation where two "species" of strategies compete: one with low rates, and one with large, unreasonable rates. As seen in Figure 2, the low-rate species quickly evolves reciprocal cooperation, the high-rate species does not, so the low-rate strategies soon get all the reproductions, the high-rate species dies out.

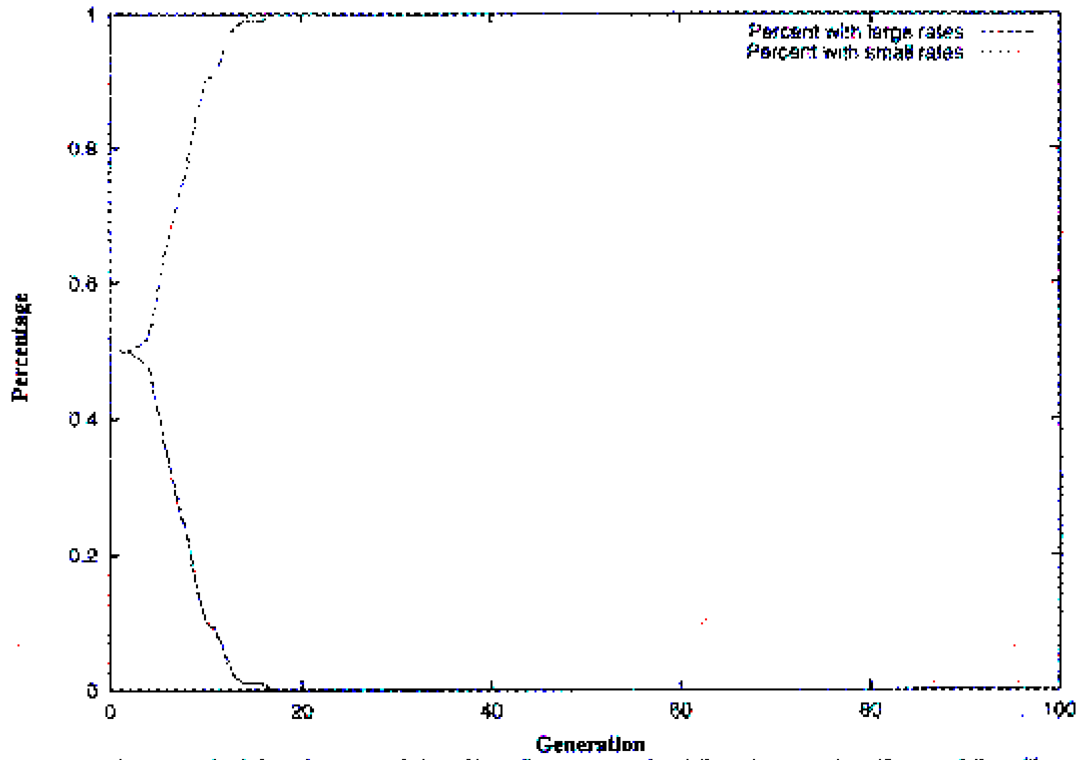

Figure 2: A simulation where a population with small crossover and mutation rates competes with a population with large rates.

chromosome can't be reached. For example, a strategy that defects after PPP, TPP, RPP, and SPP will never use its bit at PPR (because it would have had to cooperate after two Punishments to get that Reward, and this strategy never does that). Thus, two strategies that defect in those four situations but make different moves after PPR may be members of the same the same breed.

Observation 5: There is selection pressure toward having low, reasonable crossover and mutation rates.

5. EVOLVING CROSSOVER AND MUTATION RATES

We will now examine what happens when we try to let the genetic algorithm determine the crossover and mutation rates (in addition to the strategies themselves. Suppose we double the chromosome size (from 70 bits to 140 bits). These extra bits will be used to represent the crossover and mutation rates. We will use 35 bits for each rate. The genetic algorithm will operate on this chromosome exactly as it did on the previous chromosome, with the exception that before reproduction is carried out between two parents, it will read the crossover rate from each parent's chromosome, average them, and use that as the crossover rate for that reproduction operation (and do the same for the mutation rate). In other words, at the same time as the genetic algorithm is altering the strategies, it will alter the rates at which it does the altering. We expect to see that strategies with too-high crossover and mutation rates will die out because they can't copy themselves effectively, and strategies with too-low crossover and mutation rates will die out because they can't adapt to their environment, and only strategies with rates in the 1/256 to 1/20 range will be able to survive.

Suppose we interpret the 35 bits used to encode each rate as follows: the *i*th bit represents the quantity 2^{-i}. We will total all such negative powers of two which correspond to

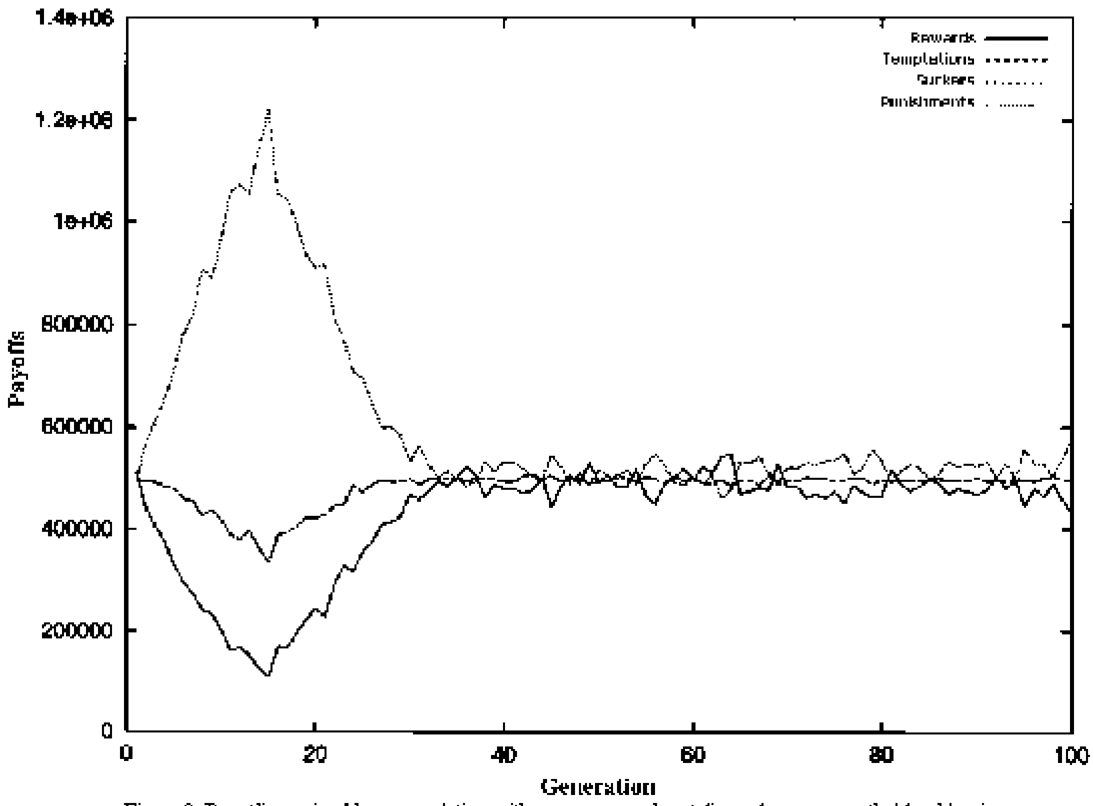

Figure 3: Payoffs received by a population with crossover and mutation rates averages that tend to $\frac{1}{2}$.

"on" bits and use that as the rate in question. Suppose further that we initialize the population with crossover and mutation rates of 2^{-4}=1/64. What we see is the average[3] crossover and mutation rates of the population quickly rise to about ½, and the population then behaves as it did when its crossover and mutation rates were fixed at ½: the cooperation rate takes a random walk around 50%, which causes the rate of occurrence of each payoff to take a random walk around 25%, a situation reflected in Figure 3.

NPT:

We know there is an evolutionary advantage to crossover and mutation rates between 1/256 and 1/20, so why does the population average tend to ½? Consider what will happen with this system (Call it the Negative Powers of Two, or NPT system) in the 35 bits used for encoding the mutation rate. We initialized this rate to 1/64. Each time we copy this region of the chromosome (35 bits) we have a slightly better than 1:2 chance of a mutation. This mutation is much more likely to turn on a bit than to turn one off (because only one bit is turned on at this point.) So the average mutation rate in the next generation will be slightly higher, because turning

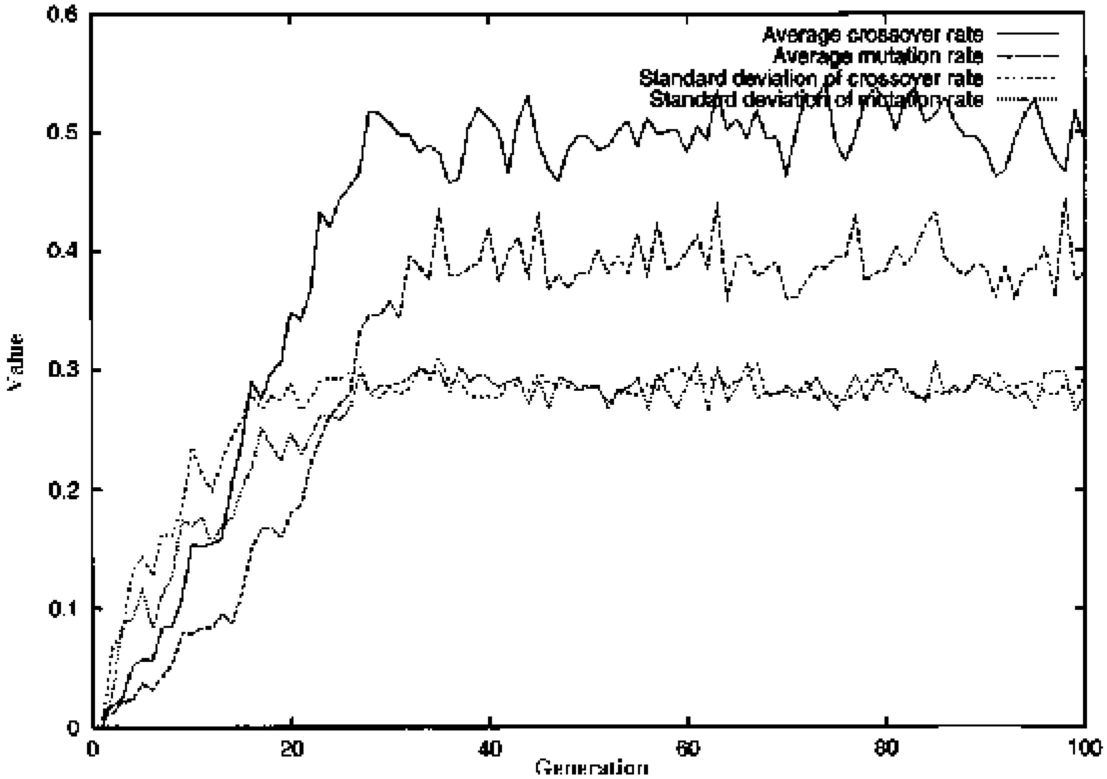

Figure 4: Averages and standard deviations of a population using the NPT system. The averages tend to approximately ½, and the standard deviations tend to 3/10.

[3]The average is a meaningful approximation of the crossover and mutation rates that are being used even when the standard deviation is high because when two strategies are paired up for reproduction their crossover and mutation rates are averaged.

bits on increases this mutation rate. This process will continue with each successive generation until about half the bits are turned on in most chromosomes, at which point further mutations are just as likely to turn on bits off as to turn off bits on. Mutation will cause a similar process to occur in the crossover rate region of the chromosome. When about half the bits are turned on in the NPT system, the population average is ½, because this system is a truncation of the series ½+¼+ **c** +...=1, so doing about half these additions at random will, on average, add up to about ½. We won't necessarily see many numbers close to ½, because some bits are weighted more than others – the bit worth ½, in particular, will cause the individual numbers to vary wildly. But in Figure 4, we see exactly what we expect from analysis of the NPT system: the average rates quickly rise to ½, and the rates' standard deviations rise quickly to approximately 3/10.

InN:

We might now believe that there is some problem with the NPT system. So we will now examine three more encoding systems to see if this bias towards half the bits being on is a general characteristic of these systems. First, consider a system where the 35 bits for each rate are interpreted as a number *n*, and the rate is then treated as *1/n*, except when *n*=0, when we instead use a very small number. Call this system the Inverse *n* system, or InN. Clearly, many of these bits have the potential to make 1/*n* very small. Once any high-order bit of *n* is turned on, the rate will get very small, and when this happens to the mutation rate, it will be very hard for this bit ever to get turned off again. Since most of *n*'s bits have this property, it is only a matter of time before enough members of the population have crossover and mutation rates of almost zero to make the crossover and mutation rates actually used in each reproduction likely to be almost zero. If we run a simulation using the InN system, we in fact see the crossover and mutation rates quickly drop to almost zero, and then we have a population that behaves as if the rates were fixed at zero. The static behavior we noticed in Observation 2 is back.

EWB:

So far, the problem seems to be that certain bits have different weights than other bits in both the NPT and InN systems. So consider the following system: if *k* of the 35 bits are on, the rate is interpreted as *k*/35. Call this the Equal-Weighted Bits system, or EWB. Even though each bit has the same value (1/35), we are going to have the same sort of problem as we did with NPT: about half the bits will get turned on until random mutations can't change the percentage of bits that are on (because they are equally likely to turn a bit on as they are to turn one off), so EWB's average crossover and mutation rates will tend to move quickly towards the value they have when approximately half their bits are on: ½. The problem was not the fact that certain bits have different weights, but that we can predict the value we get when approximately half the bits are on. (Upon reflection, we notice that InN has the same property, although we aren't likely to see half the bits get turned on, because the value of InN's rates is so strongly attracted to zero that we will have to wait a very long time to see half the bits get turned on.)

Observation 6: Despite the selection pressure noted in Observation 5, the genetic algorithm, when allowed to operate on crossover and mutation rates,

will tend to select rates according to mathematical bias present in the system used for interpreting a bit string as a real number. The population's average crossover and mutation rates will quickly become whatever the average result is when the interpretation system is used on strings with approximately half their bits turned on.

PNF:

We will now examine one more interpretation system, and see if it supports Observation 6. Suppose we use our bits as follows:

- Bits 1-4 represent +1/32, -1/32, +1/32, -1/32.
- Bits 5-8 represent +1/64, -1/64, +3/64, -3/64.
- Bits 9-12 represent +1/128, -1/128, +7/128, - 7/128.

And so on; until we have used 32 of our 35 bits (discard the remaining three). This system (call it Positive and Negative Fractions, or PNF) will produce a real number between -½ and +½, to which we add ½ to get the rate in question. This system is most like NPT; the bits have different weights (although the pattern of weights is more complicated here than in NPT), and the average value when approximately ½ of the bits are on is about ½ (since half the bits represent negations of the other half), but we will expect a large standard deviation. This is exactly what the experimental data shows us. We get a graph that is almost indistinguishable from Figure 4. Furthermore, to verify that the cause of this behavior is actually the mathematical bias present in these four interpretation systems and not some esoteric feature of the IPD, we can run simulations where *F*, the fitness function, is either a constant function or a random number generator. In both cases, the progression of the rates is not noticeably different from when the IPD games were actually played.

6. CONCLUSION

By making alterations to Axelrod's simulation, we have made six observations. Observation 1 tells us that there are other local optima in populations of IPD strategies than simple defection and reciprocal cooperation, but we have found nothing to unseat reciprocal cooperation as the optimum strategy. Observation 2 describes the static behavior we get without crossover and mutation. Observation 3 identifies a range of "reasonable" crossover and mutation rates inside which we can expect to see the evolutionary behavior as described previously, but below which we can expect to see static behavior as in Observation 2, and above which we can expect to see random walks. Observation 4 identifies the culprit in the random walk scenario as mutation; "unreasonable" rates of crossover without mutation can allow evolution which "unreasonable" rates of mutation would destroy. Observation 5 confirms that when we use both mutation and crossover, low, "reasonable" rates are better.

Observation 6 is the most complicated and important one. It identifies a severe problem with any attempt to allow a genetic algorithm-based simulation to determine its own rates of crossover and mutation: any system for encoding the rates as bit strings (so the genetic algorithm can operate on them) will have some kind of bias, and this bias is likely to be a much stronger attractor than the selection pressure towards reasonable rates. We have verified Observation 6 for four distinct rate encoding systems, and as Observation 4 would suggest, we see that mutation is responsible for the strong biases in these systems, because it tends to turn on approximately half the bits of the mutation "chromosome." The obvious next step would be to try this with mutation turned off, but it isn't clear how this could be meaningful: we cannot initialize every crossover rate to the same value (because crossover alone would not be able to change this value), but we cannot select the initial rates purely randomly either (because then we'd just start out with approximately half the bits turned on, which is what we're trying to avoid.) Some alternate method must be found.

WORKS CITED

Axelrod, Robert. *The Complexity Of Cooperation*. Princeton, NJ: Princeton University Press: 1997.

Coveney, Peter, and Roger Highfield. *Frontiers Of Complexity*. New York, NY: Fawcett Columbine, 1995.

Hardin, Garrett. *The Tragedy Of The Commons*. 1968. http://dieoff.org/page95.htm.

Orkin, Mike. *What Are The Odds?: Chance In Everyday Life*. New York, NY: W.H. Freeman, 1999.

Poundstone, William. *Prisoner's Dilemma*. New York, NY: Doubleday, 1992.

FURTHER READING

Campbell, Richmond, and Lanning Sowden. (eds.) *Paradoxes Of Rationality And Cooperation: Prisoner's Dilemma And Newcomb's Problem*. Vancouver, BC: University of British Columbia Press, 1985

Liebrand, Wim B.G., Andrzej Nowak, and Rainer Hegselmann. (eds.) *Computer Modeling Of Social Processes*. London, England: Sage, 1998.

Powers, Richard. *Prisoner's Dilemma*. New York, NY: Beech Tree Books, 1988.

Rapoport, Anatol, and Albert M. Chammah. *Prisoner's Dilemma; A Study In Conflict and Cooperation*. Ann Arbor, MI: University of Michigan Press, 1970.

Wilke, Henk A.M., Dave M. Messick, and Christel G. Rutte. (eds.) *Psychology Of Decisions And Conflict: Experimental Social Dilemmas*. Frankfurt am Main, Germany: Verlag Peter Lang, 1986.