



vrije Universiteit *amsterdam*

Institute of Exact Sciences

Course Report

Evolving cooperation in the non-iterated prisoner's dilemma: A small network inspired approach.

Nicolas Höning, Tomáš Kozelek

January 21, 2008

Supervisor:

Martijn Schut

Vrije Universiteit Amsterdam

Course: "Advanced Self Organisation" in 2007/2008

Abstract

In this report, we conduct a design study in the realm of trust in networks of agents who play the Non-Iterated Prisoner's Dilemma. We propose that if these networks are Small World Networks, agents can ask other agents they already trust if opponents can be trusted. These neighbours can in turn ask their neighbours. This process is iterated until someone knows the opponent, thus a trusted information chain is built. Such a chain has the advantage that no remote messages from untrusted sources are used to decide about someone's trustability. We show that this framework is computationally feasible. Small World Networks do in fact emerge and information chains can successfully be built within. We test a strategy using this framework against several default strategies and show that it can very well succeed.

Contents

1	Introduction	4
1.1	Objectives	5
1.2	Research Questions	5
2	Literature	7
3	Model	8
3.1	Ingredients	8
3.2	Recipe	8
3.3	Network	11
3.4	Model Categorization	11
3.5	Relation with the real-world	12
4	Implementation	14
4.1	Control loop	14
4.2	Implementation details	20
5	Experiments	22
5.1	Experimental Design	22
5.1.1	Independent Variables	22
5.1.2	Dependent Variables	22
5.2	Setup	23
5.2.1	Fixed Variables	23
5.2.1.1	Payoff matrix	23
5.2.2	Implicit Variables (Principles)	23
5.3	Results	25
5.4	Analysis	31
5.4.1	Hypotheses	31
5.4.2	Objectives	34
5.4.3	Research Questions	34
6	conclusions	36
	Bibliography	37

1 Introduction

In real interactions between humans, trust is an important notion. We feel best about an upcoming interaction if we can rely on previous interactions with that trade partner. Whenever we can't, we try to find out if someone we do already trust has experienced good interactions with him or her. In complex and dynamic markets, we seldom know our trade partners, so we try to find such chains of trust judgements every day. If we find one, using it to judge whether the interaction can be trusted might protect us from fraud.

We combine two concepts in recent scientific research to model such a world: The Non-Iterated Prisoner's Dilemma and Small World Networks. This is a design study to explore first results and pitfalls on this research path.

The Prisoner's Dilemma is a zero-sum game with two players. It depicts a situation where both opponents need to decide if they cooperate or defect. Both would achieve the most if they defect while the other cooperates (and perform worst if it is the other way around). The overall payoff is the highest for a situation where both cooperate. The clue is that a short-term rational decision is to defect, while in iterated games overall cooperational behaviour achieves best. In the non-iterated version, there is no guarantee that the players will ever see each other again. How can cooperation evolve nonetheless?

Small World Networks describe a phenomenon that seems to be inherent to human societies: The (social) network is connected in such a way that every node is reachable from every other node within a small number of steps (small when compared to the network size). This property is called the "shortest path length". Another property is the "clustering index", which is calculated by the number of neighbours that also know each other.

In our model, agents that were initialised with a small number of neighbours play the non-iterated prisoners dilemma and thereby create new neighbour relations. The general concept (of one round) is as follows :

In a sufficiently large population, two arbitrary agents are chosen and they play a single Prisoner's Dilemma game - this is repeated until every agent played once. Agents play according to (potentially) different strategies. Only some will be equipped with the strategy we propose.

The information on reputation of the opponent increases the agents ability to predict its opponents action and thus increases his payoff. The question remains: How to obtain such a reputation measure from the network?

We propose a reputation system based on "word-of-mouth" propagation of such information along chain(s) of agents. One of the key ideas is that such a chain should be very short to make this approach computationally usable. Thus, the underlying network should be of a small world type. If agents can find out about the trustability of their opponent, they can decide how to act relying on more information. Since this information was transported by agents that trust one another, it should be reliable.

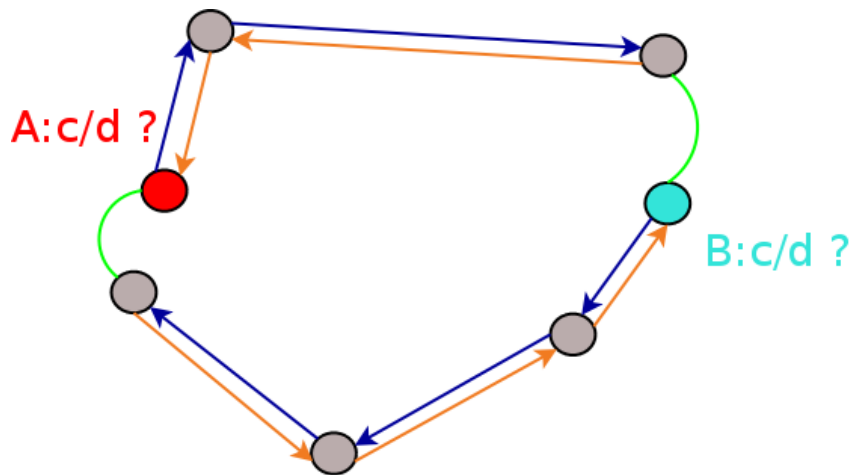


Figure 1.1: Two agents that are about to play and need to decide if to cooperate (c) or defect (d). Both manage to find a chain of trust (blue) to someone who had a previous interaction with their opponent (green). Then, the information is passed back along the chain (orange).

1.1 Objectives

- To propose a reputation based system in small world networks playing the non-iterated Prisoner's Dilemma
- To make the information exchange in the framework scale well (with respect to the size of the network).
- To test the performance of a strategy that takes advantage of this information.

1.2 Research Questions

- What does have our work in common with the previous research ?
- What are possible (dis)advantages of such a system compared to other approaches to non-iterated Prisoner's Dilemma ?
- How to obtain and deal with information about an agent through the chain ?

- How does such an informed strategy perform in a society against standard strategies ?

This document is organised as follows. Chapter 2 discusses previous research and places this approach within. Chapter 3 specifies the model of our experiments. Chapter 4 explains how we implemented this model. In chapter 5 the experiments we conducted are explained in detail and the results are discussed. We conclude in chapter 6.

2 Literature

Research in cooperation of rational individuals roots in Game Theory which was developed by Von Neumann and Morgenstern ([von Neumann and Morgenstern (1944)]). The Prisoner's dilemma is a classic Game Theory setup in which cooperation is usually studied. The history of computational cooperation research goes back to the early 1980s [Axelrod (1984)]. Axelrod held tournaments of competing strategies, from which the simple Tit-For-Tat strategy emerged as the most successful. This strategy is 'nice' in general, but will retaliate defection immediately.

In the 1990s, this research intensified as the Internet connected more computers to each other and trade networks like Ebay or Peer-To-Peer systems emerged. To prevent all-defecting strategies from winning mixed-population tournaments, the notion of trust became an important issue (e.g. [Mui et al. (2002)], [O'Riordan (2001)], [Abdul-Rahman and Hailes (2000)]). In addition, evolutionary methods have been very widely used. Oliphant([Oliphant (1998)]) showed how important the spatial organisation (e.g. the neighbouring structure) is for the outcome of such tournaments.

Lately, spatial organisation is mostly described in terms of network topology. Since the end of the 1990s, a simple kind of networks, Small-World Networks, described by Watts and Strogatz ([Watts and Strogatz (1998)]), have been very popular, also amongst cooperation researchers, e.g. [Abramson and Kuperman (2001)].

A very recent trend is to encode trust in the network (e.g. [Ellis and Yao (2007)]). For example, in Ellis' and Yao's approach, players ask neighbours of the opponent to verify his trustworthiness. They can contact these neighbours through a central authority. These approaches are effective regarding the performance, however they have some unpleasant characteristics (e.g. they require a central authority, they are potentially very vulnerable to cartels of defectors that issue each other high trustability values).

Since we have not come across an approach that utilises a potential chain connection between the player and its opponent, we decided to investigate how to obtain and utilise trust information along a chain of recent interactions. Our goal is to design a fully distributed framework that would have a potential to cope with lying agents and frauding cartels.

3 Model

Since we are working on a design study in artificial societies, it is sometimes hard whether to categorise certain pieces of information into the *Model* section or into the *Implementation* section. In this chapter we present a rather general overview of our framework. Details, together with the design alternatives, are discussed in the (following) chapter 4.

3.1 Ingredients

World In our model, the world consists of other agents and the network connecting them.

Individuals Each agent has several internal states he maintains. These are:

- The trust table realising connections to other agents. Agents can only directly contact agents that are listed in their trust table. This is crucial for agents when building an information chain to gain information on their opponent. The size of the trust table is limited.
- Total payoff the agent earned.
- Number of PD games the agent has played.

Note : We named our strategy *nuts* (Nicholas Und Tomas Strategy) and its less smart version (which doesn't make use information chains) *nuts_dummy*

3.2 Recipe

This section describes the ingredients of our agent model as proposed by [Schut (2007)]. The characteristics of our recipe are: BASIC + diversity + internal states + (non-determinism)

1. action set
 - cooperation/defection
2. observation set
 - action(cooperation/defection) of opponent
 - recommendation on opponent

3. action \rightarrow observation functions
 - cooperation/defection leads to result
4. costs for individuals
 - according to standard Prisoner's Dilemma payoff matrix
5. benefits for individuals
 - according to standard Prisoner's Dilemma payoff matrix
6. internal model set
 - trust in neighbours of social network (noted as number between 0 and 1, where 1 means full trust)
7. observation \rightarrow internal model
 - function for updating the trust in neighbours (end-weighted, s-shaped):
 - if opponent cooperates:

$$new_trust = old_trust + \frac{(1 - old_trust)}{2} \quad (3.1)$$

else (opponent defects):

$$new_trust = old_trust - \frac{(old_trust)}{2} \quad (3.2)$$

If there is no previous *old_trust* value for the given opponent in the table then the default value *old_trust* = 0.5 is used.

Figure 3.1 shows a graphical representation of the function we used for updating the trust values. Figure 3.2 shows values of trust starting with 0.1 and continuing by 10 successive cooperations.

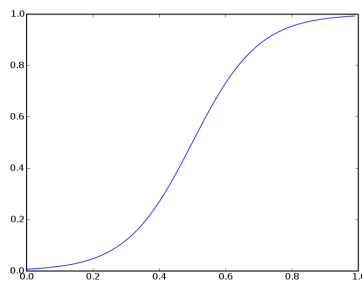


Figure 3.1: sigmoid function between 0 and 1

8. (internal) action \rightarrow observation functions

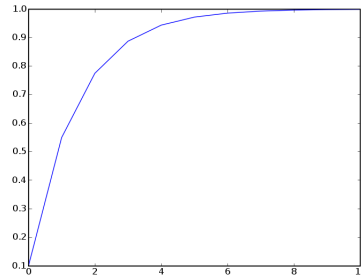


Figure 3.2: example: initial trust of 0.1 - followed by 10 cooperations

get information: who to ask (information retrieval)

- pick an agent from table which maximises following formula:

$$\alpha \cdot distance_from_target + \beta \cdot trust$$

where *distance_from_target* is derived according to the agent's position (here: the agent's id) and the trust of the agent is taken from local connections (stored in the trust table)

- What is the criteria to stop searching? The search is stopped after a specific number of steps (variable *s* - constant or logarithmic w.r.t. the size of the network).

use information (information interpretation): We obtain two pieces of information: information on the opponent itself and the (accumulated) reliability of the path (chain).

How to decide which reaction fits the obtained information ? This is different for different types of agents. Some will not use any information (e.g. *always_defect* strategy) some will (e.g. our *nuts* strategy). We will experiment with the following setup :

- Combine trust along the chain with information on the opponent provided by it and act according to this information. e.g.

if *chain_quality* < *chain_threshold* then
No path found

if *opponent_trust* > *opp_threshold* then
cooperate
 else *defect*

- This is not a trivial issue and it raises some important questions:
 - a) Should the path quality influence the formula? E.g., instead of the second formula, use:


```
if opponent_trust * chain_quality > opp_threshold then
  cooperate
else defect
```
 - b) What to do if the information is not available or it is too uncertain (i.e. the trust along the chain is too low)?

Naturally, these issues are strategy dependent and they will be discussed in more detail for our proposed strategy in Chapter 4.

3.3 Network

We would like to evolve the underlying network into a *Small-World Network* (see Figure 3.3 for a simplified example of a Small World Network). Such a structure can be characterised by relatively low average path lengths L (comparable to L of a random graph) and a relatively high clustering coefficient C (much higher than C of a random graph). Advantages of such a structure for our framework are obvious: An agent should be able to build a short information chain connecting him to the desired opponent. Every agent in the chain (except of the first) must be listed in the trust table of the previous agent. Low average path length is a first prerequisite for the possibility and feasibility of building such a chain. Since every agent in the chain chooses only one agent as a next link in the chain, it is also necessary that agents have a mean of stating which agent would be appropriate for "getting closer" to the desired opponent. This mean is in our case a defined metric over the network . A high clustering coefficient together with emphasis on locality in the agents' trust tables creates a reasonably good environment for this system to work. The closer the information chain gets to the opponent, the more probable it becomes that the opponent will be directly present in the trust table of the last agent of the chain - this means success of the information chain building process.

There are several known approaches how to "evolve" small-world networks. We experimented with two of them, e.g. the *Watts-Strogatz model* [Watts and Strogatz (1998)] and the *Kleinberg model* [Kleinberg (2006)]. More details on this are in Chapter 4.

3.4 Model Categorization

Our framework obviously doesn't fit any of specific well known models (e.g. Cellular automata, Boolean networks, Neural Networks). It is easiest to categorise it as a multi-agent system, incorporating concepts from Game and Decision Theory. Our system is dynamic from the perspective of the agents (they update their trust tables) as well as from the perspective of the world (evolutionary replacement of some weak agent is performed after a predefined period of time). Agents, depending on their strategy,

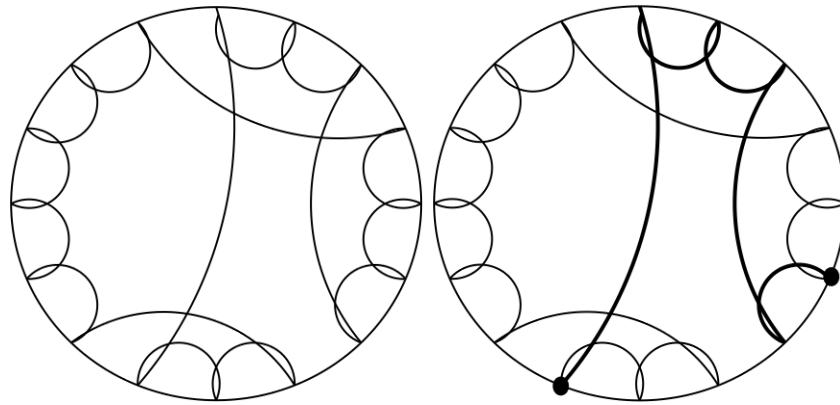


Figure 3.3: A Small World Network and search in it

might express reactive or even adaptive behaviour with respect to the network information. In our setup we experiment only with reactive strategies, having different degrees of complexity. Our model is reflecting a decentralised framework. However, to perform modelling, we naturally need a central authority for performing actions such as choosing the agents for play, etc.

3.5 Relation with the real-world

There are many real-world examples of trust-based social networks based on the repeated Non-Iterated Prisoner's dilemma, for example peer-to-peer networks, social networks behind online auctionary systems, ... On the other hand, in some of these examples the notion of trust might be implicit or might not be present at all. We decided to select one real-world-like system and try to form our model in such a way that it on a certain level reflects this system and could hypothetically be used for implementing it and providing certain benefits.

We made our model resemble an online auctionary system where users announce what they want to sell and afterwards are contacted by other users willing to buy this. However, these systems might be very complicated to resemble for various reasons (actions are continuous in time, users are more or less anonymous, etc.) Moreover, the resemblance to real-world systems is not the main topic of our work. Therefore, we decided to rather work with an hypothetical simplified structure which we call "Idealised Auctionary Network" aka IAN. It has the following characteristics:

- Every user has a unique identification
- Every user has interest in some kind of goods (e.g. comic books, old watches, etc.) and he is more likely to have/buy stuff he is interested in.
- Time is discrete and consists of timesteps.

- In every timestep, every user (A) of the network contacts at most one other user (B) and sends him a request for buying some item. If user B accepts the offer of agent A, then simultaneously user A sends user B money and user B sends user A the desired item.
- Users might fraud. Either by not sending the money for the goods or not sending the goods when expecting the money.
- Every user might ask his "contacts" about trustworthiness of the person he is going to trade with (i.e. "Will he send me the money if I send him the goods" and vice versa).

The analogies with our model are clear.

- The actions of users represent actions in the Non-Iterated Prisoner's Dilemma. Sending the money for goods is cooperation, whilst not sending money is defection. Similarly sending goods is cooperation, whilst not sending anything is defection.
- Users tend to seek other users with similar interest (which in our model is represented by a simple metric).
- Users keep information on some particular agents they have interacted with in the past (the trust table in our model).
- Users try to get information on a person they are about to trade with (the information chains in our model).

Our model introduces one more phenomenon which we haven't included in the specification of IAN. This is evolutionary replacement of weak agents by stronger ones (for further details on this see Chapter 4). However, it is not impossible to imagine that in IAN there is only a limited number of users that can participate and also there are users waiting to enter the system when some of the users in the system are out of money. This innovation could be understood as an analogy of the implemented evolutionary replacement.

We will be referencing described system simply as IAN in the rest of the report.

4 Implementation

4.1 Control loop

Since we are dealing with a simulation in the social network, the very heart of the implementation is the control loop of this simulation. We will briefly present it and then describe particular steps in further detail.

1. initialise N individuals
2. initialise world
3. for e in 1 .. total_epochs
4. for i in 1 .. N
5. select opponent to play with agent i
6. agents observations
7. agents action
8. agents internal model update
9. update the world

A further description of particular parts follows.

1. initialise individuals

Initialisation of agents consists of: assigning each agent an unique id (0 .. N-1), emptying his trust table, nullifying the variables representing the total agent's payoff and number of games the agent has played ("agent's life"). The size of the trust table is $c \cdot \log N \cdot \sqrt{\log N}$ where c is an independent variable.

2. initialise world

Initialisation of the world is basically assigning strategies to the agents according to a given scenario (a scenario describes percentage ratios for every possible strategy). Agents for every strategy are chosen uniformly randomly. In this way, there is no bias in distribution of strategies in the network. We assume that this resembles the "real world" situation well.

3. for e in 1 .. total_epochs

The whole simulation consists of epochs. Every epoch consists of a predefined number of rounds (see next item). After all rounds are finished, the world is updated and the simulation continues with the next epoch.

4. for i in 0 .. N-1

One round represents a game between two agents. First, two agents are chosen to play the game. They both are given their opponents id and are allowed to find out information about their opponent. After this, they play a single game of Non-Iterated Prisoner's dilemma game and receive the proper evaluation. Finally, both their internal models are updated and the epoch continues with the next round.

5. select two agents

Even though a little bit unsuspecting, this is one of the most crucial points in the whole design. The selection mechanism plays an important role in shaping the underlying structure which we desire to evolve into a small world network.

We list several rather general items an agents selection mechanism should fulfil. These items are stated (also) based on how we suppose IAN works, therefore we provide this motivation for every item.

- *It should be fair.*

With respect to the number of games played per agent and also (but with less emphasis) the possibility to contact anyone in the network.

Motivation: IAN is basically fair. All users are "equal" - every user can request a trade from every other user at any time.

- *It should not be deterministic.*

Motivation: IAN is in its nature a stochastic system - it is too complicated to determine which user will contact whom at a certain timestep.

- *It should resemble the metric.*

In the sense that agent A has a higher probability to play with agent B than with agent C if the distance $d(A, B)$ is lower than $d(A, C)$.

Motivation: this is a phenomenon in IAN if we consider the metric to be "interest in goods" like we mentioned when we described IAN. For instance, people interested in historical goods are more likely to contact people interested in WWII coins and banknotes than to contact people interested in items related to Star Trek.

We considered two ways of implementing the previous ideas of selecting two agents for a game in a particular round:

- Agent A (first) would be chosen randomly uniformly from the pool of agents. Agent B would be chosen randomly proportionally to the distance from agent A. After agents play the game, they are **both** removed from the pool.

This approach is absolutely fair - each agent plays exactly one game per round. Maybe it is too fair - in IAN quite often different nodes receive different number of trade requests in a given timestep (even though every user can contact only one user in each timestep). However, this significantly hurts the selection of agent B according to the metric. It would quite often happen that all agents close to agent A have already been excluded from the pool in the actual epoch. And therefore agent A's opponent would be quite far away from agent A even though it might have been possible that agent A should have played with someone close to him.

Motivation: In IAN it doesn't work like this. If some user requests the trade with another user U it is still possible for other users to request the trade with this user U as well in the same timestep (we are not taking into account technical details related to internet connection).

- Another idea is to proceed similar as the previous way, except from the fact that after the game, no agent is removed from the pool, but it is guaranteed that agent A cannot be selected as a first player again in the actual epoch. In this approach, it is not anymore important that the first agent is selected uniformly randomly. Therefore, the implementation can be straightforward: In round i agent no. i is assigned an opponent according to the metric from all other agents and plays the game.

This approach is still fair - every agent plays at least one game in every round even though some users might play more games. However, averaged over time these differences should be erased. The selection of agent B, based on the metric system now works smoothly and it resembles the IAN system quite well. Except from the fact that every agent has to play a game in every timestep (round), which is a tolerable simplification.

After taking the mentioned arguments into account, we decided to prefer this selection mechanism over the previous one.

The last issue left to fully specify the selection mechanism is to decide how agent B is selected when agent A is already known. The purpose of this part is to shape the underlying small world network. Again, we experimented with two approaches (the given descriptions of models are heavily simplified and serve only for basic orientation).

- **Watts-Strogatz model** [Watts and Strogatz (1998)]. Selects opponent uniformly with a small probability p , otherwise selects opponent "close" to the agent. The optimal value for p is believed to be approximately $p = 0.001$.
- **Kleinberg model** [Kleinberg (2006)]. Selects opponent with a probability proportional to the opponent's distance from the agent. More precisely, the probability of selecting an opponent w for the first agent v is proportional to $1/d(v, w)^\alpha$, parametrised by α . The optimal value of α to guarantee that the network will evolve in small-world network is supposed to be $\alpha = 2$.

Both of these are parametrised by one variable and are supposed to produce a small-world network for "good" values of this variable. We chose the Kleinberg model for the following reasons:

- We believe it resembles our IAN system better than the Watts-Strogatz model. If we would "translate" Kleinberg's approach to the IAN language it says: "The more similar an opponent's interest is to the one of the first user, the more probable is that he will be contacted for trade with this user." Which is exactly how we described the IAN system.
- In the Watts-Strogatz model, there is still an open question how to choose an opponent "close" to the agent. This is naturally problem-dependent and in our case, probably some proportional approach would be used. Which is actually what Kleinberg does in all cases, not only when the opponent is not chosen uniformly. In this sense, Kleinberg's model seems to us more consistent - it precisely defines how to select the opponent.

However, since we experimented with a relatively small number of agents in the network, we believe that both models would perform more or less equally.

There remains one pitfall which we realised after running some preliminary experiments (these are not listed in the results section - their purpose was to show us how the system behaves and what are the main problems). If this metric-based selection mechanism (in our case Kleinberg's model) is used with "optimal" settings, then even for a relatively small sizes of agent's trust tables, it is highly probable that agents already had some previous interactions with the opponent he was assigned. As a consequence, agents find most of their information already in their trust table and are not forced to create an information chain. In the end, strategies using information from the framework (like *nuts* or even *nuts_dummy*) performed extremely well against *always_defect*-like strategies. However, we think that such a setup is questionable from the perspective of similarity to the real-world situation since it is oversimplifying. There should be pressure on locality but not as strong that agents play most of the time with the same agents from their neighbourhood.

We solved this situation in a following manner. For every agent, the second agent to play with him is chosen according to the previously used formula $1/d(v,w)^\alpha$ parametrised by α . However, now α is used to produce a more random distribution (less biased to the distance from the first agent - values around 1.0 are used). In this way, pressure on locality is not as strong and the agents themselves must ensure that they will be able to provide information on agents in their neighbourhood. They achieve this by updating their trust tables in such a way that they keep agents closer to them with higher probability than agents further away.

6. agents observations

Every agent is given his opponents id and is allowed to look up information on this

opponent through the information chains. In our setup, this opportunity is taken only by the agents representing *nuts* and *nuts_dummy* strategies.

Moreover, *nuts_dummy* agents only look into the trust table - if the opponent is in there, then it cooperates if the opponent's trust is above a certain threshold, otherwise it defects. If the opponent is not present in the table, the default action is to cooperate.

Agents of *nuts* strategy perform information retrieval from the network in the following way: Every agent (A) can be asked by some other agent (B) about the trustability of a particular opponent (C). Together with this "question", a number representing the remaining number of questions is passed (which sets a limit on how long agents will try to find someone who knows the opponent - in our experiments we fixed this number at value 6). If that number is zero or the trust table of agent A is empty, then the information chain failed and the message that the opponent was not found is returned to agent B who asked. Otherwise, agent A returns his information on trust in a message to the sender B. Such a message consists of two parts:

- a) trust in the opponent C - which is copied from the table
- b) The trust of A in the chain - which is 1.0 because agent A took information directly from his own table

If agent A doesn't have information on agent C in his trust table, then he will ask another agent (D) from his table about it. From the trust table of A, agent D is selected by maximising the formula

$$dist_bias * distance(D, C) + trust_bias * trust_of_agent_A_in_agent_D$$

This formula balances a tradeoff between effect and safety. Clearly, the higher the distance bias, the more probable it is that the opponent will be reached soon. However, the information chain might include potentially unreliable agents (having a low trust bias). On the other hand, high trust bias might produce more reliable chains but it is also more likely to fail in finding the opponent at all. After agent D is selected, agent B asks agent D about information on agent C with the number of remaining questions decreased by one (shortening the remaining number of agents that can participate in the chain). Obviously, this is a recursive process. Therefore agent D will respond:

- a) By stating that the opponent C was not found. In this case, this message is simply forwarded from A back to B who asked in the first place.
- b) By providing a message with two pieces of information as described above. In this case, the part of the message describing the trust in the opponent is unchanged. However, the part regarding the trust in the chain is lowered by multiplying it by the number expressing trust of agent A in agent D. After this the message is forwarded from A back to B who asked in the first place.

This whole machinery starts by an agent (E) who is searching for information on his opponent. This agent simply asks himself in the way we described above. After the recursion yields a result, agent E receives information from the chain.

- a) If the result is that the opponent was not found or the trust in the chain is lower than *chain_tt* (chain trust threshold), then the default action - to cooperate - is performed. This innocent-looking statement strongly influences the behaviour and performance of *nuts* strategy. More on this in subsection 5.2.2.
- b) If the result is that the opponent was found and the trust in the chain is higher than *chain_tt* variable then the action to perform is to cooperate if the trust in the opponent (first part of the return message) is higher than *opp_tt* (opponent trust threshold). Otherwise, the action is to defect.

Obviously, the model of such a chain to retrieve information strongly abstracts over reality. Yet, it provided us with interesting experimental results.

7. agents action + determination of payoffs

Each of two playing agents announces its action. This is then evaluated and both agents are assigned their payoffs according to the standard Prisoner's Dilemma payoff matrix (for settings, see the experiment chapter).

8. agents internal model update

The agent's total payoff is updated as well as the number of games the agent has played. If the agent's opponent is already in his trust table, then his trust is updated according to the formula 3.1 or formula 3.2 (depending on opponent's action). If the agent is not in the table, the trust is counted according to same formula with value of *old_trust* set to 0.5. If the table is not full, then the new opponent is placed into the table immediately. If the table is full, then the new neighbour will be put in the table with a probability proportional to $1/d(v, w)^{\alpha_{loc}}$. If it is decided that new item will be put in the full table, then an arbitrary item from the table is removed to make room.

The parameter α_{loc} is set to strongly support locality, thus having values around $\alpha_{loc} = 2.0$ or even higher. The effect of this is that even though agents are playing against agents from the pool selected with lower bias to locality, in their tables, agents are preferring to keep information on agents from their surrounding. This increases the probability that agents will be able to provide information on agents close to them. This is a desirable property to forming information chains to gain information about the opponent.

9. update the world

Updating the world consists of performing the "evolutionary replacement". Two agents are selected uniformly and randomly from all the agents. Their average payoff per round is calculated - the number of games they played and total payoff are known. If they are from different strategies and they have different average

payoff then the one with smaller average payoff is "killed" (i.e. information on this agent are removed from trust tables of all the agents) and a new agent with the same strategy as the strategy of a winner takes his place. This new agent thus has the same id as the "killed" agent but a different strategy and it starts with initialised values of the internal variables (i.e. empty trust table, zero total payoff and zero number of games played).

This replacement is introduced because it provides a good measure of strategy successfulness - a strategy that has more members after non-trivial number of epochs is more likely to be successful in a given environment than a strategy with less members. We observed that this is a standard way how to measure success of different strategies in the literature ([Ellis and Yao (2007)],[Oliphant (1998)]). Also, it sounds natural from an evolutionary perspective. Bad performing agents (probably belonging to a strategy not fitting into the environment) are replaced by well performing agents. This process is ongoing, since changing the distribution of strategies might over time result in changing successfulness of these strategies.

When implemented like this, it is not a distributed solution. A possible distributed implementation would not erase the information on a "killed" agent from all the tables and a new agent would be given a new unique id. Then, the only pitfall would be that information on dead agents would stay in the trust tables of other agents. However this could be also simply coped with by some "ageing" mechanism. We decided to keep things simple with the centralised implementation of evolutionary replacement.

4.2 Implementation details

We implemented the whole project in Python. No specialised libraries have been used. A short description of source files for basic orientation follows:

- **main.py**

This is the heart of the whole program. The classes `Agent` and `World` are implemented here. Inside the `World` class, there is also a logging engine which produces `.csv` output files. If run as a python script, it takes the configuration file for the simulation, performs the simulation and generates as many `.csv` files as there were runs (stated in the configuration file).

- **config**

An example configuration file. For batch testing, we used the script `setup/setup.py` to generate configuration files for all different setups of variables (configurations).

- **strategies.py**

This file contains the definition of the generic class `Strategy`. This is an ancestor

class to all classes representing particular strategies: e.g. the classes `Always_defect`, `Always_cooperate`, `Nuts` and `Nuts_dummy`. These are defined in this file as well.

- **utils.py**

Supporting functions for simulation are defined here. For instance, functions for computing the average path length and clustering coefficient of the given network.

- **setup/setup.py**

This script is used to generate different configuration files for given values of variables. It also generates the `main.conf` file, which is used as a configuration file for batch runs performed by `my_run.py`.

- **my_run.py**

This script is used to run simulations for multiple configurations in a batch mode. It uses a meta-configuration file which contains links to configuration files for desired setups of simulations.

- **my_run.conf**

An example meta-configuration file for `my_run.py`.

- **my_plot_aso.py**

This script is used for the analysis of performed simulations. It can work in single (analysis of 1 configuration) or batch mode. For every configuration, it harvests data from proper `.csv` files, performs plotting via **gnuplot**, saves images and generates a little report `.pdf` file. Which information are plotted, is determined by a configuration file.

- **my_plot_aso.conf**

An example configuration file for `my_plot_aso.py`

- **avg_stats.py**

A script containing functions to perform statistical operations for `my_plot_aso.py`.

- **rgraph/rgraph.py**

A script used to count network characteristics (average path length, clustering coefficient, ratio of paths not found) for random graphs. Calculated data were used for comparison with network characteristics of evolved small-world network (see experiment section).

- **analyze/analyze.py**

A script harvesting data from all simulations and creating an HTML mockup.

5 Experiments

5.1 Experimental Design

We used **simple comparative** experimental design. The simulations were run 20 times for each experimental setting. These were the settings of independent and dependent variables:

5.1.1 Independent Variables

variable	comment	value
N	number of individuals	[150, 300]
K	number of neighbours	[$0.7 \log N * \sqrt{\log N}$, $1.1 * \log N * \sqrt{\log N}$]
α	small world network parameter	[0.9, 1.2]
α_{loc}	local small world network parameter	[1.9, 2.4]
scenario	initial distribution of strategies	[(AD, nuts) (AC, AD, nuts) (AD, nuts, nuts_dummy)]
distance_bias	bias to distance when asking (bias to trust is complement)	[0.3, 0.7]

AC Always Cooperate

AD Always Defect

nuts Proposed Strategy

nuts_dummy... Proposed Strategy with no asking (just look in the neighbour table)

Note : The runtime was $N * 4$.

5.1.2 Dependent Variables

variable	comment
L	average path length
NP	number of not connected pairs of agents
C	centrality/clustering index of network
SN	number of agents of particular strategies
SP	average payoff of the strategy

5.2 Setup

This section lists other Variables that are of interest to the experimental setup.

5.2.1 Fixed Variables

variable	comment	value
s	limit of steps in asking	6
chain_tt	expresses uncertainty	0.3
opp_tt	opponent trust threshold	0.5
E	number of epochs	4 * N
evo_step	how often the network gets evolutionary updated	3
trust_bias	bias to trust when asking	1 - distance_bias
tu	steepness of trust update function	2

5.2.1.1 Payoff matrix

shortcut	comment	player 1	player 2	value for player 1
t	trait benefit	D	C	5
r	reward	C	C	3
p	punishment	D	D	1
s	sucker payoff	C	D	0

This payoff matrix is also a fixed variable. It is a standard payoff matrix for a game of Prisoner's dilemma (taken from [Axelrod (1984)]).

5.2.2 Implicit Variables (Principles)

Besides the previously listed numerical variables with exact values, there are some more factors in our program that influence the resulting performance of strategies. These are not variables in the pure sense of the word, they are rather hard wired algorithms or principles we decided to use for our world.

nuts reasoning under uncertainty

This determines the way how the *nuts* strategy agents act when they don't have enough information from the information chain (i.e. they were unable to get any information on their opponent). This is an important part of inner *nuts* intelligence which is influencing the overall behaviour of the strategy. For the sake of simplicity and also in the sense of the paradigm "Be nice and forgiving" [Axelrod (1984)] we decided to set the implicit action to *cooperate*. This means that if an agent is unable to establish a reliable information chain, it will cooperate with its opponent. The idea is that if confronted with a betraying strategy (like *AD*) *nuts* agents will cooperate with *AD* agents from the beginning. However, after they learn they are being betrayed by certain individuals and they are able

to distribute this information over chains, they start betraying *AD* agents back. We can clearly see this behaviour on many graphs from *AD-nuts* scenarios (see section 5.3).

But there are more pitfalls when other strategies are included in the scenario. Under such circumstances this approach might strongly favour "betraying" strategies (like *AD*) against agents with the *nuts* strategy. This happens in the presence of over-optimistic strategies like *AC* or *nuts_dummy* that cooperate very often and don't learn from their mistakes as effectively as for instance *nuts* strategy. The effect is that *nuts*-agents are trying to be "friendly" and basically cooperate with these "dummy" strategies, but *AD*-agents just betray them all the time and thus gain a lot of points. *AD*-agents are "feeding" on members of these "dummy" strategies. In this case, the behaviour of the *nuts* strategy can be viewed as kind of *Tit-For-Tat* behaviour. It is important to note that the *Tit-For-Tat* strategy would show similar (poor) behaviour as described above in an analogous Iterated Prisoner's dilemma setup (with analogies of *AD* and dummy strategies).

We intended to experiment with different answers (e.g. cooperate in $x\%$ of cases and defect in $100-x\%$ cases, etc.). However, regarding the objectives this is not the main part of this report, therefore we decided to postpone this to future work.

updating a full agent's table

A new neighbour will be put in the table with a probability proportional to $1/d(v,w)^{\alpha_{loc}}$. If it is decided that new item will be put in the table, then an arbitrary item from the table is removed to make room. The parameter α_{loc} is set to strongly support locality - thus having values around 2.0. The effect is that - even though agents are playing against agents from the pool selected with lower bias to locality - in their tables, agents are preferring to keep information on agents from their closer surrounding. This way, they are increasing the clustering coefficient.

5.3 Results

Almost all the independent variables significantly influence the result of the particular experiments. Below are given some example cases with explanations. All results and graphs have been placed on our webspace [Höning and Kozelek (2008)].

Note: The independent variables can simply be read out from the titles of the examples. For example,

N150_Ksmall_DB0d70_ALP0d90_ALPLOC2d40_SCE1

depicts the following configuration: population(N) :150, size of the trust table(K): small (which is 13 for this case), distance bias (DB) : 0.7, alpha : 0.9, alpha_loc : 2.40, scenario : 1. It is important to keep in the mind that graphs represent averages over 20 runs for a given configuration. Graphs can be viewed as "simple boxplots" over time. Thus, for each X point and each "colour" interval with a centre at the mean of values, an Y size of 2 * standard deviation is plotted.

N150_Ksmall_DB0d70_ALP0d90_ALPLOC2d40_SCE1

This is a representative example for scenario 1 - an *AD-nuts* duel. In each setup for scenario 1 (averaged over runs), the *nuts* strategy wins. Usually, this victory is a decisive one. It is interesting to observe typical shapes of the curves (explained above). First, *AD* agents are successful, but very soon *nuts* agents learn to fight the *AD* agents and turn the score. It is also useful to note that information paths are not trivial - i.e. information paths of length 2,3 have a significant share of all the paths. Moreover, the communication in the net is improving over time - this is expressed by the falling number of not found information chains. For graphs, see Figure 5.1.

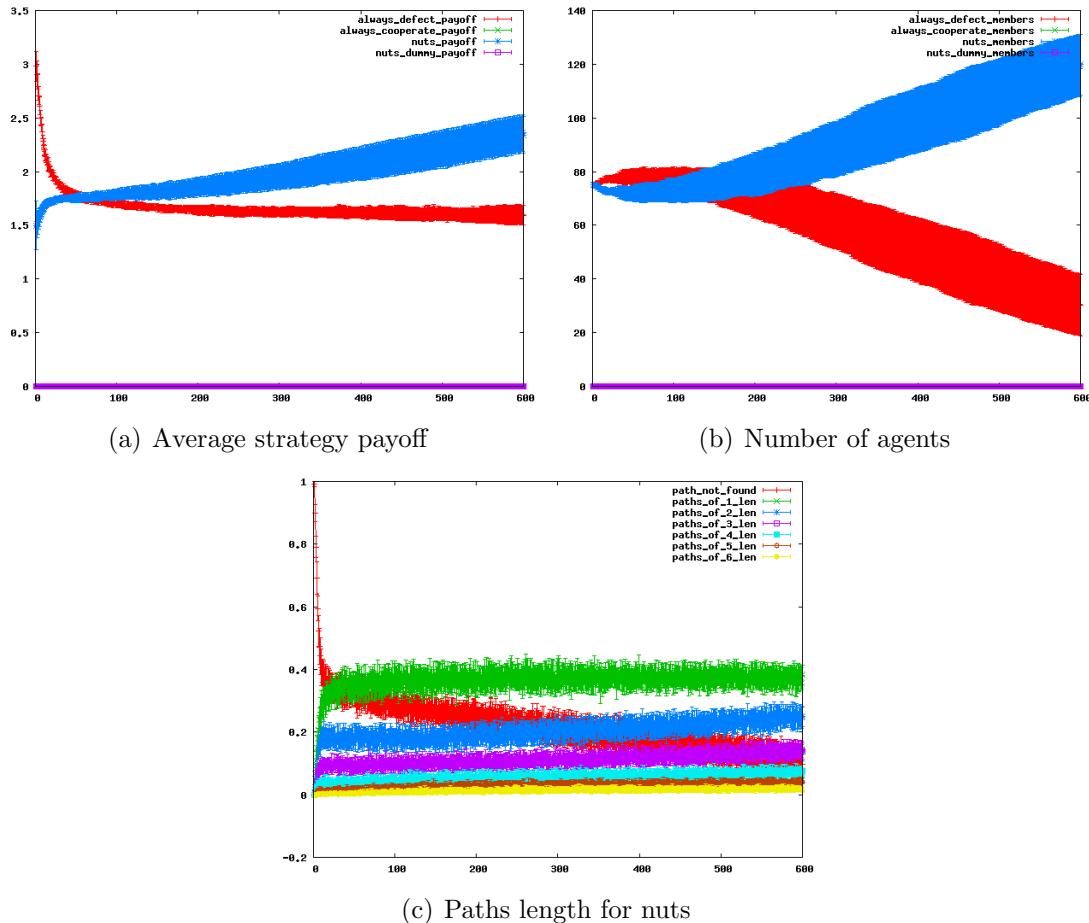


Figure 5.1: N150_Ksmall_DB0d70_ALP0d90_ALPLOC2d40_SCE1 graphs

N300_Ksmall_DB0d70_ALP1d20_ALPLOC2d40_SCE2

In scenario 2, *nuts*-agents are much less successful. This was predicted by our considerations about the exploitation of pure cooperators by pure defectors in section 5.2.2. However, still reasonable results might be achieved when circumstances are advantageous. In this example case, there is strong pressure on locality expressed by $\alpha = 1.2$ and $\alpha_{loc} = 2.4$, which partially overcomes the disadvantage from the presence of strategy *AC* on which *AD* agents simply feed. It might be possible that in more runs *nuts*-agents would catch up or outnumber *AD*-agents. Because the less *AC*-agents in the game, the better for *nuts*-agents. On the other hand, *AD*-agents already accumulated quite some profit along the time and this counts in the decision which agent is going to survive in evolutionary replacement. For graphs, see Figure 5.2.

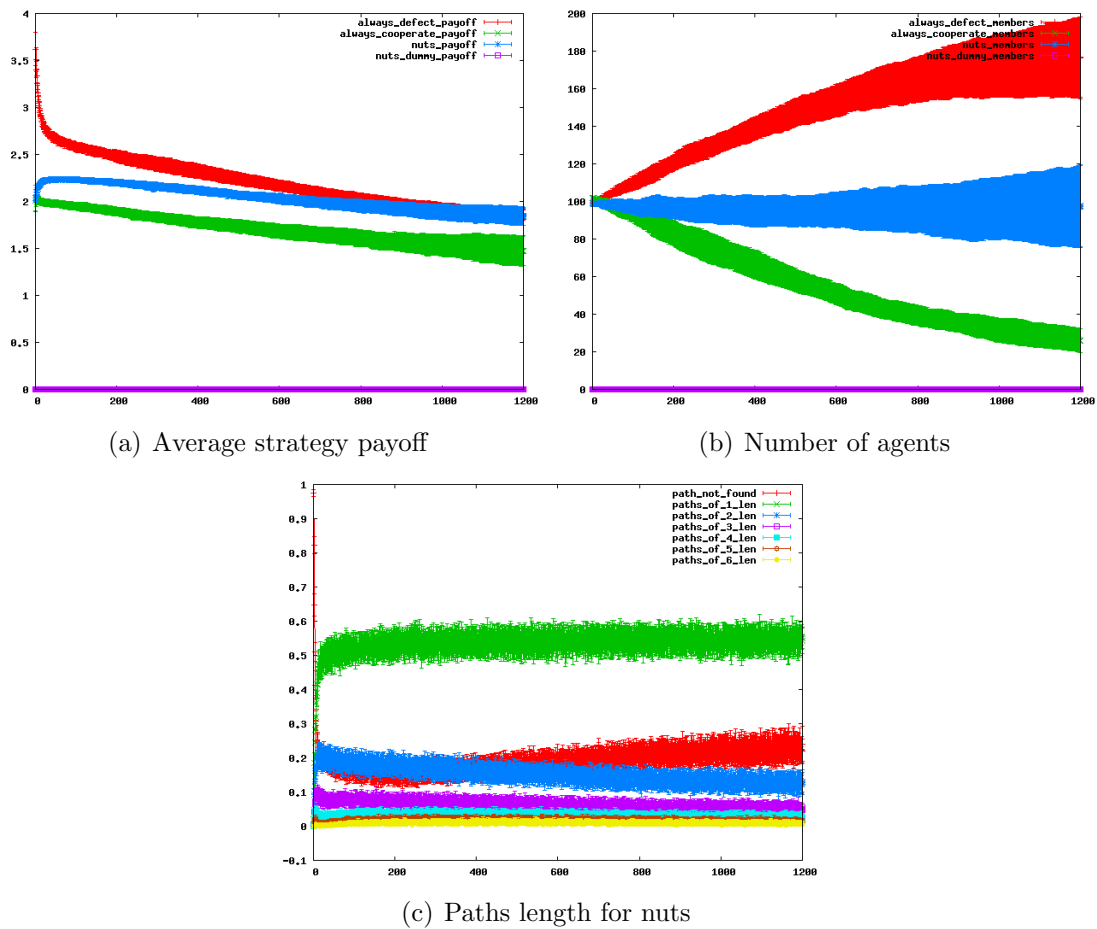


Figure 5.2: N300_Ksmall_DB0d70_ALP1d20_ALPLOC2d40_SCE2

N300_Ksmall_DB0d30_ALP0d90_ALPLOC1d90_SCE2

This setup shows the most difficult combination of parameters in scenario 2 for *nuts*-agents. The main reason for this is the weak pressure on locality. Moreover, the low *distance.bias* parameter make them prefer safety over effect. This decreases the performance (generally, less agents are found through information chain). For graphs, see Figure 5.3.

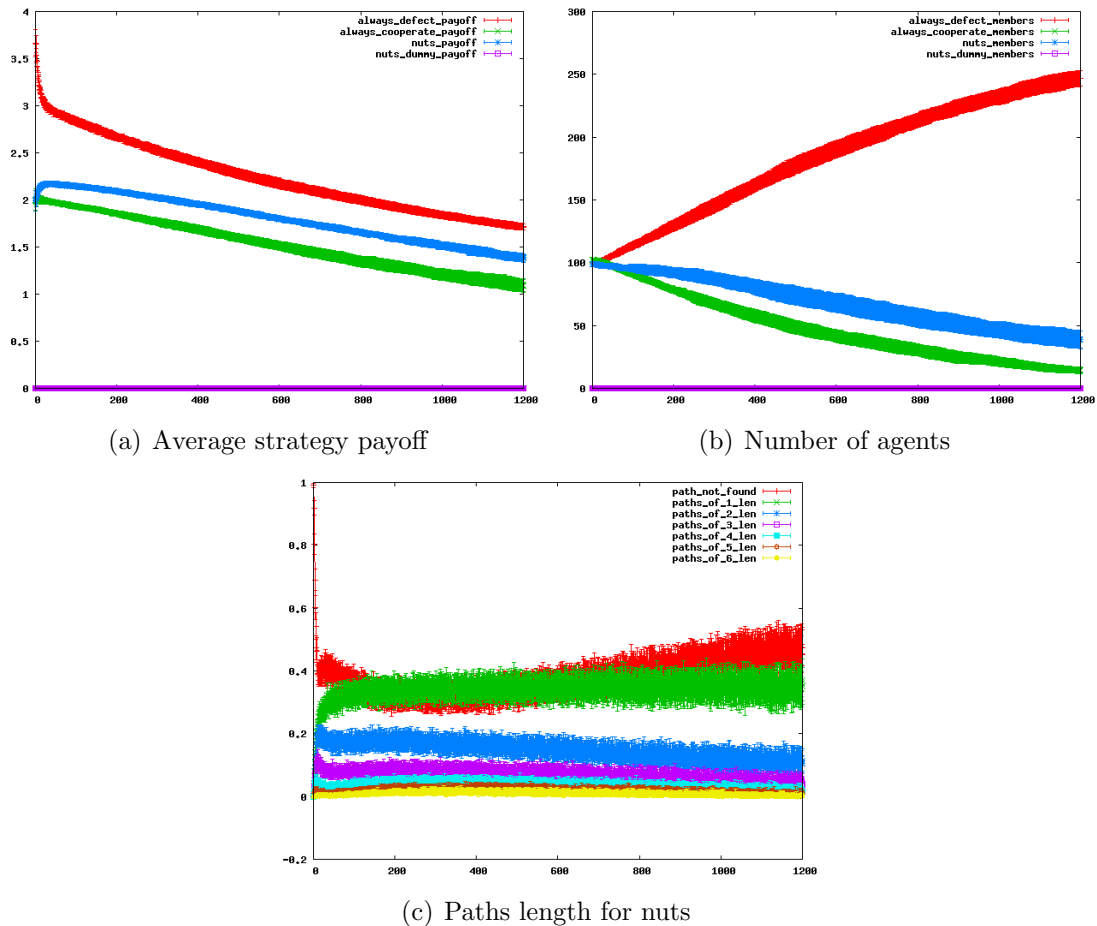


Figure 5.3: N300_Ksmall_DB0d30_ALP0d90_ALPLOC1d90_SCE2

N300_Kbig_DB0d30_ALP0d90_ALPLOC1d90_SCE3

Very strong influence is also generated by parameter K (size of the trust table). This combination presents scenario 3 with all parameters (except from K) set to "worst values" - from the point of view of *nuts*-agents. Still, *nuts*-agents dominate the field. Moreover, in the end *dummy*-agents obtain the upper hand compared to *AD*-agents as well. However, since the calculation of K for any given number of agents is (almost) logarithmic, it is reasonable to presume that these effects would partially fade out with higher number of agents (N) as the neighbourhood becomes smaller with respect to the network size. For graphs, see Figure 5.4.

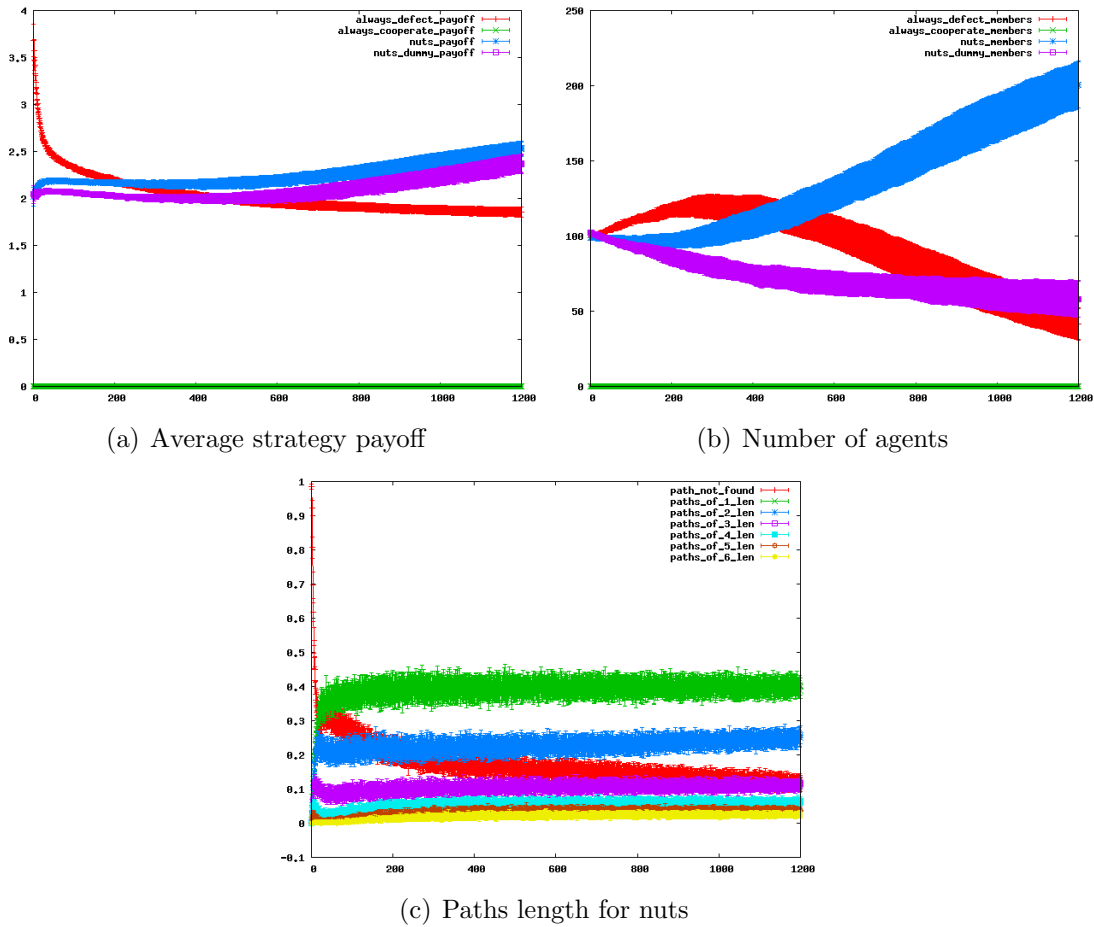


Figure 5.4: N300_Kbig_DB0d30_ALP0d90_ALPLOC1d90_SCE3

N300_Ksmall_DB0d30_ALP0d90_ALPLOC2d40_SCE3

For comparison, the same setup with small K and α_{loc} increased (for small compensation) is given. In the end of the simulation, AD is a clear winner and $nuts-dummy$ is not far from extinction. However, $nuts$ -agents are stabilised and we can presume that if the simulation would continue, they would overtake. On the other hand, the same experiment with $\alpha_{loc} = 1.9$ shows similar results in the end. However, both $nuts$ -agents $nuts-dummy$ -agents are constantly decreasing in number and it is difficult to predict what would happen next. For graphs, see Figure 5.5.

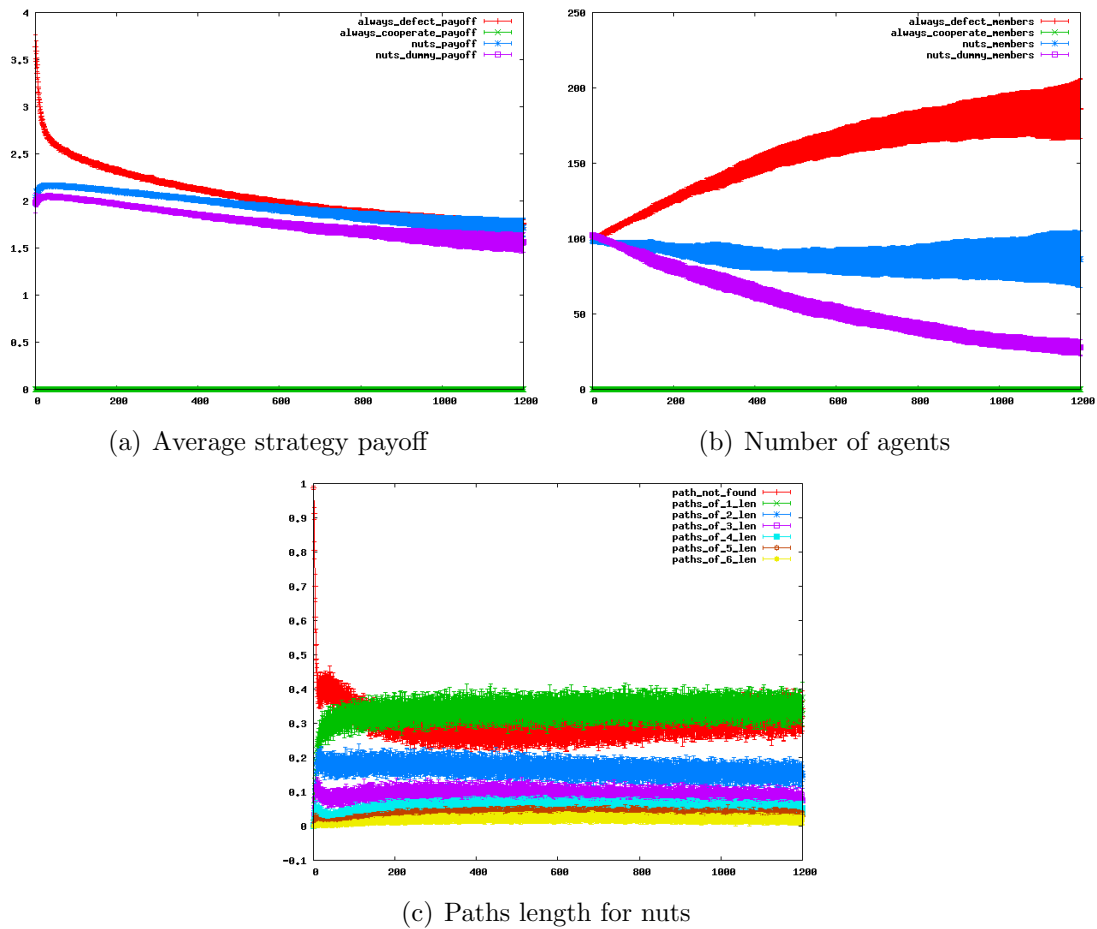


Figure 5.5: N300_Ksmall_DB0d30_ALP0d90_ALPLOC2d40_SCE3

5.4 Analysis

This section reviews our research hypotheses, objectives and states in what manner we have achieved them. Also, we discuss some basic questions that this research should address.

5.4.1 Hypotheses

- (**Hypothesis 0: In general, defectors win over cooperators.**)

Holds. We conducted a Welsh Two-Sample T-Test ($t = 52.5585$, $df = 655.931$, $p - value < 2.2e - 16$) to test on the difference of the population representation of both pure strategies (AC, AD) in the end of runs in Scenario 2 (AC, AD, nuts). This result shows the expected significant differences: The mean population size of *AD* was 142.78435 and the mean population size of *AC* was 17.06390. Figure 5.6 shows the differences over the whole runs.

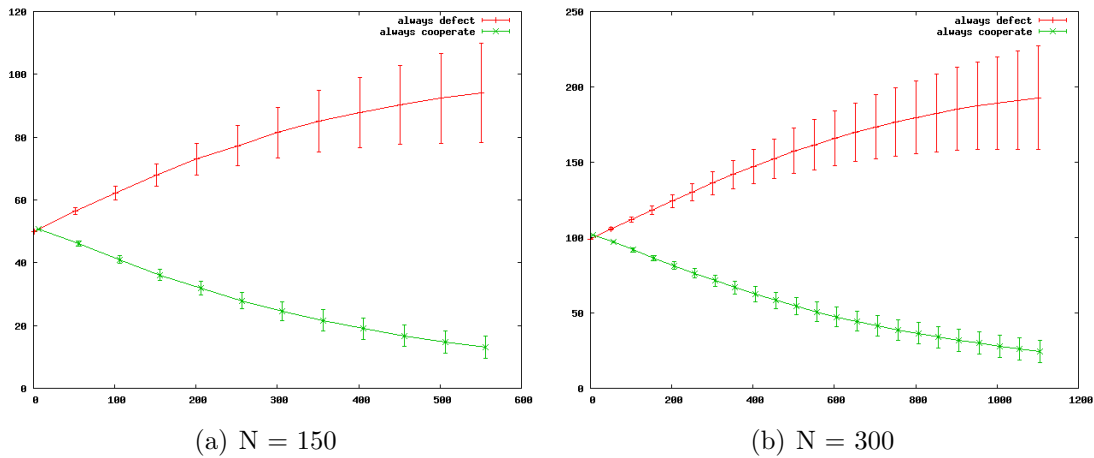


Figure 5.6: *AlwaysDefect* vs *AlwaysCooperate* in Scenario 2

- **Hypothesis 1: The underlying structure will evolve to a small-world network (which guarantees informativeness).**

Holds. In every simulation run, the network parameters achieved acceptable values. Small World Networks have $L \simeq L_{random}$, but $C \gg C_{random}$ see [Watts and Strogatz (1998)] for details. The table below shows the network parameters at the end, averaged over all our simulations.

	L	L_{random}	C	C_{random}
$N = 150, K = 13$	2.467104	2.191248	0.3098036	0.083912
$N = 150, K = 21$	2.031188	1.894476	0.2950395	0.139479
$N = 300, K = 16$	2.628051	2.339287	0.2970147	0.051419
$N = 300, K = 25$	2.207707	2.019537	0.2654894	0.082702

- **Hypothesis 2: Informed strategies will survive in various populations.**

The simple explanation is that this doesn't necessary hold. Simply averaged over all configurations, our strategy achieves to represent 55.8% of the population at the end of the run. That is not a bad value. If we look closer, we see that the success depends on the particular population.

If we only look at scenario 1, *nuts* versus Always Defect, we find that *nuts* wins by far. We conducted a Welsh Two-Sample T-Test ($t = 51.1728, df = 712.562, p - value < 2.2e - 16$) to test on the difference of the population representation of both strategies in the end of runs in Scenario 1. These results show that significant differences appeared here (The mean population size of *nuts* was 183.99818 and the mean population size of *always_defect* was 28.41058). Figure 5.7 shows the differences over the whole runs.

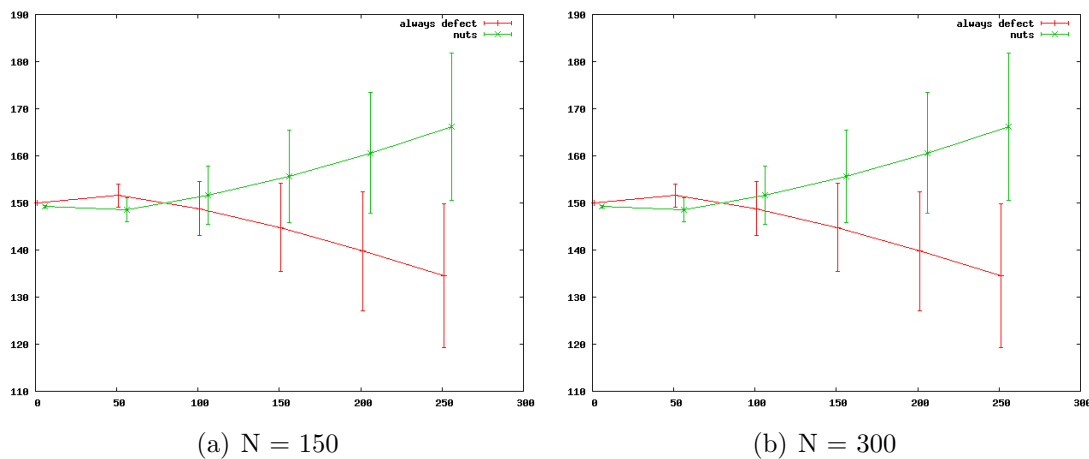
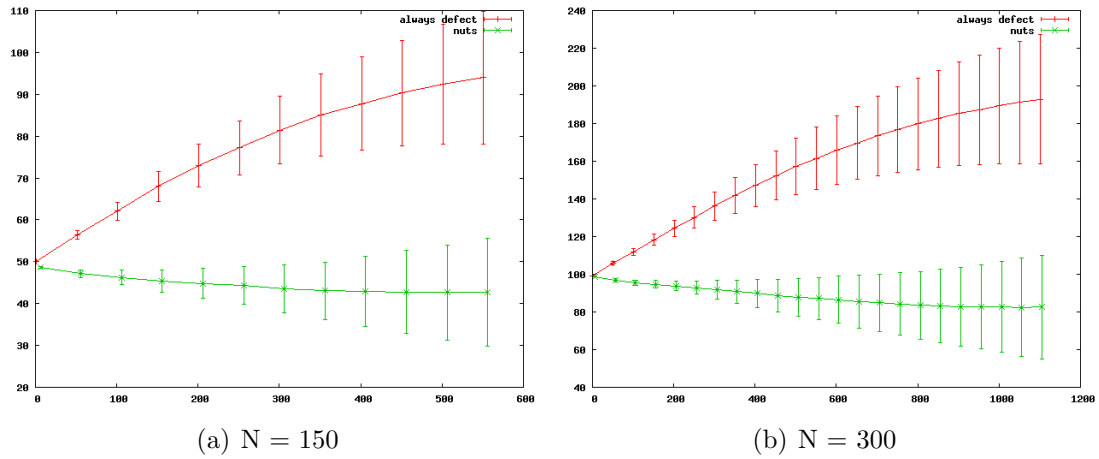


Figure 5.7: *AlwaysDefect* vs *nuts* in Scenario 1

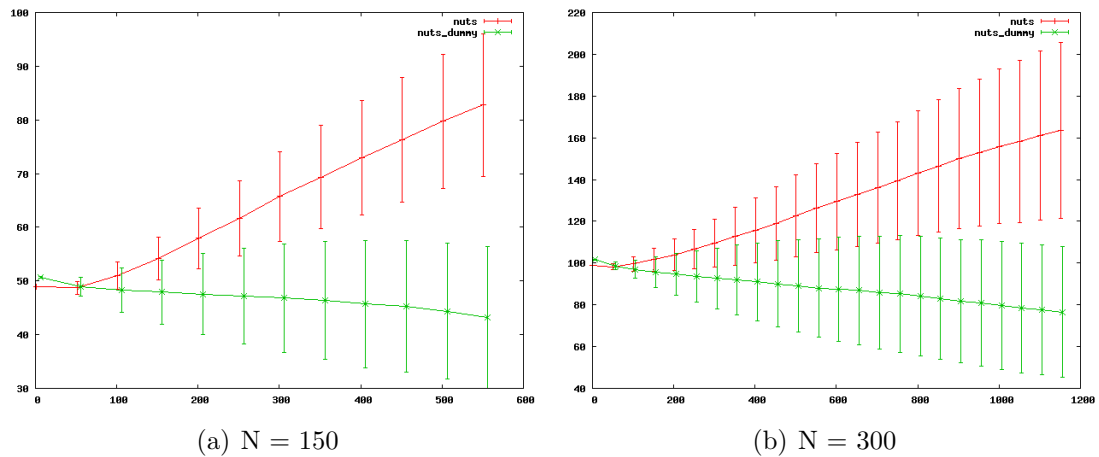
But if we look at these two strategies in the 2nd scenario, where also an always-cooperating strategy was involved, the picture is different. *Nuts* loses to always-defect. We conducted another Welsh Two-Sample T-Test like the above, only regarding the second scenario ($t = 28.7613, df = 1024.961, p - value < 2.2e - 16$). Here, the always-defect strategy will significantly outnumber the *nuts* strategy (the mean population size of *nuts* was 63.47444 and the mean population size of *always_defect* was 142.78435). Figure 5.8 shows the differences over the whole runs.

We conclude that defecting strategies profit much more from the presence of “dumb”, often-cooperating strategies than the *nuts* strategy. In the context of the Prisoners Dilemma, this means that our strategy doesn't exploit cooperators as much as defectors do. Our strategy should recover once pure cooperators die out (and then the setting is the same as in Scenario 1).

Figure 5.8: *AlwaysDefect vs nuts* in Scenario 2

- **Hypothesis 3: Using information chains increases success of informed strategy.**

This hypothesis holds. We conducted a Welsh Two-Sample T-Test ($t = 27.082$, $df = 1032.923$, $p - value < 2.2e - 16$) to test on the difference of the population representation of both *nuts* - strategies in the end of runs in Scenario 3 (*AD*, *nuts*, *nuts_dummy*). The results show with strong significance that the *nuts* strategy that makes use of the information chains will outnumber the “dumb” *nuts_dummy* - strategy. The mean of *nuts* was 125.75625 and the mean of *nuts_dummy* was 58.78437. Figure 5.9 shows the differences over the whole runs.

Figure 5.9: *nuts vs nuts_dummy* in Scenario 3

5.4.2 Objectives

- **To propose a reputation based system on small world networks in a non-iterated Prisoner's Dilemma.**

Achieved. We proposed and implemented a working reputation system based on a small world network.

- **To make the information exchange in framework scale well (with respect to the size of the network).**

Achieved. With this objective in mind, our framework was constructed. Two particular items were taken into account:

1. The information about neighbours that every single agent maintains about must be small ($O(\log N * \sqrt{\log N})$)
2. Retrieving information from network must be fast ($O(\log N)$ or constant)

- **To test the performance of a strategy that takes advantage of framework.**

Achieved. We tested the performance of such a strategy under various circumstances and showed in which cases it is successful.

5.4.3 Research Questions

- **What does our work have in common with the previous research ?**

In principle, we stuck to basic design rules of Prisoners Dilemma simulations we found in the most influential papers. The basic idea for our strategy stems from recent (and yet unpublished) research by Yao and Ellis. Also, we incorporated basic measurement techniques from Small World Network research. We differ from previous research by combining the two, which enables overall trust behaviour to use local connections, which resembles the real world better. See chapter 2 for further elaboration.

- **What are possible (dis)advantages of such a system compared to other approaches to the Non-iterated Prisoner's Dilemma ?**

Potential advantages are marked with '+' and disadvantages with '-':

- We saw in hypothesis 2 that our strategy will have problems as long as other individuals cooperate too much with defectors.

-/+ The performance of our strategy is a tradeoff between effect (a global informed strategy would perform better than ours) and safety (but it would be vulnerable to "lying attacks". e.g. cartels)

+ Our design is a distributed one and even though not explicitly tested, strategies build upon this framework should be resilient to lying and possibly to "cartel attacks"

- **How to obtain and deal with information about an agent through the chain ?**

A lot of our discussions dealt with this topic. We noticed that designers will need to find tradeoffs between

- the establishment of global versus local connections
- preferences of distance or trust when deciding who to ask.

A first analysis on the second point indicates at least a promising further research path:

We conducted a Welsh Two-Sample T-Test ($t = -2.5113, df = 1807.476, p - value = 0.01211$) to test on the difference of the population representation of both *nuts*-agents that used $D = 0.3$ versus $D = 0.7$. The results show that agents that use $D = 0.7$ will outnumber the agents that use $D = 0.3$, but not significantly (when $\alpha = 0.05$). The mean of $D = 0.3$ -agents was 117.6127 and the mean of $D = 0.7$ -agents was 126.0471.

- **How does such an informed strategy perform in a society against standard strategies ?**

The experiments proved that our informed strategy is capable of adapting in the environment of unfriendly strategies (e.g. *always_defect*). Of course, further research would deal with more enhanced strategy based on proposed framework, more various strategy scenarios and experiments with performance against "lying strategies" and possibly cartels.

6 conclusions

Our main objective was to propose a reputation based system in Small World Networks, playing the non-iterated Prisoner's Dilemma. Agents should be enabled to answer requests about the trustworthiness of other agents and thus, any agent might find out if an opponent can be trusted by building up a trusted information chain. We wanted to make the information exchange in the framework scale well (with respect to the size of the network) and test the performance of a strategy that takes advantage of this information.

Our experiments showed that such a framework is computationally feasible. We modelled a strategy for agents that worked in a pure local and bottom-up manner. Agents tried to obtain information about opponents by asking other agents they already trust who in turn ask others themselves. Small World Networks emerged by playing the Prisoners Dilemma repeatedly and thus, the agents often succeeded in establishing such an information chain with a limited length. We also showed that making use of the possibility to ask trusted neighbours for information about opponents increases payoff. Furthermore, in scenarios which did not include exploitable, always-cooperating agents, we showed that our strategy outperforms simple defector strategies.

We conclude that such a framework is a feasible and promising way to go for current network trust research. The main advantage of this approach is that trust information is less anonymous - only if agents trust one another along the chain will the information itself be trusted. Thus, this framework may be more secure against exploits like cartels. Future research will need to implement scenarios like this to look at this strategies ability to withstand.

The topic of applying trust in social networks is very vast and in our work we merely managed to introduce the basics of a new framework and perform simple testing. However, we believe that this framework might be a good starting point for further research. Especially, the following items seem very interesting to us.

1. To propose a more elaborate strategy than *nuts*. For instance, experiment with different default actions (when no information chain can be found) and their consequences on the performance of the strategy.
2. To perform testing with more clever and more various opponent strategies than just *always_defect* and *always_cooperate*.
3. To observe whether the framework provides enough capabilities for our strategy to survive in presence of liars.
4. To experiment with cartels of liars (artificially raising trust about other agents in the cartel).

Bibliography

- A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *HICSS*, 2000. URL citeseer.ist.psu.edu/article/abdul-rahman00supporting.html.
- Guillermo Abramson and Marcelo Kuperman. Social games in a social network. *Phys. Rev. E*, 63(3):030901, Feb 2001. doi: 10.1103/PhysRevE.63.030901.
- Robert Axelrod. The evolution of cooperation (summary), 1984.
- T. S. Ellis and X. Yao. Evolving cooperation in the non-iterated prisoner's dilemma: A social network inspired approach. 2007.
- Nicolas Höning and Tomáš Kozelek. Table of results, 2008. URL http://conan.sk/~tomik/stuff/aso/results_table.html.
- Jon Kleinberg. Complex networks and decentralized search algorithms. 2006.
- Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. Notions of reputation in multi-agents systems: a review. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 280–287. ACM Press, 2002. ISBN 1-58113-480-0. doi: <http://doi.acm.org/10.1145/544741.544807>.
- M. Oliphant. Evolving cooperation in the non-iterated prisoner's dilemma: The importance of spatial organization, 1998. URL <http://cogprints.org/170/>.
- C. O'Riordan. Forgiveness in the iterated prisoner's dilemma, 2001.
- Luke S., Cioffi-Revilla C., Panait L., and Sullivan K. Mason: A new multi-agent simulation toolkit. 2004.
- Martijn Schut. Scientific handbook for simulation of collective intelligence, 2007.
- John von Neumann and Oskar Morgenstern. Theory of games and economic behavior, 1944.
- J. Duncan Watts and H. Steven Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393, June 1998.