How evolutionary selection can be used to establish team behaviour in the Iterated Prisoner's Dilemma

Nicklas Lallo, Gustav Grund Pihlgren, Erik Sievers, Erik Thorsell

***Abstract*—The Iterated Prisoner's Dilemma is well known in the realm of Game Theory. In this report we present a model where self interested agents play the finitely Iterated Prisoner's Dilemma on a square lattice while also allowing for the agents to use strategies that evolve in accordance to various evolutionary selection settings. This adds a spatial aspect to the game, as well as an evolutionary. The novelty of the report is the introduction of the certain selection settings that attempts to mimic team behaviour.**

I. Introduction

The Prisoner's Dilemma is a well known game theoretical concept [1], depicting the dilemma that occurs when two arrestees are each offered a deal with their arrestor. Figure 1 shows the pay-off matrix used in Axelrod's famous tournament [1], and the version used throughout this report. The arrestees – Alice and Bob – simultaneously choose to either cooperate or defect, and depending on their choices they receive one of four expected pay-offs.

		Bob C	**Bob** D
Alice	C	(3,3)	(0,5)
	D	(5,0)	(1,1)

Fig. 1. Matrix showing the payoffs for the Prisoner's Dilemma.

A. The Iterated Prisoner's Dilemma

The Iterated Prisoner's Dilemma is a version of the original game where a finite (or infinite) number of rounds are played. The goal of the game is to maximise ones score over all rounds. Axelrod and Hamilton discussed, in their paper from 1981 [1], how advanced strategies could be used in order to maximise the score of an agent over time. A commonly used strategy (and also the strategy that won the tournament Axelrod hosted before writing the previously mentioned article) is "Tit-for-tat". An agent playing according to Tit-for-Tat will start off by cooperating but will then mimic her opponent's previous move.

B. Evolutionary models

Evolutionary models are models implementing some behaviour, similar to that of genetics, to optimise their ability to "stay alive" in a population. By encoding game theoretic strategies like a genome, and simulating natural selection and mutation, new improved strategies can be found.

C. The Iterated Prisoner's Dilemma, played in teams

Coordinated cooperation when playing the Iterated Prisoner's Dilemma has been tried before. Rogers et al. showed – by winning the 20th anniversary tournament of Axelrod's original competition – that several agents playing according to a "Master/Slave-strategy" could beat Tit-for-Tat [2]. In the tournament Rogers won, it was however not the collective team (i.e. all submitted strategies) that won, but the slave strategies fed the master strategy and pushed the master strategy towards victory. The tournament was also played as a round-robin tournament, not in a spatial environment.

D. Scope

This report examines how team behaviour can arise in a population of agents under different *evolutionary selection settings*. A team is defined as a collection of players, all playing to maximise its team's utility. In this specific case, we will model the finitely Iterated Prisoner's Dilemma, played on a lattice, where the objective of an agent is to maximise its team's coverage of the lattice. The various evolutionary selection settings will correspond to the rules for which an agent *captures* another tile on the lattice. The game will be played with a finite memory size of up to – and including – 3 bits (see Section II-B).

II. Method

The model used for simulating the Iterated Prisoner's Dilemma is represented by a square lattice containing 4096 tiles, each tile being populated by an agent. Figure 2 shows the beginning of a simulation, where the lattice is randomly filled with a set number of agents, playing according to one of four strategies:

- Always Defect (AD)
- Always Cooperate (AC)
- Tit-for-Tat (TfT)
- Anti-Tit-for-Tat (aTfT)

A more in depth description of various strategies is given in Section II-B and further discussed in Section IV-A.

Depending on the settings of the model, each tile on the lattice has one or two colours. The centre of the square depicts the team the tile belongs to whereas the border corresponds to the strategy the agent populating the tile is playing according to (if applicable).

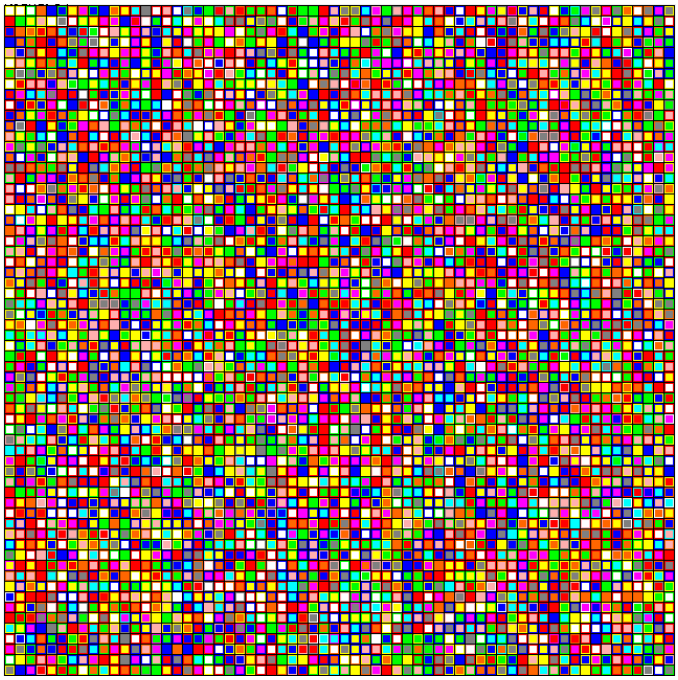

Fig. 2. A newly generated lattice, containing 4096 agents playing the Iterated Prisoner's Dilemma.

In every *generation* of the model, each agent plays each of its four orthogonal neighbours 100 times (i.e. each agent plays 400 rounds). The score of each agent is used to determine the winner of the matches. If an agent scores above all of its neighbours it remains populating its tile. If an agent's score is lower than at least one of its neighbours, the neighbouring agent with the *highest score* will take over the tile.

A. Settings of the model

The model can be tweaked in various ways. The payoff matrix can be changed, a mistake rate can be introduced, a mutation probability can be added and the team behaviour can be turned on or off (i.e. the ability for an agent to overtake a tile already populated by an agent of the same team). It is also possible to change the size of the memory used by the agents.

By introducing a mistake rate, there is a certain probability that an agent's action may be the inverse of its intended action. If the mutation probability is increased above 0, there is a probability that an agent will change its strategy by either halving or doubling the size of its strategy, changing part of its strategy by flipping one of the bits in its encoding (i.e. `0101` to `0111`) or – if team play is turned on – change which team it belongs to. The interpretation of the strategy strings are explained in detail in Section II-B.

B. Strategies and memory size

Each agent uses a finite memory to keep track of the game history. If the size of said memory is 1 bit, the agent remembers only the previous action of the opponent. This action can be either cooperate or defect (represented as 1 or 0 respectively).

As the size of the memory increases, the agent also remembers its own previous action, the opponents move of the round before that, etc. If the size of the memory is b bits, there are 2^b ways to respond to the history at hand.

Due to the binary nature of the actions, a strategy can be represented as a binary string of 0's and 1's. Figure 3 shows how a lookup table is used to map a certain history to a specific action, and how the 2^b mappings make out the agent's strategy.

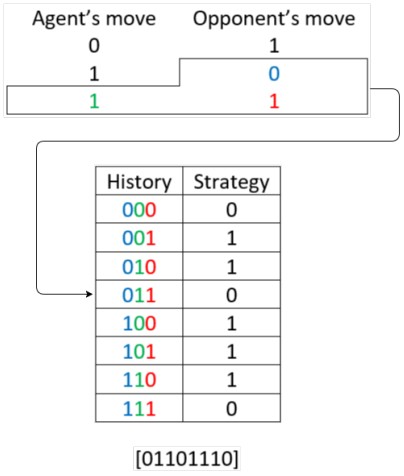

Fig. 3. A lookup table is used to map a certain history to a specific action for a player, playing with a memory size of 3 bits. The figure shows how the agent takes into account its opponent's two last actions and its own last action, mapping this history to an action in its strategy.

In the model presented, each agent has a maximum memory size of 3 bits, but each agent starts the game with a 1 bit strategy, corresponding to one of the strategies mentioned in Section II. As the game progresses the memory size may increase due to mutations, which may cause duplication of memory.

For reference, the four original strategies – and their corresponding binary representations – are:

- Always Defect (AD) – 00
- Always Cooperate (AC) – 11
- Tit-for-Tat (TfT) – 10
- Anti-Tit-for-Tat (aTfT) – 01

C. Method 1 – The control method

The first method is inspired by the model presented by Lindgren and Nordahl [3]. The agents play with their respective scores and are part of no team. Note that in Lindgren's and Nordahl's model the agents played the infinitely iterated game with their neighbours [3], whereas in this model they play the finitely iterated game.

D. Method 2 – The team average

The second method introduces teams. Each agent on the lattice plays with the average score of its team. That is, each agent plays against its neighbours and gets an individual score which is then used to calculate the team's average. The agent

then captures new tiles based on this score. This implicitly means that no agent will capture a teammate, since they both have the same score.

E. Method 3 – The team average + own score

The third method is similar to the second method, but takes into account both the performance of the team as a whole, as well as the individual agents' performance, when capturing. Each agent captures with the sum of its own score and its team's average score. Hence, the score of different agents within the same team might differ. This also means that agents can capture tiles from its teammates.

III. Result

The results presented are observations from running the model for 100 000 generations. 100 000 generations was a suitable limit, since running the simulation for longer (about 200 000 generations) showed no significant changes to the patterns or strategies observed. That is, a looping behaviour could be observed, and even though the lattice was never static, no new patterns were introduced to the loop. We call this a *stable state*, and such a state could be seen in all tests after running the simulation for 100 000 generations.

In Section IV, a thorough explanation of the more prominent strategies seen in this section is given.

A. Simulation settings

All tests were done with:

- 100 rounds of the Iterated Prisoner's Dilemma per simulation generation
- 3 bit memory size
- Mistake rate of 0.02
- Mutate probability of 0.001
    - Chance of doubling the genome size of 0.1
    - Chance of dividing the genome size by two of 0.1
    - Chance of defecting to a different team of 0.01 (when applicable)
    - If mutation neither changes the size nor team it flips a random bit in the strategy.

For the methods where teams were playing each other, there was a limit of six teams at a time.

B. Method 1 – The control method

Figure 4 shows the resulting plot after running Method 1 for 100 000 rounds. Initially (the first few hundred rounds) we see the 1 bit memory strategies `00`, `01` and `11` dominating the lattice (holding a majority of the tiles). However, almost immediately, the strategy `1001` gains a lot of ground and for the first 25 000 rounds (the only exception being a few rounds around the 12 000 round mark) `1001` dominates the lattice.

The strategy `10011001` is the duplication of `1001` and is the strategy that ultimately is able to dethrone `1001`. `1001` makes sporadic reappearances, but `10011001` dominates in the longer run.

Although the two strategies `00001000` and `00110001` never reach the top, they still manage to stay in the game indefinitely.

C. Method 2 – The team average

Adding the team element to the game, we can see how the dominating strategies change. Figure 5 shows how each strategy is either very dominating or barely staying alive. From the start we see quite a lot of noise, but after approximately 10 000 rounds, strategy `10011001` starts to dominate. We remember that `10011001` did well when running Method 1 (see Section III-B), however, after about 50 000 rounds `10011001` vanishes from the lattice and instead we see how a handful of strategies (`11011111`, `01011111`, `11111111` and to some extent `01111111`) take turns being the strategy with the most coverage.

Looking at Figure 6, we see a constant rotation of teams. What is obvious to the viewer of a live simulation becomes even more apparent when seeing the plot; no team is able to stay on top, but will dominate the lattice for a short period of time, before being corrupted and destroyed by former teammates that have mutated to another team, now exploiting its former team.

Figure 7 highlights this point even further, where instead of 100 000 rounds just 2 500 rounds are shown. Even in such a short period of time, there are no fewer than 25 clear cases of different teams dominating the lattice. More interesting still is that every team dominates the lattice at some stage during these 2 500 rounds.

D. Method 3 – The team average + own score

When each agent play with their team's score added to their own score, the spectrum of dominating strategies changes again from the previous results. After only 2000 rounds, Figure 8 shows how a clearly dominating strategy emerges – `10011001`. `10011001` has already proved itself to be a good strategy in the other tests, but when played with the settings of Method 3 it is a clear winner.

Figure 9 shows the strategies used by agents the first 2 000 rounds. The somewhat noisy start is ended with `1001` taking a clear majority of the lattice, but after only 300 rounds it is evaporated from the lattice and replaced by `10011001`.

Looking at how the teams occupy the lattice, we see an interesting difference between the result of the second and third method. In Figure 6 we saw how no team was able to conquer the entire lattice when their agents were playing according to the strategy `10011001`. In Figure 10 we do however see that most teams that dominate also conquer the majority of the lattice. In the former method, the strategy `11011001` seemed strongly correlated to `10011001`, whereas in the latter method `10011001` is so clearly dominating that no other strategy reaches the threshold of either 2 000 tiles or manages to stay alive for 50 000 rounds, while the agents are playing according to `10011001`.

Lastly, when looking at only 2 500 rounds of the simulation – in Figure 11 – it is clear that the constant rotation of teams is present in this method, as was the case in the previous one.

IV. Discussion

Below, we present a thorough analysis of the more prominent strategies seen in Section III. We also give an in-depth

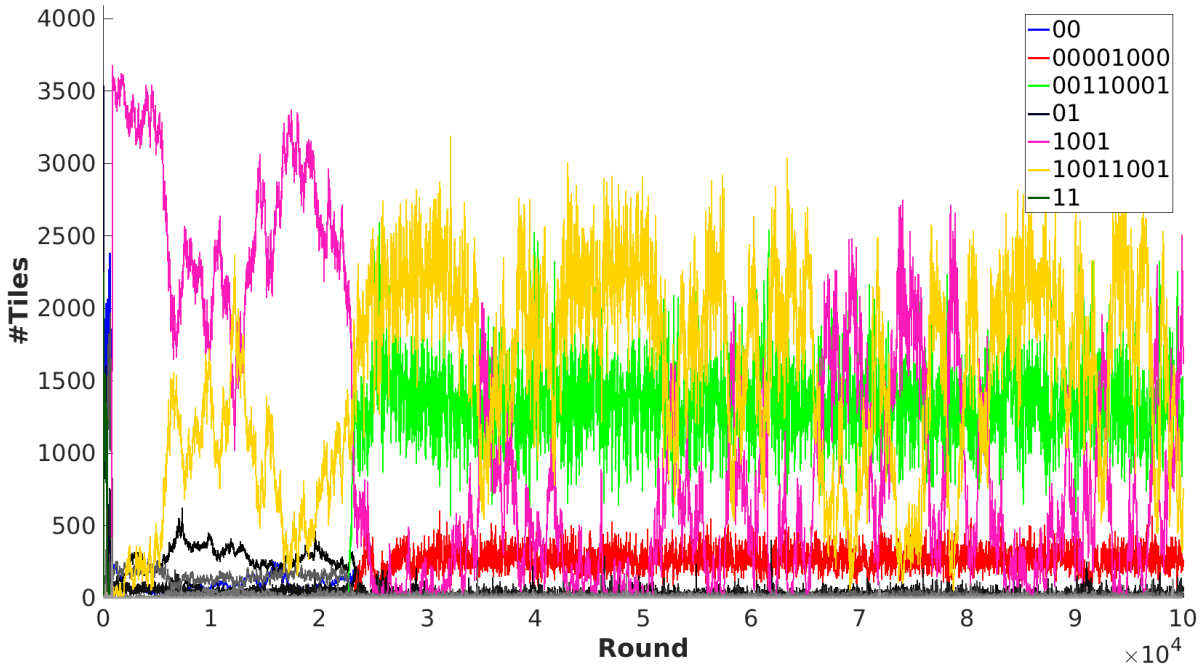

Fig. 4. Resulting plot, after running Method 1 for 100 000 rounds. It is clear that a *stable state* is reached after approximately 25 000 rounds. The legend shows the strategies which have either conquered at least 2000 tiles at some point in the game (i.e. `10011001`, or have been able to stay alive for at least 50 000 rounds (i.e. `00001000`).

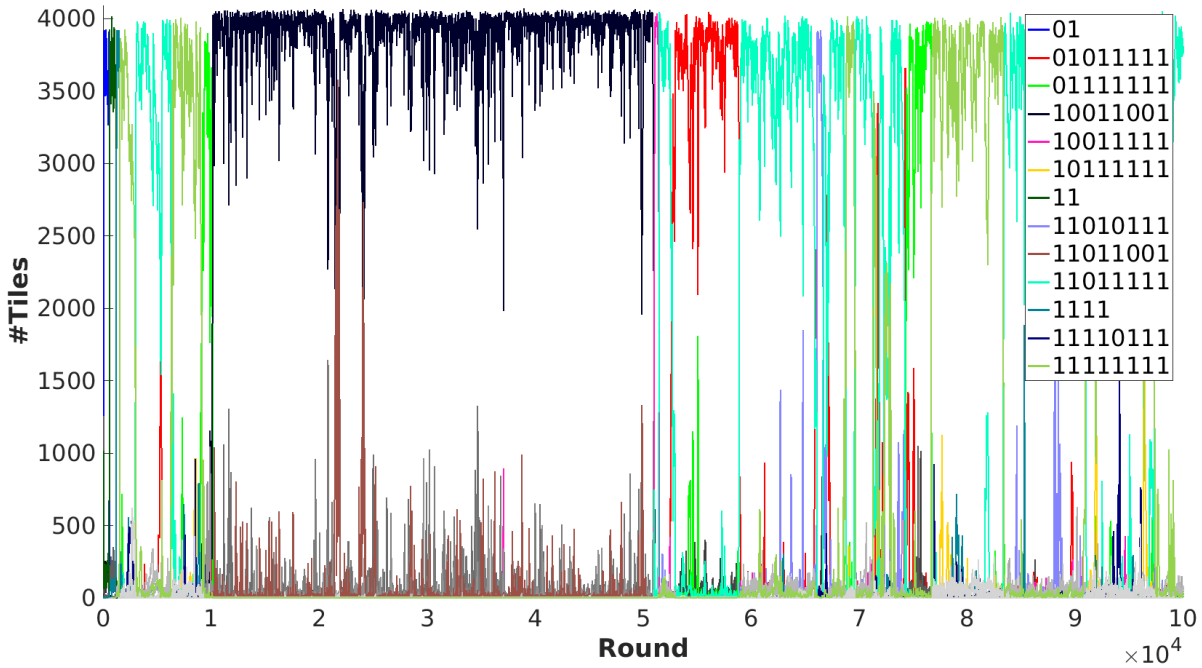

Fig. 5. Resulting plot of the strategies that occurred, after running Method 2 for 100 000 rounds. The legend shows the strategies which have either conquered at least 2 000 tiles at some point in the game, or have been able to stay alive for at least 50 000 rounds.

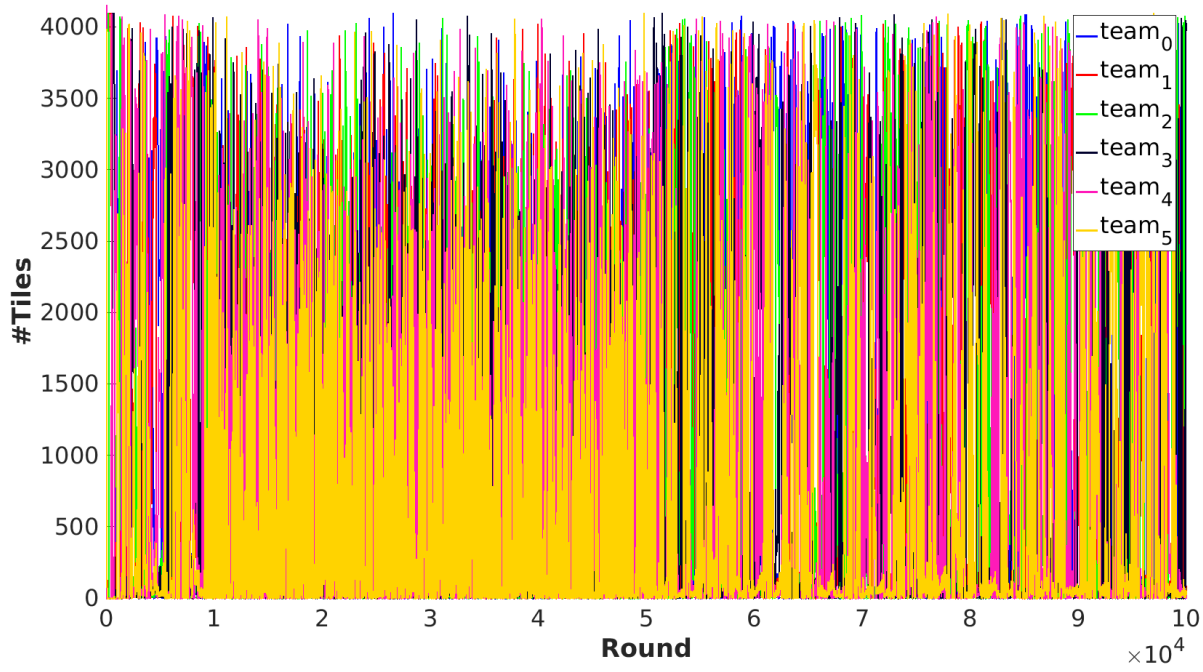

Fig. 6. Resulting plot of how the teams performed when running Method 2 for 100 000 rounds. There is no team that is able to control the majority of the lattice for any substantial amount of time.

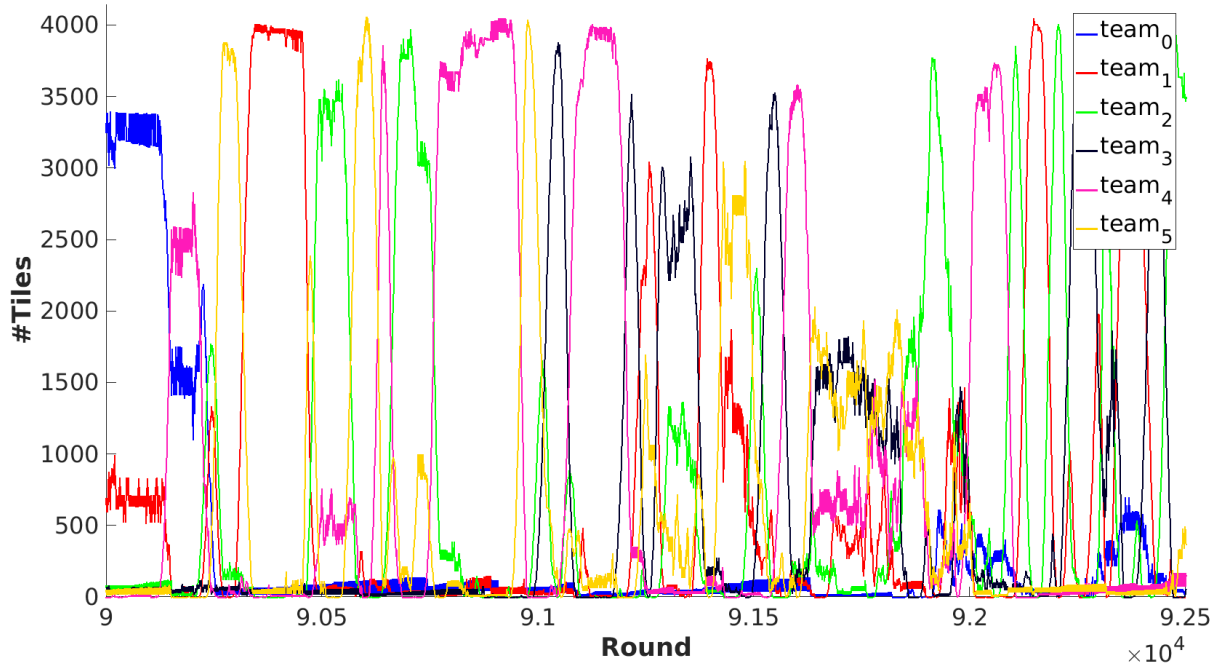

Fig. 7. A zoomed in version of Figure 6 emphasising how no team is able to control the lattice for more than a few hundred rounds.

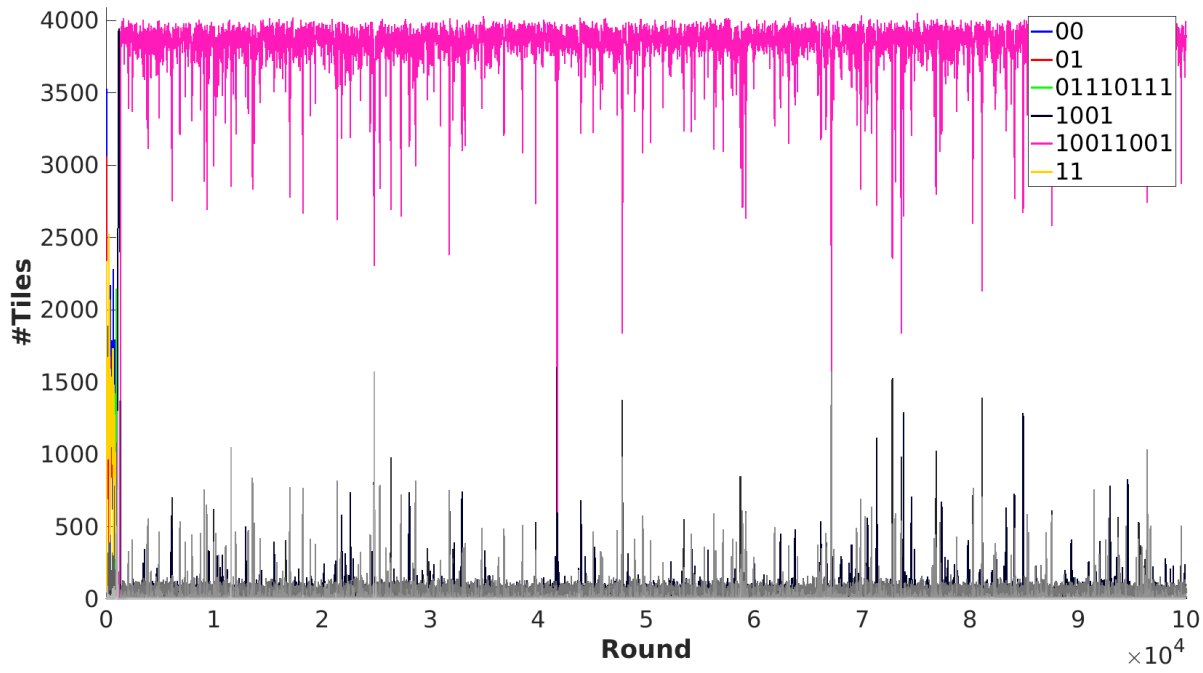

Fig. 8. Resulting plot of the strategies that occurred, after running Method 3 for 100 000 rounds. The legend shows the strategies which have either conquered at least 2000 tiles at some point in the game, or have been able to stay alive for at least 50 000 rounds.

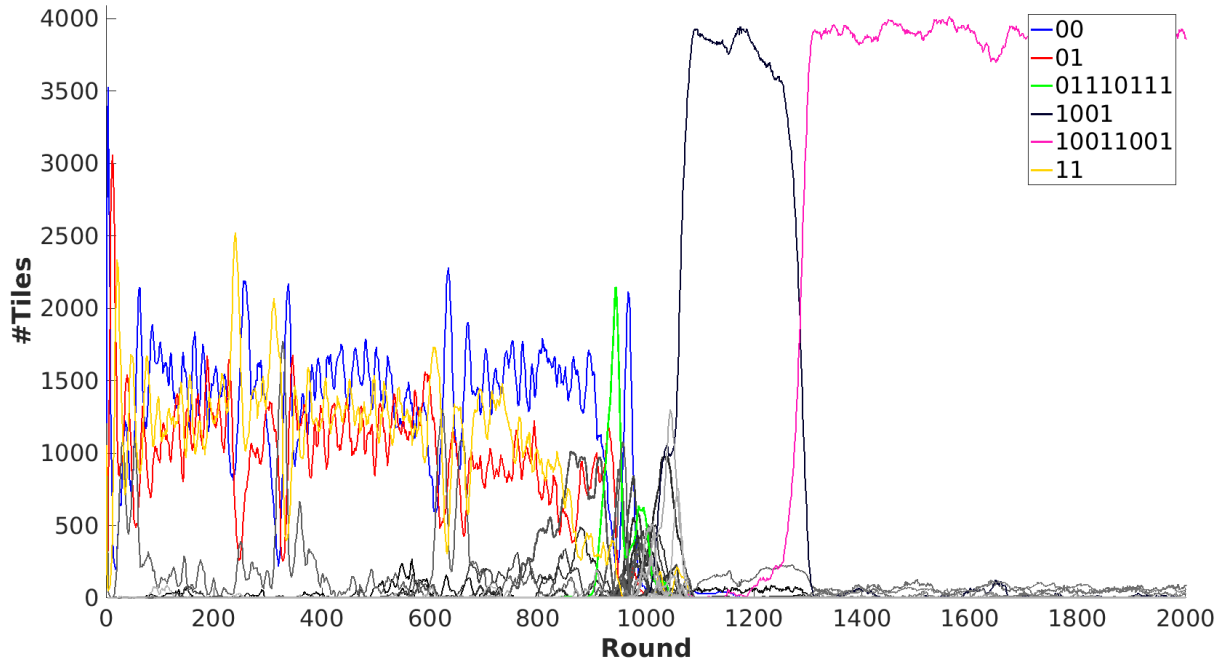

Fig. 9. Resulting plot of the strategies that occurred, when running the first 2 000 rounds of Method 3. The legend shows the strategies which at some point managed to conquer at least 2000 tiles.

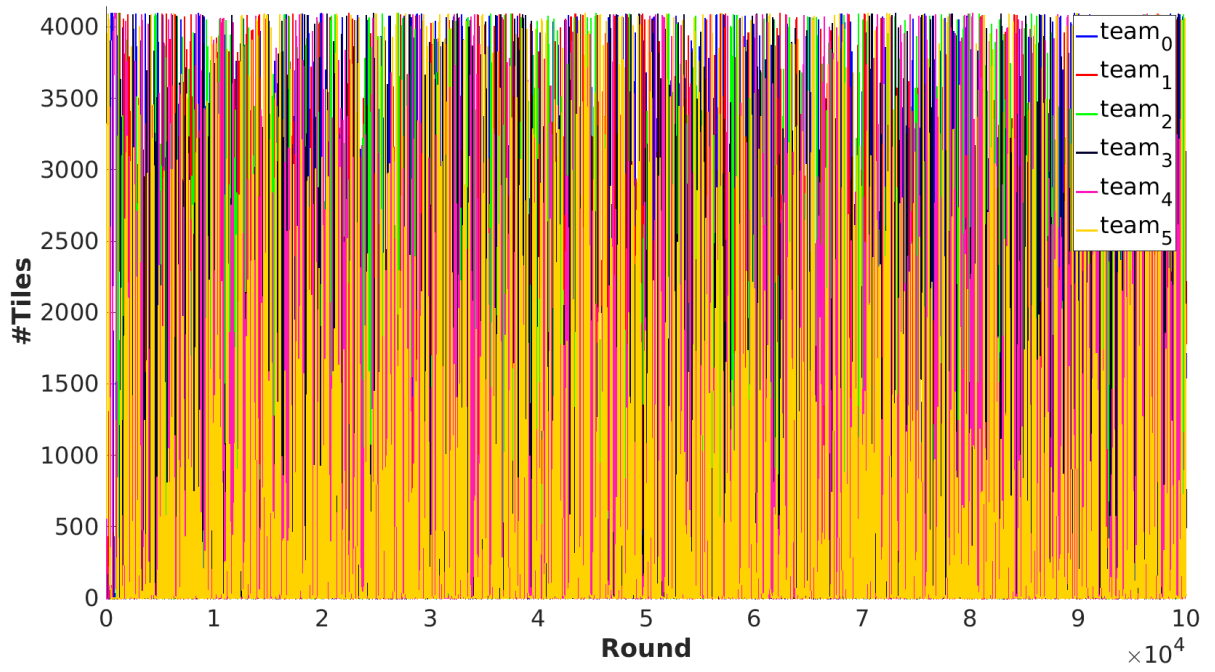

Fig. 10. Resulting plot of how the teams performed when running Method 3 for 100 000 rounds. There is no team that is able to control the majority of the lattice for any substantial amount of time.

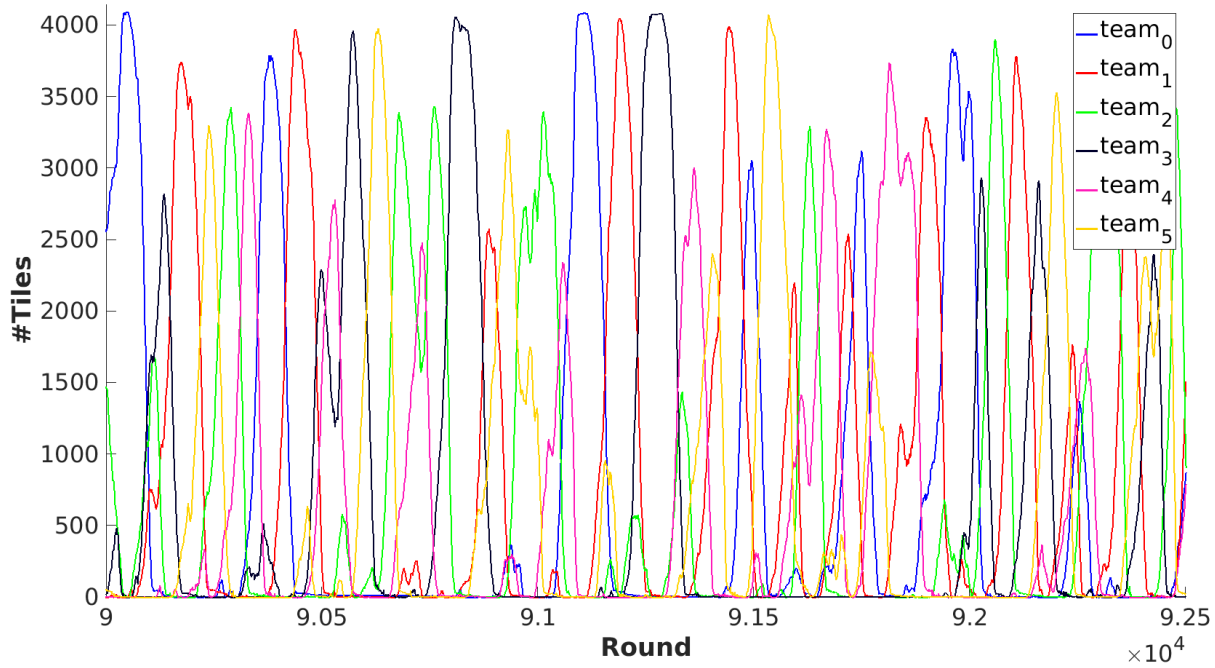

Fig. 11. A zoomed in version of Figure 10 emphasising how no team is able to control the lattice for more than a few hundred rounds.

discussion about the various simulation methods used and how their result differs from one another, as well as what this behaviour can mean.

A. Strategy Analysis

As mentioned in Section II-B, the sequences of ones and zeroes depicting the strategies in play are key in order to understand the results of the simulations. Hence, the following sections describe the more common strategies we have seen throughout our simulations.

1) Always Cooperate and Always Defect: Various versions of Always Cooperate (AC) and Always Defect (AD) show up during play of all three methods. Every strategy with only 1's is AC and every strategy with only 0's is AD.

2) 1001 and 10011001: 1001, and the functionally equivalent 10011001, both use conditional cooperative play, *Tit-for-Tat*-behaviour, and mistake correction. Cooperation arises since, if both agents cooperated the last turn, an agent playing accordingly to 1001 will decide to play cooperate the next round. If, however, any agent defected last turn, the agent will defect the next round. This is both the mistake correction and tit-for-tat behaviour at work.

1001 can however be abused, by for example AD. Due to its mistake correction, every other turn AD would gain an average score of 3, while 1001 would only get a score of 0.5 (disregarding mistakes).

3) 11110111 and 01111111: 11110111 is a highly cooperative strategy. An agent playing according to 11110111 defects only if its opponent cooperated the second-to-last turn and both agents defected the last turn. Since 11110111 starts off by cooperating, it is not difficult to draw the conclusion that this would be functionally equivalent to AC. Even if mistakes would make an agent, playing according to this strategy, defect sometimes, this has to occur exactly at the same time as the opponent defects after having cooperated, in order for the agent to defect as their actual action. However, even if this would occur the agent would return to playing cooperate the next round as its opponent no longer would have played cooperate in its second to last round.

An agent playing according to 01111111 will defect only if its opponent defected the two previous rounds, and itself defected the previous round. Hence, it can also be seen as a highly cooperative strategy.

4) 11011111: 11011111 is another highly cooperative strategy. An agent playing according to this strategy will only defect if its opponent defected the last two turns and the agent itself cooperated the last turn. This is a version of the strategy known as *Tit-for-two-Tats* that defects only if the opponent have defected twice in a row. 11011111 does however implement mistake correction, meaning that if an agent playing according to the strategy defected by mistake it will not defect the next round too; hence it will never get stuck in perpetual defection against itself.

5) 00110001: 00110001 is a version of the strategy known as *Angry Tit-for-Tat*. It is one of the three stable strategies in the control method. An agent playing according to this strategy will maintain cooperation only as long as both agents cooperate. If one agent defects the agent playing according to 00110001 will defect the following round. Additionally, if the opponent defected the second-to-last round, 00110001 will replay its previous move. Since the agent will respond to a defect with another defect it will most likely keep defecting unless mistakes allow it to resume cooperation.

6) 00001000: 00001000 is a highly defective strategy and one of the three stable strategies in the control method. An agent playing according to this strategy will only cooperate if its opponent cooperated in the second-to-last turn and then they both defected. This means that it will play somewhat cooperatively against 1001 and similar strategies as they will alternate between defecting and cooperating with one another.

B. Advantage of longer strategies

In our results we see many strong 3 bits-memory strategies that have functionally equivalent 2 bits-memory and sometimes even 1 bit-memory versions such as 10011001, 01110111 and 11111111 to name a few. In almost all of these cases the longer versions are still the more prominent ones in the stable state patterns. This would reflect on some additional benefit of having longer strategies other than the ability to play more advanced games.

One such reason could be an additional resistance to bad mutations. That is, since the strategy of the genome has 8 different components that can mutate, a mutation is less likely to change a part of the genome that is vital to its survival. Whereas if an agent has a genome with a shorter strategy, a mutation is more likely to change the strategy in such a way that it is completely ruined. In the case of longer strategies, the agent might not play considerably worse after the mutation, hence might survive a mutation from a strong strategy. If the agent can stay alive for long enough, it will likely be given the chance to spread, or mutate back into the more successful strategy, whereas a shorter strategy might have been wiped out due to a single mutation.

Another reason might be the ability to mutate from and to different 3-memory strategies. This means that if a critical mass of 3-memory strategies appears on the lattice, suddenly the longer strategy has more genomes to mutate from. While the shorter version must mutate from another 2-memory strategy (which there is fewer of since they can be less advanced and therefore not compete as well in many cases), the longer strategy can appear from a larger selection of strategies.

In Figure 4 we can clearly see an inverse relation between 1001's and 10011001's share of the lattice. This can be seen as there being a certain amount of tiles that either strategy can occupy and that the two strategies are competing for these tiles.

V. Conclusion

From the results presented in Section III and the strategy discussions in Section IV, we draw conclusions regarding the evolutionary selection settings we adopted throughout our work, and whether these settings were appropriate to mimic a team behaviour.

A. Method 1 – The control method

When simulating the control method, no team settings were used. A *stable state* was reached, containing 3 different strategies (regarding 1001 and 10011001 as equal). 10011001 held around 2000 to 2500 tiles, 00110001 held around 1000 to 1750 tiles, and 00001000 held around 250 tiles.

The control method could also been seen as a method were only *one* team is playing, each agent with its own score, and it is allowed to overtake tiles populated by teammates. When agents play according to the strategies 10011001 and 00110001, they will both cooperate at the start of the game. 10011001, having *Tit-for-Tat-like* characteristics, will keep cooperating until 00110001 defects. 00110001 is similar to *Angry Tit-for-Tat*, hence will cooperate as well – given that mistakes don't occur. As was discussed previously, an agent playing according to 00001000 is likely to play fairly well against an agent playing according to 10011001, which is why we see the strategy getting by – albeit not doing very well.

B. Method 2 – The team average

It is clear that the best possible average score a team can have is 3.0, and that this is reached when all agents in one team is cooperating. It is technically possible for a team, not dominating the lattice to have a higher average score than 3.0, but in order to dominate the lattice a team must consist of agents playing good against its own teammates – hence cooperating.

This is reflected when the average team-method is simulated. It is clear that agents playing strategies that favour cooperation are dominant. We see strategies like 10011001, 11111111 (AC) and 11011111 dominate the lattice at different points in time.

When comparing the plots in Figure 5 and 6, it is interesting to see that rounds 10 000 to 50 000 are dominated by agents playing according to the strategy 10011001, and during these rounds we observe fewer teams holding more than 3500 tiles of the lattice. This is due to the nature of 10011001 being less cooperative than for instance AC, 11011111 and 01011111. When agents play according to the latter strategies we see, in Figure 6, how more teams are able to conquer even more tiles.

We also note that during the rounds dominated by 10011001, a strategy appears that we have not yet seen, 11011001. The additional 1 in the latter means that it will cooperate ever so slightly more (if the opponent cooperated the previous round, whilst the agent itself defected, and the opponent also defected the round before that) and an agent playing accord to this strategy is likely too good against 10011001 to be wiped out, hence steals some small part of the lattice, but not good enough to take a significant chunk from 10011001.

Simulating the team behaviour we also see how the mutation probability affects the teams. It is clear from Figure 7 that no team is able to dominate the lattice for more than a few hundred rounds. This is because a team, cooperating well, is easily exploited if an agent's genome mutates into a different team and starts defecting against its old team.

C. Method 3 – The team average + own score

As can be seen in Figure 8 it takes only a few thousand rounds for the agents playing according to strategy 10011001 to establish said strategy as the dominant strategy. As this occurs, we also see how the simulation reaches its stable state. Barring only one minor fluctuation, during which 1001 takes some part of the lattice 10011001 is completely dominant perpetually.

When, in the previous method, agents playing according to the "not as good, but still viable" strategies managed to stay alive due to their teams' average score. It is clear from these results that 10011001 is the more beneficial strategy. Figure 9 shows how the start up phase of the simulation plays out. A bit of turbulence is abruptly ended by 1001 which in turn is exchanged for 10011001 as the clearly dominating strategy. 10011001's ability to cooperate well with itself, correct occurring mistakes, and punish whoever tries exploiting it makes it a superior strategy.

D. Strategies suitable for teams

From our simulations it is clear that a strategy suitable for adoption by a team should be able to cooperate well with itself. The strategy must however also be able to withstand the competitiveness that occurs in the early phase of the game, i.e. the part of the game where its team does *not* hold the majority of the lattice. Last, but not least, it is of utmost importance that it reestablishes cooperation quickly after any mistake occurs. That is, if an agent playing according to the strategy for some reason defects against its team mate, the strategy must allow the agent to start cooperating again as soon as possible.

The results from our simulations are clear, when agents play for no team, or as a part of a team with intentions to eliminate weaker parts of the team, 10011001 is an excellent strategy to adopt. When agents play as part of teams who averages their score, and are unable to eliminate weaker parts of their team on their own, 10011001 is still a viable strategy, but as we have shown it will likely be exchanged for even more cooperative strategies as the game progresses.

E. Methods reflecting teamwork

In this project we have implemented and analysed two different versions of the Iterated Prisoner's Dilemma played on a lattice that attempts to reflect teamwork using different selection settings. We have compared these to a control method and one another to see how the behaviour of the model changes under these new selection settings. The question that remains is whether or not any of those settings can be said to actually reflect teamwork.

1) Team average: When each agent plays with their team's average score we see how more cooperation is better. This could be said to reflect a team, since everyone is cooperating and not punishing their teammates for mistakes. Punishing teammates would lower the average score of the team, hence not be beneficial for the team. On the other hand, we see teams play according to cooperative strategies against opposing teams as well, which is the complete opposite of what the teams

were supposed to do in Rogers et al. [2]. Instead, each agent cooperated with its teammates and defected against all other agents.

2) Team average + own score: Giving the agents the chance to add their own performance to their teams' allowed `10011001` to become the dominant strategy. Although this is still more cooperative than the control method, where the less cooperative strategies `00001000` and `00110001` survives, the method still only gives rise to mistrusting cooperation with slower mistake correction than playing solely with team average does.

3) Comparing methods: Additionally one can, when comparing Figure 7 and Figure 11, observe that the dominating teams in Method 2 holds the lattice longer than the teams in Method 3. This could be a sign that capturing solely with team average is a better representation of teamwork than capturing with the teams average added to the agent's own score.

REFERENCES

[1] Axelrod Robert, Hamilton William D. (1981) *The Evolution of Cooperation*, Science, New Series, Vol. 211, No. 4489. (Mar. 27, 1981), pp. 1390-1396. Available: http://links.jstor.org/sici?sici=0036-8075%2819810327%293%3A211%3A4489%3C1390%3ATEOC%3E2.0.CO%3B2-6&origin=JSTOR-pdf

[2] A. Rogers, R.K. Dash, S.D. Ramchurn, P. Vytelingum, N.R. Jennings (2007) *Coordinating team players within a noisy Iterated Prisoners Dilemma tournament* Available: http://eprints.soton.ac.uk/263238/1/journal_paper.pdf

[3] Lindgren Kristian, Nordahl Mats G. (1994) *Evolutionary dynamics of spatial games*, Physica D, Vol. 75, No. 1-3. (Aug. 1, 1994), pp. 292-309