Project Plan

James King

September 2018

Abstract

For 6 marks copy from the project list.
For 15 marks describe and motivate the final project.
For 30 marks describe and motivate, and also give well thought out individual project goals.

Background

The Iterated Prisoner's Dilemma uses a mechanism known as direct reciprocity to aid in the evolution of cooperation, this was popularized in Axelrod's tournaments and research on it [1]. There is a well tested library that is able to run matches, tournaments and the Moran Process for the Iterated Prisoner's Dilemma [2]. This is not the only mechanism used to aid in the evolution of cooperation. Nowak presents 4 alternative methods [3] including: kin selection, group selection, network reciprocity and indirect reciprocity.

Of these indirect reciprocity stands out as an interesting model for interaction between humans or intelligent agents. Pairs of agents are chosen at random to interact (they may not necessarily interact again), in each pair one agent is the donor and the other is the recipient. A donor chooses whether to defect or cooperate, defecting gives them a higher payoff so is always tempting, however the model utilizies the idea of reputation (also known as image score [5]) to create a condition where an agent may want to cooperate. If the donor cooperates their reputation will be enhanced, if they defect it will fall. Some strategies known as discriminators, will choose to defect if the recipient has a low reputation or cooperate if they have a good reputation.

Nowak and Sigmund present the idea of onlookers [5] this limits the amount of direct observers of interactions. As is noticeable in real life, direct observation of other's interactions is not always possible, so this seems to more closely match real life. In societies with advanced communication however direct observation is not the only method of finding out about interactions, gossip has been put forward as an alternative form [6]. Previously the outcome depended upon the probability of witnessing another's decision, now with gossip it depends upon how the information has been spread and how agents base their behaviour on gossip. Ralf D. Sommerfeld et. al state that gossip needs to truthfully describe other's behaviour, and when hearing correct gossip an agent should act accordingly or gossip will fail. It is also suggested that gossip can be kept within the boundaries of truth if spreading misinformation leads to a loss of reputation.

In thinking about how to implement this model, we will consider using the logic programming language Prolog - because it has the potential to provide transparent decisions - to implement an agent's decision making component was most appropriate, whilst implementing the environment itself in a procedural language would be more appropriate (Python has been chosen). The issue with this is interfacing between the two languages, there are libraries such as PySwip [8] that are able to do this but are limited. For instance PySwip struggles working in multi-threaded environments.

Project Description

I will explore indirect reciprocity, by both researching and reporting on Nowak and other authors research on the mechanism, drawing on their research and possibly adding on to the ideas of onlookers and gossip to create a model for implementation. I shall be using the Axelrod-Python library to create a proof of concept that I am able to create a web application that allows users to set up and run games and tournaments. This web application will be hosted on heroku and created using Flask and Python.

The solution that we are exploring for the interfacing issues between Python and Prolog is to create a web service for the agent's decision making component. This will have a Prolog server containing "agent's minds". The minds themselves will represent information on the state of the environment using an efficient version of the events calculus that Kostas has worked on (mvfcec). Percepts can be added to the agent's mind, and based on these percepts and agent will be able to decide on an action, and act on it by returning an action to the environment. Separating out the head and body as in Stathis *et al.* 2002 [7], though this solution would take this separation further, having the head based in the Prolog service and the body in the environment.

The capabilities of the Prolog service are to be demonstrated on the web application, which will run indirect recirprocity tournaments creating the environment that agents reside in and containing the body of the agent. The web application will use the service to input percepts and get actions from.

These have been the core parts of the project, however I could extend this to the other mechanisms presented by Nowak [3] if I have time. Axelrod-Python has an implementation for network reciprocity - which would be interesting to implement so I could attempt to create an especially nice frontend for the patterns created in the population [4]. Due to the nature of group selection working on top of direct reciprocity [9] it would be possible to implement it on top of the Axelrod-Python library. Leaving kin selection to implement using the Prolog service.

Goals

- To create a Prolog service that is able support the decisions of a game-theoretic agent to interact with any other system that is capable of creating an environment for that agent to work in, and contacting the service.
- To explore in a game-theoretic context a mechanism to aid in the evolution of cooperation known as indirect reciprocity.
- To create a web application that allows users to set up and run game-theory tournaments of both direct and indirect reciprocity.
- To use the web application to demonstrate an environment that combines the Prolog service and a set of game-theory strategies and evaluate the results.

Motivation

The development of the Prolog service will provide a solution to problems interfacing between Prolog and other languages, meaning it may be a useful model for intelligent agents and multi-agent systems research. It will develop my logic programming skills, this is key to my interest in intelligent agents, aiding my aim to secure a job in the AI sector.

Understanding and modelling interaction between agents is important to the development of AI systems, indirect and direct reciprocity are two aids to this, thus understanding and implementing them will aid me in my aim to secure a job in AI. Not only this but due to my research I have developed a keen interest in game theory and studying interactions between biological agents, especially human interaction.

Web development is an area of interest for me as well and is a sector in which I'd be keen to go into.

Developing a web application will help me develop my skills as a developer in the full stack, whilst the Prolog service will augment my skills in developing web services.

Timeline

For 8 marks copy milestones from the project list without dates.
For 20 marks copy milestones from the project list but with dates or extra milestones.
For 30 marks milestones are adequate, with dates but without sufficient motivation.
For 40 marks a good list of well explained milestones.
Early Deliverables
Familiarisation with the idea of strategies and the identification of set of them.
Implementation of strategies and the definition of an interface that allows agents to be selected and play against each other
Definition of a tournament
Report on the design of a tournament.
Implementation of a tournament, where a user selects the number of agents, associates them with existing strategies, specifies the number of iterations and deploys a tournament
Report on the Iterative Prisoner's dilemma problem
Report on previous work with the Iterative Prisoner's dilemma

Final Deliverables
A program that allows us to simulate tournaments of the iterative prisoner's dilemma.
A GUI that allows a user to create and manage tournaments.
GUI should provide summary statistics that are tournament specific, i.e. a summary for each player/strategy and how it is compared with the other players/strategies in the tournament.
The report should describe some of the theory behind strategies in cooperative settings.
The report should provide an analysis and an evaluation of the different strategies, including an indication of which one is the best strategy, if any.
The report should contain a design of the program and a discussion of any software engineering issues encountered.
The report should describe any useful and interesting programming techniques that have been employed to develop the final prototype.

Suggested Extensions
Make use of multi-agent system platform to support the tournament in a completely distributed setting.
Use logic-based rules to specify the strategies and wrap them in the Java implementation.
Use strategy recognition techniques so that agents can detect what strategies opponents are playing, and adapt their behaviour accordingly.
Create a methodology for developing tournaments of this type.

First Term

Report on the design of the Prolog service and Agents, and the web application and agent's environment
Including an agent definition and the inner workings of an agent's decision making component, a definition of the structure of the Prolog service and management of agents (as well as the design of the API), a definition of the agent's environment and how this will work with the Prolog service, and a design of the web application.
Date: Method: Explanation and motivation: Link to other milestones:

Milestone: Report on the evolution of cooperation in relation to game-theory and different game-theoretic mechanisms to aid it
Report on past work on this area such as Axelrod's tournaments [1] and Nowak's five rules for the evolution

of cooperation [3].
Date: Method: Explanation and motivation: Link to other milestones:

Milestone: Report on indirect reciprocity, strategies for agents and development of a concrete model to implement
Report on past work on indirect reciprocity including the strategies agents have employed playing this game. From this collated work I shall define a model which I am to implement and a number of key strategies.
Date: Method: Explanation and motivation: Link to other milestones:

Milestone: A proof of concept program for running a Prolog web service
I will be using tutorials online to learn how to create a Prolog web service, and from this learning I will attempt to create my own service. This service should resemble a very early version of the Prolog service I wish to provide.
Date: Method:
Explanation and motivation: This will prove to myself that I can create a Prolog web service and gain me confidence, but also reduce the risk of the project failing if I cannot create one.
Link to other milestones:

Milestone: A proof of concept web application to run direct reciprocity games and tournaments and provide statistics on them
The environment for the agents will be on a web application, this proof of concept will show that I am capable of creating a web application that can run similar games without the need to program the environment inside it yet. I will be using the Axelrod-Python library to run the games for a convincing mockup.
Date: Method: Explanation and motivation: Link to other milestones:

Milestone: Design and implementation of a GUI for indirect reciprocity tournaments
This will follow on from my proof of concept web application, to which I will add a GUI to to set up, run and provide analysis on indirect reciprocity tournaments. This will help motivate my creation of the backend for running the environment and the Prolog service in turn.
Date: Method: Explanation and motivation: Link to other milestones:

Milestone: Design and implementation of the structure of the Prolog service
Taking inspiration from my proof on concept Prolog service and previous architectures for intelligent agents and multi-agent systems [7] I will define the structure of the Prolog service.
Date: Method: Explanation and motivation: Link to other milestones:

Milestone: Design, implementation and testing of the management of agents in the Prolog service
Once the structure of the Prolog service has been implemented I can then develop the components that will manage agents in the system. The system will have to deal with the management of multiple agents all at one time so this must be carefully implemented and thoroughly tested (I will need to learn how to test a Prolog system for this).
Date: Method: Explanation and motivation: Link to other milestones:

Milestone: Design and implementation of agent templates in the Prolog service
For agents to work correctly in the system they will have to follow constraints (naming and functionality). These constraints I will define in a template.
Date: Method: Explanation and motivation: Link to other milestones:

Milestone: Design, implementation and testing of the environment for a simple indirect reciprocity tournament
The environment is a key aspect of the overall product. It will run the cycles of the game and record information about the game. This will be written in the web application and needs to be thoroughly tested.
Date: Method: Explanation and motivation: Link to other milestones:

Milestone: Design, implementation and testing of a defector and cooperator strategy for a simple indirect reciprocity tournament
We need strategies in the Prolog service to play indirect reciprocity tournaments. The simplest of which are defector and cooperator strategies, which is why I shall start with them.
Date: Method: Explanation and motivation: Link to other milestones:

Milestone: Design, implementation and testing of a discriminator strategy for a simple indirect reciprocity tournament
Discriminator strategies make indirect reciprocity tournaments interesting, and are likely to be the next simplest to implement after defector and cooperator.
Date: Method: Explanation and motivation: Link to other milestones:

Milestone: Linking together the Prolog service and web application environment
After the implementation of a subset of indirect reciprocity functionality without onlookers and gossip I should now be ready to link together the tested web application environment and well tested Prolog service.
Date: Method: Explanation and motivation: Link to other milestones:

Second Term

Milestone: Design, implementation and testing of the onlooker and gossip features in the web application environment
The model that I'm exploring becomes more interesting with onlookers and gossip features, with the simple version of the web application and Date: Method: Explanation and motivation: Link to other milestones:

Milestone: Design, implementation and testing of the onlooker and gossip features in the Prolog service
Date: Method: Explanation and motivation: Link to other milestones:

Milestone: Design, implementation and testing of a development upon current strategies to play the gossip and onlookers meta-game
Date: Method: Explanation and motivation: Link to other milestones:

Milestone: Relinking the Prolog service and web application environment with the onlooker and gossip meta-game implemented
Date: Method: Explanation and motivation: Link to other milestones:

Milestone: Implementation and testing of further strategies for an indirect reciprocity tournament
Date: Method: Explanation and motivation: Link to other milestones:

Enhanced analysis post-tournament for indirect reciprocity
Be able to query the motivation behind an agent's decisions. Graphical display of analysis. Date: Method: Explanation and motivation: Link to other milestones:

Enhanced GUI features
Use React.js, bootstrap and CSS for a better GUI Date: Method: Explanation and motivation: Link to other milestones:

Milestone: Development of content for the web application to make it into an educational system for the evolution of cooperation
Date: Method: Explanation and motivation: Link to other milestones:

Milestone: Design and implementation of a database to record results of indirect reciprocity tournaments
Date: Method: Explanation and motivation: Link to other milestones:

Risk Assessment

For 3 marks general risks are given such as "may get behind".
For 6 marks risks are associated to deliverables and show clear thought.
For 10 marks risks provided with reasonable mitigations.
For 15 marks a good case is made for understanding how the project may fail to proceed and what should be done about it. Some risks have likelihoods and importance.

Risk: Using SWI-Prolog to run a web service

This is new to both me and my supervisor. It has been done before and there are resources online, but it is still new territory to me. This could delay the delivery of the agent mind as a service system.
Mitigation: If I fail to be able to deliver this on time I shall switch my plan to one of the following two options: write a service in Python to replace the Prolog service or forget the service and code the program to run indirect reciprocity directly in the web application.
Likelihood: Possible
Importance: Moderate

Risk: Failure to understand and use mvfcec

Mvfcec or the multi-value fluent cached event calculus, is a version of the event calculus where querying is faster. I have knowledge on the event calculus itself, but this is a more complicated form of it. If I fail to understand and use it, I cannot write a report on it and either the Prolog service will be far slower or I will fail to deliver it
Mitigation: I shall attempt to avoid this by working with Kostas to understand it. However if I still fail to grasp mvfcec, I shall either switch to using the simplified events calculus, or use one of the two mitigations from the risk: Using SWI-Prolog to run a web service. Likelihood: Possible
Importance: Moderate

Risk: My inexperience of writing large Prolog base applications

I have used Prolog in the AI course in second year, however these applications were nowhere near as large or complex as the Prolog service will have to be. I am doing it this way to push myself, but recognise this could cause a failure to deliver the service.
Mitigation: switch from Prolog to Python for writing the service or simply include the agent decision making component in the web application.
Likelihoood: Possible
Importance: Moderate

Risk: My inexperience in writing large web applications

In first year we used HTML, CSS, Javascript etc. to learn about writing web applications. However writing a large complex back and front end is new to me. Meaning there will be obstacles that I have not yet anticipated.
Mitigation: I have taken the Miguel Grinberg Flask Mega-Tutorial, to further practice my skills in writing web applications. I feel that my practice in this has allowed me to give a low likelihood that this will affect my project, and due to the resources available for Flask the importance of any obstacles to my project will be minor.

Likelihood: Unlikely
Importance: Minor

References

[1] Robert Axelrod and William D. Hamilton. The evolution of cooperation. *Science*, 211:1390–1396, 1981.

[2] Vince Knight; Owen Campbell; Marc; eric-s-s; VSN Reddy Janga; James Campbell; Karol M. Langner; T.J. Gaffney; Sourav Singh; Nikoleta; Julie Rymer; Thomas Campbell; Jason Young; MHakem; Geraint Palmer; Kristian Glass; edouardArgenson; Daniel Mancia; Martin Jones; Cameron Davidson-Pilon; alajara; Ranjini Das; Marios Zoulias; Aaron Kratz; Timothy Standen; Paul Slavin; Adam Pohl; Jochen Müller; Georgios Koutsovoulos; Areeb Ahmed. Axelrod: 4.3.0. `http://dx.doi.org/10.5281/zenodo.1405868`, September 2018.

[3] Martin A. Nowak. Five rules for the evolution of cooperation. *Science*, 314, 2006.

[4] Martin A. Nowak and Robert M. May. Evolutionary games and spatial chaos. *Nature*, 359:826–829, 1992.

[5] Martin A. Nowak and Karl Sigmund. Evolution of indirect reciprocity by image scoring. *Nature*, 393:573–577, 1998.

[6] Ralf D. Sommerfeld, Hans-Jürgen Krambeck, Dirk Semmann, and Manfred Milinski. Gossip as an alternative for direct observation in games of indirect reciprocity. *Proceedings of the National Academy of Sciences of the United States of America*, 104:17435–17440, 2007.

[7] Kostas Stathis, Antonis C. Kakas, Wenjin Lu, Neophytos D emetriou, Ulle Endriss, and Andrea Bracciali. *PROSOCS: a platform for programming software agents in computational logic*, pages 523–528. 4 2004.

[8] Yuce Tekol. Pyswip. `https://pypi.org/project/pyswip/`, 2018.

[9] Arne Traulsen and Martin A. Nowak. Evolution of cooperation by multilevel selection. *Proceedings of the National Academy of Sciences of the United States of America*, 103:10952–10955, 2006.

For 3 marks just include web pages.
For 6 marks clear use of more than one source, not just web sites.
For 10 marks nice evidence of good research and understanding how this motivates the timeline.
For 15 marks items are very good and each has a short relevant discussion of it's value to the project.