Classical Prisoner’s Dilemma Strategies Simulation

Zeying Li
Department of Economy
Email: zeyli@ucdavis.edu

Overall Introduction

The classical Prisoner’s Dilemma (PD) has been the standard cooperation research problem for decades, with its central question being: why cooperate when one could get more personal benefit from betraying? In this final project, we resolve that very dilemma by running a series of agent-based simulation experiments with a version of the MASON modeling toolkit modified for this purpose. We systematically vary parameters in five themes of simulation: payoff values, mutation rates, grouping behavior, and memory size. We want to test how several strategies ranging from simple Defectors and Cooperators to more advanced ones like Walkaway and Tit-for-Tat perform as they vie for resources and reproduction opportunities.

This work is particularly remarkable since collaboration is at the core of so many types of sophisticated social organisation, ranging from bacterial colonies through to human civilization. The PD model is a perspicuous one in which minimal payoff structure adjustments or gradual environmental variation (for example, local reproduction or limiting movement) have profound effects. Through experiment in silico, we shall determine under which conditions cooperative activity will arise and persist in spite of the pervasive incentive to defector.

Aside from simply transforming the payoff matrix, these experiments add agent motion patterns, group (or patch) formation, and local reproduction—hybrids of previous research, e.g., Aktipis’s. Particular concerns are whether it allows Defectors to survive when given a partial payoff for mutual defection (compared to zero), whether a non-zero mutation rate avoids strategy ”lock-in,” and how punishing naive Cooperators (with a negative sucker’s payoff) influences total cooperation. Finally, we will attempt to gather results on all five subjects, seeing how a number of parameters simultaneously promote or discourage cooperation and contrast them with famous results in evolutionary game theory.

All the action happens on a two-dimensional, toroidal grid, meaning the edges wrap around and neighbors can come from any direction. Depending on how the simulation is set, each cell can only host one agent at a time or potentially host multiple agents together. At the very start, a certain number of agents—like 2,500 or 5,000—get randomly placed on this grid, each assigned one of several strategies (such as always cooperate, always defect, a walkaway variant, or something more conditional like Tit-for-Tat or Pavlov). Agents that use Tit-for-Tat or Pavlov maintain a small memory of how they and their opponent behaved the last time they met, so they can adapt on the next encounter. Whenever two agents share a cell, they play one round of a Prisoner’s Dilemma and update their resources according to the outcome. If an agent’s resources climb above a reproductive threshold (for example, 30) and the environment’s overall population limit (carrying capacity) isn’t exceeded, it replicates; but if its resources dip below zero (assuming the zero is fatal) or it reaches a maximum age, it disappears from the grid. Movement also varies by strategy: “mobile” agents (like a mobile defector or a mobile Pavlov) search for someone to interact with and may randomly relocate if they fail, “walkaway” agents leave if they’re exploited by a defector, and “stationary” agents never move at all. Each time step, every agent takes a turn to possibly interact, move, age, check for reproduction, and potentially mutate its strategy in offspring if the mutation rate is nonzero. Throughout the run, the simulation’s “Experimenter” can log how each strategy’s population size evolves, reflecting which strategies earn enough payoffs to thrive and which fade out under different conditions (for example, whether the sucker’s payoff is large or small, whether grouping is allowed, or how big the memory size is).

I. Topic 1

A. Introduction (Topic 1)

In the first topic, we explore how defender Defectors (who always defect) could play versus defender Cooperators (who always cooperate). When we vary the sucker’s payoff (the penalty for cooperating against a defector), defender Defectors will play versus defender Cooperators in a Prisoner’s Dilemma game. It is a significant question in the sense that a small change could make or break the emergence of widespread defection or continued cooperation. We are assuming that when the payoff to a sucker is weakly negative, Defectors will exploit Cooperators considerably. But as it is strongly negative, Cooperators will have a good chance of gaining the initiative—especially if circumstances allow them to cluster and limit their contact with defectors.

B. Methods (Topic 1)

We have two strategies—COOPERATOR and DEFECTOR—each started in about equal numbers in an Environment that determines the Prisoner's Dilemma payoffs. The parameter sweeps include varying the sucker's payoff (e.g., -1, -5, or -10), enabling/disabling grouping via the `groups_or_patches` setting, and testing different `mutationRate` values (0.0 or 0.01). All agents start with resources in some range, invoke `Agent.play()`, and replicate if they've accumulated adequate resources without exceeding `carryingCapacity`. The simulation is run for 2501 time steps, and the `Experimenter` class records the number of Cooperators and Defectors after every run.

C. Results (Topic 1)

With standard PD payoffs (T = 5, R = 3, P = 0, and variable S), Cooperators and Defectors behaved normally when we tested various `mutationRate`, `useLowerBoundsSurvival`, and the mutual defector payoff (`defect_defector`). With a favorable mutation rate of 0.01 and properly negative S, Cooperators remained in dominance all the time, particularly with `groups_or_patches = True`. The ability to group allowed Cooperators to buffer themselves from being taken advantage of by free-rider Defectors. With ungrouped populations, Cooperators needed S to be very negative (e.g., -5 or worse) before they could catch up, but they were successful in dominating Defectors over the long term. As soon as mutation was disabled (0.0), whichever strategy won out early would generally wipe the other out entirely. This demonstrates how mutation can prop up a losing strategy, sometimes enabling a turnaround. In addition, disabling lower-bound survival rules enabled resource-scarce Defectors to survive, thwarting Cooperators' typical dominance. Providing mutual Defectors with a lucrative payoff (e.g., `defect_defector = 1.0`) significantly boosted Defectors' prevalence in the population since they no longer exterminated one another.

D. Discussion (Topic 1)

All these findings verify some previous results about the Prisoner's Dilemma. First, Cooperators perform optimally if the payoff to the sucker is very unfavorable and if they can bunch (in clumps), avoiding as much as possible contact with Defectors. Second, adding mutation guarantees that an approaching extinction strategy may occasionally reappear, which means that early leaders cannot totally trap. Third, environmental rules (e.g., lower-bound survival) and mutual-defection payoffs can tip the balance substantially—if Defectors receive a positive payoff when paired against themselves, their worst flaw is removed, and that is more likely to trigger collapsing cooperation. Primarily, even slight punishing of Defectors—e.g., a negative sucker's payoff or zero pay for mutual defection—offers cooperation. Future research can build on this by introducing more advanced payoff structures or by mixing mobility and memory in naive Cooperators or Defectors to determine if cooperation would endure when Defectors have mutual payoffs.

II. Topic 2

A. Methods (Topic 2)

We only use Walkaway Cooperators (WCoop) and Walkaway Defectors (WDef) in this case. We begin each run with 2500 WCoop and 2500 WDef agents on a 100×100 grid (5000 total). We alter the sucker's payoff to -10 and keep T=5, R=3, and P=0 fixed for the PD. We also reverse `groups_or_patches` to see if clustering has any impact on the dynamic between these two mobile strategies. Simulation steps ranged from 50 or 100 up to 2500 and some number of replicates (1 to 5) for replication. All of the experiments measure the fraction of WCoop vs. WDef at the final time and output who is winning in varying scenarios.

B. Introduction (Topic 2)

In the second issue, we substitute naive strategies by two ”walkaway” types, i.e., to see how Walkaway Cooperators fare against Walkaway Defectors. The walkaway model allows the agents to walk away from losing interactions, and thus the question is whether this form of partner choice gives cooperators a clear benefit or, vice versa, allows defectors to exploit cooperators over and over again and walk away.

C. Results (Topic 2)

Replacing naive Cooperators and Defectors with walkaway equivalents revealed that Walkaway Defectors had a definite advantage, especially when the sucker's payoff was not quite terrible or when grouping was disabled. Although Walkaway Cooperators can make an exit from cheating relationships, Walkaway Defectors also benefit from mobility, roaming around to find isolated cooperators. When `mutationRate` had been 0.01, minority strategies would occasionally re-emerge, building a tenuous balance which tended to work in favor of WDef if its intermediate PD conditions allowed it to exploit WCoop. When mutation was 0.0, whatever walkaway strategy emerged early in the run won out. In certain group runs, however, isolated groups of WCoop formed stable clusters that were invulnerable to penetration.

D. Discussion (Topic 2)

Mobility is considered here to be a double-edged sword. While WCoop keeps defectors from repeated exploitation, WDef also can ”walk away” from losing encounters and exploit weak cooperators elsewhere. Clustered environments can tip the balance in favor of WCoop by enabling big cooperative clusters, but invasion by defectors, particularly when the sucker’s payoff is low, can render invasion easy for defectors to perform. The presence of a small but nonzero mutation rate prevents both strategies from being driven to extinction, at times allowing cooperators (or defectors) to recover. Future releases may allow for more evolved partner-choice conventions or memory-based strategies that enhance walkaway cooperators’ effectiveness in clustering.

III. Topic 3

A. Introduction (Topic 3)

We introduce memory-based cooperation by Tit-for-Tat (TFT) mobile and compare it with Walkaway Defectors. The issue is whether punishment in unison—punishing defectors and rewarding earlier cooperative players— is strong enough to beat the mobility of walkaway defectors. We test if memory capacity and sucker punishment size matter in this game, hypothesizing that with adequate memory capacity and an effectively negative cost to cooperate against defectors, TFTMobile will effectively deter defectors.

B. Methods (Topic 3)

Here, `TFT_MOBILE` and `WALKAWAY_DEFECTOR` agents play repeated times. TFTMobile remembers every opponent’s previous move in a memory (`memorySize` 1 to 3) and cooperates or defects accordingly. Walkaway Defectors always defect and walk away if they receive small payoff from a meeting. We sweep parameters over `memorySize` (1, 2, 3) and sucker’s payoff (e.g., -1 vs. -10). We also experiment with `groups_or_patches` and `mutationRate` and check final strategy distribution after 2501 steps.

C. Results (Topic 3)

TFTMobile vs WDef, played pairwise, was also highly sensitive to sucker’s payoff and memory size. With S= -1 and `memorySize`=1, WDef tended to win because TFTMobile was unable to retaliate reliably when defected upon. But a 2 or 3 memory size gave TFTMobile the advantage in most runs, especially in clustered settings where proximity to known defectors made it feasible to punish in revenge. At S=-10, this trend was even more pronounced: WDef could outperform a starting `memorySize`=1 TFTMobile, but with `memorySize` being 2 or 3, TFTMobile consistently kept defectors at bay, often dominating as overwhelming majorities or virtual monopolies.

D. Discussion (Topic 3)

The results validate that memory capacity and sucker’s payoff size matter in resisting defection. Short memory window means that Tit-for-Tat cannot punish repeated cheaters, but `memorySize`=2 or 3 provides enough history to retaliate against repeated cheating. In addition, making the sucker’s payoff strongly negative makes cheating more expensive, so that TFTMobile ”learns” who cheats and defectors will be punished. The non-stationary environment emphasizes this because neighbors are met more than once. Future research can experiment with partial recall or add some naive strategies as a percentage of the game and see if it changes TFTMobile’s dominance when faced with walkaway-type defectors.

IV. Topic 4

A. Introduction (Topic 4)

Under our fourth theme, we allow a variety of strategies to co-exist at the same time within the same structure. All eight simulation code strategies—Cooperator, Defector, Walkaway Cooperator, Walkaway Defector, immobile and mobile Tit-for-Tat, and immobile and mobile Pavlov—are vying for the same resources with the same PD payoffs. Our primary aim is to see which strategies survive or co-exist at the end of simulation, as a function of sucker’s payoff, memory capacity, and agglomeration.

B. Methods (Topic 4)

We create an equal number of each strategy (eight total) on a 100×100 grid using `Environment.makeAgents()`. We cycle `memorySize` (1, 2, 3) for TFT and Pavlov strategies, sucker’s payoff (e.g., -1 vs. -10), and reverse `groups_or_patches`. Each agent reproduces when their resources are greater than `resoucesToReproduce`, up to `carryingCapacity`. We run 2501 steps and record the end ratios of all strategies to provide a complete picture of who ”wins” with each parameter setting.

C. Results (Topic 4)

For a payoff to the sucker of -1, naive Defectors and Cooperators scored worse than other higher-scoring strategies like Pavlov or Tit-for-Tat. Walkaway Defectors and Cooperators collaborated but were eclipsed when memory length was more than 2 or 3. Pavlov and TFT (fixed or floating) generated clusters that eliminated naive or all-walkaway types in clustered runs, but holdout enclaves of walkaways remained. When the sucker’s payoff dropped to -10, memory-based strategies took over again, building a large lead over naive strategies while decreasing walkaway types. Pavlov and TFT had a good opportunity to retaliate or defect-punish Defectors, and Pavlov's "lose–shift" logic functioned beautifully in groupable situations.

D. Discussion (Topic 4)

As long as all relevant strategy types must play against one another, conditional cooperators (Pavlov, TFT) will dominate naive or walkaway only strategies, provided that there is enough memory or repeated play. Naive Defectors and Naive Cooperators will only survive in a forgiving environment but lag behind experience-learning agents to adjust their moves. Mobility gives walkaway agents room to move in search of new encounters, but after Pavlov or TFT stabilize in clusters, Defectors can no longer retain resources. A higher negative sucker's payoff further establishes memory-based cooperators' dominance. Futures might involve the provision for endogenous development of payoffs or memory or investigation of partial movement rules constructing complex interaction "dances" between such advanced strategies.

V. Topic 5

A. Introduction (Topic 5)

For our last topic, we think about a mobile version of Pavlov—singleton as PavlovMobile—to compete against Walkaway Defectors when the memory world of their rivals is diverse, and their rewards are dissimilar. The past papers anticipate that Walkaway Defectors play reasonably well when they are opposed by naive or low-memory players. Otherwise, memory-based strategies like Tit-for-Tat will punish a defector if it is costly to punish suckers. PavlovMobile plays a "win–stay, lose–shift" strategy, cooperating or defecting based on payoff history in past interactions, while Walkaway Defectors defect and shift when there is no sufficient reward in exploitation. Varying the reward to the sucker (1 vs. 10) and memory size (1, 2, 3), we explore if flexibility of behavior (Pavlov) works better than mobility-based exploitation (Walkaway Defect).

B. Methods (Topic 5)

Every run of the simulation starts with 2500 PavlovMobile agents and 2500 Walkaway Defectors on a 100×100 grid, and none of the other strategies. PavlovMobile's memory capacity may be 1, 2, or 3. We run two payoff sweeps: one with S=1 and one with S=10 (with T=5, R=3, P=0). We also flip `groups_or_patches` between true and false. Other parameters are `mutationRate`=0.01, `resoucesToReproduce`=30.0, `carryingCapacity`=5000.0, and `useLowerBoundsSurvival`=true. Each simulation lasts for 2501 time steps, with five runs per condition. We save the final ratios of PavlovMobile and Walkaway Defectors.

C. Results (Topic 5)

Sweep #1 (S=1).
With on-grouping and `memorySize`=1, Walkaway Defectors dominated because PavlovMobile could not successfully switch strategy upon defection. At `memorySize`=2, PavlovMobile did significantly better, lasting around 70–80% of the population several times. By `memorySize`=3, it lived around 85–90% as frequently, which means that multi-round memory helped PavlovMobile punish persistent defection. Ungrouped runs also followed the same pattern but to a lesser extent—Walkaway Defectors had a higher percentage at `memorySize`=2 (about 40–50%) but were still losing at `memorySize`=3.

Sweep #2 (S=10).
When sucker's payoff is -10, the risk of defection is much greater. Walkaway Defectors continued to dominate in `memorySize`=1, but in `memorySize`=2, PavlovMobile would rise to about 80% or higher, and with `memorySize`=3, it rose well over 90% in all but a few runs. Grouping made it worse by having repeated interaction allow PavlovMobile to switch over and repeatedly punish defectors.

D. Discussion (Topic 5)

These findings indicate that PavlovMobile can catch up to Walkaway Defectors if it has ample experience (`memorySize`2) and punishment for cheating is enough. `memorySize`=1 is still too "memory-less," enabling defectors to exploit them repeatedly. But `memorySize`=2 or 3 renders PavlovMobile's "lose–shift" strategy significantly more effective, rapidly punishing repeat defectors and starving defectors, especially in a clustered environment. The large reward of -10 again increases the stakes: after several rounds of defectors, Walkaway Defectors cannot recover as PavlovMobile starts to retaliate with defection consistently. The memory size is the deciding factor: for `memorySize`=2 and higher, Pavlov agents are able to coordinate sufficiently to drain Walkaway Defectors' resources, particularly in patches. Future research can explore hybrid strategies combining mobility and constrained Pavlov logic or involve moving ecological parameters that occasionally raise defection to see if PavlovMobile will still reign supreme.

VI. General Discussion

Across these simulations, our results are consistent to a great extent with Athena Aktipis's (2004) argument that ”walk away” behavior is a good strategy for encouraging cooperation by enabling agents to avoid repeated exploitation.

Even in plain situations (Topic #1) where no walk-away mechanisms were defined, Cooperators employed clustering to keep Defectors at arm's length whenever the sucker's payoff was heavily negative. This again supports Aktipis's overall argument that partner selection—learned by movement and clustering—unstable undercuts exploitative relationships and fosters cooperation. Nevertheless, we discovered situations where defection dominated, i.e., where mutual cheating was punished or when there were constraints that allowed Defectors to starve. If it was very unpleasant to cheat and be caught, naive memory-based cooperators would convincingly dominate if they punished or excluded Defectors, especially in repeated games.

Memory-based approaches like Tit-for-Tat or Pavlov performed especially well once memory capacity was two or three steps. That was especially true when the world had options for local interaction that wove over and over the same neighbors. In other worlds, naive Cooperators or short-memory strategies fared poorly in worlds where Defectors could move freely. But when Defectors too had to worry about running out of resources as well, certain reactive or cooperative strategies were sufficient to turn around the fortunes against them. This added another level of complication, illustrating how survival or environmental problems could be as captivating as the payoff matrix itself.

Constraints (e.g., `useLowerBoundsSurvival`=true) allowed cooperators to ”starve out” Defectors, whereas eliminating survival ceilings allowed suffering Defectors to survive. Capacity within the carrying also regulated whether cooperators would or would not explode once they got ahead. Thus, in ecologically realistically separated systems, cooperation can be even stronger—or, conversely, defection will be available if resource limitations are eliminated. Directions for future research have two broad paths.

First, adding hybrid strategies such as partial walk-away and partial memory would lead to more similar behavior for actual agents who can leave exploiters and recall past experience. Second, incorporating dynamic payoffs or seasonality could reveal how defectors and cooperators learn to accommodate shifting situations. Overall, our results accord with Aktipis and other evolutionary theorists: cooperation is stable in certain ecological and cognitive environments (large enough negative payoffs to cheating, adequate memory for retaliation, or ability to dump poor partners), but defection recurs in certain guises as soon as one of those props is removed. Employing these agent-based models, we find that cooperation will generally succeed, but by no means universally so—its success is founded on the delicate interplay among payoff matrices, mobility and memory choices, and environmental constraints.

VII. Appendix : graphs

TABLE I: Simulation Results: Groups/Patches vs. No Groups/Patches (Topic 1)

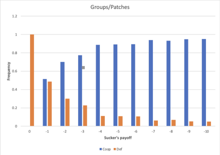
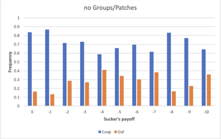
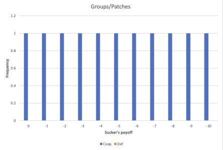
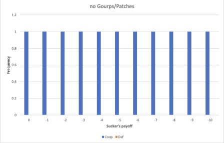
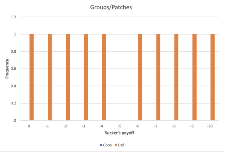
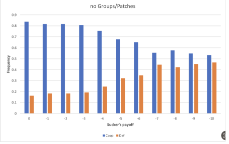
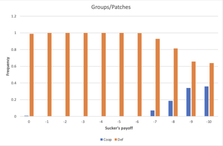
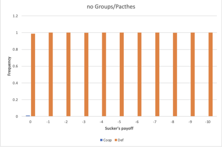
Sweep	Groups/Patches	No Groups/Patches
Sweep 1		
Sweep 2		
Sweep 3		
Sweep 4		

TABLE II: Simulation Results (Topic 2)

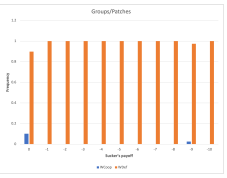
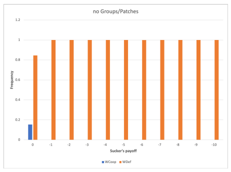
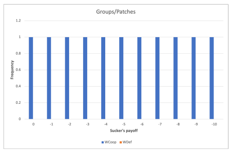
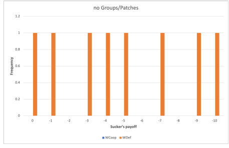
Sweep	Groups/Patches	No Groups/Patches
Sweep 1		
Sweep 2		

TABLE III: Simulation Results (Topic 3)

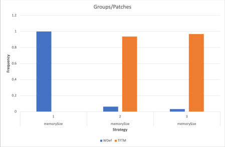
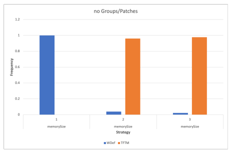
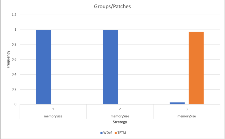
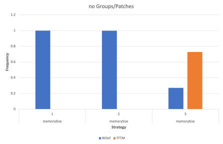
Sweep	Groups/Patches	No Groups/Patches
Sweep 1		
Sweep 2		

TABLE IV: Simulation Results (Topic 4)

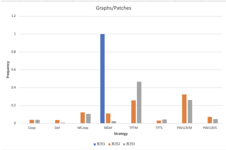
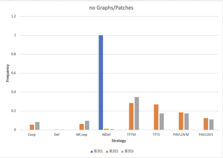
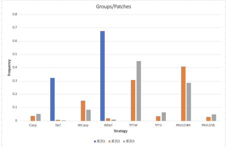
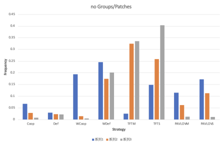
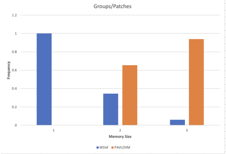
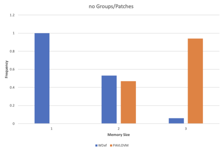
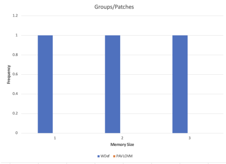
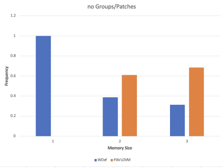
Sweep	Groups/Patches	No Groups/Patches
Sweep 1		
Sweep 2		

TABLE V: Simulation Results (Topic 5)

Sweep	Groups/Patches	No Groups/Patches
Sweep 1		
Sweep 2	 Groups/Patches Frequency Memory Size	 no Groups/Patches Frequency Memory Size