The Prisoner's Dilemma, Memory, and the Early Evolution of Intelligence

Mikaela Leas[1,4], Emily L. Dolson[2,3,4], Riley Annis[2,4], Joshua R. Nahum[2,3,4], Laura M. Grabowski[1,4] and Charles Ofria[2,3,4]

[1]College of Engineering and Computer Science, University of Texas Rio Grande Valley
[2]Department of Computer Science and Engineering, Michigan State University
[3]Ecology, Evolutionary Biology, and Behavior Program, Michigan State University
[4]BEACON Center for the Study of Evolution in Action
mikaela.leas01@utrgv.edu

Downloaded from http://direct.mit.edu/isal/proceedings-pdf/alif2016/28/408/2326031/978-0-262-33936-0-ch068.pdf by guest on 22 May 2025

Abstract

Memory is an essential component of intelligence as it enables an individual to make informed decisions based on past experiences. In the context of biological systems, however, what selective conditions promote the evolution of memory? Given that reliable memory is likely to be associated with costs, how much is it actually worth in different contexts? We use a genetic algorithm to measure the evolutionary importance of memory in the context of the Iterated Prisoner's Dilemma, a game in which players receive a short-term gain for defection, but may obtain greater long-term benefits with cooperation. However, cooperation requires trust; cooperating when an opponent defects is the worst possible outcome. Memory allows a player to recall an opponent's previous actions to determine how trustworthy that opponent is. While a player can earn a high payout by defecting, it will likely lose the trust of an opponent with memory, yielding a lower long-term payout. We determined the value of memory in the Iterated Prisoner's Dilemma under various conditions. When memory is costly, players reduce their available memory and use short-term greedy strategies, such as "Always Defect". Alternatively, when memory is inexpensive, players use well-known cooperative strategies, such as "Tit-for-Tat". Our findings indicate that organisms playing against a static opponent evolve memory as expected. However, memory is much more challenging to evolve in coevolutionary scenarios where its value is uneven.

Introduction

Biological evolution has produced our only examples thus far of general intelligence. As such, understanding the evolutionary process–both how it occurred in nature and how we can replicate it in a computer–may prove important on the path to developing artificial intelligence. One important component of such research is understanding the role of memory. Memory is the foundation of learning, allowing an individual to alter its future behavior based on prior stimuli (Sherry and Schacter, 1987). As such, memory is critical for such behaviors as navigating, tracking, foraging, avoiding predators, hunting prey and cooperating with others (Dunlap and Stephens, 2009; Grabowski et al., 2010; Liverence and Franconeri, 2015; Kraines and Kraines, 2000; Soto et al., 2014). These behaviors are sufficiently beneficial to fitness that memory is advantageous to many individuals despite the associated biological costs (Barton, 2012; Dukas, 1999; Mayley, 1996). Understanding the importance of memory and the conditions under which memory evolves is crucial as it is a fundamental component to both real and artificial organisms.

To study the selective pressures that lead to the early evolution of memory, we need a way to measure their impact on memory's value. Here, we propose a technique for performing a cost-benefit analysis of memory via a simple evolutionary simulation. As an environment for this simulation, we will use the the classic game theoretic problem, Iterated Prisoner's Dilemma (IPD). Game theory provides a tractable framework for studying the value of memory in social contexts. IPD specifically is an ideal choice, because it is well-understood, requires memory for optimal performance, and is commonly used as a model system for studying cooperation (Axelrod, 1987; Crowley et al., 1996; Kraines and Kraines, 2000; Golbeck, 2002). In this game, two players repeatedly interact; at each step, they may cooperate with or defect from each other, and are rewarded according to the Prisoner's Dilemma payout matrix (see Table 1). The fact that IPD is so well-studied allows us to thoroughly validate this approach to studying memory. At the same time, we can gain useful insights into a relatively intuitive system before tackling more complex ones.

To assess the value of memory in this environment, we use a genetic algorithm to evolve strategies for playing IPD. Strategies in this algorithm are allowed to use memory, but at a cost. They must sacrifice part of their payout to have and use memory. By imposing a series of different memory costs and observing under which memory-using strategies evolve, we can measure the value of memory in this evolutionary context. Allowing evolution to generate novel strategies, rather than hard-coding in well-known strategies and allowing them to compete, ensures that we are not inadvertently introducing our own biases to the study system. To further ensure the validity of our system, we initially test it in a static environment where all players compete against a fixed set of three strategies. Overall, this system should

allow the evolution of individuals that use successful strategies in IPD, allowing us to determine the value of memory.

Iterated Prisoner's Dilemma

Three commonly-employed strategies for IPD are Always Defect, Always Cooperate, and Tit-for-Tat (Brunauer et al., 2007). The first two are the repetition of one action (defect or cooperate, respectively), while Tit-for-Tat is a strategy that repeats whatever action a player's opponent performed last. Always Defect and Always Cooperate do not require memory, as they do not rely on the history of a player's actions or those of its opponent. Tit-for-Tat, however, does require memory. In a single iteration of Prisoner's Dilemma, the best possible strategy is Always Defect; regardless of the opponent's decision, defecting will always yield a higher payout on a given iteration than cooperating would have (see Table 1). This consistent benefit makes Always Defect a selfish/greedy strategy (Axelrod, 1987).

	C	D
C	R = 3	S = 0
D	T = 5	P = 1

Table 1: **Payouts To Row-Player for Prisoner's Dilemma** Fitness is determined based on this matrix. Four payouts are possible: Reward (R), Sucker (S), Temptation (T), and Punishment (P). These payouts are a result of whether the player and the opponent each cooperate (C) or defect (D). In a single iteration, T is the highest payout for a single player. However, when playing repeated iterations of Prisoner's Dilemma, players can retaliate against each other, yielding lower payouts for both than if they had cooperated consistently.

When playing multiple iterations, cooperative strategies, such as Tit-for-Tat, outcompete the Always Defect strategy by allowing for the higher rewards associated with long-term cooperation (Axelrod, 1987; Crowley et al., 1996; Golbeck, 2002). To be successful, cooperative strategies must, among other things, be **forgiving** and **retaliating**; both of these attributes require memory (Axelrod, 1987). **Forgiving** strategies (eventually) cooperate in response to their opponents cooperating, even if the opponent defected in the past. Conversely, **retaliating** strategies (eventually) defect in response to their opponents defecting. Both of these strategies are only possible if the player is able to remember the opponent's actions. Thus, we can reasonably expect memory to be worth sacrificing some percentage of a player's payout, an assumption which is born out by prior research (Crowley et al., 1996).

Methods

Our system is a genetic algorithm, where fitness is based on the cumulative payout of IPD. A genetic algorithm is a method for computationally solving problems that maintains and generates a population of potential solutions by selecting the most successful ones and allowing them to reproduce (Goldberg and Holland, 1988). There are four important components within a genetic algorithm: representation of a genotype, the initialization of the population, mechanism for selecting the next generation, and mutation operators (Mitchell, 1996). To facilitate validation of our approach via comparison to the results of previous research, we based our system off of systems that have successfully been used to study IPD in the past (Axelrod, 1987; Crowley et al., 1996; Kraines and Kraines, 2000). Crowley et al.'s set-up was a particularly strong influence, as their system allowed for flexible evolution of memory-using strategies. Our implementation is open source and available on GitHub: https://github.com/mikaelaleas/ChangingEnvironmentGA.

Strategy	AD	TFT	R	Average
AD (0)	1.00	1.06	3.00	1.69
TFT (1)	0.98	3.00	2.24	2.07
TTFT (2)	0.98	3.00	2.60	2.19

Table 2: **Payouts for Optimal Strategies for the Static Environment** A player's payout is determined from the Prisoner's Dilemma matrix (Table 1). In our static environment, the player competes against three static strategies: Always Defect (AD), Tit-for-Tat (TFT), and Random (R) over 64 iterations. The player's optimal strategy is dependent on the size of its memory. The AD strategy uses zero bits of memory, while the TFT strategy uses one bit of memory. When the player has one bit of memory, the best strategy is Tit-for-Tat. In this environment, the optimal strategy when a player has two bits of memory is to first cooperate and then defect any time the opponent has defected in the player's memory. This strategy is called Two-Tits-for-Tat (TTFT).

Representation of Genotype

Genotypes in our system are closely based off of those used by Crowley et al. (1996). An individual's genotype has three components: (1) the amount of memory it uses, (2) the initial state of its memory, and (3) its decision list. (1) The size of an individual's memory is the number of previous iterations for which it can remember its opponent's actions (as a simplification, organisms are unable to remember their own actions). Each bit of memory can hold information about one iteration. Since the decision list grows exponentially with the amount of memory used, we limit individuals to have no more than four bits of memory; that is, individuals can remember up to four iterations of their opponent's actions. (2) Next, since the memory is supposed to be a list of the opponent's actions, its initial state (before the opponent has actually played any iterations) biases the early decisions

Downloaded from http://direct.mit.edu/isal/proceedings-pdf/alif2016/28/408/2326031/978-0-262-33936-0-ch068.pdf by guest on 22 May 2025

made by an individual. The initial state of this memory is allowed to evolve. The individual's memory is subsequently updated every iteration of IPD, with the oldest past action being removed and the most recent action being added (see Figure 1). (3) Finally, the decision list is used to specify which action an individual will take, given a particular memory state (see Figure 2). The length of the decision list is 2^n, where n is the number of bits of memory, with an entry corresponding to every possible state of memory. The initial population, of size 500, was composed of individuals with each of these components randomly selected. Populations were allowed to evolve for 500 generations.

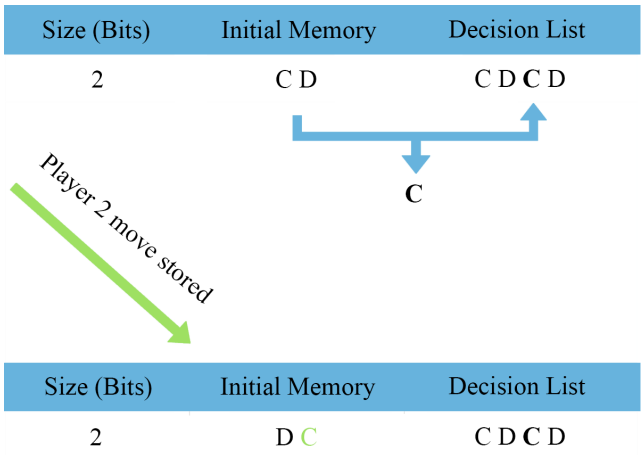

Figure 1: **Single Iteration of Prisoner's Dilemma** The player has three components: size (in bits), initial memory, and decision list. In a single round, a player will use the initial memory and decision list to decide whether to cooperate (C) or defect (D). A player's initial memory is updated every round to store the opponent's last action. The decision list does not change during an individual's lifetime. Here, player 1 cooperates with player 2 and player 2 cooperates with player 1. Player 1's initial memory is updated to reflect player 2's cooperation.

Selection of Next Generation

To select which individuals contribute offspring to the next generation: (1) a fitness score is generated for each individual, and (2) the population participates in a tournament. To determine a fitness score, individuals play 64 iterations of Prisoner's Dilemma (1 game) against competitors. In the static environment that we use to validate this approach, these competitors have three predetermined strategies: Always Defect, Tit-for-Tat, and Random. These three strategies were chosen to keep simplicity of the model, allowing for a focus on the evolution of memory-using strategies. In the coevolutionary environment, these competitors are randomly chosen from the population. Based on the IPD payout matrix, each individual is awarded a payout. This payout is

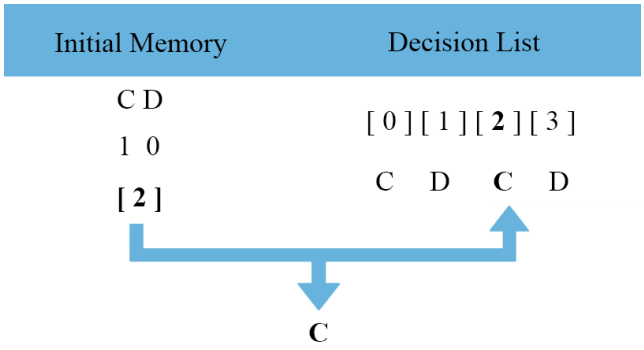

Figure 2: **Initial Memory and the Decision List** During a single iteration of Prisoner's Dilemma, a player chooses to cooperate (C) or defect (D) based on its decision list. Defect is represented with a 0 and cooperate with a 1. In this example, the initial memory is CD, which is represented as the binary number 10 (i.e. 2, in decimal). This points to index 2 in the decision list, which contains a C, so this player will cooperate in this iteration.

multiplied by the difference between 1 and the total cost of memory (accounting for all of the bits). The result is the fitness score.

$$fitness = payout(1 - cost * size) \quad (1)$$

The fitness score calculation determines how the cost of memory affects the fitness of an individual. The cost of memory is fixed prior to the experiment. Finally, an average fitness for each individual is calculated. The next generation is produced through a tournament-style selection. The population is divided into subgroups of 10 individuals. The best half of the group–those with the highest fitness scores–are selected for the next generation. Note that this is a slightly gentler selection scheme than the one used by Crowley et al. (1996); we chose it because we felt that the reduced elitism was a better analog for the biological systems we are ultimately interested in understanding.

Mutations

Mutations occur probabilistically after the next generation is selected. There are three classes of mutations that can occur, corresponding to each of the portions of the genome: (1) size mutations, (2) initial memory mutations, and (3) decision mutations. All three types of mutations have a fixed probability of 0.01 of occurring when offspring are created. (1) A size mutation will increase or decrease the size of an individual's memory by 1 bit. This change affects the length of the decision list and the initial memory state of the individual. If the size of the memory is increased, the decision list will be duplicated meaning that increasing memory has no immediate effect on behavior unless one of the other types of mutations also occurs. However, if the size of the

Downloaded from http://direct.mit.edu/isal/proceedings-pdf/alif2016/28/408/2326031/978-0-262-33936-0-ch068.pdf by guest on 22 May 2025

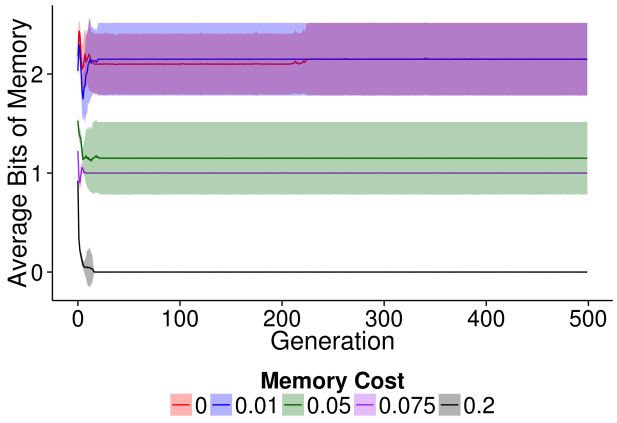

Figure 3: **Average Number of Bits of Memory Used By Cost of Memory** Shaded area represents standard deviation for each line. The cost of memory had a strong impact on the average number of bits of memory used by the population (Kruskal-wallis test, chi-squared = 168.44, df =8, $p < .0001$). When the cost of memory increases, the average number of bits of memory decreases (Post-hoc Wilcoxon Rank-sum test with Bonferonni correction). The average number of bits used at each memory cost are consistent with the predicted values from Table 3.

memory is decreased, the decision list is halved by removing the least significant bits (most distant in the past memory position). (2) The memory mutation affects the initial state of memory. This mutation will randomly choose an index of the initial memory and toggle the action (cooperate or defect) at that position. (3) The decision mutation targets the decision list. This mutation will randomly choose an index of the decision list and toggle the action (cooperate or defect) at that position.

Results and Discussion

Static Environment

To verify this system's efficacy, we started out by allowing strategies to evolve in a static environment in which each player competed against three static strategies: Always Defect, Tit-for-Tat, and Random. In this scenario, we can deterministically calculate how much a bit of memory should be worth in each context. The expected fitness and the highest memory cost for which players evolve to use memory is calculated from the Prisoner's Dilemma payout matrix, the individual's size, and the memory cost. The individual plays 64 iterations of IPD against each of the three strategies and receives payouts accordingly. The payouts are then adjusted according to the individual's size and the cost of memory, to determine the individual's fitness (Equation 1). Using more bits of memory allows the player to recall more previous actions of the opponent and thus determine which

Strategy	Cost	AD	TFT	R	Average
AD (0)	0.01	1.00	1.06	3.00	1.69
TFT (1)	0.01	0.97	2.97	2.22	2.05
TTFT (2)	0.01	0.96	2.94	2.55	2.15
AD (0)	0.075	1.00	1.06	3.00	1.69
TFT (1)	0.075	0.91	2.78	2.07	1.92
TTFT (2)	0.075	0.83	2.55	2.21	1.86
AD (0)	0.2	1.00	1.06	3.00	1.69
TFT (1)	0.2	0.78	2.40	1.79	1.65
TTFT (2)	0.2	0.59	1.80	1.56	1.32

Table 3: **Expected Average Fitness by Cost of Memory in the Static Environment** This table shows the expected average payout per iteration for the optimal strategies for 0, 1, and 2 bits of memory, adjusted by various costs of memory. The parenthetical next to each strategy name denotes the number of bits of memory that it uses. Here, we show three costs, each of which favors a different strategy: Always Defect, Tit-for-Tat, or Two-Tits-for-Tat.

strategy the opponent is using. Once an individual is able to determine its opponent's strategy, it may alter its future actions to increase its payout. This enables the evolution of better strategies that are able to retaliate against opponents if exploited. For example, an individual using the Always Defect strategy receives an average payout per iteration of 1.69 (see Table 2). If the cost of memory were 0.01 and the individual had one bit of memory, that payout would be reduced to 1.67. Using two bits of memory would further decrease the payout to 1.65. When there is no fitness cost, the optimal strategy is to start out cooperating, use the maximum allowed amount of memory, and defect any time an opponent has defected within memory. This will result in an individual always defecting after the first iteration against Always Defect, cooperating with Tit-for-Tat, and recognizing Random as frequently as possible. However, there are diminishing returns to adding additional bits of memory (see Table 2); in this simple setup, the greatest fitness improvement comes from adding the first bit, making Tit-for-Tat a possible strategy.

When a cost is applied to memory, the optimal strategy may change (see Table 3). If our system is accurately measuring the value of memory, we would expect to see Always Defect be the dominant strategy when the cost per bit of memory is 0.18 or greater, Tit-for-Tat be dominant when the cost is between 0.18 and .065, and so on. This result is almost exactly what we see in practice (see Figure 3). As predicted, this shift seems to be driven by an increase in Tit-for-Tat-style strategies as the cost of memory decreases (see Figure 4).

The one slightly unexpected result is that, even when

Downloaded from http://direct.mit.edu/isal/proceedings-pdf/alif2016/28/408/2326031/978-0-262-33936-0-ch068.pdf by guest on 22 May 2025

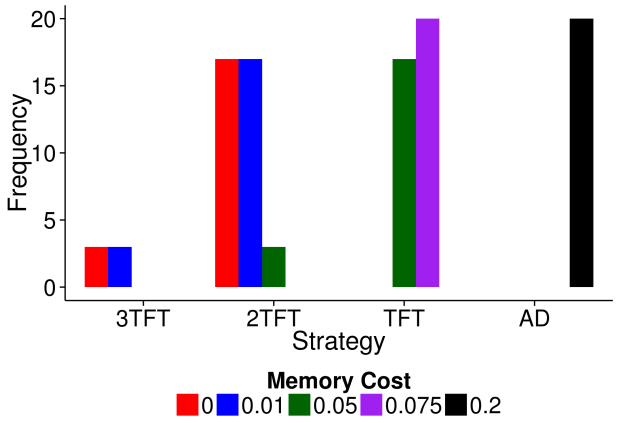

Figure 4: **Most Common Strategies By Cost of Memory** We calculated the most commonly used (dominant) strategy in each of the 20 replicates within each of the 5 memory cost conditions. The four dominant strategies we observed were 3TFT (Three-Tits-for-Tat, the optimal strategy with three bits of memory), 2TFT (Two-Tits-for-Tat, the optimal strategy with two bits of memory), TFT (the optimal strategy with one bit of memory), and AD (the optimal strategy with no memory). As expected, the dominant strategy depended on the cost of memory. Increasing the cost of memory increases the frequency with which less memory-intensive strategies are dominant.

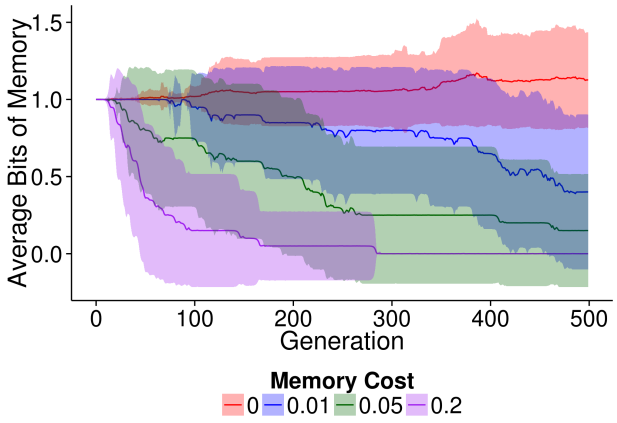

Figure 5: **Memory Usage in Coevolutionary Environment (Low Mutation Rate)** Shaded area represents standard deviation for each line. Memory use consistently evolved only when memory had no cost; the average amount of memory used in this condition was significantly different from the amount used in all of the other conditions (Kruskal-wallis test and post-hoc Wilcoxon rank-sum test with Bonferonni correction, chi-squared $= 55.93$, df $=5$, $p < .0001$). In all of the other conditions, the average amount of memory used gradually declines over time.

memory has no cost, strategies don't tend to use much more than three bits of memory. We hypothesize that this is due to the following mechanism: Every additional bit of memory doubles the size of an individual's decision list. An excessively large decision list is at increased risk of experiencing genetic drift away from the optimal values. Thus, the potential fitness gain from adding a fourth bit of memory may not be worth the increased risk of the lineage making incorrect moves later on. Such a scenario would be consistent with the decreased recognition accuracy found by Crowley et al. (1996).

Coevolutionary environment

Having demonstrated that our methodology accurately measures the value of memory in a system, we can now move on to a more interesting case. Instead of placing solutions in a static environment, we can allow them to compete against each other. This scenario introduces complex coevolutionary dynamics that would normally confound attempts to measure the value of memory. In this setup, the population is initially populated with Tit-for-Tat (one bit of memory) and each individual plays IPD with each other individual in its tournament to determine its fitness. Like before, the top-half of each tournament is allowed to reproduce. We ran this treatment at two different mutation rates: low (.01 for each mutation type) and high (.1 for each mutation type).

At a low mutation rate, memory proves far less useful in this more complex environment, as evidenced by the fact that it is not consistently used if it has any cost associated with it (see Figure 5). As in the previous experiment, increasing the memory cost increases the percentage of replicates in which Always Defect, rather than Tit-for-Tat, becomes the dominant strategy. When examining individual runs, a common pattern takes place. The initial population of Tit-for-Tat is frequently invaded by Always Cooperate. Always Cooperate can displace Tit-for-Tat (in the absence of other competitors) because it receives the same payout, but does not have to pay any cost for memory. Once Tit-for-Tat is extinct (or nearly so), Always Defect arises and quickly displaces Always Cooperate. In the low mutation rate replicates, Tit-for-Tat rarely is generated via mutation from Always Defect, leading to a stable population that is trapped at a sub-optimal strategy. Although Crowley et al. did not analyze the strategies that evolved in their system, these results are consistent with theirs in that they too observed that applying a cost to memory resulted in decreased cooperation (Crowley et al., 1996).

Interestingly, memory use in a coevolutionary context increases at the higher mutation rate (see Figure 6). When memory is free in this treatment, strategies quickly evolve to use the maximum allowed amount of memory, suggesting that the implicit costs of making use of a large memory are

Downloaded from http://direct.mit.edu/isal/proceedings-pdf/alif2016/28/408/2326031/978-0-262-33936-0-ch068.pdf by guest on 22 May 2025

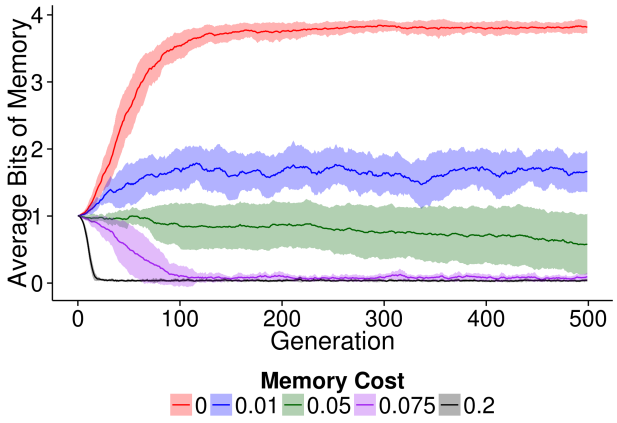

Figure 6: **Memory Usage in Coevolutionary Environment (High Mutation Rate)** Shaded area represents standard deviation for each line. Populations evolved to use more memory at lower costs. At memory costs of .075 and higher, the average amount of memory used by the population after 500 generations was not significantly different from 0 (Kruskal-wallis test and post-hoc Wilcoxon rank-sum test with Bonferonni correction, chi-squared = 95.24, df =5, $p < .0001$).

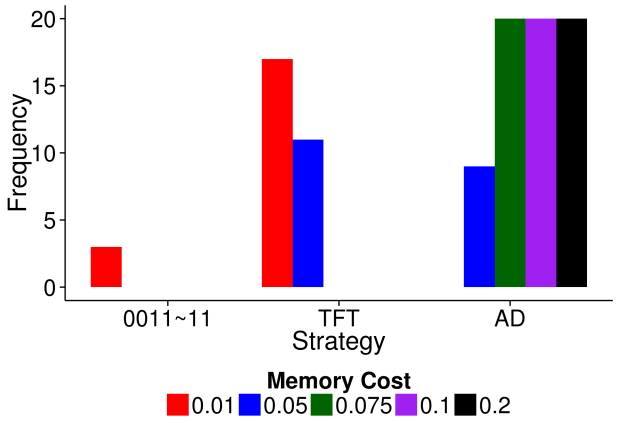

Figure 7: **Most Common Strategies in Coevolutionary Environment (High Mutation Rate)** Again, Tit-for-Tat is more frequently the dominant strategy at lower memory costs. Note that this figure does not include the strategies used when there was no cost to memory, because there were too many of them. Approximately half of the replicates in the 0 cost condition of this treatment used Tit-for-Tat, and the other half each had a different dominant strategy (although most of the dominant strategies were not dramatically more prevalent than other strategies in the population). Also note that the strategy on the far left, 0011∼11 is denoted only by its genotype (decision list∼initial memory), as it does not correspond to a well-known named strategy. It cooperates initially, and any time its opponent cooperated two iterations ago.

overwhelmed by coevolutionary selective pressures. Alternatively, the large decision lists that individuals with a lot of memory have may serve to increase mutational robustness. This effect would be in contrast to the results observed in the static environment and in previous research (Crowley et al., 1996). Understanding the relationship between these factors would be an interesting direction to explore in the future.

In the condition with no memory cost, Tit-for-Tat is the most common strategy in approximately half of the replicates, a finding which is consistent with Tit-for-Tat's dominance in the Axelrod tournament (Axelrod, 1987). Among the other half of the replicates there is an incredible diversity of most common strategies - only two of the other replicates have the same most common strategy. Applying any cost to memory causes the population to converge to well-known strategies (see Figure 7). These results align with Mayley's finding that applying a cost to learning (analogous to memory, in our case) substantially inhibits the exploration of strategies that would require it (Mayley, 1996).

Conclusion

We demonstrated the evolutionary value of memory by using a genetic algorithm that awards fitness based on the results of many iterations of the Iterated Prisoner's Dilemma. Under static environmental conditions, the population often evolved to use memory, despite it being costly, as long as it provided a substantial gain in payout. In fact, the extent to which memory was used aligned nearly perfectly with theoretical predictions about the costs and benefits of memory in this system. This result demonstrates that the technique proposed here is an effective way to quantify the value of memory in evolutionary contexts. By simply giving memory a fitness cost and observing whether memory evolves we can assess its importance in complex scenarios.

In more dynamic environments, we observed that memory was valuable when there were no costs because it enabled cooperation. However, it was easily evolved away under high memory costs (where Always Defect could rapidly overtake Tit-for-Tat) or low mutation rates (where Always Cooperate could outcompete Tit-for-Tat and subsequently be outcompeted by Always Defect). While this phenomenon illustrates the difficulty of measuring the value of memory in an environment where that value keeps changing, our results were consistent with the findings of prior research and we were able to more fully investigate the mechanisms behind them.

While we were able to show the value of a single bit of memory, the evolutionary dynamics explored here generally did not provide a substantial benefit to having larger amounts of memory. In light of these early findings, we plan to extend this research, both in static environments (to test our

Downloaded from http://direct.mit.edu/isal/proceedings-pdf/alif2016/28/408/2326031/978-0-262-33936-0-ch068.pdf by guest on 22 May 2025

analytical predictions of the value of memory) and in dynamic coevolutionary environments (to study the practical evolution of memory in realistic scenarios).

For static environments, we plan to explore the evolutionary response of players to imperfect opponents, such as those that attempt to engage in Tit-for-Tat, but make occasional errors. A single mistake can spiral into a high level of defection and much lower overall payouts, but if a player uses larger amounts of memory, it will be able to recognize and forgive mistakes for a longer-term benefit. We will also explore introducing longer-term memory that the player can set as it chooses. We will provide these players with combinations of opponents that require long-term memory to receive optimal payouts, such as Always Cooperate and Tit-for-Tat. In such cases, a player with long-term memory will be able to initially probe to determine whether its opponent responds negatively to a defection. If so, it can play Tit-for-Tat from then on (starting with a cooperation). On the other hand, if the opponent does not retaliate, the player knows that it can play Always Defect from then on out for a larger payout.

Dynamic environments have an even wider potential for helping us learn more about the evolution of memory. As of now, it is challenging to evolve cooperative strategies *de novo*. They require memory to increase–immediately incurring a cost–but no gain is realized until a cooperative strategy is in place and multiple players are using it and interacting. We plan to explore structured populations with smaller, local groups where kin selection effects can dominate and selection is weaker, allowing these strategies to more easily come into play. We plan to also explore more stabilizing forces once players are engaging in cooperation so that it doesn't evolve away as easily as we saw here.

Overall, this work is an important step in studying the early evolution of memory utilization, and insights from it are likely to be valuable in informing other real and artificial life studies involving the evolution of intelligence.

Acknowledgments

We extend our thanks to Michael Wiser, Alexander Lalejini, and Anya Vostinar for their comments on early drafts of this manuscript. This research has been supported by the National Science Foundation (NSF) BEACON Center under Cooperative Agreement DBI-0939454, by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1424871 awarded to ELD, and by Michigan State University through computational resources provided by the Institute for Cyber-Enabled Research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

Axelrod, R. (1987). The evolution of strategies in the iterated prisoners dilemma. *The dynamics of norms*, pages 1–16.

Barton, R. A. (2012). Embodied cognitive evolution and the cerebellum. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1599):2097–2107.

Brunauer, R., Lcker, A., Mayer, H. A., Mitterlechner, G., and Payer, H. (2007). Evolution of Iterated Prisoner's Dilemma Strategies with Different History Lengths in Static and Cultural Environments. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, SAC '07, pages 720–727, New York, NY, USA. ACM.

Crowley, P. H., Provencher, L., Sloane, S., Dugatkin, L. A., Spohn, B., Rogers, L., and Alfieri, M. (1996). Evolving cooperation: the role of individual recognition. 37(1):49–66.

Dukas, R. (1999). Costs of memory: ideas and predictions. *Journal of Theoretical Biology*, 197(1):41–50.

Dunlap, A. S. and Stephens, D. W. (2009). Components of change in the evolution of learning and unlearned preference. *Proceedings of the Royal Society B: Biological Sciences*, 276(1670):3201–3208.

Golbeck, J. (2002). Evolving strategies for the prisoners dilemma. *Advances in Intelligent Systems, Fuzzy Systems, and Evolutionary Computation*, 2002:299.

Goldberg, D. E. and Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99.

Grabowski, L. M., Bryson, D. M., Dyer, F. C., Ofria, C., and Pennock, R. T. (2010). Early Evolution of Memory Usage in Digital Organisms. In *Artifical Life XII*, pages 224–231.

Kraines, D. P. and Kraines, V. Y. (2000). Natural Selection of Memory-one Strategies for the Iterated Prisoner's Dilemma. *Journal of Theoretical Biology*, 203(4):335–355.

Liverence, B. and Franconeri, S. (2015). Human cache memory enables ultrafast serial access to spatial representations. *Journal of Vision*, 15(12):1292.

Mayley, G. (1996). Landscapes, learning costs, and genetic assimilation. 4(3):213.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA.

Downloaded from http://direct.mit.edu/isal/proceedings-pdf/alif2016/28/408/2326031/978-0-262-33936-0-ch068.pdf by guest on 22 May 2025

Sherry, D. F. and Schacter, D. L. (1987). The evolution of multiple memory systems. *Psychological Review*, 94(4):439–454.

Soto, D., Rotshtein, P., and Kanai, R. (2014). Parietal structure and function explain human variation in working memory biases of visual attention. *NeuroImage*, 89:289–296.

Downloaded from http://direct.mit.edu/isal/proceedings-pdf/alif2016/28/408/2326031/978-0-262-33936-0-ch068.pdf by guest on 22 May 2025