University of Southampton
Faculty of Engineering and Physical Sciences
Electronic and Computer Science

# Negotiation intelligent agent with evolutionary approach

by
Pornphapat Innwon
September 2020

Supervisor: Dr. Enrico Marchioni
Second Examiner: Dr. Erisa Karafili

A dissertation submitted in partial fulfilment of the degree
of MSc Artificial Intelligence

# Abstract

Autonomous negotiation agent is one of attention in e-commerce because it can negotiate on behalf of humans. This agent can do bargaining like a human negotiator and can surpass human in term of performance given the efficient algorithm. With the negotiation system, these agents can trade with other agents autonomously with their preference setting and its goal in mind. This property means each agent have different behaviours to find the most preferable offer for its client. With that, agents need to learn and adapt a strategy to fight with their competitors. Moreover, the agent usually has a partial preference from the clients because they do not know how exactly what they want each resource. Also, the real-life negotiation is complex in general and need an advanced mechanism to make a good strategy within time constant. With these problems combined, the traditional negotiation agent should not be used effectively. As such, this paper presents a negotiation agent model using the genetic algorithm to deal with this dynamic and complex environment. The genetic algorithm is an optimization technique based on the evolution of life which technically used in the combinational problem. As such the algorithm can help improving agent performance. The implementation of the negotiating agent using the genetic algorithm is done in this project to learn the mechanism behind each component. In this project, the result is presented from the experiment for evaluating the agent model and studying the interaction between the agents in the negotiation.

# Acknowledgement

I would like to express my sincere thank of gratitude to my supervisor, Dr Enrico Marchioni, for guiding and supporting me during the project process. I am very grateful to the examiner, Dr Erisa Karafili, who gave me guidance on writing for this dissertation.

**Statement of Originality**

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

***You must change the statements in the boxes if you do not agree with them.***

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

**I have acknowledged all sources, and identified any content taken from elsewhere.**

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

**I have not used any resources produced by anyone else.**

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

**I did all the work myself, or with my allocated group, and have not helped anyone else.**

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

**The material in the report is genuine, and I have included all my data/code/designs.**

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

**I have not submitted any part of this work for another assessment.**

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

**My work did not involve human participants, their cells or data, or animals.**

*ECS Statement of Originality Template, updated August 2018, Alex Weddell aiofficer@ecs.soton.ac.uk*

# List of Contents

# List of Figures

# List of Tables

# Chapter 1 Introduction

## 1.1 Background

As the world has limited resources and people can get different resources depending on the living environment, they need to negotiate to get resources they want. For example, people need to trade something valuable for food to survive. This reason is why negotiation is important since the ancient era. Negotiation system has been developed constantly to improve human quality of life. Nowadays, people use money as a currency to exchange items such as meals, clothes and houses. Different negotiation systems are made to provide a fair and secure environment. The online digital market is one of the alternatives that people comfortably use to purchase things with their technical devices and their digital currency. Lately, the autonomous negotiation agent is a growing interest topic to this e-commerce as it can be beneficial to this market. The negotiating agents have been developed and improved over the past decade. The performance record of these agents demonstrated that they can outperform human negotiator with the condition that they have a great understanding of negotiation space according to Bosse paper [1]. These agents can make a mutual agreement with other agents autonomously using client interest as the preference to give an offer that is favourable to its client. In other words, the agent can act to fulfil their objective goal under the negotiating setting it employs. The reason behind of behaviour can give insight into how the human mind works. This information is precious to artificial intelligence research.

The negotiation in this scenario is about agents creating an offer that is acceptable to their competitors. The offer is an agreement made by one of the agents. The detail of the agreement describes how resources are allocated between two agents. During the bargaining process, there are a limited number of possible offers that an agent can make. Each agent then must find an offer that satisfies its preference from this space. The counter-offers occur when one agent does not accept an offer which its competitor make as that offer is unacceptable in term of market perspective. Several rounds of offers are made and exchanged between both agents in one negotiation until it found a mutually accepted agreement or the negotiation is terminated due to time constraint.

Many factors must be considered to make an efficient negotiation agent. First, the agent does not have information about opponent preference due to information privacy. This secrecy is normal in real-life negotiation so the agent must acquire and understand this knowledge and through bargaining. The agent should able to find an acceptable offer to the opponent as each agent mostly act on their self-interest. Second, the agent only has its partial preference because the client has uncertainty about their desire. This problem is called user uncertainty in negotiation. Thus, the agent must make a strategy that can deal with this problem. Finally, the negotiation in real-world is very complex as the domain consists of many issues and has a limited time constraint to negotiate. Each issue has difference multiple values based on its type like a combinational problem. With limited computation power and vast domain space, finding the optimal offer can be a challenge to the negotiating agents. As such, the agent must be able to handle these problems efficiently and successfully.

For dealing with these problems, this paper introduces a genetic algorithm approach which is considered as the optimization technique. The mechanism is to find satisfying offers or candidate offers in negotiation setting based on natural evolution. The nature of this algorithm generally uses to find an optimal solution in complex and wide problem space effectively [2]. It can continuously adapt to the negotiation process. The algorithm can be used to empathize the opponent by finding an offer that optimizes both payoffs. This reason is to increase the chance that the competitor will accept the offer because agents are self-interest to their task in general. As such, this algorithm can be used to improve negotiation performance.

## 1.2 Project Aim and Objective

The main purpose of this paper is to study the mechanic behind the negotiation agent and the principle of genetic algorithm. This aim also includes implementation of the negotiating agent model. The motivation comes from the ongoing interest in the electronic market. The negotiation agent is capable to do a negotiating task without human guidance. With high applicability, the company can use these agents to save human resource. The standard negotiating agent has difficulty to find an optimal agreement on large-scale negotiation due to limited computational resource and a lack of knowledge. With the genetic algorithm, the agent should able to find a valuable offer by utilising knowledge from the negotiation and recognising main issues in the bargaining which are user uncertainty, partial information of the opponent, time pressure, and the complexity of the negotiation domain.

The main objective of the project is to design and implement a discrete multi-issue negotiation agent using a genetic approach by reviewing several literature papers. Concerning the challenges in a traditional negotiation agent system, the genetic algorithm is used to search and find optimal offers from all possible offers in negotiation domain. The development of the agent model is done by using genius platform [4][15]. The platform generally uses to develop the negotiation model. Another objective of this project is to evaluate the agent model to have a better understanding in the components of the agent. Regarding to the evaluation, the genius platform also consists of several negotiation scenarios and famous agent models from the Automated Negotiating Agents Competition (ANAC) tournament [26]. Many negotiation scenarios and other famous agents from the tournament can be used to evaluate agent performance. Note that, ANAC tournament is an event that encourages researchers to develop an agent model using knowledge of previous agents.

## 1.3 Paper Structure

The project separates into multiple chapters to give a better explanation. For chapter one, the outline of the project is explained. On chapter two, some main concepts of a related topic are revised. This chapter includes the notion of negotiation agent. Moreover, the concepts of approaches such as genetic algorithm and linear programming that are used in this project are examined. On chapter three, the challenges for building the negotiating agent is outlined to give general ideas. On chapter four, the detail of an approach that is used to create the negotiation model is explained. The alternative approaches are also explored to give the motive behind the chosen method. On chapter five, the platform used to build the negotiating agent is described. For chapter six, the experiment is used to evaluate the agent model. The performance of the agent is presented and is discussed in critical analysis. Lastly, the conclusion, future work and project management of this project is reviewed in chapter seven. Some further improvement of the agent and is also proposed in this chapter.

## 1.4 Project Contribution

Although the implementation of the genetic algorithm is done in a negotiating agent, most of the works are relied on its search function without considering opponent preference in the negotiation. Additionally, the uncertainty problem in the negotiating agent is new in recent year. In this project, the genetic algorithm is implemented in the agent using the opponent model to perceive the opponent information and to give further research on the genetic algorithm dealing with the user uncertainty in the automated negotiation.

# Chapter 2 Literature Review

## 2.1 Intelligent Agent

The intelligent agent is a system that able to do a task on their own depending on its goal without human order. The agent uses the information that receives from the event or environment it detects to perform an appropriate action for a task. It uses sensors to percept the environment and gives the values that use as guidance to make an action. For example, the air conditioner increases or decreases the power depending on the current temperature to set a suitable atmosphere in the room. The general purpose of this machine is to research how cognitive science work by observing the behaviour of the agent. The agent gives research value to develop ideal artificial intelligence. It has a lot of potentials to create more advanced applications according to the simplified concept in the paper [5]. This reason is mainly due to the wide usability of the agent. Examples of the applications are Siri or Google assistant that use the information received by voice recognition and data to give useful feedback to the users.

There are different types of intelligent agent based on the purpose of its goal. The utility-based agent is one of intelligent agent that will be used in this project. Different from normal agents, this agent focuses on the path or direction that leads to the goal [14]. The goal is just the requirement of this agent. For instance, a user wants to find the fastest route to the restaurant using the google map. The application then measures each possible path using information from that request time to give the best route to a user. The congestion of traffic road, the occurrence of accidents and festival events during that time are used to evaluate the best route from the starting point. The reason for using this agent is that the goal in negotiation is to find the most satisfying agreement from the competitor. Therefore, the utility value is used as a measurement in order to find the best possible agreement in the negotiation.

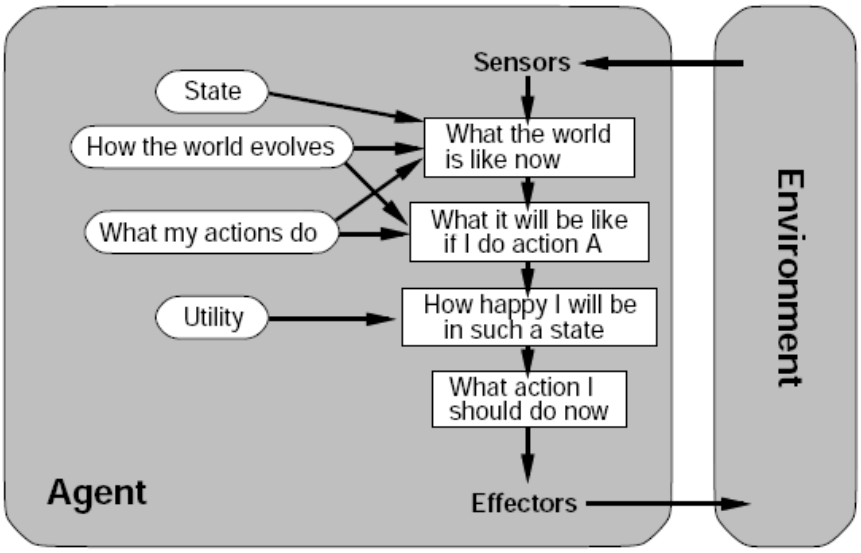



*Figure 1 Utility-based agent [14]*

## 2.2 Autonomous Negotiation Agent

For the autonomous negotiation agent, it is capable to bargain other agent using its preference to achieve a consensually accepted offer without direct support from humans. The result of the comparison between the agent and human negotiators indicated that the agent can exceed humans in term of performance with sufficient information [1]. It also shows the room of agents' potential as shown in Beam's paper [7]. As a result, this agent can be beneficial to e-commerce as it can compensate human negotiators.

There are types of agents such as cooperative and competitive agents for completing its task. The agents in this project primarily focus on self-interest agents which only pursue its own goal. The strategies need to be developed as the agent need to fight each other. According to the papers, the game theory techniques are useful for these agents as they can plan their decisions during the negotiation [3][9]. The game theory is the mathematical study to find the rationality of the players when they make a choice. It normally used in economic due to its usefulness in the bargaining problems.

The negotiation for this project will occur between two parties only which is called bilateral negotiation. The negotiating agents work inside a system like a negotiation setting shown in figure 2. The main core of the negotiation setting can be split into three major parts which are negotiation domain, negotiation protocol and preference profiles. Note that, the negotiation scenario contains negotiation domain and preference profile.

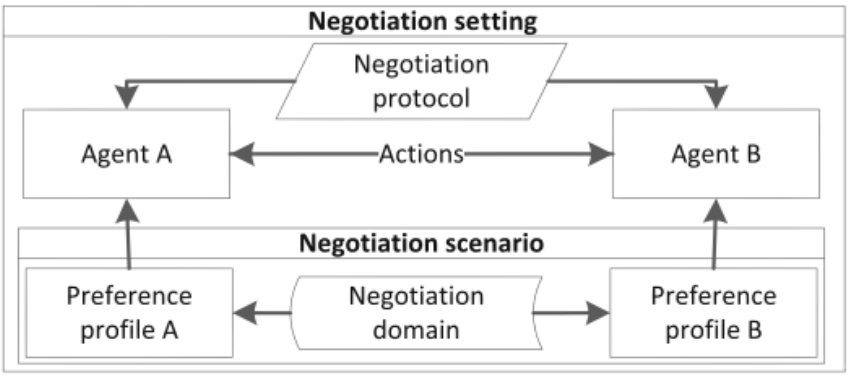



*Figure 2 Outline of the autonomous bilateral negotiation [21]*

### 2.2.1 Negotiation Domain

Like a real-life case, the negotiators should understand about the negotiation detail as much as possible to gain an advantage. The negotiation domain is a set of all possible offers as each offer consists of issues and attribute values relating to each issue. For example, one issue can be manufacture name and attribute values inside a manufacturing issue can be manufacture names. The agent can choose only one attribute value in each issue for creating an offer. In the negotiation, the agent must create an offer from this negotiation domain to send it to an opponent agent for making an agreement.

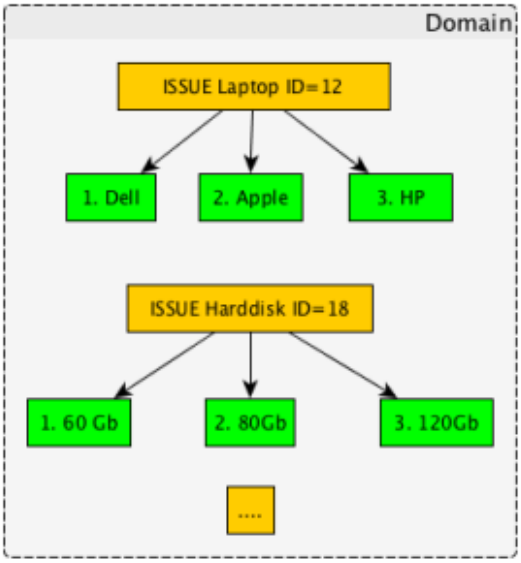



*Figure 3 Example for laptop negotiation, with orange boxes are issues and green boxes are attribute values [15]*

The domain can have one or more issues depending on the number of resource types. The negotiation with one issue is called a single-issue negotiation while negotiation with two or more issues is called a multi-issue negotiation. The issue also can be a continuous value or discrete value. The continuous value is a value that can be any value while the discrete value is a value that can be within a set of distinct values. Example of continuous and discrete values are price value and colour of a car respectively. This project will focus mainly on the discrete multi-issue negotiation. The negotiating agents can be specialized in some specific types of negotiation domain. In other words, these agents use a strategy that is efficient only in specific domains. Another type of negotiation agent is a general-purpose agent which is made to be compatible in various negotiating domain settings. These agents are called domain-independent agents and will be implemented in this project. The reason for these types will be explained more in the preference profiles section. Also, the agents in some cases need to consider discounting and reserving values so that they do not make a loss from the agreement. However, they are ignored for this project for research purpose.

### 2.2.2 Negotiation Protocol

The negotiation protocol using in the paper is known as alternating-offers protocol [8]. It is widely used in the negotiation due to its concept. In each round, the protocol lets an agent choose one action which is to create an offer or accept the offer that the opponent created. When the agent creates and sends a new offer to the opponent, the opponent then can choose one action like that agent after that. The offers are called counter-offers when the agent make an offer instead of accepting a competitor's offer. In this system, both agents will send offers and counter-offers alternatively in each round until they have a joint agreement, or the negotiation time run out. This protocol gives fairness between each agent so no one can have a better advantage.

### 2.2.3 Preference Profile

Each agent has its preference profile which is its preference information considering all possible offers in the negotiation domain. It ranks offers in the negotiation domain based on preference order. The preference profile between both agents can be similar or different depending on the negotiation scenario. The profile can be used to calculate the utility value of an offer from an evaluation technique. The higher utility value means the agent prefer that offer than lower utility offer. In other words, the agent will use its profile as a reference to find a satisfying offer during the negotiation. Considering its profile, the utility value of an offer depends on two factors. Two factors are the evaluation of attribute value in each issue and the importance of the issue. For the mechanism, each agent creates an offer using an attribute value in each issue. In scientific term, the issues $I$ and attribute values $a_j^{i_k}$ inside the domain $D$ are denoted as equation (1). Note that n is the number of all issues and m is the number of all attribute values in that issue.

$$I \epsilon D, I = \{i_1, i_2, \dots, i_n\} \; for \; each \; i_n = \{a_1^{i_n}, a_2^{i_n}, \dots, a_m^{i_n}\} \tag{1}$$

Each issue has different priority depending on the agent's preference. The negotiation in this scenario has a linear additive utility function which means the utility $u$ between each issue is independent and can be indicated as equation (2). Note that, the utility of issue is also known as issue weight.

$$u(i) \rightarrow [0,1], u(I) = \sum_{j=1}^{n} u(i_j) = \sum_{j=1}^{n} w_j = 1 \tag{2}$$

Where $u(i_j) = w_j$

The evaluation of attribute value can be any value within the range between zero and one. But the evaluation sum of all attribute values in one issue is also always one so:

$$u(a_j^i) \rightarrow [0,1], u\left(a_j^{i_h}\right) = \sum_{k=1}^{m} u\left(a_k^{i_h}\right) = 1 \; with \; i_h \; \epsilon \; I \tag{3}$$

The offer can be written in the form of:

$$o = \{a_a^{i_1}, a_b^{i_2}, \dots, a_m^{i_n}\} \; where \; a_m^{i_n} \; \epsilon \; i_n \; for \; each \; attribute \; value \; a \tag{4}$$

The utility value of an offer $o$ can be measured as:

$$u(o) = \sum_{j=1}^{n} w_j * u\left(a_k^{i_j}\right) \tag{5}$$

where $a_k^{i_j}$ is the attribute value of the issue j in the offer o.

The agent can use this equation in order to calculate and find a suitable offer to itself. The agent prefers an offer i over offer j when $u(o_i) > u(o_j)$ which can be denoted as $o_i \succcurlyeq o_j$. The highest utility offer of the agent also known as o* is the most preferred offer to that agent. The utility of all offers between both agents can be presented in the figure 4.

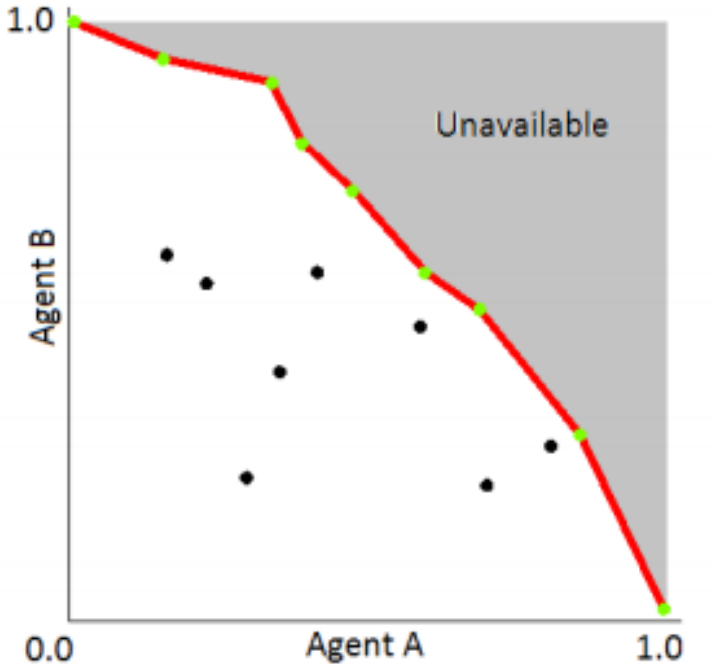



*Figure 4 Utility value of each offer between two agents [15]*

In the figure, the relationship between the two agents is shown using the utility value of all possible offers. As stated before, the utility value measures the desirable value to an agent. The dot in the graph represents the utility of an offer between two agents. The white area under the curve red line in the graph contains a list of all offers or feasible offers in relation to both agents' utility. The grey area over this red line does not consist of any offers as it is not possible to create such an offer. The red line in figure 4 depicts the Pareto frontier. The offers lie underneath this line are considered as Pareto optimal. It means that there is no offer that the utility of one agent increases without decreasing other agent utility. They also contain a Nash bargaining solution which maximizes the product of both agents' utility. Additionally, they meet a requirement of social welfare because they have high joint utility. With these properties, the agent normally wants to find an offer near this line to maximize its gain and increase the probability that other agent will accept the offer due to its high utility value of that offer. For this reason, the different types of negotiating agent are made to deal with different negotiation scenarios such as an example in figure 4.

## 2.3 Genetic algorithm

The genetic algorithm is one of the evolutionary algorithms which mimics natural evolution, a theory by Darwin [10]. The theory stated that a normal living thing has some resemblance to its parents since life creation due to the recombination of chromosomes. The structure of organisms in the population have minor change over time from the reproduction and the inheritability. The reason for this is to adapt in the harsh environment so they can increase the survival rate and produce more offspring. This algorithm is useful on doing an optimization task as it performs a heuristic search from all available actions [11][16]. This search is a technique using information from the task to evaluate the best possible path. The algorithm is also well-known using operators like reproduction process to find an optimal solution to pursue its goal. The noteworthy operators are selection, crossover and mutation [16]. The fundamental structure of this algorithm is explained in figure 5.

```
Initialize population
Evaluate chromosomes in the population
While new population is not full do
        Select chromosomes to reproduce depending on fitness value (selection)
        Apply genetic operators to create new offspring (crossover, mutation)
        Put offspring into new population and evaluate again using fitness function
```

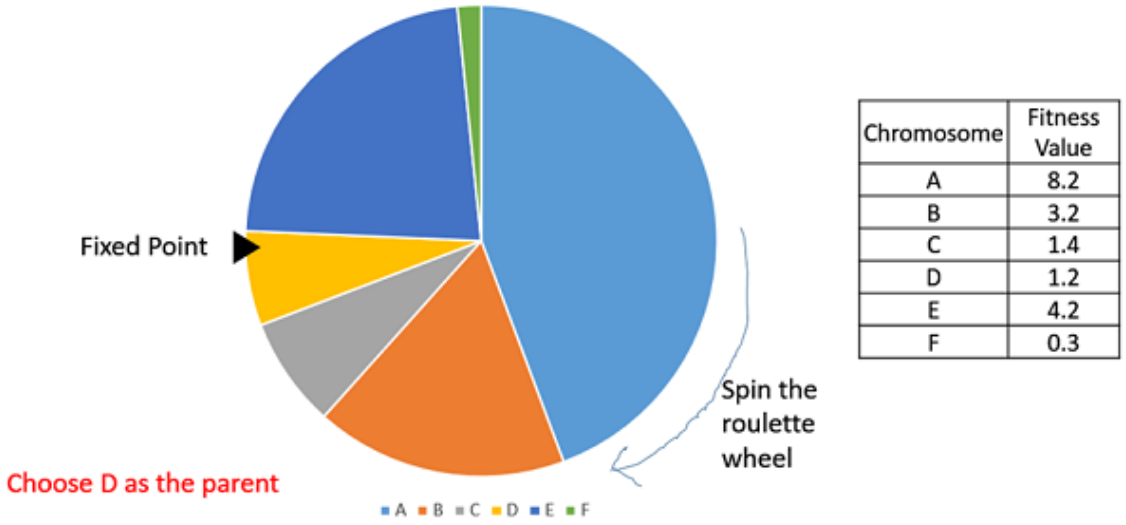
*Figure 5 Pseudocode of the genetic algorithm*

For this negotiation, the offers are created and served as chromosomes in the population to mimic the reproduction process. Each offer in the population is considered as a member or a candidate for discovering an ideal solution. Before that, each offer is evaluated using a fitness function. The fitness value from the fitness function implies how an offer is appropriate to use in negotiation by considering the environment variables. The fittest offer means that the offer has high probability to be accepted in the negotiation while being self-interest. The mechanism of operators can filter defective offers and evolve fitting offers in the new population. The main operators in the genetic algorithm are selection m, crossover and mutation.

### 2.3.1 Selection

Even though the selection is not directly helping create a successor, the operator is used to favour choosing offers in the population that have high fitness value evaluated by a fitness function. This operator process is to increase the overall fitness in the new population as it makes sure that the offers that have low fitness value have less probability to show comparing to other offers in the next generation. The candidate offers are put into a mating pool for doing generic operators. There are different types of selection depending on the objective.

- **Roulette wheel selection**




| Chromosome | Fitness Value |
|---|---|
| A | 8.2 |
| B | 3.2 |
| C | 1.4 |
| D | 1.2 |
| E | 4.2 |
| F | 0.3 |

*Figure 6 Roulette wheel selection*[1]

This selection is also known as fitness proportionate selection. The process uses the fitness value of each offer as the probability find a candidate offer in the population. This mechanic is similar to the roulette wheel game. The probability of each offer is defined as equation (6).

$$prob_i = \frac{fitness(i)}{\sum_j^n fitness(j)} \qquad (6)$$

Where n is a number of all offers in the population and fitness(i) is the fitness value of the offer i.

The mechanism means that the offers with high fitness value have more slice than low fit offers. In other words, the most fitting offer tend to get selected than the least fitting offer.

[1] https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_parent_selection.htm

- **Tournament Selection**

The mechanism of this selection is to select k number of the offers in the population randomly. Then, it runs a tournament to find one offer out of k offers which have the highest fitness. The chosen offer will be used to reproduce a new offer. Note that, the number k is the tournament size.

### 2.3.2 Crossover

The operator uses two offers that are chosen in the selection operator to create new offers. Based on natural reproduction, it uses portions of both offers to combine making new offers out of them. In this case, crossover uses the attribute values of each issue as the genes. Based on Holland's theory, successors or the new offers tend to have higher fitness than its predecessors [12][16]. the theory implies that small adequate portion of genes in a chromosome can act as a building-block to create even greater building-block. The variations of this operator are one-point crossover, two-point crossover and n-point crossover.

- **One-point crossover**

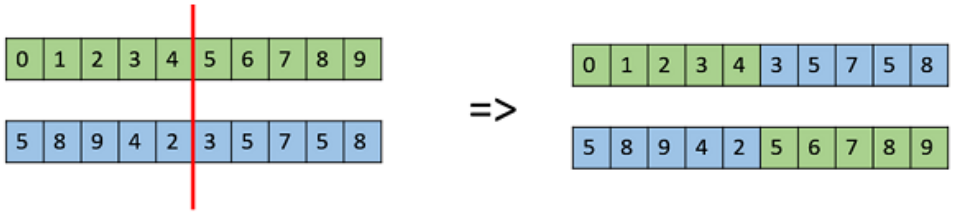



*Figure 7 One-point crossover*[2]

The operator chooses one position in the chromosome in random and swaps both agents' genes using that position. The position is also known as a crossover point.

[2] https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_crossover.htm

- **Two-point crossover**

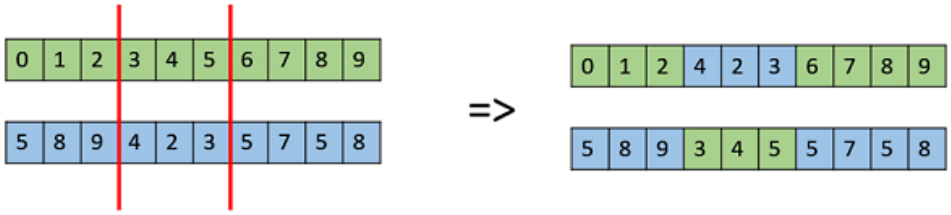



*Figure 8 Two-point crossover*[2]

The operator selects two crossover points in random and swaps both agents' genes between these points.

- **N-point crossover or uniform crossover**

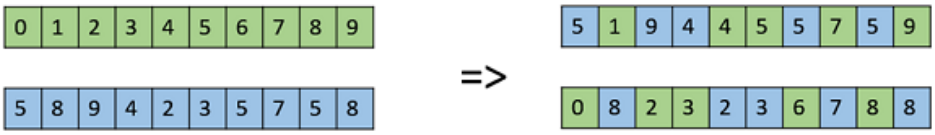



*Figure 9 Uniform crossover*[2]

Each gene in an offer has a low chance to be exchanged between both agents. The probability is independent between each gene.

### 2.3.3 Mutation

In the genetic algorithm, the mutation is used to increase the variability of the successors while it preserves some structure of predecessors. The reason is to avoid being struck at local optima as all members in the population can have defective genes in one position of a gene. The crossover operator cannot remove these defectives efficiently in this case, so the mutation is needed. There are also many approaches to do the mutation. The examples of these approaches are uniform mutation, non-uniform mutation and random resetting mutation.

- **Uniform mutation**

For this case, the operator always does a mutation in a random position. The operator makes that each chromosome has a low probability to change gene value. For this project, each issue in the offer has a low chance to be altered with other attribute value.

- **Non-uniform mutation**

The mechanism is similar to uniform mutation but the mutation rate decreases over time in order to reduce the chance to mutate to the defective values as the population progress.

- **Random resetting mutation**

In this mechanism, one or more issue is chosen, and each chosen issue changes their attribute value randomly.

There are still other mutation operators but most of them are used in more simplify values such as binary bits and independent values, unlike attribute values which are dependent on the issue in the negotiation.

## 2.4 Linear programming

Linear programming is another optimization technique. The concept of this technique is to represent a complex problem into a linear relationship and find an optimal solution while satisfying several constraints [13]. In other words, the aim is to find optimal values of the variables that maximize or minimize the objective function subject to various constraints. The objective function is one of the major components in linear programming. It is a linear function that is used to optimize and find the optimal solution. For linear programming, Other major components are decision variables which contain independent variables in the solution and constraints which are linear functions used as a set of rules to restrict the objective function. These components can be visualized like in figure 10.

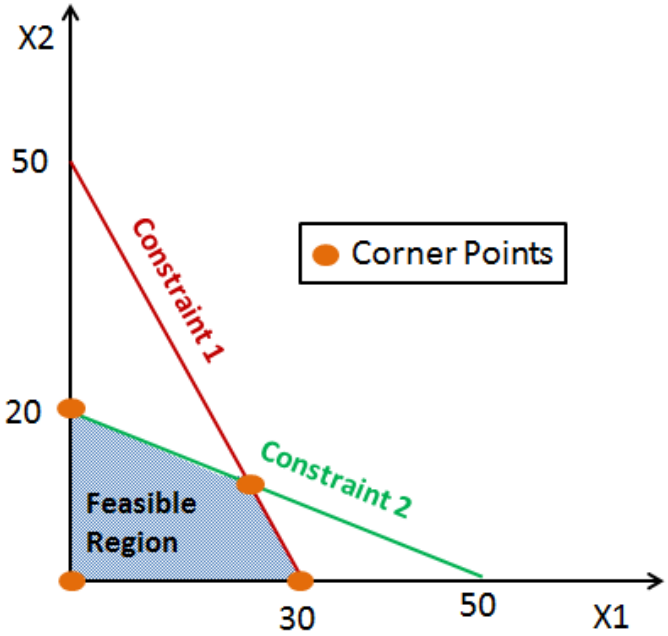



*Figure 10 Visualization of the linear programming*[3]

The blue area in the graph represents a feasible region which covers all possible decision variables that meet the requirement of all constraints. The area outside this region is called the infeasible region as it violates one of the constraints. The constraints are red and green lines in the figure. All possible values in the feasible region can be used to find the solution using the objective function.

[3] https://www.datacamp.com/community/tutorials/linear-programming-with-spreadsheets

There are also four assumptions about linear programming [13].

**1. Proportionality** – The contribution of any variable related to objective function and constraints is proportional to its variable due to the linear relationship. This means that if one value of the variable is changed, objective function and constraints related to that value is also changed. For example, the variable 4x is always double the value of 2x.

**2. Additivity** – The total value of the objective function and each constraint is the sum of the independent contribution of each variable. This means there is no dependent value. For example, x + y = z then z is not less than or more than x + y but is always equal to x + y.

**3. Divisibility** – The decision variable can be any real number within a specified range.

**4. Certainly** – All parameter values are known as certainly and always be constants. This means that solution will give little value in a solution given that some of the parameter values are wrong.

# Chapter 3 Challenges for building Negotiation Agent

The strategy of each agent depends on the programming and the environment of the negotiation. The environment, in this case, consists of its preference profile in negotiation, the opponent preference and the time pressure. In real-case negotiation, the traditional agent model has low performance due to these problems. The agent must have a deep understanding of these variables to improve its performance. The obstacles can be grouped into four major issues which as user uncertainty, opponent preference, time pressure and domain complexity.

- **User uncertainty** – The agent only knows partial information about user preference. The reason is because of the uncertainty of the user's desire. This problem makes the agent difficult to estimate the real utility of an offer and make an acceptable agreement to its user. In the normal scenario, the agent usually gets its preference in term of the preference list from a client. The list includes some offers in the negotiation domain order by preference. The agent should be able to use this knowledge to estimate the utility value of each issue and attribute value using the mathematical or statistical technique. The model for calculating agent utility is called user model.

- **Opponent preference** – In real negotiation, knowledge about the opponent is almost non-existent because of data privacy. With each agent aims to find an offer that gives maximum payoff for itself, it is difficult to find a satisfying offer for both parties if the preference conflicts with each other. The agent can use the offers received from the opponent to create a model for opponent preference. The number of interactions between agents in negotiation can secure the high performance of the agent [6]. This knowledge allows the agent to create an offer that fulfils the competitor desire. The model for calculating opponent preference is called the opponent model.

- **Time pressure** – Negotiation always has a time constraint. Both parties must find a jointly accepted agreement before the negotiation end due to deadline. Because the agent mostly gets no benefit from a failed negotiation, the agent must make a concession strategy to reach the consensus with the opponent while under time pressure. As such, concession strategies must be made in the negotiation for reaching an agreement. Note that, the concession means conceding some resources to get acceptance negotiation which usually used in economic. Nevertheless, the opponent model can be applied to reduce time pressure and improve the ability to make a favourable offer.

- **Domain Complexity** – In real-life, the negotiation has a complex negotiation space. The difficulty to find a high payoff offer for both agents can grow exponentially due to this reason and its computational limit. Thus, the agent should able to make an efficient strategy that reduces the computation stress as much as possible to increase the interactions between improving agent efficiency. The more cycles it gets from the negotiation, the more preference information comes out of the competitor agent.

By solving these problems, the successful agent model can be made.

# Chapter 4 Agent Model Design

For the design of the agent model, various techniques are used to solve the main issues of the negotiating agent. The agent model for this project aims to find offers near the Pareto frontier to maximize its payoff and social welfare. Moreover, the agent will try to find offers near the Nash bargaining solution. This reason is that it increases payoffs for both agents in the negotiation and is preferred from the opponent perspective due to payoff. The detail of the techniques used in the model is explained in the subsections below. Note that, most of the techniques are considered based on the time limit to implement the model within three months as there are a lot of backgrounds on each technique.

The overall structure of the agent in this project can be shown in figure 11. The agent pickups the information from the negotiation scenario to learn about its preference. The negotiation protocol acts as a rule throughout the negotiation. During agent turn, it performs an action using its components to justify which offer it should use. The main components of the negotiation model are user model, opponent model and offering strategy.

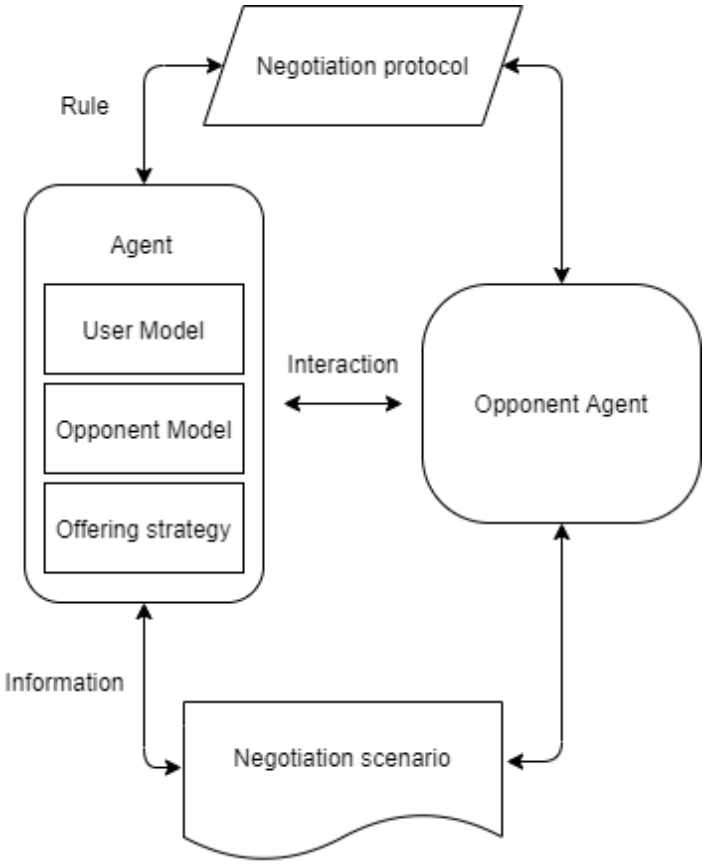



*Figure 11 Structure of the agent*

## 4.1 User Model

As the agent generally does not have incomplete information about its preference, the linear programming based on the Tsimopukis' paper is used as the user model to estimate agent utility [19]. Other techniques like elicitation of user preference by Baarleg [17] or gaussian approximation by Leahu [18] are also explored. The mechanism of elicitation is not compatible with genetic algorithm much because there is no need to elicit an offer as the genetic algorithm in this project always search for a suitable offer every round. Another reason is that there is also elicit cost which is the negative cost for bothering the user. For the definition, elicitation means that sending offer information to the user to receive the preference knowledge and update the user model. Nevertheless, it is still an attractive alternative for updating the user model while negotiating. The Gaussian model is similar to the linear programming that has an assumption about normal distribution. The model concept can deal a wide range of negotiation scenarios and is better than the linear model. However, linear programming is chosen because it can be easily implemented within three months. Note that, in the development platform, the agent also has a default model for the user model which is a very simple heuristic search. However, it tends to overestimate the utility of the agent due to its simple estimation, so it is not used in the project.

The linear programming technique transforms a list of offers ordered and received by a client into a linear problem to approximate the utility value of each issue and attribute value. The technique uses the utility value equation which is the equation (5) to make a linear function. Because the technique cannot create the linear function with the multiplication of two parameters which are attribute value and issue weight, new values are made using the combination of these two values. The new equations for these values are:

$$u(o) = \sum_{j=1}^{n} v_k^j \tag{7}$$

$$with\ v_k^j \rightarrow [0,1]\ , v_k^j = w_j * u\left(a_k^{i_j}\right) \tag{8}$$

The set of new variables for linear programming is:

$$V = \{v_1^1, v_2^1, \dots, v_{m_1}^1, v_1^2, \dots, v_{m_2}^2, v_3^1, \dots, v_{m_3}^3, \dots, v_{m_n}^n\} \tag{9}$$

Where $o$ is an offer, $w_j$ is issue weight, $v_k^j$ is an attribute value of issue j and $v_{m_n}^n$ is the issue n combined with last attribute value $m_n$ of the issue n.

The technique uses the fact about the list that the client gives for their preference. The fact is that the offers are ordered in term of user preference. This means that a high-ranked offer always has higher utility than a low-ranked offer. Using the pairwise comparisons between two offers in the list $(o, o') \in L$ and the equation from (7), it can be denoted that:

$$u(o) - u(o') = \sum_{j=1}^{n} \left(v_m^j - v_{m'}^j\right) \geq 0 \; with \; v_m^j \, , v_{m'}^j \in V \tag{10}$$

Where $v_m^j$ is a value for an offer $o$ and $v_{m'}^j$ is a value for an offer $o'$ and the offer $o$ is higher than the offer $o'$ in term of ranking.

The short term for this equation is:

$$\Delta u_{o,o'} = \sum_{j=1}^{n} \left(v_m^j - v_{m'}^j\right) \; with \; \Delta u_{o,o'} \geq 0 \tag{11}$$

Because the equation is inequalities, the slack variables $z$ are added to transform functions into equality. The number of slack variables is equal to the number of pairwise comparison in the offers list that the agent gets. The linear program can be created as:

$$\text{Minimize:} \quad S = \sum_{(o,o') \in L} z \tag{12}$$

With the constraints:

$$z_{o,o'} + \Delta u_{o,o'} \geq 0, \tag{13}$$

$$z_{o,o'} \geq 0 \; for \; (o, o') \in L, \tag{14}$$

$$v_m^j \geq 0 \; for \; all \; v \; in \; V \tag{15}$$

Note that, the highest offer $o^*$ in the list is used to as a constraint in the linear program. This reason is because the program will typically solve the solution with all variables equal to zero.

$$u(o^*) = \sum_{j=1}^{n} v_{m^*}^j = 1 \tag{16}$$

The summary of the linear programming used to estimate the utility is:

***Table 1: Linear Programming summary***

| Objective function | Decision Variable | Constraints |
|---|---|---|
| (12) | (9) $\cup \; z_{o,o'} \mid (o, o') \in L$ | (13), (14), (15), (16) |

## 4.2 Opponent Model

For finding opponent preference, Jonny black model from ANAC 2015 is used as the opponent model [20]. Before that, the Bayesian opponent modelling is an alternative technique that can be used for finding opponent utility [21][22]. The technique is a famous approach to estimate the opponent utility based on the probability of each hypothesis. This technique holds hypothesis space consists of all hypothesis as each hypothesis represents one possible opponent preference based on the negotiation domain. The mechanism of the technique is that it tries to identify a most likely hypothesis using offers that opponent sends. The opponent usually sends offers based on their preference on self-interest for maximizing its payoff. Given this reasoning, the model can give a better estimation of opponent utility. Even so, the disadvantage is that the Bayesian model used a lot of computation cost. This cost is due to the calculation of each hypothesis in the hypothesis space [22]. If the negotiation domain is large with multiple issues, the computation complexity grows exponentially. Another problem is to define the hypothesis space for this model. It is difficult to implement for the general-purpose agent. Nevertheless, the agent model in this project works mainly on the evaluation mechanism of the genetic algorithm so the opponent model does not contribute much.

The Jonny Black Opponent model is based on simple mathematical technique and some assumptions. The model uses a record of all offers that the opponent sends in the negotiation. The assumptions of this model are:

- The most preferred attribute value appears more frequently than other values in the offers that the opponent sends.
- The opponent is less likely to change an attribute value in the most important issue as it can diminish its payoff greatly.

For the mechanism, the preference order of the attribute value is based on the frequency of the attribute value in all received offers. The attribute value with the highest frequency means that is the most preferred option in that issue from the assumption of the opponent model. For example, assume there is an issue called ‘colour’ with three attribute values in the negotiation and all three attribute values are red, blue, green. The opponent sends three offers which have blue, blue and red attribute values respectively. This means the preference order isblue > red > green based on the assumption. The equation of estimating the utility value for each attribute value in the issue is:

$$u\left(a_k^{i_j}\right) = \frac{l - m_a + 1}{l} \tag{17}$$

Where $l$ is the number of all possible attribute in that issue, $m_a$ is the rank of attribute value ordered by frequency of all attribute values. Note that, the estimation of the utility value for the most frequent value is 1 while the least frequent value is $\frac{1}{l}$

The frequency of the attribute value can be used to find the weight issue from the assumption. Based on one assumption that the opponent unlikely to change attribute value on the most important issue, the model use Gini Index which is impurity measure to estimate the issue weight. Using equation (1), the equation of this calculation is:

$$\hat{w_j} = \sum_{a_k^{i_j} \in i_j} (\frac{f_a}{r})^2 \quad (18)$$

Where $f_a$ is the frequency of an attribute value and $r$ is the total number of received offers.

As the weight is not normalized, the normalization of the weight from previous equation is:

$$w_j = \frac{\hat{w_j}}{\sum_{i_h \in I} \hat{w_h}} \quad (19)$$

With these equations, the agent can estimate the opponent utility. Note that, during the negotiation, the opponent model always updates the opponent utility after the agent gets the offer form the opponent all the time.

## 4.3 Offering Strategy

Considering the time pressure and domain complexity, the genetic algorithm is used as a strategy to find a suitable offer during the negotiation. The algorithm is implemented using Lau's works as references [23][24]. In the paper, it uses fitness function to evaluate all offers in the population to find the most appropriate offer using the observation on the negotiation time and the similarity of the received offer. The similarity measure is used in this algorithm to search offers near the latest received opponent's offer for optimizing its payoff and reaching the mutual agreement. Nonetheless, it improves on the concession strategy using the genetic algorithm without considering the opponent utility of an offer. As such, the project will use the opponent model as an extension to further improve the agent model. The small adjustments are also made during the implementation and testing to be more compatible with the negotiation scenarios in the experiment.

For the strategy, the agent will choose one of two actions based on the received offer if there is one during the negotiation. Otherwise, the agent will always send the highest utility offer to the agent. The actions that the agent can perform are to accept an opponent's offer or send a counter-offer to the opponent.

### 4.3.1 Offer Acceptance Strategy

For criteria on accepting the opponent's offer, the agent will accept opponent's offer given its offer utility is higher than all proposed offers by its agent. The second condition is that a received offer is one of the already proposed offers. This reason is that offer utility is higher than the offer that agent will propose in the next round based on concession behaviour. To put it another way, all already proposed offers are always more preferred than offers that do still not propose in the negotiation. Also, the agent will accept an offer if the utility of this offer is higher than the utility of a counter-offer it proposes.

### 4.3.2 Counter-Offer Proposal Strategy

If the agent rejects the opponent's offer, it will perform a genetic algorithm to find a suitable offer on each round. Before that, the fitness function needs to be discussed first as the algorithm uses this function as a measurement to search for appropriate offers. The fitness function for this agent considers four main factors which are the time pressure, the utility of an offer, the similarity of the last opponent's offer and the Nash bargain solution.

**1. Time pressure** – Time pressure is an important variable for calculating the concession. During the negotiation, the agents are more presumably to concede as time goes by because they want to find a mutually accepted offer. The concession between each agent can be different depending on the agent strategy. The agents that eager to concede quickly to find an agreement are called Conceder agents while the agents that like to hold their self-interest are called Boulware agents. The time pressure function is made to create an agent's attitude regarding the negotiation time.

$$TP(t) = 1 - (\frac{\min(t, t_d)}{t_d})^{\frac{1}{beta}} \quad (20)$$

Where t is a current time of the negotiation and $t_d$ is the deadline with the interval of [0,1] and $beta$ is an eagerness value of the agent towards the time.

The agent can adjust the value of $beta$ to change its behaviour in the negotiation. The agent is Boulware when the $beta$ is valued between zero and one is Boulware agent while the agent is Conceder when $beta$ is valued more than one.

**2. Offer Utility** – Concerning the offer utility, the agent uses the offer that gives the highest payoff to measure the utility of candidate offers. Because the maximum utility value that agent can get is different in various negotiation scenarios.

$$u^{norm}(o) = \frac{u(o)}{u(o_{max})} \quad (21)$$

**3. The similarity of opponent's offer** – The agent uses weight Euclidean distance for measuring the similarity between the candidate offers and the opponent's offer. The opponent's offer, in this case, is the latest offer the opponent sends to the agent. The Euclidean distance is the proximity measurement normally used to find a distance between two points in the graph. As the attribute values are all discrete values, the agent uses the user model to estimate the utility value of attribute values for finding the distance. Note that the weight issues are also used as the priority measure for this function.

$$wED(o,o') = \sqrt[2]{\sum_{j}^{n} w_j (u\left(a_k^{i_j}\right) - u\left(a_{k'}^{i_j}\right))^2} \quad (22)$$

Where $a_k^{i_j}$ is the selected attribute value of a candidate offer and $a_{k'}^{i_j}$ is the selected attribute value of the latest received opponent's offer.

**4. Nash bargaining solution** – For this project, the concept of a Nash bargaining solution is applied to find offers near Pareto frontier. As mentioned before, it maximizes both agents' payoff which makes the opponent more likely to accept the offer. The opponent model and user model are used in this function to evaluate the candidate offers. Note that, there is no disagreement value in this project. This value means the minimum value that agent is willing accept the offer.

$$NashPoint(o) = u(o) * u'(o) \quad (23)$$

Where $u'(o)$ is the opponent utility estimated by opponent model.

Using equation (20), (21), (22) and (23), the fitness function for this agent is:

$$fitness(o) = (alpha * TP(t)) * u^{norm}(o) + \left(1 - alpha * TP(t)\right) * \left(1 - wED(o,o')\right) + NashPoint(o) \quad (24)$$

Where the variable $alpha$ is used to control agent attitude during the negotiation. The value can range from being charitable ($alpha = 0$) to being self-interest ($alpha = 1$). With this fitness function, the agent will search offers based on these factors.

As mentioned before, the genetic algorithm uses offers in the preference profile list as a population for creating preferred offers. Then, the agent will perform operators such as selection, crossover and mutation after evaluating offers like figure 5 however there is additional method apply in this agent. The detail of the genetic algorithm for this agent is outlined in figure 12.

Initialize population $P_0$ with list of offers ordered by preference which has N size and contains $o_{max}$

While negotiation is not end and the agent want to send a counter-offer.

Evaluate all offers in $P_i$ with fitness function

$P_{i+1}$ = Result of elitism method on $P_i$

$MP$ = Result of selection method on $P_i$

$List1$ = Result of crossover method on $MP$

$List2$ = Result of mutation method on $List1$

$P_{i+1} = P_{i+1} \cup List2 \cup List1 \cup MP$

Evaluate all mutated offers in $P_{i+1}$ with fitness function

*Figure 12 Pseudocode of the agent's genetic algorithm*

From the pseudocode, the genetic algorithm uses offers in the user preference list as a population $P_0$. The population has limited N size and $o_{max}$ is an offer that gives maximum payoff. For each round that the agent rejects the opponent's offer, all offers in the population are evaluated using fitness function which is equation (24). Then, the algorithm performs elitism method which chooses n% most fitting offers and put it in the new population $P_{i+1}$ directly. The reason is to preserves the suitable offers and prevent them to being mutated from the generic operators. Next, selection function is performed and put candidate offers in the mating pool $MP$. The selection method that is used in this agent is tournament selection with a tournament size of two. The reason for using tournament selection is that it surpasses roulette wheel selection according to Zhong's paper [25]. After that, Two-point crossover is performed for the agent to find candidate offers. The function is executed until the mating pool is full. Note that, crossover rate is used as the probability to perform crossover offers. Furthermore, uniform mutation is used for this project. Later, offers from results in the mutation, crossover and selection operations are put in the new population $P_{i+1}$ in corresponding order until the new population is full. Offers in the population are always unique offers. Finally, the offers that are changed from the operators are re-evaluated again for deciding a counter-offer. The agent will randomly choose one of the top three offers with the highest fitness value to show the its preference to the opponent because the agent likes to send the same offer almost all the time. With a decided offer, the agent will check with latest opponent's offer again whether it is better to accept the offer based on the utility. The offer is rejected if the utility than the utility of counter-offer.

# Chapter 5 Implementation

As stated before, the genius platform is used to develop a negotiating agent [4][15]. It can simulate negotiation sessions with other agents and provide a visualization of the result for analysing the agent model. The platform also has a library containing agent models and negotiation scenarios from ANAC tournament which can use to calculate agent performance. As the programming, the agent itself is all written in Java language. Another external tool uses in this project is SCPsolver[4] which is an application that can perform linear programming for the user model in the negotiation agent.

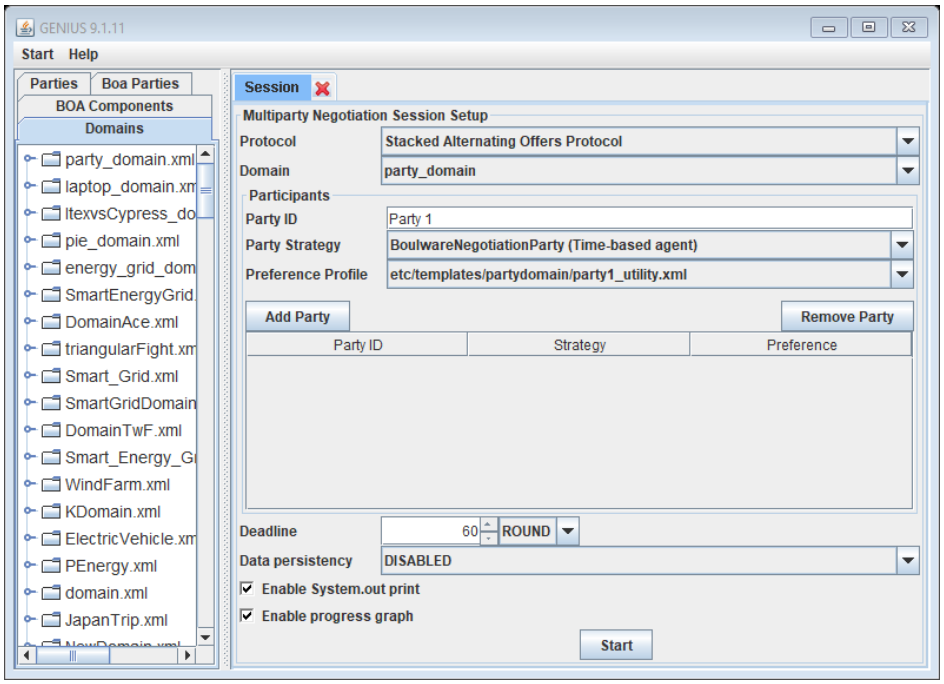



*Figure 13 Genius platform user interface*

[4] http://scpsolver.org/

# Chapter 6 Experiment and Result

The experiment is carried out to test the performance of the agent model. This experiment aims to evaluate the agent model whether it can perform in different negotiation scenarios efficiently. As mentioned before, the negotiation scenarios from the genius platform and agent models from the ANAC tournament are used to evaluate the agent performance [26].

As the preference profiles and negotiation domain in a negotiation scenario can greatly affect the negotiation outcome, four different negotiation scenarios are chosen for this experiment in term of negotiation size. Each negotiation scenario contains a different number of issues and attribute values. In other words, the number of all possible offers on each scenario is different ranging from small to a large number. Note that, all attribute values are discrete. The details of negotiation scenarios are in the table below.

***Table 2: Negotiation Domains for the Experiment***

| Domain ID | Domain Name | Number of Issue | Size |
|---|---|---|---|
| 1 | laptop_domain | 3 | 27 |
| 2 | SmartEnergyGrid | 4 | 625 |
| 3 | party_domain | 6 | 3072 |
| 4 | WindFarm | 7 | 7200 |

For other agent models in the experiment, the agents that are in ANAC finalists are used as the opponent's agent to evaluate the implemented agent. They are selected from the tournaments between the year 2015 and year 2017. These agents from these tournaments are used to focus on learning opponent model and conceding for favouring their payoff but they are not made for user uncertainty problem. The reason is that this problem is new in the recent year and there is no agent with similar circumstances in the genius platform. As such, other agents' performance can be worse compared to the ANAC tournament. Nevertheless, it can be used to evaluate general performance such as the attitude of the agent towards an opponent and the offering strategy.

***Table 3: Opponent Agent Models for the experiment***

| Agent Model | University | ANAC year |
|---|---|---|
| PonPokoAgent | Tokyo University of Agriculture and Technology, Japan | 2017[5] |
| Caduceus | Ozyegin University, Turkey | 2016[6] |
| ParsAgent | University of Isfahan, Iran | 2015[7] |

For the negotiation setting, the negotiation protocol is an alternating-offers protocol. The experiment will run a bilateral multi-issue negotiation with only discrete attribute values. Each agent will negotiate with other agents in each domain within one minute. Also, each agent will use both preference profiles in the domain to evaluate its performance. Moreover, each agent only knows about 10% of the user preference in each domain.

The different measures are also used to evaluate the quality of the accepted offers from negotiation sessions. The evaluation measures are the average utility of final offer, the average distance to Pareto frontier, distance to a Nash bargaining solution and the joint utility of both agents.

[5] http://web.tuat.ac.jp/~katfuji/ANAC2017/
[6] http://web.tuat.ac.jp/~katfuji/ANAC2016/
[7] http://web.tuat.ac.jp/~katfuji/ANAC2015/

## 6.1 Result

The result from each domain is presented in each table. Note that, the agent that is implemented in this project is called GAagent. The parameters that is used for this agent are: population size = 100, mating pool size = 80, elitism rate = 10%, crossover rate = 60%, mutation rate = 5%, alpha = 0.5 and beta = 0.8.

***Table 4: Result of 'laptop_domain' Domain***

| | Average utility | Distance to Pareto frontier | Distance to Nash | Joint utility |
|---|---|---|---|---|
| **GAagent** | 0.81895 | 0.06307 | 0.25187 | 1.67417 |
| PonPokoAgent | 0.88142 | 0.03154 | 0.22804 | 1.7372 |
| Caduceus | 0.88142 | 0.03154 | 0.22804 | 1.7372 |
| ParsAgent | **0.88512** | **0** | **0.1856** | **1.78523** |

***Table 5: Result of 'SmartEnergyGrid' Domain***

| | Average utility | Distance to Pareto frontier | Distance to Nash | Joint utility |
|---|---|---|---|---|
| **GAagent** | 0.61266 | **0.19112** | **0.31388** | **1.25566** |
| PonPokoAgent | **0.67604** | 0.20066 | 0.38468 | 1.23084 |
| Caduceus | 0.54716 | 0.22765 | 0.35440 | 1.20527 |
| ParsAgent | 0.37543 | 0.54335 | 0.65674 | 0.73080 |

***Table 6: Result of 'party_domain' Domain***

| | Average utility | Distance to Pareto frontier | Distance to Nash | Joint utility |
|---|---|---|---|---|
| **GAagent** | **0.84001** | **0.05691** | **0.18384** | **1.55609** |
| PonPokoAgent | 0.75662 | 0.11134 | 0.29095 | 1.43081 |
| Caduceus | 0.52789 | 0.22316 | 0.42443 | 1.24784 |
| ParsAgent | 0.64211 | 0.23081 | 0.32437 | 1.29853 |

***Table 7: Result of 'WindFarm' Domain***

| | Average utility | Distance to Pareto frontier | Distance to Nash | Joint utility |
|---|---|---|---|---|
| **GAagent** | **0.88011** | **0.04578** | **0.21509** | **1.60067** |
| PonPokoAgent | 0.50411 | 0.41950 | 0.61900 | 1.01407 |
| Caduceus | 0.32252 | 0.56807 | 0.76783 | 0.79690 |
| ParsAgent | 0.40534 | 0.55856 | 0.75424 | 0.81254 |

The highlighted value describes the value that has the highest quality of a metric in that domain. These tables can also be illustrated into graphs with each metric as shown below.

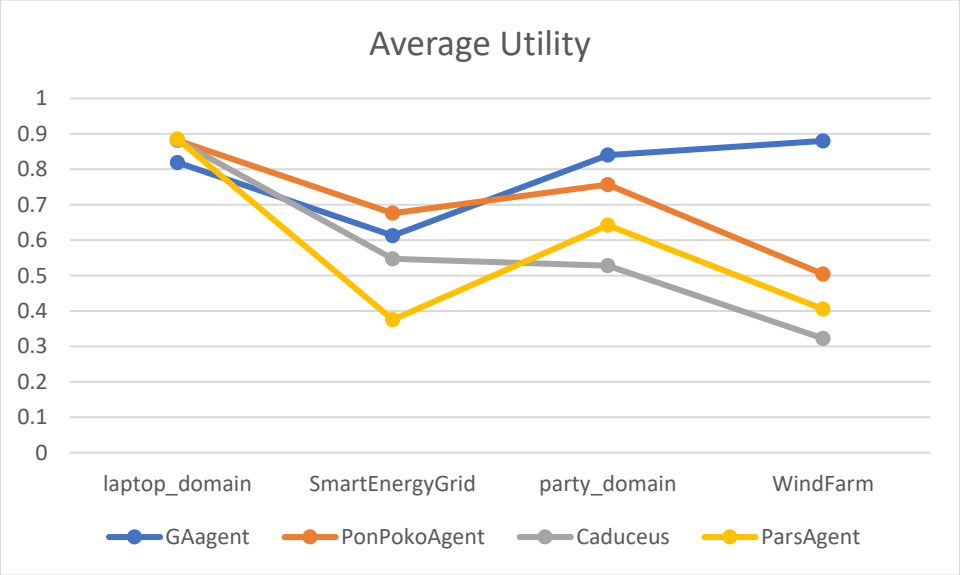



*Figure 14 Average utility of each agent*

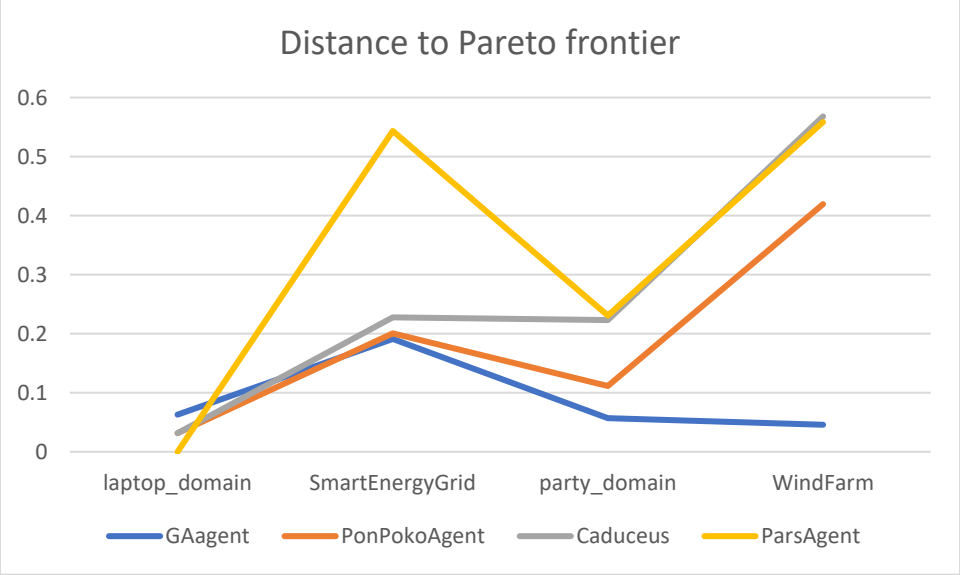



*Figure 15 Distance to Pareto frontier of each agent*

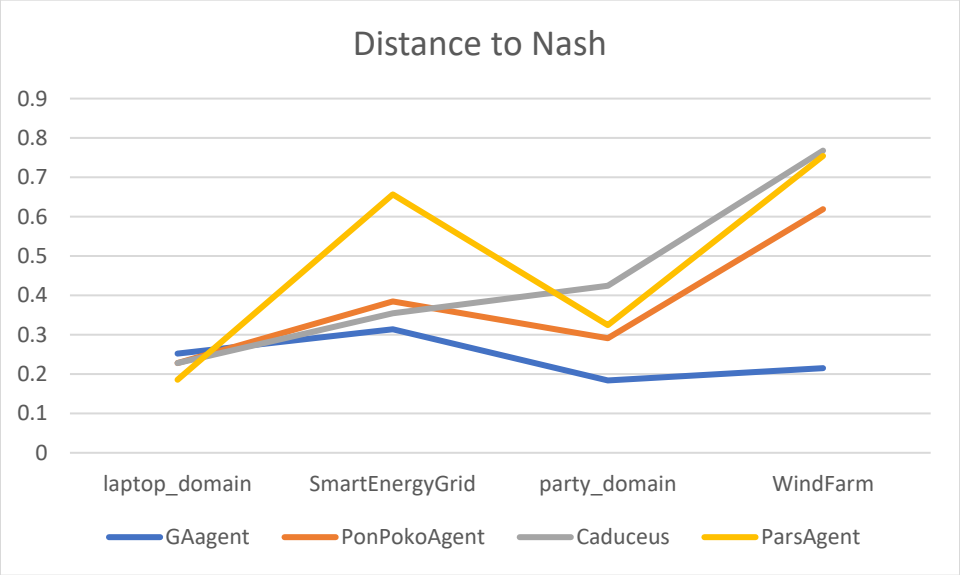



*Figure 16 Distance to Nash bargaining solution of each agent*

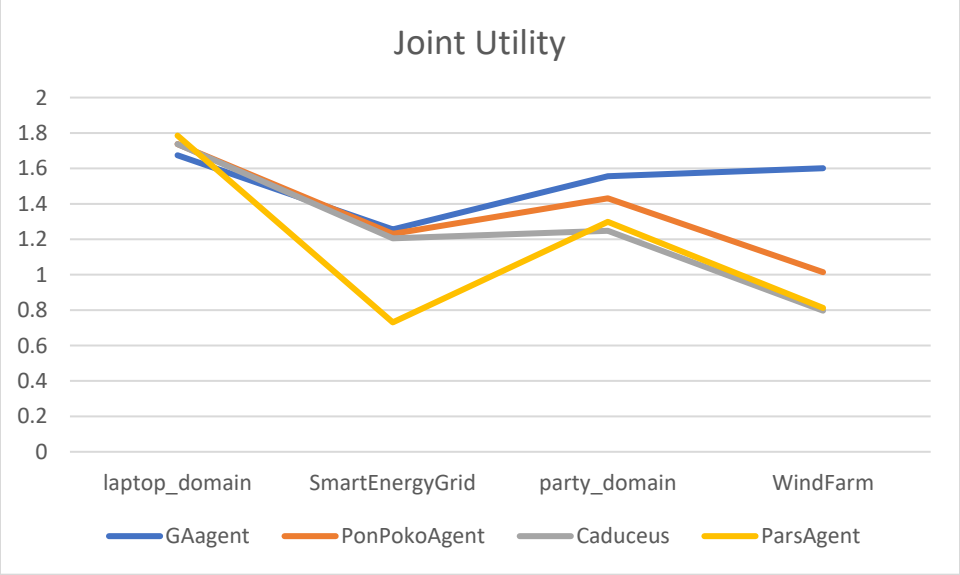



*Figure 17 Joint utility of each agent*

## 6.2 Evaluation

In general, the agent performs well in the large domains as the overall performance exceed other agents' performance with the highest average utility in the largest negotiation scenario according to figure 14. From the figure 15, the agent has a very low distance to Pareto frontier and Nash bargaining solution when comparing to other agents in the large domain. This result means that the agent itself try to search and accept the offers near Pareto frontier as much as possible. Moreover, the result shows that the agent likes to find an offer with the high joint utility. The reason for this is because PonPokoAgent has higher average individual utility but the joint utility of its agent is lower than GAagent. Other agents' performance is considerably low on the larger domains (party_domain and WindFarm). The ParsAgent performance on the SmartEnergyGrid domain is especially bad according to these figures. This result can be due to the negotiation space of the domain as all agent performance in this domain is very low comparing to other domains.

On the other hand, the overall agent performance on the smaller domains (laptop_domain and SmartEnergyGrid) has bad performance when comparing to opponent agents. The distance to Nash bargaining solution and distance to Pareto frontier is large while other agents can find an agreement near the Pareto frontier. Especially, the agent on the laptop domain has the worst performance out of all agents. ParsAgent dominates all aspects of the performance in this domain. This result can be shown that the concession of the agent has substantially bad performance.

These results can be visualized in the negotiation graph like in figure 4 by using the genius program. The reason for this is to give more insights on how each agent performs during the negotiation. Some examples of the negotiation are shown to explain the result.

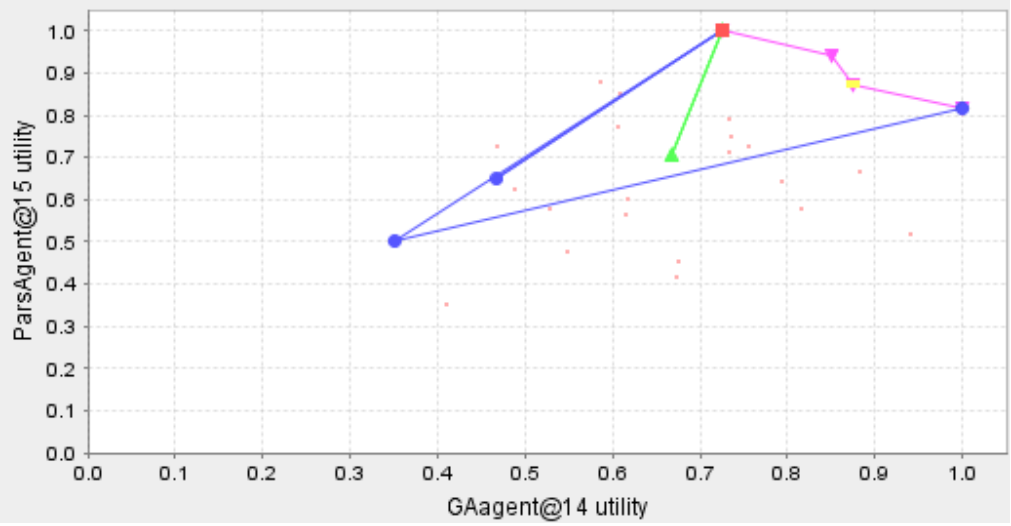



*Figure 18 GAagent(blue) vs. ParsAgent(green) in laptop domain*

Figure 18 represents a negotiation process between GAagent and ParsAgent in laptop domain. The line represents the path that the agent sends the offer and the red dot depict the mutually accepted offer in the negotiation. The pink line represents the Pareto frontier. The blue colour is the agent implemented in this project while the green colour is ParsAgent from the tournament. It can be seen a blue line in the figure that the agent sent offers that are unacceptable to both agents in term of utility value. The reason can be because the agent always randomly chooses the top three offers with high fitness value as the candidate offer based on the strategy. With small domain size, the offers that agent sends in each round can have a huge difference in term of utility value due to the low population of offers. This result allows the opponent to wait and accept the conceded offer.

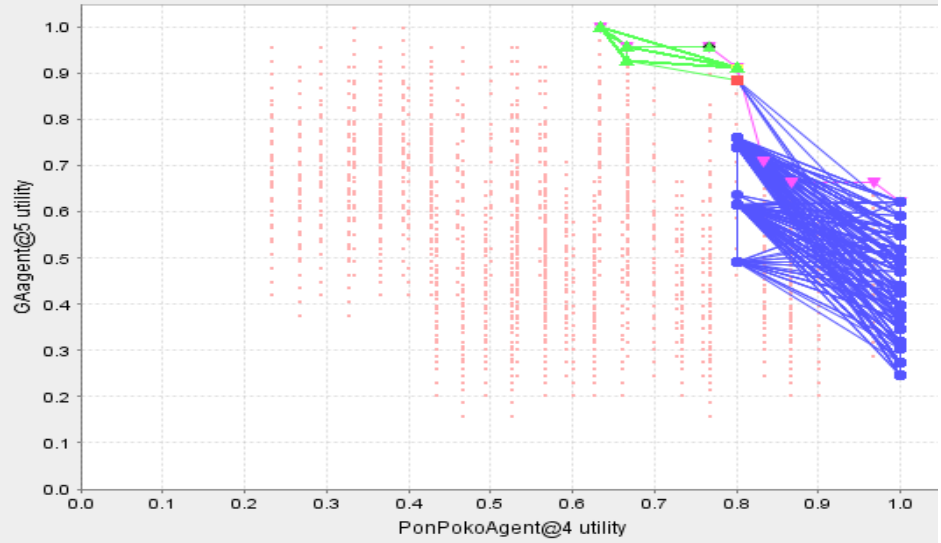



*Figure 19 GAagent(green) vs. PonPokoAgent(blue) in WindFarm domain*

Figure 19 illustrates the negotiation between GAagent and PonPokoAgent in WindFarm domain. The green colour is GAagent while blue colour is PonPokoAgent. From the graph, the agent tends to send offers that are near Pareto frontier by looking at the green line. This behaviour means the genetic algorithm performs very well in a large domain contrary to the small domain in figure 18. From the algorithm, the fitness function is capable to find offers near Pareto frontier. The negotiation is also ended with an agreement on this line. This agreement is also near a Nash bargaining solution which is a black dot on the Pareto frontier. The figure shows that the agent can find this offer depicting by the green line over the black dot. From these pieces of knowledge, it can be concluded that the agent has a good understanding of the negotiation domain in the large domain.

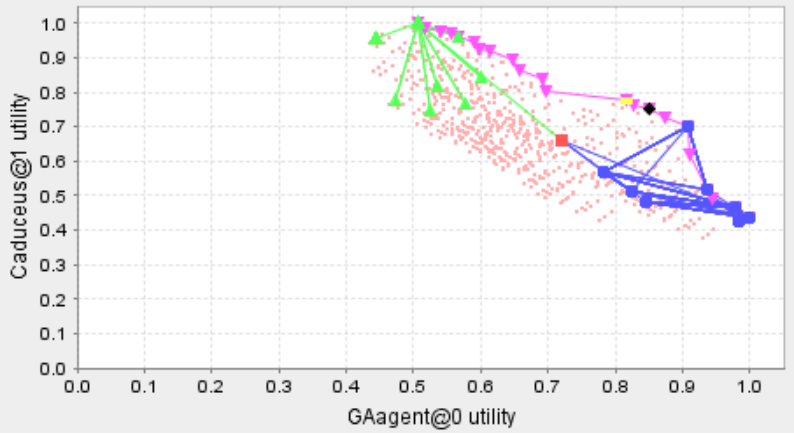



*Figure 20 GAagent(blue) vs. Caduceus(green) in SmartEnergyGrid domain*

Figure 20 shows the negotiation between GAagent and Caduceus in SmartEnergyGrid domain. The blue colour is GAagent while green colour is PonPokoAgent. The graph reveals high competitive trait of the domain because all possible offers depicting by red dots are bundled together like a downward line. As a result, the overall performance of all agents is lower than other domain. The agent in the figure can find the agreement in the middle of this line. The result shows the agent struggles to find a solution as the agreement is very far away from the Pareto frontier and the Nash bargaining solution. It can mean that the agent performance is low on the competitive environment.

With the analysis from the experiment, some general summaries of the agent can be explained in term of advantages and disadvantages.

For the advantages, the agent performance is high in the large domain. The genetic algorithm can search for offers that are near Pareto frontier and are also favourable to its agent. It shows a good understanding of the negotiation domain. From the result in table 2, the agent also attempts to find high joint utility offer as the average individual utility is lower than opponents.

On the flip side, the agent does not work well in the small domain as it is mainly because of the offering strategy. Based on the experiment, the agent has the lowest performance in the laptop domain. The negotiation from figure 18 and the result from table 2 show bad performance in term of concession when comparing to other agents. Moreover, the agent also has low performance in the competitive environment from the figure 20.

# Chapter 7 Conclusion

The automated negotiating agent is implemented in this project. The genetic algorithm is used as the main component in this agent. Given the genetic algorithm property, its function can be capable to find a suitable agreement from the negotiation. The details of other agent methods and alternative approaches are also outlined to give the reasons for the design of the agent. The implementation is done with the design that can be done within the time limit. The experiment is made to evaluate the agent using negotiation domains and other agent models. This experiment gives me insights into the components in the negotiating agent and the mechanism of the genetic algorithm in the real application. The experiment is also presented a better understanding of the concession strategy for the negotiation agent. The result of the experiment shows that the genetic algorithm can be used in the negotiation efficiently. However, the concession strategy of the agent is poor when comparing to other agent models and looking at negotiation progress.

## 7.1 Future Work

For the further adjustment of this project, the agent needs more room of improvement on concession strategy as it does not perform the concession well in the small domain size. The agent should be to learn about the concession rate from the opponent to apply an appropriate strategy. This rate can be calculated by checking the change of opponent utility on each round or estimating the average of opponent utility.

For the genetic algorithm, the generic parameters can be adjusted based on the average of fitness value in the population for reducing mutated offers. The reason is for preserving optimal offers. This change can increase offers that are near Pareto frontier and can give the agent more suitable choices during negotiation.

As the agent needs to be implemented and evaluated within the time limit, the user model and opponent model are chosen from this constraint. As suggested before, the Bayesian model can be implemented as an opponent model for better estimation of the opponent utility. The Gaussian model for the user model is also an interesting model to implement.

## 7.2 Project Management

The table below presents the planning schedule for this project. Each process of the project is listed and ordered in this table within thirteen weeks. The number 1-13 represents each week. In general, the project took longer than I am expected due to the difficulties in the first two phases and mainly the current situation this year. The detail of some challenging progress also is outlined.

***Table 8: Gantt Chart***

| # | Task | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.1 | Background research on negotiation agent and genetic algorithm | | | | | | | | | | | | | |
| 1.2 | Review of other techniques used in the negotiating model | | | | | | | | | | | | | |
| 2.1 | Design of agent model | | | | | | | | | | | | | |
| 2.2 | Simple implementation | | | | | | | | | | | | | |
| 3.1 | Genetic algorithm implementation | | | | | | | | | | | | | |
| 3.2 | Optimization | | | | | | | | | | | | | |
| 4 | Performance evaluation | | | | | | | | | | | | | |
| 5 | Report Writing | | | | | | | | | | | | | |

In phase 1, the research review of literature took three weeks. Most of the time is used to learn about the introduction and the concept of other techniques because the genetic algorithm cannot solve all the issues of the negotiating agent. The genetic algorithm can mostly standalone by itself, but it will worsen the agent performance. Other techniques must be used in the combination with the algorithm to create the agent model. The main challenge in this part is to understand the fundamental of the techniques and apply them into the agent model. Another challenge is to find the techniques that can be implemented within the time limit of the project.

For phase 2, the agent model is designed by using techniques from the literature review for one week. The implementation takes four weeks more than it should be due to the change of the agent design because of the low performance of a technique from the implementation. The genetic algorithm is implemented in the last two weeks of the phase 2 to test compatibility of the techniques.

In phase 3, implementation of the genetic algorithm is done for this project. The structure of the algorithm used a lot of computation cost, so refinement of the programming is also done in this part. As there are many parameter variables using in this algorithm, several numbers of tests are needed to optimize the model.

On phase 4, evaluation of the agent is done using negotiation scenarios and agent models from ANAC tournaments. The result performance of the agent is recorded to receive a deeper understanding of the agent model and its techniques.

# Reference


[1] Bosse, T. and Jonker, C.M., 2005, July. Human vs. computer behavior in multi-issue negotiation. In *Rational, Robust, and Secure Negotiation Mechanisms in Multi-Agent Systems (RRS'05)* (pp. 11-24). IEEE.

[2] Ahmad, I. and Dhodhi, M.K., 1995. Task assignment using a problem-space genetic algorithm. *Concurrency: Practice and Experience*, *7*(5), pp.411-428.

[3] Kraus, S., 2001, July. Automated negotiation and decision making in multiagent environments. In *ECCAI Advanced Course on Artificial Intelligence* (pp. 150-172). Springer, Berlin, Heidelberg.

[4] Lin, R., Kraus, S., Baarslag, T., Tykhonov, D., Hindriks, K. and Jonker, C.M., 2014. Genius: An integrated environment for supporting the design of generic automated negotiators. *Computational Intelligence*, *30*(1), pp.48-70.

[5] Winikoff, M., Padgham, L. and Harland, J., 2001, December. Simplifying the development of intelligent agents. In *Australian Joint Conference on Artificial Intelligence* (pp. 557-568). Springer, Berlin, Heidelberg.

[6] Zlotkin, G., Consenting Agents: Designing Conventions for Automated Negotiation.

[7] Beam, C. and Segev, A., 1997. Automated negotiations: A survey of the state of the art. *Wirtschaftsinformatik*, *39*(3), pp.263-268.

[8] Aydoğan, R., Festen, D., Hindriks, K.V. and Jonker, C.M., 2017. Alternating offers protocols for multilateral negotiation. In *Modern Approaches to Agent-based Complex Automated Negotiation* (pp. 153-167). Springer, Cham.

[9] Binmore, K. and Vulkan, N., 1999. Applying game theory to automated negotiation. *Netnomics*, *1*(1), pp.1-9.

[10] Smith, J.M. and Maynard, S.J., 1993. *The theory of evolution*. Cambridge University Press.

[11] Bartz-Beielstein, T., Branke, J., Mehnen, J. and Mersmann, O., 2014. Evolutionary algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *4*(3), pp.178-195.

[12] Holland, J.H., 1992. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

[13] Lewis, C., 2008. Linear programming: theory and applications. *Retrieved August*, *3*, p.2018.

[14] Russell, S., Norvig, P. and Intelligence, A., 1995. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, *25*(27), pp.79-80.

[15] Baarslag, T., Pasman, W., Hindriks, K. and Tykhonov, D., 2018. Using the Genius Framework for Running Autonomous Negotiating Parties.

[16] Mitchell, M., 1998. *An introduction to genetic algorithms*. MIT press.

[17] Baarslag, T. and Kaisers, M., 2017, May. The value of information in automated negotiation: A decision model for eliciting user preferences. In *Proceedings of the 16th conference on autonomous agents and multiagent systems* (pp. 391-400).

[18] Leahu, H., Kaisers, M. and Baarslag, T., 2019, May. Preference Learning in Automated Negotiation Using Gaussian Uncertainty Models. In *AAMAS* (pp. 2087-2089).

[19] Tsimpoukis, D., Baarslag, T., Kaisers, M. and Paterakis, N.G., 2018, December. Automated negotiations under user preference uncertainty: A linear programming approach. In *International conference on agreement technologies* (pp. 115-129). Springer, Cham.

[20] Yucel, O., Hoffman, J. and Sen, S., 2017. Jonny black: a mediating approach to multilateral negotiations. In *Modern Approaches to Agent-based Complex Automated Negotiation* (pp. 231-238). Springer, Cham.

[21] Baarslag, T., Hendrikx, M.J., Hindriks, K.V. and Jonker, C.M., 2016. Learning about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems*, *30*(5), pp.849-898.

[22] Hindriks, K. and Tykhonov, D., 2008, May. Opponent modelling in automated multi-issue negotiation using bayesian learning. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1* (pp. 331-338).

[23] Lau, R.Y., 2005, January. Towards genetically optimised multi-agent multi-issue negotiations. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences* (pp. 35c-35c). IEEE.

[24] Lau, R.Y., Tang, M., Wong, O., Milliner, S.W. and Chen, Y.P.P., 2006. An evolutionary learning approach for adaptive negotiation agents. *International journal of intelligent systems*, *21*(1), pp.41-72.

[25] Zhong, J., Hu, X., Zhang, J. and Gu, M., 2005, November. Comparison of performance between different selection strategies on simple genetic algorithms. In *International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06)* (Vol. 2, pp. 1115-1121). IEEE.

[26] Jonker, C.M., Aydogan, R., Baarslag, T., Fujita, K., Ito, T. and Hindriks, K., 2017, February. Automated negotiating agents competition (ANAC). In *Thirty-first AAAI conference on artificial intelligence*.