**Abstract**

Maintaining cooperation when group interests are at odds with individual ones constitutes a major challenge. However, previous research has shown that reputation effects may constitute an important means to combat cooperation problems. According to the framework of indirect reciprocity, actors are more likely to cooperate with partners who have shown cooperative behavior in the past. Thus, cooperative individuals, are more likely to receive cooperative acts than uncooperative ones, keeping overall cooperation constant and high. Our experiment builds upon Milinski et al’s (2002) work, which amongst others, showed the benefits of implementing reputation to solve public dilemmas. However, we focused on comparing against a control group, where building a reputation was not possible at all. In our experiment both groups played alternately one round of a public good game followed by an indirect reciprocity game.

**1 Introduction & Theoretical Background**

The tragedy of the commons is a dilemma described in an influential article titled “the tragedy of the commons” written by ecologist Garret Hardin and published in 1968 (Hardin 1968). The Dilemma is caused by actors, who share a common good but act rationally according to each self interest. Thus, they behave contrary to the best interests of the whole community which leads to depletion of the good. Nevertheless, in everyday life, persistent cooperation amongst humans seems to be a common phenomenon. Social Dilemmas predominantly have been studied via public good games (Sylwester & Roberts 2013). Typically, cooperation declines after a few rounds of playing public good games (Fehr & Fischbacher 2003). Nevertheless, it showed that some game features can foster cooperation and even maintain it (Sell & Reese 2014). Amongst other factors, research on reputation in social dilemmas has shown, that it can improve the likelihood of cooperation and thus preventing recources from depletion (Milinski et al. 2002). In their experiment the authors used two game sequences, where the first sequence consisted of 16 rounds of a public good game alternated with an indirect reciprocity game. In the second sequence, participants played eight consecutive rounds of public goods games, followed by eight rounds of indirect reciprocity. In the first sequence the levels of cooperation remained high throughout the 16 alternating rounds, while during the second sequence, cooperation declined over the course of the eight repeated public goods game rounds and was restored over the course of the indirect reciprocity games. Subsequent studies pointed in the same direction, by supporting Milinski et al.’s (2002) findings in showing, that participants will cooperate more often, when an incentive to gain reputation exists (Milinski et al. 2006, Seemmann et al. 2005).

**2 Research Design**

We therefore, build upon Milinski et al’s (2002) work, however we focus on the reputational effect and try to isolate it. Therefore, contrary to Milinski et al. (2002) we do not change the sequence of the experiment as a treatment. We focus on the effect of the reputation by introducing a control group where the players do not know the history of other player's decision. We expect that this group will be more likely to run into a tragedy than the other group, where the individuals are interested in maintaining their individual reputation. To implement our research design in our online experimental environment, we first of all divide participants into two groups. The treatment group gets to see the history of their respective partners in the game, while the control group does not and they get informed by their respective instructions about whether their decisions will be visible for others or not. Subsequently, both groups play alternately one round of a public goods game (PGG) followed by a round of an indirect reciprocity game (IRG). However, contrary to common PGG we implemented a resource pool, from which each individual draws from. That is, in the first part, players have to decide whether they want to cooperate or not, by deciding to draw a lower (cooperate) or higher (not cooperate) quantity of the resource, which is then saved in their private account. In the second part, players get randomly assigned a partner, and get asked if they want to donate a certain amount of their private account to their respective partner. At this point in time, the treatment group will be able to see past decisions of their assigned partners, while the control group does not. In the next round, the resource pool gets refilled by a certain quantity, such that if all of a group cooperated, the pool remains at the same level and does not go extinct. However, consequently if not all members cooperate, the pool shrinks overtime and allows groups to go bankrupt. That is, if a group depleted their resource, they are no longer able to draw from it, and only can switch tokens between themselves.

**3 High-level description of our software implementation**

In this game we implement some features - for example: stages, widgets, decisions, allocation, a common and a private account. For each new page is a new stage defined and in these stages are functions written. The stages determinate the order in which the pages appears. The first step contains the instruction, the second step contains a quiz to the instructions, the third step contains the choice of the public goods game (“how much do you want to remove out of the common pool?”), the next step displays the results of these decisions, the fifth stage contains the decisions of the exchange game (indirect reciprocity), the next step displays the results of these decisions and the last steps displays some feedback options and the total win by each player. The within functions manage the communication between the logic-file (server level) and the players-file (local level), displays the page in the browser as well as handling the input by a user. So in these stages is most of the code embedded, like all the following descriptions. The widget displays different things, like a visual timer, the round where the user is located or a counter to display your payoff. These three widgets are implemented in the code as well as a quiz widget, a feedback (+ total win) widget and a widget to end the round (the so called “Done Button”). Moreover the user can make decisions in this game such as how many fishes the user want to catch (five or ten) or how many coins the user would like to exchange (zero or two point five). Implemented therefore in the code are buttons which the user can click. The decision what button the user choose has an impact on the private and/or the common pool. Furthermore the choices of the button are allocated in the background by the code and communicate between the logic- and the players-file. Supplementary the users can extract an amount (defined by us (five or ten)) out of a common pool. This pool increases every round by the minimum choice the users have. But the pool can probably decreases by the players decisions and can pass zero but then the players fails and cannot extract an amount anymore. The players can only play with their private accounts. This account includes all removal out of the common pool plus the choices if they exchange coins or not.

*3.1 What problems did you encounter and how did you solve them?*

Of course there were some issues at the coding time, for example to display the updated common pool by each round. We do not know how we should code the subtraction by every user from the common pool without to define for each user a new variable for the value of the last removal decision. So we are asking for some help and our lecturer gives us the example of a for-loop. This for-loop solves our problem. So we define a variable only in the for-loop and this match to the players identification. Also we refill the common pool in this for-loop and the common pool is displayed equal to each player. Furthermore there was the question what is happened to the game if the pool extinct? In theory the public goods game will be skipped. The players have now only the opportunity to play with their private accounts and exchange some coins in the reciprocity game. We try some things but they want not work. But in the end we found a solution. We create a new HTML page which appears if the common pool extinct as well as the choice of the removal out of the common pool will be skipped. Both approaches are determinate by if-conditions. The decision-step of the public goods game will be skipped and the new HTML page will be displayed at the result page of the public goods game. There will be written some information about your loosing.

In the indirect reciprocity game, we encountered an error that only occurred seemingly randomly during our tests: The decision about the donation was sometimes displayed to the wrong player, and the coins were also taken away from the wrong players account. We found out that the IDs of the players we had used to send out the decision back to the correct player were stored in an array which remained fixed during the experiment. The decisions however, were saved in the order they arrived, which was sometimes equivalent to the order of the players, (when the player who logged in first, first clicked a button) and sometimes not. We solved this problem by not using the IDs from the game’s history. Instead, we used the players’ IDs that were being sent together with the incoming offers. These IDs were then used to correctly specify the receiver (the player who had made the decision himself). Generally, it was challenging and error-prone to deal with the decisions in the indirect reciprocity game, because the same decision of a certain player had to be processed in two different ways. The decision had to be returned to the player himself and forwarded to the exchange partner after being manipulated. Last but not least, having different treatments beared a challenge for the indirect reciprocity game: Some contents had to be displayed only for one of the two treatment groups. We decided to move the statements from the htm-files to the player-client to be able to trigger them with an event listener in case they were needed (see chapter “Lessons learnt”).

*3.2 Are there some open problems in your implementation that you were not able to code?*

Say what and why you could not code them. The biggest problem we were not able to solve is related to our treatment. As explained above, we were planning on displaying the history of all previous decisions of the “partner” in the common pool game stage to the players of the treatment group. It would have been necessary to generate a sort of table containing the decisions of the other player in the first step of the reciprocity game. This table would have been updated with every round, with only one decision displayed in the first round, and six in the last round. We were not able to code this, because we did not know how to access an array of all decisions of a player in the previous rounds. In the code elements we used as the starting point for our code, there were only functions that stored the players’ decision of the current round in the memory, in the next repetition of the respective game, this variable was being overwritten by the code. Hence, it would have been necessary to store the decisions not only player-, but round-specific. We had no idea how to code this, which is why we decided to circumvent this by changing the experimental design such that only the most recent decision of the other player is displayed now. For a critical discussion of this simplifying decision, see the conclusions.

**4 Conclusions**

The implementation of our research design was almost completely successful. It is now possible to let real participants play the game. Our treatment makes it possible to investigate the isolated effect of reputation on a version of the classical “tragedy of the commons”. The results of repeated executions of the experiment could be compared using a single parameters: The share of groups that managed to avoid running into disaster by erasing the fish population. We would expect this parameter to be higher in groups with the treatment. We tried to make our design as robust as possible by staying close to the methodology used by Milinski et al. (2002). For example, the number of coins to be distributed in the indirect reciprocity game is identical. The research design is guided by the aim to provide a high external validity. Hence, we introduced both a refill mechanism of the common pool and the possibility for it to become empty. This in turn could compromise the internal validity of the experiment. The reason is that there are two different motives for not choosing the maximum amount of fish: The fear of the pool running out or the reputation and feared consequences of it. The most important weakness of our study is the simplified treatment (see description of implementation). Because only the last decision is being displayed, one could argue that contrary to a real history of decisions, now we simply provide the players with the possibility to punish deviant behavior. Thus, statements about the effect of reputation have to be made with caution.

**5 Lessons Learnt**

Reflect about what you knew before lecture 1, and what you have achieved at the end of the course. We were all some beginners in programming with JavaScript (especially in Node.js) and HTML. But one group member has some school knowledge in JavaScript so that was a little bit helpful for understanding what is happening in the code. Moreover we developed an elaborated online experimental design on the basis of scientific literature. Also we started almost from the scratch and in the end; we ended up with an online experiment that runs without major problems. 6.2 What recommendations would you give to a future participant of this course and to the lecturer? First at all, this entire seminar was very helpful in having a look into coding some JavaScript and HTML elements and displays them into a local browser. We, as the group, have really some fun at coding the online experiment as well as there were some situations which were very frustrating, but in the end of each day there was always a lucky moment if something works. Additionally, at the beginning we have some issues to understand the syntax. Therefore we had to do some JavaScript coding at home, so we have can play a little bit around. Furthermore, the lecturer has can provide students with some easy homework (not mandatory) to get some experience how sampling coding works, especially for some basic Node.js and JavaScript functions. For example some simple tasks to work in the nodegame environment, like “Take the dictator basic game and change some simple details”.

**Bibliography**

Fehr, E., Fischbacher, U., 2003. The nature of human altruism. Nature 425, 785–791. <https://doi.org/10.1038/nature02043>

Hardin, G., 1968. The Tragedy of the Commons. Science 162, 1243–1248. <https://doi.org/10.1126/science.162.3859.1243>

Milinski, M., Semmann, D., Krambeck, H.-J., 2002. Reputation helps solve the ‘tragedy of the commons.’ Nature 415, 424–426. <https://doi.org/10.1038/415424a>

Milinski, M., Semmann, D., Krambeck, H.-J., Marotzke, J., 2006. Stabilizing the Earth’s climate is not a losing game: Supporting evidence from public goods experiments. Proceedings of the National Academy of Sciences 103, 3994–3998. <https://doi.org/10.1073/pnas.0504902103>

Sell, J., Reese, B., 2014. Social Dilemma Experiments in Sociology, Psychology, Political Science, and Economics, in: Laboratory Experiments in the Social Sciences. Elsevier, pp. 225–245. <https://doi.org/10.1016/B978-0-12-404681-8.00010-8>

Semmann, D., Krambeck, H.-J., Milinski, M., 2005. Reputation is valuable within and outside one?s own social group. Behav Ecol Sociobiol 57, 611–616. <https://doi.org/10.1007/s00265-004-0885-3>

Sylwester, K., Roberts, G., 2013. Reputation-based partner choice is an effective alternative to indirect reciprocity in solving social dilemmas. Evolution and Human Behavior 34, 201–206. <https://doi.org/10.1016/j.evolhumbehav.2012.11.009>