Investigation into Optimization of Agent-based VRP using User Input via a Game Interface

Grant Smith - 40111906

Submitted in partial fulfilment of
the requirements of Edinburgh Napier University
for the Degree of
BSc (Hons) Games Development

School of Computing

April 20, 2017

Authorship Declaration

I, Grant Smith, confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others this is always clearly attributed;

Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;

I have acknowledged all main sources of help;

If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;

I have read and understand the penalties associated with Academic Misconduct.

I also confirm that I have obtained informed consent from all people I have involved in the work in this dissertation following the School's ethical guidelines.

Signed:

G. Smith

Date:

20/4/2017

Matriculation no:

40111906

Data Protection Declaration

Under the 1998 Data Protection Act, The University cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please sign your name below one of the options below to state your preference.

The University may make this dissertation, with indicative grade, available to others.

The University may make this dissertation available to others, but the grade may not be disclosed.

The University may not make this dissertation available to others.

Abstract

The aim of this project was to investigate the application of games as a tool for optimization in vehicle routing. A player bidder was implemented with a computational institution and some market-based control features. Experiments were ran with a number of participants including both experts and non-experts in the area of vehicle routing. The findings showed that the introduction of people did not have a strong impact on the solution costs. Further investigation was done into the individual routes of the results to determine what caused this outcome. It was discovered that whether the players route was good or bad, the computer agent would compensate either way. This was unexpected but opens the area for further research. The institution was shown to have little affect on the routes of non-experts but in its current state negatively impacted routes of experts. This requires further investigation but it is believed to be caused by the selection for the institution rather than the institution itself.

Contents

List of Figures

Acknowledgements

I would like to thank the following people for without them this project would not have gone so smoothly:

All of the participants that took part in the experiments. There help supplied this project will the needed results.

Dr Neil Urquhart, for his outstanding supervision and continual input to the project.

1 Introduction

Vehicle routing has been an on-going problem for decades now with numerous approaches to solving it. The majority of vehicle routing solvers make use of a centralised method, processing the problem from start to finish linearly where one process manages the decision making of the vehicle routing system. In more recent years, many researchers taken a more decentralised approach. This usually involves the use of agents who autonomously solve the vehicle routing problem. The decentralisation leads to the empowerment of each agent, spreading the decision making amongst several processes, commonly known as heterarchy [3]. As each agent is managing itself, there is an aspect of competition where agents can be incentivised to focus on improving their own solution. This solution in many cases is only part of the complete solution and in most cases represents a single route or vehicle.

Although agent-based solutions have changed the way we think about vehicle routing, many of the previous approaches are still used in some form. Methods such as the sweep and components of it have been used as preliminary stages to agent-based vehicle routing solvers [10]. Many algorithms use the split phase of sweep to divide the problem into sub problems which are then distributed amongst a number of agents. Others such as [15] have used the K-means algorithm to partition the nodes. In some cases, techniques used before agent-based systems are used purely to quantify the results of an agent-based approach as done in [18]. Due to the new perspective that comes from using agents to solve vehicle routing problems, solutions involving Ant Colony Optimisation [18] have become increasingly popular attributable to the simplification of such algorithms incurred by agent-based implementations.

This dissertation investigates the application of using games to optimize an agent based vehicle routing solver. It incorporates aspects of market-based control to manage the bidder agents through the alteration of incentives such as currency rewards/penalties. These incentives will be governed by a computational institution [6] which controls the values of such rewards/penalties. The institution will be managed by either a user. Users will compete with the software agents in attempt to get the best score or route. The results of each solution, which can be thought of as one game, will be logged for analysis to determine optimizations, if any, that can be made from what is learned about the winning user's actions.

The proposed investigation will be implemented using a framework developed

by myself and Dr Neil Urquhart during my internship at Napier University. The current framework uses JADE [20] to implement an agent-based auction solution to vehicle routing problems. This will be used as the baseline for all of the results to evaluate the effectiveness of the market-based control component of the system.

1.1 Aim

To develop a system for investigating the use of games as a way to optimize an agent-based vehicle routing solver.

1.2 Objectives

The current agents within the framework will be extended to incorporate Market-based control. The functionality of the Market-based control will be used to drive the decision making of bidder agents controlled by the players. The scoring system within the game will be managed by the market-based control, displaying information such as balance, profit, and cost to the player. To further drive the player towards optimal solutions, a computational institution will be implemented that can directly affect the net profit/loss of a completed bid. As this system is player driven and the computational institution has many possible combinations for each problem, the best solution to each problem will likely not be found within the course of this work. Although the best solution will probably not be found, this project should give an insight into the applications of games to vehicle routing.

After the execution of each problem, the system will output a number of files documenting the actions taken by each bidder (computer and player) and their resulting routes. These files can then be used to evaluate the solutions against a baseline of all computer controlled agents.

1.3 Research Question

1. Can people replace computers in vehicle routing to improve solutions?
2. Does the players have a strong impact on other routes?
3. How do the solutions of expert and non-expert players compare?
4. How does the average cost per visit for players compare to that of non-players?
5. Does the institution improve the routes of the players?

1.4 Scope

Factors that will be of the most interest in the evaluation of the market-based control component of the system will be the cost, distance, time, and

C02 emissions of the routes. The time taken to find solutions is not of relevance to this investigation as the system will be slowed down to enable users to interact with it in a turn based manner. Problems will be ran using either car or public transport as the mode of travel. There will be no problems ran using a mix of the two as this may scew the results of the area being investigated.

1.5 Constraints

The main constraint for this system will be the ability to gather results. This is due to how heavily it relies on the people to use the system to produce the results.

1.6 Sources

The sources for this project come from articles, journals, and books available online or from the University library. Used sources will be covered in the Literature Review in Section 2.

1.7 Chapter Outline

The remainder of this report will cover the methodology and implementation in Section 3, an analysis of the findings in Section 4, and an evaluation of the work under taken in Section 5. Following these sections will be a discussion on possible future work in Section 6 and a conclusion summarising the project in Section 7.

2 Literature Review

2.1 Vehicle Routing With Agents

Agent-based systems have been used to solve problems from many different areas. These areas include vehicle routing, factory scheduling [3], and disease simulations. As for vehicle routing, an agent-based approach has become relatively common over the last few years. Many of these projects have used common Vehicle Routing Problem (VRP) algorithms with altered implementations to research the applications of an agent-based approach.

The most commonly used algorithm in the recent years for solving agent-based VRP stems from the sweep algorithm, as used by [15] and discussed in [10], which initially splits the problem down into clusters of sub problems. Which are each solved by one or more agents. Usually this involves an algorithm for reordering the nodes within each cluster so that they are arranged more efficiently. Clark and Wright's savings or more complex algorithm are used in most cases but crude approaches have been favoured for their speed by a number of developers. The speed of the cruder approaches usually come with suboptimal results as they are more simplistic such as the Nearest Neighbour method of solving VRPs. As stated by [11] Nearest Neighbour does not necessarily generate routes that are geographically closed as the choosing of the next node only considers the current. On the occasion where Nearest Neighbour has produced good results it is used iteratively to converge on a solution which leads to a loss of some or most of the performance gain of a simplistic algorithm. [18] does this with the addition of Ant Colony Optimization to determine the best route through tracking of frequently used connections.

Another technique used in multi-agent systems involves genetic algorithms. [8] uses this as one of their metaheuristics where they use the genetic operators of crossover, mutation, and reproduction to alter a sample of routes until a stopping criteria is met. The continual alterations gradually converge on an efficient solution. The downside of this process is that although it can converge on a solution quickly, there is not necessarily a guarantee it is reasonably optimal.

Ant Colony Optimization is another popular method which is frequently used in conjunction with the sweep algorithm [15]. Ant Colony Optimization incorporates a method of tracking the optimal routes through a problem. This is done by treating the problem space as a graph and leaving weightings known as pheromones on the edges/connections. These pheromones indi-

cate the popular and ultimately most efficient connections between nodes as decided by the agents (representing ants/vehicles). Since Ant Colony Optimization relies on determining popular connections, it is an iterative process and as such usually involves the reallocation of visits between agents to calculate the optimal connections.

As can be seen in [3], [4] and [14], the implementation of a bidding process in conjunction with an agent-based approach is relatively popular. Bidding can have a substantial overhead depending on the communication methods used to pass messages between agents, this is fairly noticeable with JADE [20] due to the verbose messages (uses FIPA). This overhead is negligible due to the other benefits that come with a bidding process such as flexibility and scalability. These benefits come from the variable number of bidding agents within a bidding process. As the number of bidding agents can be changed, the system can be easily modified to manage a varying size of vehicle routing problems from very small to very large. [3] uses this to allow for modularity in the system so that additional agents can be added, or even removed, if deemed necessary for the system. In their case this could mean that additional storage agents could be added with ease if they had a bigger factory than the system was originally developed for.

[21] proposes a system for the American Red Cross which aims to increase the available blood for platelet extraction. They do this by reducing the total travelling distance of each vehicle and by focusing the agents on the strictest time window of the blood to incentivise the agents to meet the time frame for platelet extraction. [21] uses Column Generation to reduce the number of possible solutions to a more manageable amount in attempts to reach an optimal solution at a faster rate. In more recent years as more and more problems have included time windows, many approaches have shifted away from Column Generation [12] for more approximate techniques which within a smaller time frame can still produce solutions of a high quality. Through incentivising a short trip, [21] claims they managed to reduce the travel distance by 60% and increase the extractable platelets. The claims don't hold much validity though as they are simply claims, the only result listings are the timings of system execution.

2.2 Vehicle Routing Without Agents

Many Vehicle routing solvers focus on relatively small problems usually consisting of a number of nodes within part of a city. [11] expands upon the Nearest Neighbour algorithm, as used by [18], to develop what they call the

Route-Nearest Neighbour algorithm. This algorithm aims to find solutions to large-scale vehicle routing problems which in their test involved the deliveries to be made for a cigarette company across Suizhou city. They did not go into a great amount of detail about the algorithm but essentially it divided the problem space up into regions that would each be delivered to over the course of a working week. Once divided, the routes within these sections would be organized using their altered version of Nearest Neighbour. Their Nearest Neighbour algorithm took into account the entire route when selecting the next node rather than just using single nodes to determine the acquisition of a node. To do this, they calculate an arc vector between nodes which are used as the cost of adding nodes and additionally to prevent connections that would cause the route to be geographically closed. The results they produced seem rather insignificant as they do not show whether or not the routes are geographically closed due to the fact that there are no connections drawn between them.

[17] investigates an arguably more interesting, more comprehensive approach to solving large-scale VRP. They incorporate an element called time geography where the time windows and locations of nodes are visualised in the same coordination system. Using this, the dividing of nodes into subsets of the overall problem can be done at a finer detail to the common technique of simply using locations to split a problem. Solving similar problems to [12] but on a much larger scale, by dividing a problem using the time windows as well, the results are much more efficient. The problem where nodes are put together simply because they are near each other geographically even though they are to be fulfilled at completely different times does not occur as it is handled at the splitting stage. Using time as a third dimension to the two dimensional locations allows for an intriguing method of calculating optimal neighbours involving the angles within the three dimensional perspective of the problem.

2.3 Sequential and Dynamic Vehicle Routing

As with sequential implementations of VRP solvers, agent-based solvers can be used to find solutions to both static (SVRP) and dynamic (DVRP) vehicle routing problems. [3] is a great example of a dynamic vehicle routing problem. DVRPs differ from SVRP in that SVRPs have the entire problem from the start and are able solve the problem without the uncertainties that would arise in a DVRP. A DVRP may or may not start with any part of a problem to solve. As a DVRP system is running, information is added in, such as visits to be made. Such a system then has to handle this information dynamically in some cases making assumptions, as in [15] where the capacity

requirement of each visit is unknown until a vehicle has reached it. [3] is a DVRP system where there is no initial problem to be processed, each task is scheduled as it supplied to the system by the user. DVRPs usually result in solutions that are less efficient, and more 'brittle' [19], than those created by SVRP systems but this is expected as the goal of a DVRP system is to find an effective solution quickly, not an optimal one.

The aim of a dynamic vehicle routing solver is to handle the allocation of dynamically introduced nodes. It can be seen in [4] that DVRP and SVRP can be combined. Their solver takes a static problem and solves that using the split stage of the sweep algorithm and an algorithm similar to Nearest Neighbour but with the addition of angles. The second stage of [4] is dynamic and processes the nodes passed in using a bidding process where each agent informs the request manager about the cost of adding the node to the end of their route. This method shows similarities to the process used by [15] where there is also an initial solution but in their case, the solution may be the final solution unless a capacity constraint is broken which results in the reallocation of remaining nodes.

2.4 Agent-based Systems

Not all agent-based vehicle routing systems are for determining the optimal route(s) for one or more vehicles. In the case of [9], their aim was to determine how a number of factors affect C02 emissions and travel time. They examined the cumulative effect of flexitime, urban concentration, the new Aberdeen bypass, and cyclists and drivers sharing the road. Their system simulated a number of commuters going to and from work each day, running the course of 60 simulated days (the first 20 of which were ignored as burn in). Compared with the typical TCA model, space within the simulation was modelled at a much coarser grain of (100m), than the usual 7.5m. This could lead to a potentially significant degree of inaccuracy in their results. Additional factors that could affect congestion were also ignored such as business vehicles (eg deliveries) and traffic lights which would be relatively strong factors affecting congestion especially on such a coarse simulation. Public transport as a whole was also ignored but a reason for doing so was given:

'Moreover, compared with larger cities such as Edinburgh, the use of public transport is lower in Aberdeen. Rail, the only other public transport option in the area, is also ignored' - pg 6 (section 3.15)

From their results it can be seen that flexitime definitely reduces C02 emissions and overall congestion, most significantly when the flexitime ranges

from 0 to 60 minutes. Urban concentration reduced C02 on average as people were more likely to walk/cycle as on average they lived nearer their work. The new bypass increased the overall C02 emissions as the agents who drove were more likely to pick it as it was the faster route. Finally, cyclists sharing roads with cars did not affect the congestion on the roads which in turn meant they did not affect the C02 emissions or commute time of agents who drove. [9] even with its inaccuracies, shows promising results, and shows yet another area of use for agent-based systems.

One useful application of agents is for the scheduling of a factory where no predefined schedule or list of jobs is supplied. Instead, jobs are supplied to the factory and each agent, representing a machine, bids with each other for resources. This can include one machine doing a job for another, making an order for materials, or holding materials/parts for future tasks. Each agent quantifies its decisions using real world currency and competes with other agents in order to win a task. This is commonly known as market-based control or a market-driven contract net architecture. Through this process of controlling a factory, the schedule is built up using heterarchy rather than a hierarchy as no agent is ranked lower than another and there is no central decision maker scheduling the machines. This leads to improvements in the effectiveness and scalability of the system, as said in [3], 'Agent software offers all the advantages of object-orientated programming with the additional benefits of fault tolerance, modularity, local empowerment, and overall flexibility'. [14] also agrees with this 'they can provide flexible, extendable, and fault tolerance control and management systems.'.

2.5 Market-based Control

The use of an auction house paradigm is a common approach to agent-based systems. [13] uses such an approach to distribute a cluster of nodes to agents that represent routes. Each agent bids for nodes to create their own optimal route. This approach does not use a centralized decision maker (auction house) as many others do. Instead, a Consensus-based Decentralized Auction is used where agents agree as a whole as to who wins the bid. This is similar to [3] which manages the scheduling of a factory using agents that bid against each other for work from a client. Having agents compete in a virtual marketplace develops near optimal solutions to problems where different components have their own motives. This can apply to many areas whether it is an agent who wants the best route for themselves or even an agent that wants to do all of the client's tasks.

[6] covers an approach to controlling agents through the use of a compu-

tational institution. Although this is not necessarily market-based control, they can be coupled to supply overall management of an agent-based system. Using an institution, norms can be used to control agent's actions to drive them towards an institutional goal as well as their own goal. In vehicle routing an institutional goal could be an optimal overall solution and each agent's goal would be their individual route. For the purpose of this project, the approach of [6] is overly complex and consists of many agents with various roles that depending on the problem may be completely unnecessary.

Market-based control in relation to agent-based systems have been used for a variety of applications which in the case of [16] involves task allocation of numerous surveillance robots. Similar to that of [15], pheromones are left by the robots (agents) to add weightings to segments of the problem. A map denoting the pheromone value of each part of the problem space is used to determine which areas need to be visited next. Using market-based control, [16] saw a significant decrease in low pheromone zones compared to their crude greedy allocation approach. They found market-based control proved very successful at the allocation of tasks between agents just as found by [3].

[14] uses multiple different agents to develop a system for managing the operation of a micro grid. Similar to [3], each agent represents a physical component, or components, of the system, acting on their behalf. [3] and [14] are actually fairly alike. They both consist of agents that can complete tasks, agents that can store produce for later consumption, and both follow a bidding process to determine the agents to use. [14] found that the use of market-based control could allow for the price of electricity from a micro grid to remain low through the management of swapping power generators as decided by agents bidding.

2.6 Optimisation through games

'Systems that can learn interactively from their end-users are quickly becoming widespread.' [2]. In more recent years, machine learning has been used to improve the functionality of software. [5] developed a simple game for tagging music as "an integrated approach to annotating multimedia content that combines the effectiveness of human computation". The tags that the users set for the media was stored to refine a system for searching through a massive database of multimedia. Using the results, the system can make assumptions on future audio based on what it previously learns from user interactions.

[5] was inspired by the ESP game [1] and as such is very similar in its ap-

proach. The ESP game was a competitive web game where players tagged images to improve the accuracy of image searches. ESP took inspiration from spammers using bots to bypass captchas by putting the same captchas on their site and waiting for a user to answer it for them. The game was symmetrical where each player was to enter what they saw in an image or what it made them think without knowing what the other player was entering. The aim was to have a player enter a word that the other had entered. This would result in the tagging of an image which could later be used in the searching process. The ESP game also allowed for a single player and no player mode. Single player would pair a player up with a recording of one side of a previous, improving on some of the already acquired tags. The no player mode would take this even further by using two recordings from separate games to improve the tags by favouring frequent matches.

Games have been used within the last decade to take advantage of human computation to solve tasks a long side computers or even to solve those that computers can't in the case of ESP [1]. [7] uses human computation to tackle computationally difficult tasks through the use of puzzles. These puzzles are made for non-expert users to advance the scientific domain by using their own problem solving to discover new optimal ways of manipulating proteins.

2.7 JADE

The Java Agent Development Framework (JADE [20]) is a FIPA compliant Java framework by TILAB. It is used to simplify the implementation of peer-to-peer agent-based applications. JADE provides libraries to develop agents and a runtime to execute them which must be active on a device before agents can be instantiated. Asynchronous message passing is used to communicate between agents which may be on a single processor or multiple connected via a network. A common use for these messages is to make negotiations between agents in order to solve a complex problem using distribution. JADE is very comprehensive and can be used to build a wide variety of software. This lends it to be a great tool for prototyping but for many problems can also cause it to not be the most efficient solution. The main issue with JADE is the overhead supplied by it being a jack of all trades, there are a lot of standards that just are not needed for every problem but are bundled into your code anyway.

2.8 Current System

The framework this project is being built on uses the Java Agent Development Framework (JADE [20]) to solve vehicle routing problems using an agent-based auction house solution. Agents within the system use asynchronous message passing, supplied by JADE, to communicate information

such as the negotiation of bids and the data required to calculate the cost of accepting visits. These costs are in the form of real world currency or C02 emissions.

The system consists of four types of agents; MapServer, Auctioneer, Bidder, and CarShareServer. The MapServer manages the loading and storing of essential information about the travel costs between all of the nodes. This information includes the time taken to get between visits and the respective C02 emissions. Other agents can make requests to the MapServer for information relating to two nodes. This agent does cause a bottleneck in the system while all of the other agents are requesting information as they end up waiting in a queue for their responses. This could have been improved using a method of caching information on agents receiving the responses. The reason only one agent is used for this task is purely for the use of modularity. It currently loads the information from a CSV file but could easily be extended to use another method of acquiring information.

The Auctioneer is the core of the system. It controls the entire bidding process and stores the visits to be auctioned. While the Auctioneer has visits to auction, it sends out calls for proposals to all of the bidder agents, waits for their responses, and then selects a bidder to allocate the visit to. During this time, the Bidders accept the visits, determine if they can allocate them to their route without breaking time constraints, and then calculate their best bid. If the Bidder agent cannot allocate the visit, then it sends a refusal to the Auctioneer. Otherwise it sends the calculated bid as a proposal. During each bid, the agent may also send a visit to be returned to the Auctioneer as part of their bid (this is usually in the form of 'I can do this task if I don't do this other one'). The calculating of a bid can also incorporate car sharing negotiations. In a simplified way, this is where one agent attempts to join another agents route for a few visits so that they can reduce their own C02 emissions.

Once all visits have been distributed, the Auctioneer sends a request to each Bidder to return their route. On receiving all of the routes, the Auctioneer combines them into a solution consisting of the total costs and a list of routes.

This system has a number of ways that it can run. These include setting a minimising factor; reducing the real world cost or reducing the emissions. These can also include setting the allowed transport modes; car, public transport, or a mix. If a mix of public transport and car is allowed, then car sharing can also be enabled. Even if car sharing is enabled, it does not necessarily mean there will be car shares as it may be less efficient to do so. This is

almost always the case when the minimising factor is cost.

There is a great room of improvement to be made with this system. Namely, the bidding process as it is simplistic in how it determines the best point in the route to add a visit. At the core it basically checks if time windows will be broken and if not then the bid is equal to the inverse of the distance between visits A and B plus visits B and C minus the distance between visits A and C. This results in a low bid for a high difference and high bid for a low difference.

3 Implementation / Methodology

As discussed in Section 1 the software developed was written using two pre-existing frameworks; JADE [20] and framework developed during an internship in the previous year. JADE was required by the underlying framework as it is built upon it but it is also a popular, robust option for developing agent based software in java. The use of JADE allows the software to be separated into multiple, modular components which leads to a highly scalable solution and lends itself well to an auction-based solution. This separation means that with the use of JADE's messaging system, each component could be running on a different machines. This is obviously not always beneficial as the overhead incurred by the communication time could lead the system to run slower but in certain scenarios this overhead is acceptable. In the case of this project, the communication overhead is not apparent as some components are controlled by people. As such a few milliseconds is unnoticeable in comparison to the decision making time required per turn for a person.

The developed software is intended for use by researchers primarily as a means of investigating the applications of games to vehicle routing. The software could be used for other purposes such as a prototyping tool for new vehicle routing solvers. This software would be a beneficial tool in that case as a researcher would be able to step through their algorithm by hand to get a rough idea of how effective it is. Doing so could supply information that would lead to alteration to the algorithm that would not have previously been found until many hours of implementation have been undertaken.

Before the implementation for the developed software could begin, there were changes to the underlying framework that needed to be made. As this framework was only recently completed before the start of development, there had not been any other software that used it. This meant that issues that would only found through developing with it had not been resolved. The main changes that were required were related to the implementation of the communication since it was implemented using only local identifiers. This would have severely limited the developed software as it would not be possible to scale it across multiple machines. Resulting in a limitation to the number of players that could take part purely due to the restriction of a single machine. To resolve this, the communication implementation of the framework was updated to use the DFService supplied by JADE. This service enables agents to register themselves with the service so that other agents can search for them. Agents can then communicate with them regardless of what machine they are on. Other minor changes were made to the framework so that inheriting classes could override or extend upon more of the functionality of their parent class. Without these minor changes, the

control over the framework was restrained, limiting the potential of software developed with it.

3.1 Structure

The developed software consists of four types of agents. The auctioneer which can be seen as the main controller within the system, managing the auctioning of visits and the initialisation of each problem. There is a map server agent which stores the information regarding travel costs between each location to any other location and is used as a central data store for each agent to request said information. A computational institution was implemented as an agent which can control the incentives towards visit, directing the bids to and from certain visits. Finally, the bidder is the core processing agent which decides on the ordering of routes based on which visits the auctioneer tells it that it has won. The bidder is the only agent that has multiple instances running at one time during a problem as one agent is used per route. For this solution there is a second type of bidder which is controlled by a player. A player controlled bidder has similar functionality to a regular bidder but as the name implies, the decision making is handled by a person. A class diagram for this structure can be seen in Appendix D.

3.1.1 MBCAuctioneer

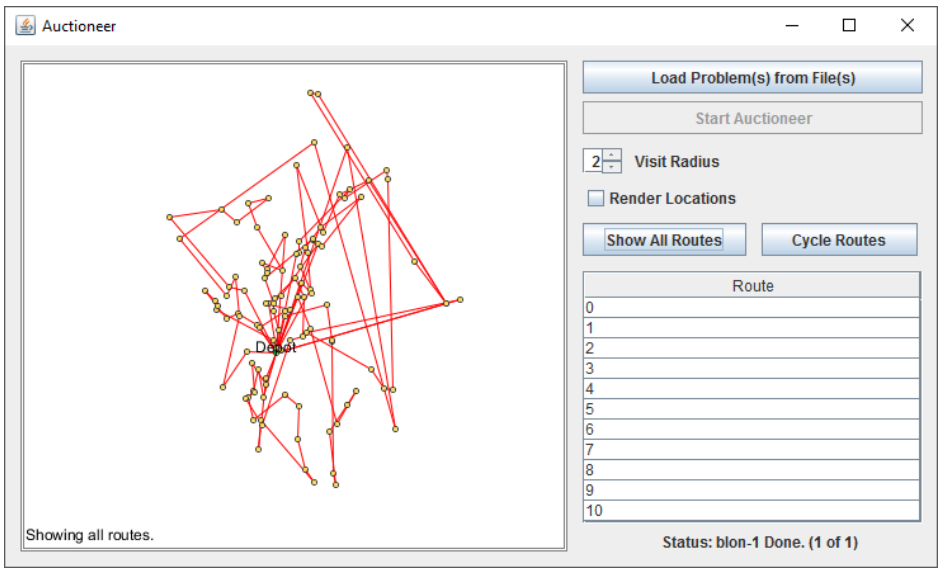


Figure 1: Screenshot of the Auctioneer's GUI.

The Auctioneer is the core component of the system, handling the bidding system, the loading of problems, and the saving of results. It is the central

point within the system that all agents must communicate with and this is the reason that all output of the system is sent to the Auctioneer. The Auctioneer was implemented using a state machine to handle the flow of the bidding process. Three major states are defined; initialise, bidding, and finishing. The bidding state consists of two minor states, one for requesting bids / auctioning visits and the other for waiting on the responses.

Initialisation State The initialisation state as the name suggests is used primarily for the setup of the auctioneer and all other necessary agents. At the start of this state the Auctioneer determines which bidders to use in the bidding process. This is done by checking how many agents are available on the DFService. Depending on how many agents are available and how many are needed for the current problem, the Auctioneer can create additional agents to make up the difference. This functionality is used heavily to allow for a dynamic number of players. Player controlled bidders are registered with the DF service while non-player controlled bidders are created at runtime by the Auctioneer. Once the necessary bidders have been found or created, the Auctioneer begins to send out initialisation messages to the relevant agents. Every bidder is sent a copy of the 'bidder-arguments' which defines the number of returnable visits, the factor to be minimised (emissions or cost), and the transport mode(s) allowed. Bidders are also sent a copy of the depot to inform them of the start and end location of their route.

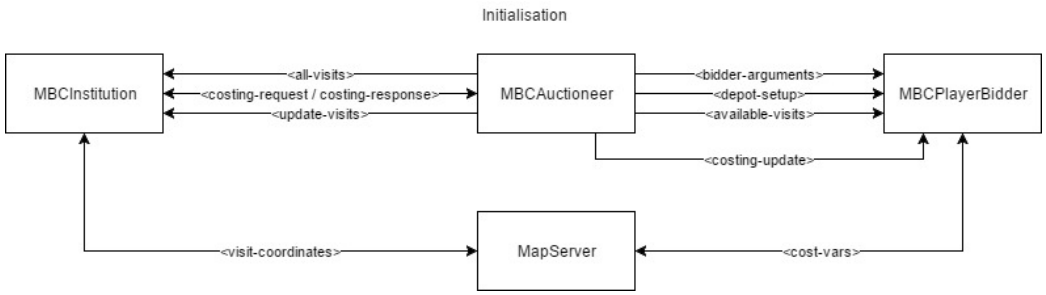


Figure 2: Communication overview during initialisation stage.

Before leaving the initialisation state, the Auctioneer sends three messages to the Institution and two message to the bidders to prepare them for the current problem. Initially an 'all-visits' message is sent to set the Institution's knowledge of the problem space. This is followed by an 'update-visits' message to refresh the available visits of the Institution which at this point is all visits. Every bidder is then sent a copy of all of the visits in an 'available-visits' message which updates their copy of available visits and all visits since at this stage it is also empty. Lastly, the Auctioneer requests the costing information from the Institution using a 'costing-request' message.

On receiving a 'costing-response', the costing information is forwarded to the bidders and the Auctioneer enters the auctioning visits stage of the bidding state. An overview of the communications for this state can be seen in Figure 2.

Bidding State

This state is split up into two minor states, one where a visit is auctioned off and bidders begin their bidding process and the other where the agent waits and receives incoming bids. This state is repeated while the Auctioneer has visits that still need to be auctioned and as a result is the main body of the program. Each iteration when auctioning a visit, the Auctioneer supplies some initial information to each agent involved. Before starting, this agent checks for any 'balance-inform' messages from previous winners, updates the balances if necessary and send a 'all-balance-inform' out to each agent containing the updated balances. For the first turn, there are no balance-inform messages as each agent starts with the same balance. Following this, the Auctioneer makes a 'cost-vars' request to the MapServer to determine the maximum bid a bidder agent can respond with to the next visit being auctioned. Once this value has been calculated, the agent sends out a 'visit-max-bid' message to each bidder agent to inform them of their bidding limit.

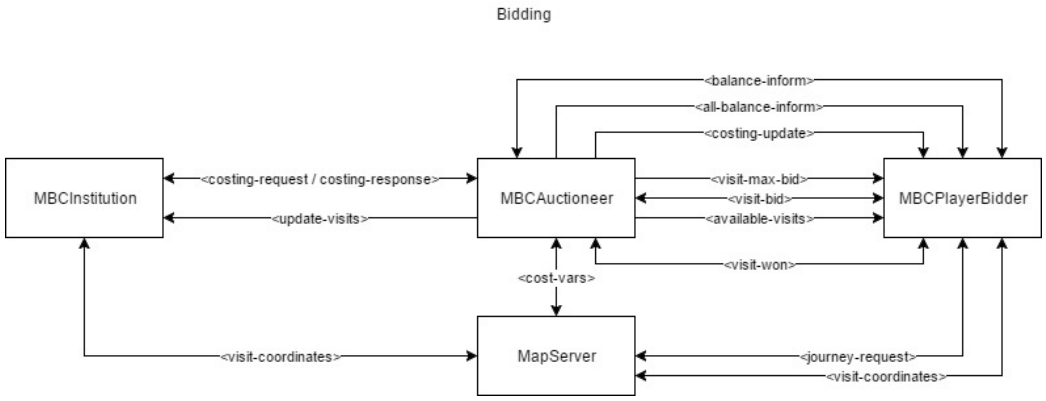


Figure 3: Communication overview during bidding stage.

After doing some pre-turn initialisation the Auctioneer begins the bidding by removing a visit from its list of visits to auction and sends it to the bidders using a 'visit-bid' request. At this point, the auctioning minor state ends and the agent enters the waiting minor state. During this state, the agent waits for all bid responses to come in from the bidder agents. Once all responses have been received, the agent sends updates to the Institution so that it can prepare for the next turn. These updates are similar to those found in the initialisation state, updating the available visits using a 'update-visits' inform

followed by a request for the newest costing values using a 'costing-update' request. This request is done in a blocking manner to stop the Auctioneer until the institution has finished being updated and has responded to the request. The Auctioneer additionally sends an 'available-visits' inform to update each bidder before deciding which bidder has the optimal bid at which point it sends a 'visit-won' inform to the respective bidder and begins the next turn. Once the Auctioneer has emptied its list of visits to auction, it enters the finishing state. An overview of the communications for this state can be seen in Figure 3.

Finishing State

This is the final state for the Auctioneer before the its behaviour is reloaded for the next problem. The finishing state is intended to combine all outcomes of the solution for saving. The agent initially sends a 'bidding-finished' request to each bidder, expecting a response containing the route they have put together. This message is followed by another request, 'bidder-log', for the bidder's log data that they have accumulated during the processing of the completed problem. Once the Auctioneer has received a response to both of these messages from all the bidders involved, the information is saved into timestamped files with the relevant problem name and identifier (eg 'res' for results, 'log' for the log). The Auctioneer then makes multiple 'visit-coordinates' requests to the MapServer to get the required information for rendering the routes and then displays the output along with the resulting costs. An overview of the communications for this state can be seen in Figure 4.

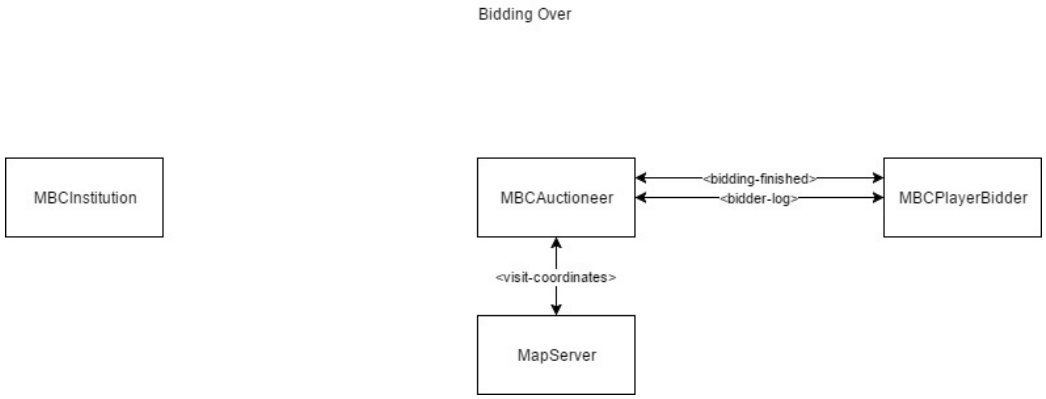


Figure 4: Communication overview during finishing stage.

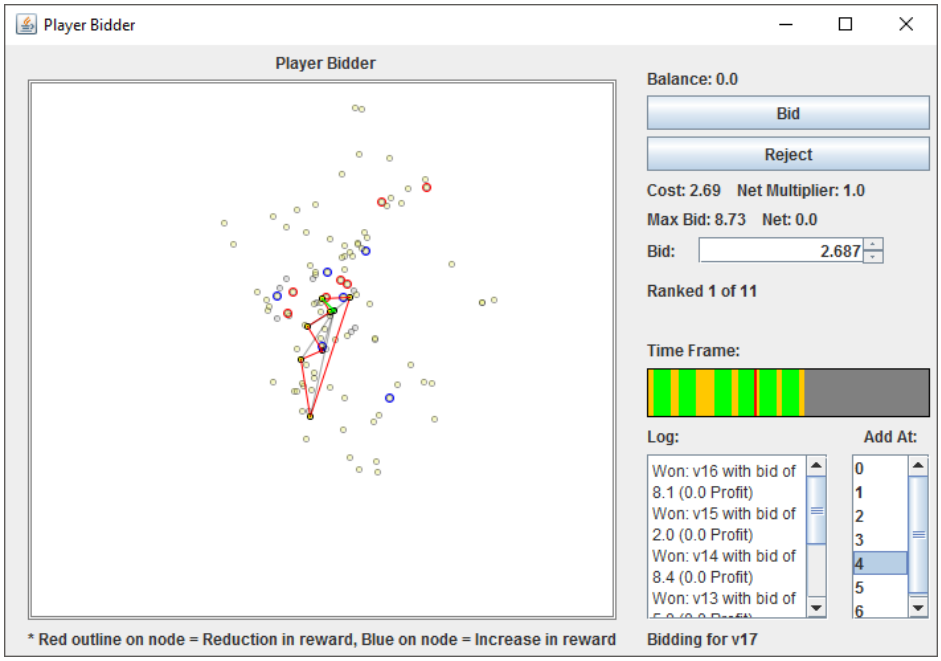


Figure 5: Screenshot of the PlayerBidder's GUI.

3.1.2 MBCPlayerBidder

Like the Auctioneer, the player bidder was implemented using a state machine consisting of three states. These states match those found in the Auctioneer (see Section 3.1.1). This was required to organise the communications between these two closely coupled agents. This is especially important as there can be many bidder agents. Without the state machine, it would be difficult to say with absolute certainty what each agent is doing at any specific time. Each bidder agent is treated as a worker within the problem space. Their objective is to develop a route for themselves while keeping the cost of the route to a minimal.

The player bidder extends the bidder class and as such covers all of the communications used by it. Because of this and the fact that the bidder class is from the underlying framework, the details of the bidder class alone will not be covered. The main difference between the the player bidder and the non-player bidder is the GUI used by the player bidder. Additionally, the player can modify their bid to accept visits at a profit or loss if they so choose. Non-player bidders will always bid at cost so that they never make a loss.

Initialisation State When the player bidder is started, it waits in this state until the Auctioneer communicates with it. Before the bidder can

truly begin, it requires the Auctioneer to send four messages. The first is the 'bidders-arguments' inform which defines parameters such as the factor to be minimised and the transport mode to be used. The next message is a 'depot-setup' which informs the bidder where its route should start and end. The next two messages are the 'available-visits' and 'costing-update'. The 'available-visits' inform sets up the list of all visits the first time it is received. Subsequent 'available-visits' messages update the available visits list instead. These are used by the player's GUI to show what visits are still to be auctioned and what have already been won by another bidder agent. After receiving all messages from the Auctioneer, the player bidder sends a 'cost-vars' request to the MapServer and waits for the response. This is done so that during the bidding state, the bidder can calculate the cost for adding the auctioned visits to its route. Once the response has been received, the bidder moves to its bidding state. See Figure 2 for a visual overview of all messages during this state.

Bidding State As with the Auctioneer, this is the main state within the state machine and is repeated until all visits have been auctioned. At the start of each iteration of this state, the bidder receives a 'visit-bid' cfp (call for proposal) from the Auctioneer. The bidder then determines if the incoming visit can be inserted into the route without breaking time constraints. This is done by sending a number of 'journey-request' messages to the MapServer and summing up the times required by each visit in the route. If the visit cannot be allocated to the route then a 'visit-bid' response is sent to the Auctioneer containing a rejection. If the visit can be allocated then some additional information is received before passing control over to the player. This includes a 'visit-max-bid', 'available-visits', and 'costing-update' inform from the Auctioneer. A 'all-balance-inform' is also received to show the player how they rank in terms of their balance.

The bidder's GUI is updated by requesting new coordinate information using a 'visit-coordinates' request to the MapServer. Control is then passed to the player where they decide whether to bid or reject the visit. The details of the player interaction with the GUI is covered in Section 3.2. Once the player has decided what to do with the visit (bid or reject), a 'visit-bid' response is sent to the Auctioneer containing whether the visit was bidded on (a bid amount is included if this is the case) or rejected. If the bidder wins the visit then a 'visit-won' inform will be received and the visit will be inserted into the route where previously decided. The winner will then send a 'balance-inform' to the Auctioneer so that it can update all of the bidders with the correct balances of all other bidders. If the bidder receives a 'bidding-finished' message then they will change to the finishing state and handle the response process. For a visual overview of this state, see Figure

3.

Finishing State After receiving a 'bidding-finished' inform, the bidder enters this state. This state is used for combining the results of each bidder into one solution held by the Auctioneer. The bidder calculates the cost of its route for each parameter (emissions, time, distance, and monetary cost). After doing so it responds with its route and the costing parameters it just calculated. Finally, the player bidder sends an inform message ('bidder-log') to the Auctioneer containing the log that it has accumulated during the problem that was just completed. This includes the strategy that the player entered using the GUI and the bids the player made for each visit. From here, a normal bidder shuts itself down so that the Auctioneer can recreate them as needed. If the bidder is a player bidder then the agent does not shut down, instead it calls its reset function to clear out problem specific information. Figure 4 shows an overview of all communications during this state.

3.1.3 MBCInstitution

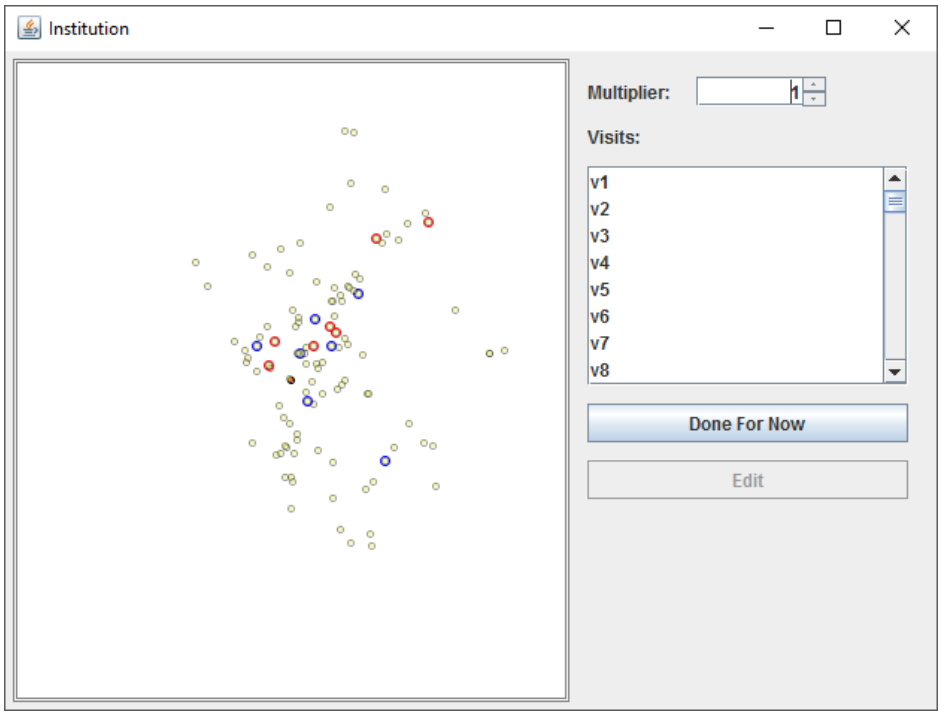


Figure 6: Screenshot of the Institution's GUI.

A computational institution was developed to allow for passive control over the bidders through the use of profit incentives. These incentives come

in the form a multiplier assigned to each visit within the problem. As it is, this value can range from 0 to 2 with 0 being no profit or loss and 2 being double profit or loss. The intention is to direct the solution towards certain visits by either supplying the bidder with a potentially higher profit (multiplier greater than 1) or a lower loss (multiplier less than 1). Deciding what value each visit should be allocated is a non-trivial task in itself but if used effectively, could significantly improve a solution.

The Institution unlike the Auctioneer and Bidder agents, only consists of two states. It does not require the finishing state as it is automatically reset when the Auctioneer sends its initialisation messages. The first state, initialisation, is controlled by the Auctioneer which sends two informs 'all-visits' and 'update-visits'. These messages set the Institutions values for the all visits list and available visits list which are necessary for the modification of the multipliers assigned to each visit. Once both of these messages are received, the initialisation state ends and the running state begins. As with the Bidder agents, the Institution is turn based but it is required to always go before the bidders. This is because any modifications made by this agent could affect all of the bidder agents. If the Institution was to go after the bidders then all modifications would be delayed by a turn and if it was to go at the same time, some bidders would receive the modifications and others wouldn't.

During every turn, this agent receives two messages and sends two. The first message, a 'update-visits' inform, is from the Auctioneer to keep this agents list of available visits up to date. The second is repeated for each visit and is a 'visit-coordinates' request to the MapServer in order to obtain the information required to render the visits. Before the end of a turn, the Institution receives a 'costing-request' from the Auctioneer. At this point, the entire system is waiting on the Institution. As the institution is user controlled, the system can be paused to allow for a user to make modifications to the multipliers at the start of any turn during a problem. Once done, the Institution will then release its lock on the system and respond to the Auctioneer with a 'costing-response'. From here, this agent waits for input from the Auctioneer before it begins the next turn.

3.1.4 MapServer

The MapServer is the simplest of all of the agents but is essential to the system for all information relating to journeys between two locations. The journey information could have been stored by every agent within the system that required this information. This would however be greatly inefficient in terms of memory usage. It would also require every agent to have a means of

loading this data, which as in this case is a file and that would be problematic if the journey information needed to be updated should the agents be running on different machines. Lastly the system would also be hindered in terms of its modularity as to change the method used to access the data if each agent had their own copy would require each agent to be extended. As this implementation uses a separate agent to handle the journey information, other agents only have to send requests to receive the information they need from a centralised resource. Additionally, the current implementation uses a file which is loaded by the MapServer upon initialisation and if this were to change to an online resource, only this agent would need to be extended, all other agents would be unaffected.

Unlike the other agents within the system, the MapServer has no predetermined order as to how it runs apart from the loading of the journey information in this scenario. Once the journey information has loaded, the MapServer follows a Client-Server architecture, waiting for requests from the other agents. These requests all related to the journey information as this is the scope of the agent. Agents can make three types of requests labeled as 'journey-request', 'visit-coordinates', and 'cost-vars'. A 'journey-request' message must contain to and from locations which are used by the MapServer to identify the requested data. The response to this message will contain a JourneyData object consisting of all information related to the journey between the two locations, including the emissions, distance, and travel time for both car and public transport. A 'visit-coordinates' message contains a single location and is responded to with the X and Y location of an individual visit and is used purely for the rendering of visits. Finally, a 'cost-vars' message requires no content and signals the MapServer to send cost variables as defined within the journey information file for this implementation. This information defines the values used for deriving a monetary value for each route, for example staff cost per hour.

3.2 Overview

Starting the system

Once the Auctioneer, MapServer, Institution, and necessary Player Bidders have been started, the system is ready to be set up. For the MapServer, the file containing the journey data is selected and the start button is clicked on the MapServer's GUI, that is the extent of user input for the MapServer. The Auctioneer requires the selection of at least one problem file and for each one selected, the number of bidders required must be entered. At the same time, for all of the problem files selected, the minimise factor (cost/emissions) and the transport mode (car, public transport, mixed) should be set. Once all problems have been selected, the start button is clicked on the Auctioneer's

GUI and all input for the Auctioneer is complete for now.

Running of the system

With the MapServer and Auctioneer ready, it is time for the Institution to set any necessary costing multipliers. The rest of the system is paused until the 'Done For Now' button on the Institution's GUI is clicked. At the start of every turn, the Institution can gain control over the system if the costing multiplier of the visits need updated. After the Institution yields control, the bidders are requested for their bids on a visit. The computer controlled bidders attempt to determine the optimal position to insert the visit into their route, responding with the cost of doing so as their bid. This means that for every computer controlled bidder, their balance will stay at 0 as they will bid at no profit/loss.

Figure 7: **Player Bid Range** - Illustrates the bidding range for a player. Minimum value is always 0 and the maximum is set by the auctioneer. The initial bid is set by the cost which can be within the range or clamped by the limit in some cases.

For the Player Bidders, there is no way to state how exactly they decide to bid as this is the area of research for the optimisations. All that can be covered is what they have to help them decide and in the case of the results section, an insight into the decision making process of a number of users. During each turn, the Player Bidder is given two options, to bid for or to reject a visit. If they choose to reject, they just skip the turn and wait for the other bidders to complete their turn. If they choose to bid, the value set as their 'Bid' will be sent to the Auctioneer. The player may modify their bid within the range of 0 and the value of the 'Max Bid' (see Figure 7) which is set by the Auctioneer. Additionally, the player can change the location at which the new visit will be inserted into their route. By changing this value, the GUI will update showing how this will affect their route and the cost of inserting the new visit in said location. The player is also shown a visual representation of their time frame to allow them to estimate how much more time they have left in their route. The player's 'Bid' value is initially set to cost when a turn begins leading to no profit/loss if left unchanged. Bidding low will net a loss for the player but will increase their chances of winning the visit. Bidding high will have the opposite effect, netting a profit but having a lower chance of winning. If the Institution has modified the

costing multipliers of some visits then the player can take advantage of these to reduce or increase their profit/loss. Every time the player updates their 'Bid' or the insert location of the new visit, they are informed of the resulting net profit/loss. Once the player has decided what to bid and where to insert the visit if it is won, the player clicks the 'Bid' button, ending their turn. After all bidders have ended their turn, the Auctioneer selects the bid it deems to be the best, lower bids are better, and informs the winning bidder of the visit they have one. They then insert the visit into the their route as they decided earlier and then the next turn begins.

As discussed earlier, once all visits have been auctioned, the Auctioneer requests information from the bidders such as the route and log. The Auctioneer then processes the data and saves it to a file. All bidders created by the Auctioneer then stop execution and the rest of the agents return to how they were before running a problem. For example the Auctioneer needs to have the problem files selected again.

3.3 Overview of Experiments

For this project, participants were required so that the effects of introducing players to a vehicle routing solver could be analysed. This section covers the participants that took part and the procedure followed to gather the results.

3.3.1 Participants

To answer the research questions proposed in Section 1, results were gathered using a number of participants. Two types of participants were defined, experts and non-experts. The results produced by each group were compared with each other and the baseline. The baseline was acquired by running the system using only non-player bidders and the same number of bidders. For gathering results, there were four groups. Two groups consisted of two non-experts, one group of two experts, and a final group of just one expert. More groups would have been beneficial but due to the limited number of participants, this was not possible. Running an all player bidder test would have been interesting but was not feasible with the available participants.

The different groups were used to determine if the introduction of players to the solver had a positive or negative impact on the solutions. The two types were introduced to investigate how experience in the area can affect the resulting solution. From the results produced, the routes were analysed to see how much a player's route can impact the route of another bidder (player or non-player). Additionally, from the results, the average cost per visit was calculated to examine how the route of a player compares to that of a non-player.

3.4 Ethics

As this project required the input of participants, it was mandatory for each person to sign a consent form. These forms can be seen in Appendix E.

3.4.1 Experiment Setup

During the experiments, the participants were supplied with a short document explaining the use of the system (see Appendix F). This covered the bidding process and the details about each component of the player's GUI. At any point, the participants could ask about the system if they needed to but no input was given as to how they develop their route. Each group used one computer for the server and one additional computer for each user. These were all connected using JADE which is the underlying system used for the agents and their communications.

Each group completed a number of problems using the solver. The problems ran and their parameters were kept the same for each group. This was done so that the results could later be compared to see how expertise in the area affects the results. The baseline also used the same configurations minus of course the player bidders. From each complete problem, quantitative results were produced that illustrate the four costing values of each route and the contents of them. These solutions will be discussed in Section 4 covering the findings of their analysis. In addition to the results produced by the system, the users were also asked to provide a statement about their strategy for route allocation. This statement is used as a method of understanding the reasoning behind the choices made by the users of the system.

4 Results

The results for this system were gathered with the help of participants playing the roles of the player bidders. The details of the result gathering process is covered in Section 3.4.1. There are a number of types for the results which identify the groups used to produce them. The types are as follows:

- Two Expert Players
- Two Non-Expert Players
- One Expert Player
- One Non-Expert Player
- Non-Players (baseline)

The rest of this section will cover the results produced by each group type. The significance of the results and the relationship between the groups will be discussed. Factors such as the overall solution costs and average cost per visit will be analysed to determine the effect of introducing people into the bidding process of an auction-based vehicle routing solver.

4.1 Solution Costs

Before gathering result, it was expected that the introduction of player bidders would improve the solutions. With experts it was believed that the solutions would always be significantly better than a solution produced with just non-player bidders. Solutions with non-experts were thought to have resulted in solutions similar to those created by non-player bidders.

4.1.1 Blon-1

Five sets of results were gathered for the 'Blon-1' problem. These results show the costings produced with one expert, all computers, two experts, and two sets of two non-experts. The results can be seen in Figure 8.

The use of one expert player resulted in the best solution but the costs of that solution and the all non-player solution were almost identical. Two experts were not very successful with costs that were either almost the same as the computers or were slightly worse. It was expected that two experts would be better than just one but this was not the case. This could be explained by a conflict in strategy where each of the experts is after the same visits. Finally, the two non-expert groups were significantly worse than

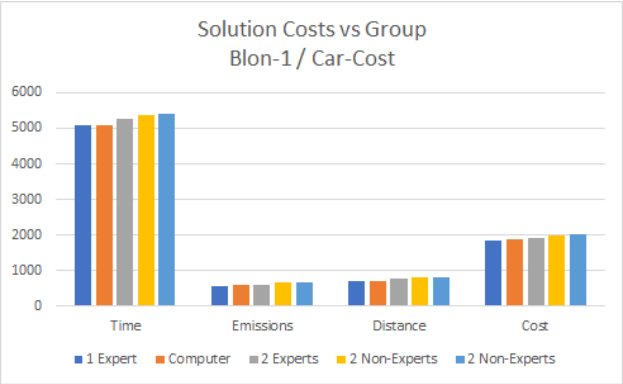


Figure 8: **Results for Multiple Blon-1 Solutions** - One expert was found to be the best overall with computer only slightly behind.

the computer solution. This was sort of as expected since the non-experts did not have a great understanding of the area prior to the experiment.

The aim of introducing the institution was to see if the incentives added by them would direct the players towards better routes. It was noted before this experiment that the results would be highly dependant on placement of positive and negative visits. As a results, the solutions may not represent the best or worst impact the institution can have.

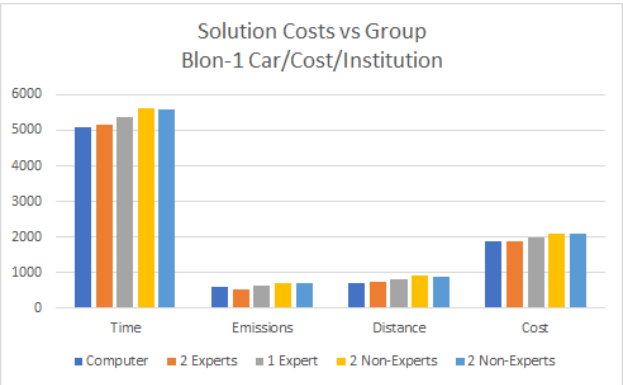


Figure 9: **Results for Three Blon-1 Solutions (with institution)** - The computer produced the best costs for the most part but was not much better than the players.

By comparing the results in Figure 8 and 9 it is clear that the institution caused the players to produce less efficient routes. As mentioned before, this could have been due to the placement of the positive and negative visits and could be improved with further experimentation. The institution in

itself is an area worth researching on top of just the application of human computation. Strategies could be devised that consistently lead players to good or bad solutions.

Interestingly, the institution seemed to cause the group of two experts to create a solution that was better than that created a single expert. This is closer to what was expected between the two groups but in this case the computer was still better than them both.

4.1.2 Cluster-1

The problem 'Cluster-1' was selected as it is slightly different from the other problems. Instead of a large scattering of visits, it has very visible clusters of visits. This solution is well suited to the computers as they can easily pick out these clusters and this results in the computers each taking a cluster each.

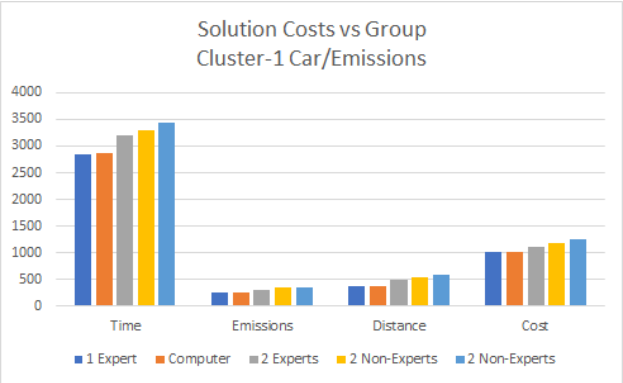


Figure 10: **Results for Multiple Cluster-1 Solutions** - The computer and one expert solutions were almost indistinguishable. All other solutions were noticeably worse.

As with the results for 'Blon-1', the single expert was the best overall but closely followed by the computers. In this scenario the group of two experts was almost as bad as the non-experts implying that these groups had a conflict in strategy or attempted to develop a route consisting of too many clusters.

4.1.3 Lon-1

The results of two individual experts were compared against a computer and a non-expert solution. This experiment was done to see if a single non-expert could improve upon the solution compared to non-bidders. The other results

show that generally a single expert produces a solution that is better than the computer and the group of two experts.

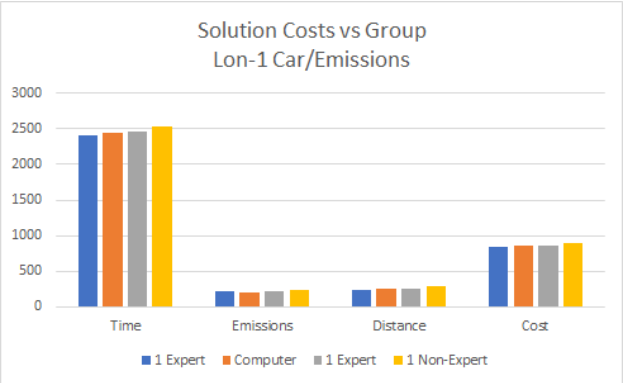


Figure 11: **Results for Multiple Lon-1 Solutions** - One expert was found to be better and worse than the computer on different runs. Non-expert continues to be worse than all.

From Figure 11 it can be seen that the non-expert is still worse than the expert groups which was as expected. Sadly one non-expert instead of two still did not reduce the solution cost compared to the non-player solution as we had hoped.

4.1.4 Summary: Solution Costs

A few of the research questions were answered from the overall solution costs. Relating to the first research question, replacing computer bidders with players did not significantly improve the solutions. It was believed that players would improve the solutions but in actual fact they had very little impact. As for the third research question, it was discovered that non-experts did produce solutions that were worse than experts which as expected. Finally the fifth question was partially answered as from the solution cost results it seems that the institution caused players to develop worse routes. Further investigation will be done in Section 4.2 to answer the remaining research questions and to receive more insight into why the solutions did not drastically improve.

As a whole, the results were interesting, revealing that players can be inserted in a vehicle routing problem without having a drastic impact on the efficiency of the solution. Knowing this could open up another area of researching into human computation to solve other problems such as those involving preferential data (eg A customer's delivery driver preference). This

area could be researched knowing that doing so would have little impact on the cost of the solutions.

Although the results for solution costs did not show a significant decrease in the costs in any case, the results themselves are still interesting. The fact that the findings contradict the expectations is noteworthy and worth further investigation. Section 4.2 covers further investigation into why the results were not as expected.

4.2 Average Cost Per Visit

This section covers the average visit cost per route within a solution. The aim is to determine if there is a significant difference in averages between players and the computer bidders. The cost of the player routes could be lower than the bidders routes but some non-player routes could be bringing the entire solution down as a result of introducing differing bidding strategies. The results in this section include a bar for each route representing the averages for time, emissions, distance, and cost. Additionally, the bar contains the route size which is denoted by the dark blue portion at the top of each bar. This part does not count towards the average cost of the visits and is there purely for an understanding of the route size. As such this portion will be excluded when comparing the averages.

4.2.1 Blon-1

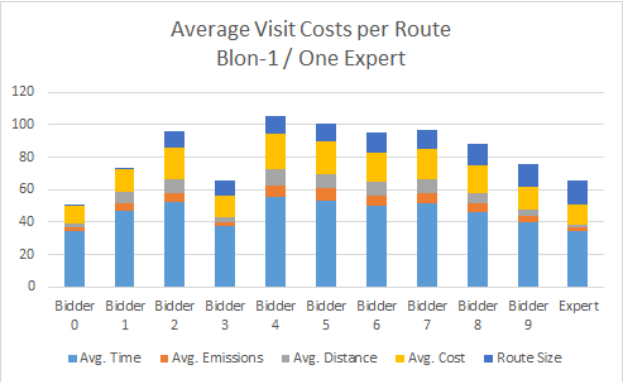


Figure 12: **Blon-1 with One Expert** - The expert bidder has one of the best averages, especially for how many visits it has.

The graph in Figure 12 shows that using just one expert, the player is significantly better than most non-players. The non-player bidders that have a similar average to the player have far fewer visits in their route. Their low average could then be due to them grabbing visits near the depot rather than

the ordering of their visits being well optimized.

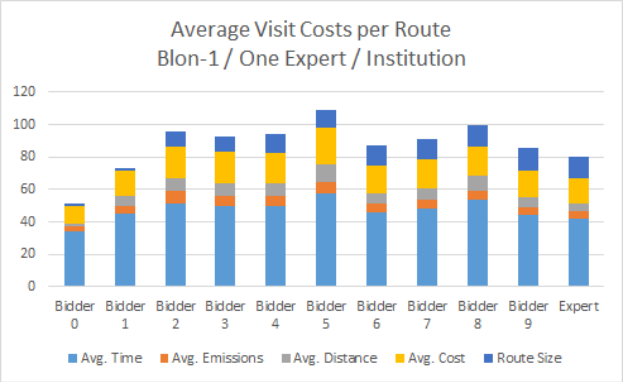


Figure 13: **Blon-1 with One Expert and Institution** - Results for every bidder seemed to get worse with the instition.

By adding the institution, the results for one expert got worse. By comparing Figure 12 and 13, the averages show that the institution caused the player to make a less efficient route. The players decisions also seemed to have a knock on effect which lead to the bidders also having worse routes.

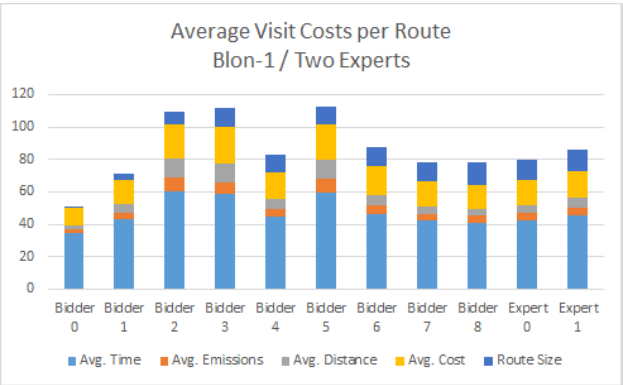


Figure 14: **Blon-1 with Two Experts** - The expert bidders are between the best and worst non-player bidders. Neither good or bad.

The use of two expert players produced averages that were similar to the average non-player bidders (see Figure 14). The similarity of the bidders results is unexpected as a single expert player was far better than most non-player bidders. It was believed that adding an additional expert would reduce the overall solution cost as it would bring the average cost per route down. After looking at the routes created by the two experts, it could be

seen that their routes crossed over a lot (see Figure 15). This explains why the introduction of two experts did not noticeably alter the results of the solution as a whole.

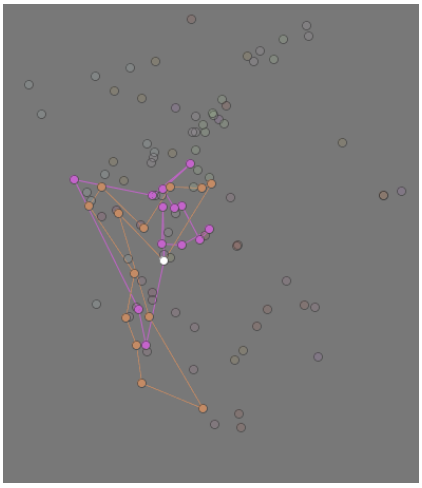


Figure 15: **Blon-1 with Two Experts** - This figure shows the crossover between the two expert routes in pink and orange.

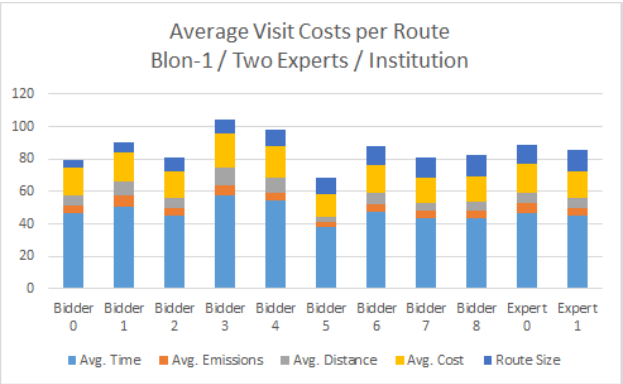


Figure 16: **Blon-1 with Two Experts and Institution** - Less variance but still averages around 80 per visit.

From comparing Figure 14 and 16 the results show that the institution decreased the variance in the average visit cost making the solution slightly worse due to the previously low averages being slightly higher. This could also be caused by the increase in the cost of one expert's route. The change to their route likely impacted the routes of the non-players.

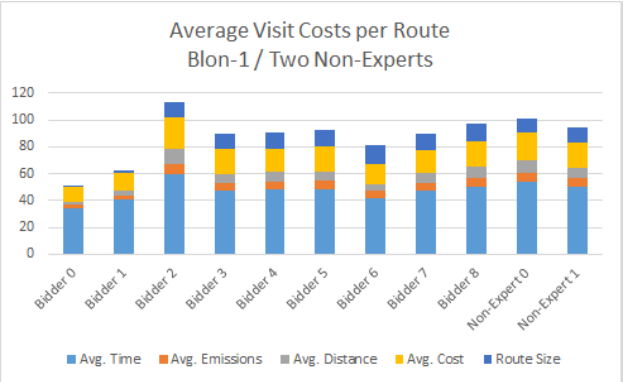


Figure 17: **Blon-1 with Two Non-Experts** - The non-experts were among the worst in their solution with only one non-player worse than them.

The non-expert players were found to produce suboptimal solutions. Their routes were among the worst in their solutions. Figure 17 shows that the non-experts were only better than one computer bidder. The fact that the non-experts compare so badly to the non-players makes it clear as to why the solution got worse overall. These results reveal that the player's do in fact need some experience in the area to produce good results. These results could potentially be improved if the non-expert players spent more time playing with the system. This would be due to the player learning what makes better solutions. Like in many games, the more a person plays, the better they tend to get.

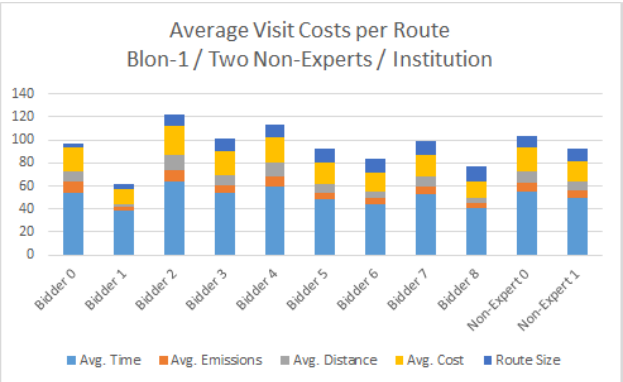


Figure 18: **Blon-1 with Two Non-Experts and Institution** - Very little change can be seen compared to the costs without the institution.

Non-experts stood out from the expert groups when it came to introducing the institution. There is very little difference in the results shown in

Figure 17 and 18. This implies that the decisions of the non-experts were not influenced by the institution. This definitely was not expected especially from how much the institution affected the routes of the expert groups.

4.2.2 Cluster-1

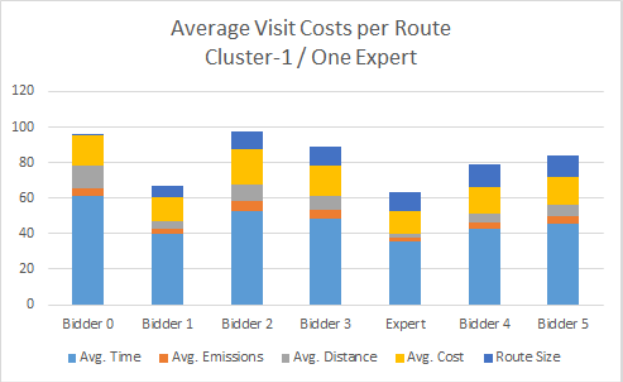


Figure 19: **Cluster-1 with One Expert** - The one expert is better than all other bidders. This player should improve the overall solution but the non-players could be hindering it.

One expert was significantly better than all bidders in the results for Cluster-1 (see Figure 19). In this case, it is actually the computer bidders that cause the solution to not appear far better than the computer only solution. Compared with the non-expert results in Figure 21, the expert player's route is very efficient having almost half the average cost per visit. This route should decrease the cost of the solution as a whole but it does not. As a result, the computer bidders must have been negatively impacted by the player's route.

Non-player bidders are very similar to the player bidders in this solution, excluding bidders with very few visits. Figure 20 shows that the two experts created the best routes within this solution but are not significantly better than the non-players. 'Expert 0' is in fact very similar to the non-player bidders which would lead you to believe that this player is unnecessary in the solution and could be replaced by a bidder. Although the two experts were the best in the solution, the overall costs were still worse than an entirely non-player solution. This implies that the players have in this case negatively impacted the routes of the computer bidders.

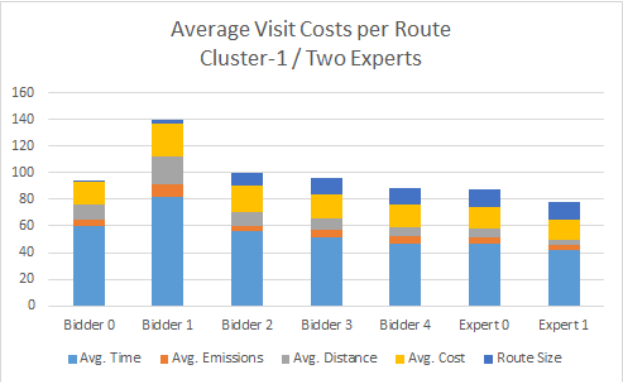


Figure 20: **Cluster-1 with Two Experts** - The two experts produce the best routes in the solution, closely followed by the other bidders with large routes.

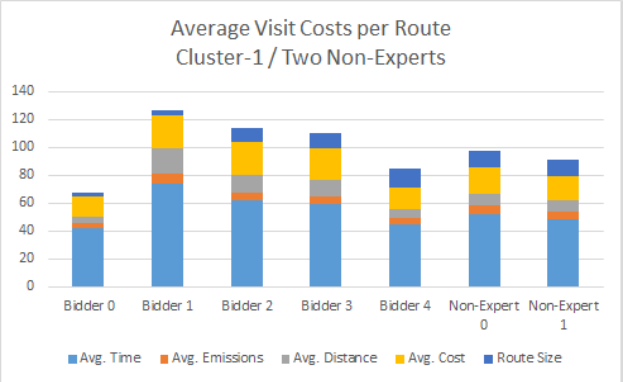


Figure 21: **Cluster-1 with Two Non-Experts** - Non-experts did not create great routes but they were not the worst bidders in the solution.

Two non-experts on the Cluster-1 problem did not create routes that were exactly good or bad. They fall around the mean in terms of average visit cost. Ignoring the routes with very few visits in them, Figure 21 reveals that the routes of the two non-experts likely did negatively impact the solution. Compared to Figures 19 and 20, the solutions for the non-experts noticeably worse as expected from the cost of the solution as a whole.

4.2.3 Lon-1

For the problem Lon-1 with one expert, the route created by the player is very similar to the average of the routes created by the computer bidders.

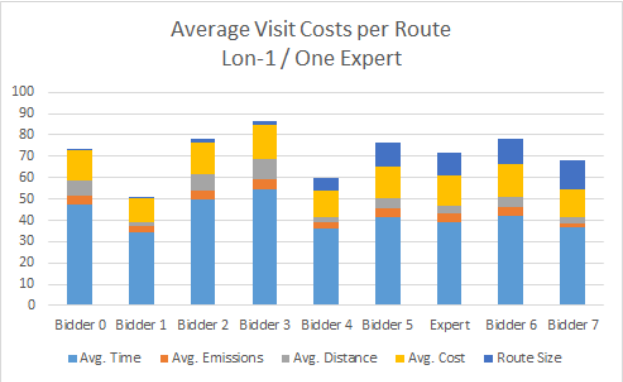


Figure 22: **Lon-1 with One Expert** - The expert produced a route that was not necessarily bad but in the end, increased the cost of the solution by affecting the other bidders.

Comparing 22 and 23, the expert was definitely better than the non-expert but neither produced an efficient route. The non-expert produced an exceptionally bad route. Their route was the second worst in their solution with a total average very close to the worst in the solution containing the expert. The routes in both solution are very similar in cost for the most part (if you account for the scaling of the vertical axis). It would appear that the non-expert caused 'Bidder 1' to create a very inefficient, small route with almost double the average cost of 'Bidder 1' in the expert solution.

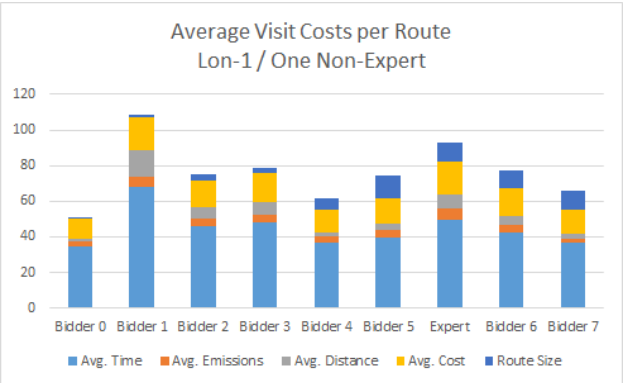


Figure 23: **Lon-1 with One Non-Expert** - An expensive route was created by the non-expert and appears to have also impacted the other bidders negatively.

4.2.4 Summary: Average Visit Cost per Route

By taking a closer look at the individual routes of each solution, it was found that the routes created by the players were not necessarily the issue. The cause of the increase or very subtle change in solution was due to the interaction between routes. Although players may have developed more efficient routes than the bidders, the bidders themselves ended up with less efficient routes as a result of competition with the players.

Enabling the institution for the most part had a negative impact on the routes of the players. The Non-experts were unlike the experts in that they did not appear to be affected by the institution. This could be caused by the non-experts ignoring the highlighted visits or by filling their routes before reaching them. With more participants, the institution could have been investigated further to determine if the selection of visits needed some improvement.

As mentioned in Section 4.1.4, the introduction of players to a vehicle routing problem appears to have very little impact on overall solution cost. By looking at the averages of each route, it is clear that there is some form of balance between the players and the computers resulting in a near identical solution (at least in the case of experts).

Relating back to the research questions proposed in Section 1.3, the average visit cost per route reveals some information that was not uncovered by solution cost alone. As mentioned, there seems to be a balance between the computer and player bidders. This means that players can replace the computer bidder but will not necessarily improve the solution (Research Question 1). As seen from the results in this section, the player can have a strong impact on the routes of the other bidders which leads to a balance maintaining steady solution costs (Research Question 2). The third research question was answered from solution cost alone but the average visit cost per route gives more clarity to why the solutions of the non-experts are worse than those of experts. The main reason the solution costs of the non-experts are not as good as the solution costs of the experts is due to the higher average cost for a visit (Research Question 4). As for the fifth research question, the institution had a minimal impact on the cost of a player's route but the change in the route of the player resulted in the non-player's routes being more expensive.

4.3 Results Summary

The results that were produced did not match the original expectation laid out prior to implementation. It was found that overall the solution costs were generally worse when two expert players were used alongside non-player bidders. It is believed that this is potentially due to a conflict of interest

between the participants. Their routes cross over a fair amount meaning that they could have lead to each others routes being suboptimal.

Two non-expert players alongside non-players resulted in solutions that were always worse that the solutions found by all non-player bidders. Non-experts were expected to be similar to the bidders but in actual fact they negatively impacted the solution.

The results for one expert player was more promising than the results of the other groups. In most cases, the results were slightly better than those of all non-players. The difference between solution costs were still not as significant as expected though.

With further research into the cause of why the solutions did not drastically improve, the finding for average visit cost per route were interesting. There appeared to be some form of balance between both the players and non-players. This was the case specifically for the expert groups where if an expert created an efficient route, the bidders routes were slightly worse and vice versa. This shows some relation between the routes and that they do in fact impact each other. As a result there was very little change to solution costs meaning that players can be inserted but not really for improving costs, there would need to be some other motive such as preferential decisions.

Solutions created by the non-experts were found to be worse than both the computer and expert solutions. This was as expected as the non-experts did not have much prior in the area. Although, in most solutions it was found that all player routes were similar in cost to those of the non-players. Expert's routes were usually better than the average non-player bidder while non-experts ranged between slightly better and slightly worse. With the addition of the institution, solution costs went up and variance in the average visit cost per route was reduced. The change in cost was minimal and likely could be improved if the selection of positive and negative visits was better. Investigation into the selection of these visits is outwith the scope of this project.

5 Evaluation

Through the experiments ran using the implemented system, the research questions in Section 1.3 were answered. Surprisingly some of the findings (which can be seen in Section 4) were not as predicted and as a result the answers to the research questions were unexpected.

For the first research question, replacing computers with people was expected to improve the solution costs significantly. In Section 4.1 it was found that the solution cost as a whole was almost unchanged from a non-player solution. With non-experts it was found to be worse but that was expected to some extent. Further investigation was done (see Section 4.2) to see why the solution costs did not improve. This proved useful in supplying a greater understanding as to why the players had little impact on the solution costs.

The answer for the second research question was as anticipated. Players were found to have a strong impact on other routes whether it be two players competing for visits or a player and a computer. The findings in Section 4.2 showed that there was a balance between the routes of the players and non-players. If the player created a good route then one or more bidders would create a bad route as a result and vice versa.

As with second research question, the third was answered as expected. The solutions involving experts were always found to be better than those with non-experts. If the results in Section 4 had shown otherwise, this would have been a confusing but very interesting outcome.

In Section 4.2 it was found that the average cost per visit for each route was closely connected. As a result, the fourth research question does not necessarily have a strict answer. For example if the player develops an efficient route then they will have an average much lower than the non-players. Although if the player develops an inefficient route, then their average will be higher than most computer bidders. The average visit cost for a player is entirely determined by how well they do but in most cases it can be said that the player will produce a route that has a lower or equal average to that of a non-player.

For the fifth and final research question, it is believed that more investigation needs to be done to determine if the institution can improve the player's routes. The results found so far indicate that with the experts their routes get slightly more expensive. As discussed in Section 4.3 this could be down to the selection of the visits being modified by the institution. As a result, investigating selection methods could lead to more efficient selections that reduce solution costs. One interesting discovery was found relating to the non-experts with the institution. Their results showed little change as if for the most part they just ignored the modified visits or filled their routes

too early.

The results that were gathered required the participation of people to use the system. Because of this, the system was to some extent user tested. No issues in the functionality of the system were found during the experiments and all results were gathered successfully without any problems. On that note, there were some minor bugs that were found but none of which affected the users ability to use the system. These bugs were graphical such as the built-in tool tips for a JFrame leaving strange artifacts on the window. This was likely due to the automatic repaint function being disabled so that the player agent had full control over when to render.

After the experiments, there was an issue found in the outputted log file from each set of two player results. Luckily the log files were completely salvageable, some data was just offset incorrectly in the csv file because one agent's log was longer than another. This could be resolved by printing blank entries if one agent's log file is shorter than another.

The system can be ran without the use of a player but still use most of the functionality (minus the player gui and agent). Because of this, the system was ran numerous times on a variety of problems even those not ran for results to guarantee that it would run without error. The system never ran into issues in this case. This of course does not test all of the system as the player is excluded but gives a good reference on how stable the majority is. With more experiments, the stability of the entire system could be proven further.

The experiments did not go exactly as planned due to a smaller number of participants than originally expected. In response to this, fewer problems were ran on the system and the problems were reordering to obtain the most necessary results. As there were less participants than anticipated, the problems were also limited in how many people were playing at once. It would have been interesting to see the results of an all player solution but this was not feasible with the available people.

As stated by some of the experts during the experiments, it would have been useful to be able to reorder the route after visit acquisition. If this was available to them then the expert's results could have potentially been better. This could have even significantly lowered the solution costs. If this was implemented though, the computer bidders would also require the logic to do this themselves otherwise the comparison would be unfair. The underlying framework did originally do this during an earlier stage of its development but it caused the system to take an exceedingly long time to run. This falls under the area of Future Work and as such is discussed in Section 6 along with other related topics.

6 Future Work

A key limitation for this project was the number of participants. Ideally, there would have been enough participants to run a problem with all bidders being players. This was not feasible due to the number of people willing to take part. If more people were available then more result would have been gathered which could lead to greater insight into the reasons why the solutions did not improve. More participants would have also allowed for more of the problems to be ran as the results in Section 4 do not over all of the problems that could have been ran.

As stated by two of the experts during the experiments, it would be beneficial to be able to reorder the visits within a route. This would allow the players to make their routes more efficient in light of new information such as acquiring a visit that they do not feel is in the correct place in their route. This would require the computer bidders to have the same functionality for the results to be fair. If the player could reorder the route but the computers could not then if the results showed an improvement, it could be due to the reordering alone and nothing to do with the fact players were involved.

Returning visits would be a good addition to the player agents especially alongside route reordering. This functionality is already implemented in the underlying framework to some extent but is unused for the purpose of this project. If the player could return visits then it would be more likely that they would reach the end of the bidding process without filling their routes. Currently the non-expert players bid for almost all visits that they are presented. This leads to very inefficient routes but if they could return them then they could potentially develop lower costing routes.

With the results obtained from the group of two experts, there was a clear competition between their strategies. Both experts were essentially fighting over the same visits and ended up with routes that crossed over a lot (see Figure 15). To resolve this, the rendering code could be updated to show a representation of the other bidders routes. In addition to the rendering code being updated, additional messages would be required to pass this information between all bidders. By allowing bidders to see each others routes, they can choose to actively avoid one another.

The institution was found to be ineffective at reducing the cost of the solutions. Further investigation into the selection of visits to be modified by the institution could prove beneficial in meeting the expectations set prior to the project. Additionally, supplying the users with information about the net multiplier value of these visits before the bidding process reaches them could influence whether or not the player will wait for said visit.

After finding that players have little impact on the solution costs, it would

be interesting to see how well preferential data could be processed by the players. The institution could be extended to contain net multipliers for each route/driver. This could then be used to indicate to the user, information such as the driver's preferred area to work and the customer's preference in driver.

The computer bidders supplied by the underlying framework are able to handle problems which allow them to use more than one mode of transport. This was not used within this system as the aim was more to investigate visit selection and ordering rather than the additional parameters such as transport mode. With the findings showing that the players have little impact on the solution costs, it would be interesting to see how transport selection being controlled by the player would affect the solution costs.

As stated earlier, there was a limitation inflicted by the number of participants. If there had been more, other factors would have been worthwhile investigating. This is by no means the most important of them all mentioned in this section but it could have a strong impact on the solutions. As the system is, every time a problem is ran, the order the visits are bidded on is fixed. There is no pre-processing to reorder the visits prior to the bidding stage. The order is purely determined by the order in the problem file. Ordering methods such as closest to depot or farthest to depot first could sway the results positively or negatively. For the gameplay side of this project, a more randomly driven approach would be good for the re-playability. If there were enough players, a random approach could find an efficient solution over a number of runs. A random approach would require a lot of runs for the results to be valid though as the randomness itself could impact the solutions drastically.

A potentially useful application for this tool could be in prototyping. By using this to run through the pseudo code of an algorithm, researchers could get insight into the effectiveness of their algorithm before implementing it. Doing so could potentially save many hours of implementation by enabling a researcher to test and, if necessary, redesign the algorithm before even starting to implement it. The impact on the solution costs would likely be minimal as found from the current results. If however they are better than that of the current computer bidder, it is highly likely that replacing these bidders with the new algorithm would in fact reduce the solution costs.

7 Conclusion

The main objective of this project was to determine if people could replace computers in an auction based vehicle routing solver. It was believed that by doing so, the results would improve by a substantial amount. Market-based control elements such as currency were added along with a computational institution. These were intended as gameplay elements that would drive the human players towards efficient solutions. After implementing the system using the underlying framework and JADE[20], experiments were ran with the aid of participants who took the place of player bidders. The details of this process can be seen in Section 3.3. The detailed analysis of the results obtained from these experiments can be seen in Section 4.

As a whole, the introduction of people to the system was expected to reduce the solutions costs. More specifically, the expert players were assumed to be significantly better than the computers while the non-experts were expected to be similar to the computers. The results produced were intriguing, showing that replacing computer bidders with people did not have the expected outcome. Rather than improving the solutions, in some cases they got noticeably worse. This was always the case for the non-expert players. The expert players occasionally produced better results but not by a very large margin.

With further investigation into the solutions it was found that player bidders had a strong impact on the routes of the other bidders. This resulted in solution costs balancing around those already found by the non-player bidder solutions. Good player routes resulted in bad non-player routes, bad player routes resulted in good non-player routes. For the most part, the expert players consistently produced routes with a much lower average visit cost than the non-players. This suggests that the experts were developing more efficient routes but the solution costs as a whole was almost unaltered due to the impact they had on other routes.

From the current findings, the computational institution supplied no positive impact on the solutions. It is believed that further investigation into the selection of visits for it to modify could change this. Non-expert results have shown little change when the institution is introduced implying that they either ignore the incentives or develop a completed route before the affected visits are to be auctioned.

Although the results were not as expected, the outcome is definitely interesting. The fact that people have very little impact on the solution costs implies that people could be used in such system for other means. As mentioned in Section 5, the small impact on solution costs could allow for human computation as a method of handling data that is otherwise more difficult

to process such as preferential data.

The findings of this project open up more opportunities for investigation in the area of games as a tool for vehicle routing. A drastic improvement in the solutions would have been great but the consistently low impact brought on by player bidders is potentially more useful. If the solutions could be far better than those produced by the computers, they could also be far worse. As the solutions are almost unchanged no matter what, this allows for research into other uses of human computation in the area without the potential to ruin solutions.

References

[1] Luis von Ahn and Laura Dabbish. "Labeling Images with a Computer Game". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '04. Vienna, Austria: ACM, 2004, pp. 319–326. isbn: 1-58113-702-8. doi: 10.1145/985692.985733. url: http://doi.acm.org/10.1145/985692.985733.

[2] Saleema Amershi et al. "Power to the People: The Role of Humans in Interactive Machine Learning". In: *AI Magazine* (Dec. 2014). url: https://www.microsoft.com/en-us/research/publication/power-to-the-people-the-role-of-humans-in-interactive-machine-learning/.

[3] Albert D. Baker. "Market-based Control". In: ed. by Scott H. Clearwater. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 1996. Chap. Metaphor or Reality: A Case Study Where Agents BID with Actual Costs to Schedule a Factory, pp. 184–223. isbn: 981-02-2254-8. url: http://dl.acm.org/citation.cfm?id=225624.225649.

[4] D. Barbucha and P. Jedrzejowicz. "Multi-agent platform for solving the dynamic vehicle routing problem". In: *2008 11th International IEEE Conference on Intelligent Transportation Systems*. Oct. 2008, pp. 517–522. doi: 10.1109/ITSC.2008.4732573.

[5] Luke Barrington, Douglas Turnbull, and Gert Lanckriet. "Game-powered machine learning". In: *Proceedings of the National Academy of Sciences* 109.17 (2012), pp. 6411–6416. doi: 10.1073/pnas.1014748109. eprint: http://www.pnas.org/content/109/17/6411.full.pdf. url: http://www.pnas.org/content/109/17/6411.abstract.

[6] Giacomo Cabri, Luca Ferrari, and Rossella Rubino. "Building computational institutions for agents with RoleX". In: *Artificial Intelligence and Law* 16.1 (2008), pp. 129–145. issn: 1572-8382. doi: 10.1007/s10506-007-9058-9. url: http://dx.doi.org/10.1007/s10506-007-9058-9.

[7] Seth Cooper et al. "The Challenge of Designing Scientific Discovery Games". In: *Proceedings of the Fifth International Conference on the Foundations of Digital Games*. FDG '10. Monterey, California: ACM, 2010, pp. 40–47. isbn: 978-1-60558-937-4. doi: 10.1145/1822348.1822354. url: http://doi.acm.org/10.1145/1822348.1822354.

[8] F. C. Fernandes et al. “A multiagent architecture for solving combinatorial optimization problems through metaheuristics”. In: *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*. Oct. 2009, pp. 3071–3076. doi: 10.1109/ICSMC.2009.5345934.

[9] Jiaqi Ge and Gary J. Polhill. “Exploring the Combined Effect of Factors Influencing Commuting Patterns and CO2 Emissions in Aberdeen Using an Agent-Based Model”. In: *Journal of Arti cial Societies and Social Simulation* 19.3 (2016), p. 11. issn: 1460-7425. doi: 10.18564/jasss.3078. url: http://jasss.soc.surrey.ac.uk/19/3/11.html.

[10] B.L. Golden, S. Raghavan, and E.A. Wasil. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Operations Research/Computer Science Interfaces Series. Springer US, 2008. isbn: 9780387777788. url: https://books.google.co.uk/books?id=-3ta5ne3-owC.

[11] R. He et al. “A Route-Nearest Neighbor Algorithm for Large-Scale Vehicle Routing Problem”. In: *Intelligent Information Technology and Security Informatics (IITSI), 2010 Third International Symposium on*. Apr. 2010, pp. 390–393. doi: 10.1109/IITSI.2010.144.

[12] Petr Kalina, Jiří Vokřínek, and Vladimír Mařík. “Agents Toward Vehicle Routing Problem With Time Windows”. In: *Journal of Intelligent Transportation Systems* 19.1 (2015), pp. 3–17. doi: 10.1080/15472450.2014.889953. eprint: http://dx.doi.org/10.1080/15472450.2014.889953. url: http://dx.doi.org/10.1080/15472450.2014.889953.

[13] Elad Kivelevitch, Kelly Cohen, and Manish Kumar. “A Market-based Solution to the Multiple Traveling Salesmen Problem”. In: *Journal of Intelligent & Robotic Systems* 72.1 (2013), pp. 21–40. issn: 1573-0409. doi: 10.1007/s10846-012-9805-3. url: http://dx.doi.org/10.1007/s10846-012-9805-3.

[14] R. Mehta et al. “Market based multi-agent control of microgrid”. In: *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on*. Apr. 2014, pp. 1–6. doi: 10.1109/ISSNIP.2014.6827704.

[15] S. K. Nambiar and Sumam Mary Idicula. “A multi-agent vehicle routing system for garbage collection”. In: *2013 Fifth International Conference on Advanced Computing (ICoAC)*. Dec. 2013, pp. 72–76. doi: 10.1109/ICoAC.2013.6921930.

[16] A. Pustowka and E. F. Caicedo. "Market-Based Task Allocation in a Multi-robot Surveillance System". In: *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*. Oct. 2012, pp. 185–189. doi: 10.1109/SBR-LARS.2012.37.

[17] Mingyao Qi et al. "A spatiotemporal partitioning approach for large-scale vehicle routing problems with time windows". In: *Transportation Research Part E: Logistics and Transportation Review* 48.1 (2012). Select Papers from the 19th International Symposium on Transportation and Traffic Theory, pp. 248–257. issn: 1366-5545. doi: http://dx.doi.org/10.1016/j.tre.2011.07.001. url: http://www.sciencedirect.com/science/article/pii/S1366554511000846.

[18] Ichiro Shigaki and Masami Konishi. "Decentralized Probabilistic Algorithm Using a Multi-Agent System for Vehicle Routing Problems". In: *International Journal of Smart Engineering System Design* 5.4 (2003), pp. 241–249. doi: 10.1080/10255810390245564. eprint: http://dx.doi.org/10.1080/10255810390245564. url: http://dx.doi.org/10.1080/10255810390245564.

[19] Jingyan Song et al. "Re-optimization in dynamic vehicle routing problem based on Wasp-like agent strategy". In: *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.* Sept. 2005, pp. 231–236. doi: 10.1109/ITSC.2005.1520131.

[20] tilab. *Java Agent DEvelopment Framework (JADE)*. 2016. url: http://jade.tilab.com/.

[21] Jinxin Yi. "Vehicle Routing with Time Windows and Time-Dependent Rewards: A Problem from the American Red Cross". In: *Manufacturing & Service Operations Management* 5.1 (2003), pp. 74–77. url: http://EconPapers.repec.org/RePEc:inm:ormsom:v:5:y:2003:i:1:p:74-77.

Appendices

A Project Overview

Grant Smith 40111906

Initial Project Overview

SOC10101 Honours Project (40 Credits)

Title of Project: Investigation into Optimization of Agent-based VRP using User Input via Game Interface

Overview of Project Content and Milestones

This project aims to investigate the research application of user interaction with an Agent-based vehicle routing solver. An interface representing a game will be developed for an existing system and results from the user interaction will be gathered. These results will be evaluated to determine the effectiveness of such an approach. This evaluation will include comparisons to the results of the existing system to show the effect of user interaction. Market-based control will be used to add incentives for the agents and the user. The user will control an agent in attempt to achieve better routes than the other agents. If the user achieves the best or one of the best routes then their interactions will be logged for further research which could include improving the agents.

The Main Deliverable(s):

- A, simplistic (unless there is time), game for researching the use of people to offload decision making within the area of vehicle routing.
- An evaluation on the effectiveness of the approach and the corresponding results.

The Target Audience for the Deliverable(s):

This project will be useful to researchers in the area of vehicle routing as it will supply researchers with a tool for investigating the use of user interaction in the area.

The Work to be Undertaken:

- Collection of the results produced by the current system.
- Design the game for user interaction
- Design the Market-based control features of the bidding process
- Implement basic version of interface for user interaction
- Implement the Market-based control features
- Update the interface for Market-based control
- Test the system
- Collect results produced
- Compare and evaluate the results

Games Development 15/09/2016

Additional Information / Knowledge Required:

Knowledge of the Java Agent Development Framework (JADE) and vehicle routing using agents. Additional information from research into the areas of vehicle routing, market-based control, and gameplay mechanics. This project also requires vehicle routing software supplied by Dr Neil Uruqhart.

Information Sources that Provide a Context for the Project:

The Java Agent Development Framework (JADE) is a framework implemented in Java to simplify the implementation of multi-agent systems. JADE is used by the existing system to enable many agents to be creating, each managing the bidding and acquisition of visits for a route. By using multiple agents, a vehicle routing problem can be solved by utilising all the cores on a system.

Market-based control (Scott H. Clearwater) supplies an understanding of applying competitive interactions between agents in a number of areas ranging from mock market to managing the work schedule at a factory. The logic of the bidding process could have aspects that follow similar ideas found in this book, namely the component of relating the costs of tasks to real world costs.

The Importance of the Project:

Implementation of Market-based control into the bidding process should improve upon the overall results of the existing system. This will be due to the agents being self driven towards obtaining the optimal results through competition with other agents and a penalization and reward system for agent actions. The interface will supply an area for further research into the application of user interaction through games in attempt to optimise vehicle routing.

The Key Challenge(s) to be Overcome:

The most challenging part of this project will likely be the designing of the Market-based control features and obtaining users to test the game component of the project.

B Second Formal Review Output

C Diary Sheets

Week starting Details

12/9/2016

Discussed the IPO with my supervisor at our first meeting. Went over the contents of the document such as the milestones, deliverables, audience, and some of the work to be undertaken.

19/9/2016

Work on the Initial Project Overview (IPO) document. Followed by a meeting with my supervisor to ensure the contents of the IPO met the specification.

26/9/2016

Research for the literature review.

- Market-based control with notes
- JASSS article with notes
- Gamification - J. Dale with notes
- Stored PDFs of more articles to read

Meeting with supervisor.

3/10/2016

Further research for the literature review including some write up

10/10/2016

Made a slight modification to the project so that it matches my degree course.
Meeting with supervisor.

17/10/2016

More research including into new areas covered by the change from last week.
More work on the literature review.
Meeting with supervisor.

24/10/2016

Write up for the literature using the research from prior weeks.

31/10/2016

Completed my dissertation draft and sent it to my supervisor on the Wednesday.
Had my interim review meeting on the Friday.

7/11/2016

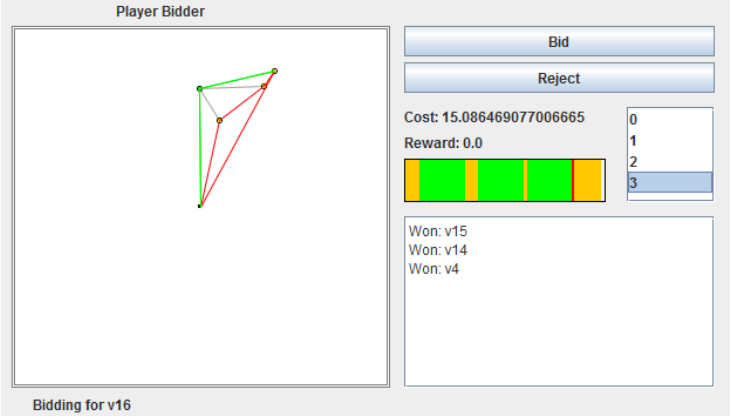
Gathered baseline results using the updated journeyData file from my supervisor.
Meeting with supervisor.

14/11/2016

Created a simple player bidding agent; the player can select whether they want to bid or not and where to insert the visit if they win it.
The GUI for this currently only shows a scaled version of the agent's route and some input options. A representation of the route's scale in comparison to the overall problem is needed as it is hard to determine how large the route it during creation.

Updated the GUI:

- The route scales relative to all visits seen, not just the ones owned
- The GUI now shows the cost of adding a visit
- The GUI now shows the timings of the route in a simplistic/abstracted manner



Managed to improve upon the baseline using one player and the rest as agents. The first row is the baseline, all bold entries below are improvements compared to that row. All of these results are for the problem ób lon-1ô with 10 routes, 110 visits, 5 return limit, car as the transport mode, and minimising cost.

Cost	Emissions	Time	Distance	
1955	635	5300	790	All Agents (Baseline)
1961	626	5278	781	One Player
1957	634	5306	792	One Player
1935	630	5213	755	One Player

21/11/2016

Updated the system to show the player all of the visits and which are still available. The player agent now has a balance and gains a reward set by the auctioneer and pays depending on the cost for adding to the route.

The representation of the route timeframe needs to be updated to show the route in terms of the available time and not just the relative time of each visit to the route.

The bidding should be updated according to the notes below after the discussion with my supervisor.

Notes

Work on the bidding. Make reward the auctioneer's limit for a visit. Make cost the default bid but allow the player to bid lower or higher (if cost is higher than the limit then use limit as the default). Bid low to win more visits and high to get a better balance. When a bid is won, the cost of that bid is subtracted from the balance. May be beneficial to accept one visit at a loss in attempt to collect a cluster of visits at a profit.

Start all agents with a balance of 0 and they should aim to be in profit by the end. Largest profit is expected to correlate with the better routes.

12/12/16

- Updated time frame rendering
- Improved the UI so make it easier to understand
- Updated player bidding
  - Max bid set by auctioneer
  - Adjustable bid
  - Suggested bid from cost of adding
- Balance is adjusted depending on the adjusted bid - cost of adding (profit and loss is possible)
- Balance starts at 0

2/1/17

- Implemented the start of a log for the player bidder (needs to be combined at the auctioneer).
- Updated the library being used and my code to enable the bidders to be ran in separate JVMs. This should also mean that bidders can be ran on separate machines but has not been tested yet.
  - The AID of agents is now acquired using the DF Service rather than creating a new AID when starting a communication. Doing so leads to the ability to communicate across JVMs (and should also allow for cross machine communication).

- Updated bidders so that they can be reused and consequently updated the player bidder to make use of this (multiple problems can now be ran).
- Bidder arguments were updated in the library to be received as a message. Bidder agents can now be created outwith the auctioneer as they do not need to be passed parameters such as minimiseFactor when being instantiated. This helped with the implementation of the networking functionality as the previous method of setting the arguments would not have allowed for reusability and instead agents would need to be destroyed and recreated.
- Bidder agents now store the AID of the Auctioneer, CarShareServer, and MapServer so that they can communicate with these agents without creating an AID each time which would require the Bidder to know the ip of these agents.

9/1/17

- Players now send their log to the auctioneer when finished. The auctioneer then combines all logs into one file and saves it (using the problem name and timestamp to uniquely identify them)
- Players are now prompted for an IP address when executed rather than connecting to localhost.
- Fixed a bug where bidders did not terminate when their window was closed.

16/1/17

- Users Strategy is now added to the log file
- Updated the saving of the files resulting from a completed problem, matching how the new log file saves, so that multiple of the same problem can be ran without overwriting each other. This is necessary for the result gathering that is planned.

23/1/17

- Started implementing the computational institution

30/1/17

- Continued working on the computational institution and finished of its functionality including a GUI. The GUI needs to be improved to make it more usable at this stage.

6/2/17

- Updated the Institution including the visit selection method so that you can click on visits you wish to alter rather than scrolling down a list of all available visits. It is now much easier to use.

13/2/17

- Make changes to the player bidder to make it easier to understand for someone who has not used it before
  - Additional labels and tooltips were added to explain elements of the interface
- Implemented a Ranking system based on the bidder's balance
- Create a small document to explain how to use the system and the aim
  - This document was given to the users for the result gathering and they were asked to read through it before the program was ran.
- 10 sets of two player results were gathered.
  - Blon-1 - Car - Cost - 11 Bidders (Dry Run)
    - 3 Groups Completed
  - Blon-1 - Car - Cost- Institution - 11 Bidders
    - 3 Groups Completed
  - Cluster-1 - Car - Emissions - 7 Bidders
    - 3 Groups Completed
  - Cluster-1 - Car - Cost - Institution - 7 Bidders
    - 1 Group Completed

20/2/17

- Took some new baseline results to match the specifications of the problems ran last week.
- Wrote a tool using processing (processing.org) to process/view the results.
  - Currently shows:
    - Routes
    - Totals
      - Time
      - Emissions
      - Distance
      - Cost
    - Bar Chart Summaries
- In hindsight I would have used processing to develop the frontend for the player as it abstracts a lot of the excess code required by java to draw basic objects to the screen. This would have made the code for rendering much shorter and easier to read.
- From the analysis so far:

  The results seem to show that the bidder is better than the player in some cases. This appears to be where the visits are already in defined groups such as those in the cluster problem type. There could be room for improvement in the bidder's ordering of the nodes but that is hard to tell with the current results as most players go for more than one cluster while the bidders only go for one. If players were to only go for one cluster and allocate their visits efficiently, there may be room for improvement.

  With other problem types where there is less structure to the positioning of the visits, players tend to do better than the bidders. In some cases the difference is quite significant. This is likely due to the bidders being given less direction as

they are not forcefully separated due to visit distribution. Players do better as they are able to pick out areas within the problem space that they believe could be connected efficiently.

27/2/17

- Further analysis of the results gather from previous weeks and of new one player results.

6/3/17

- Worked on the methodology section of the dissertation

13/3/17

- Started work on the poster for the poster presentation

20/3/17

- Poster presentation and work on honours dissertation

27/3/17

- Work on methodology section of the dissertation

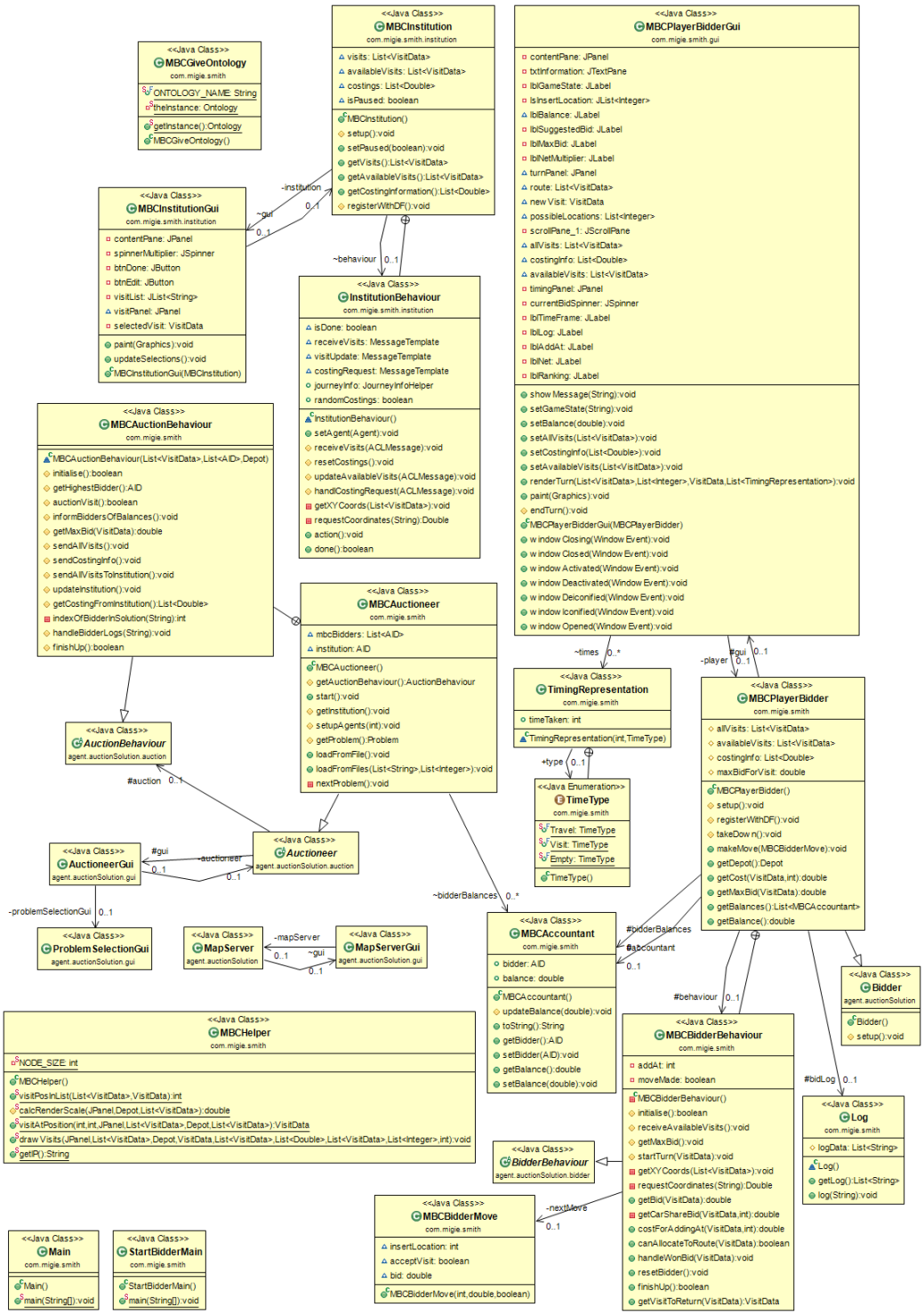
3/4/17

- Work on results section of the dissertation

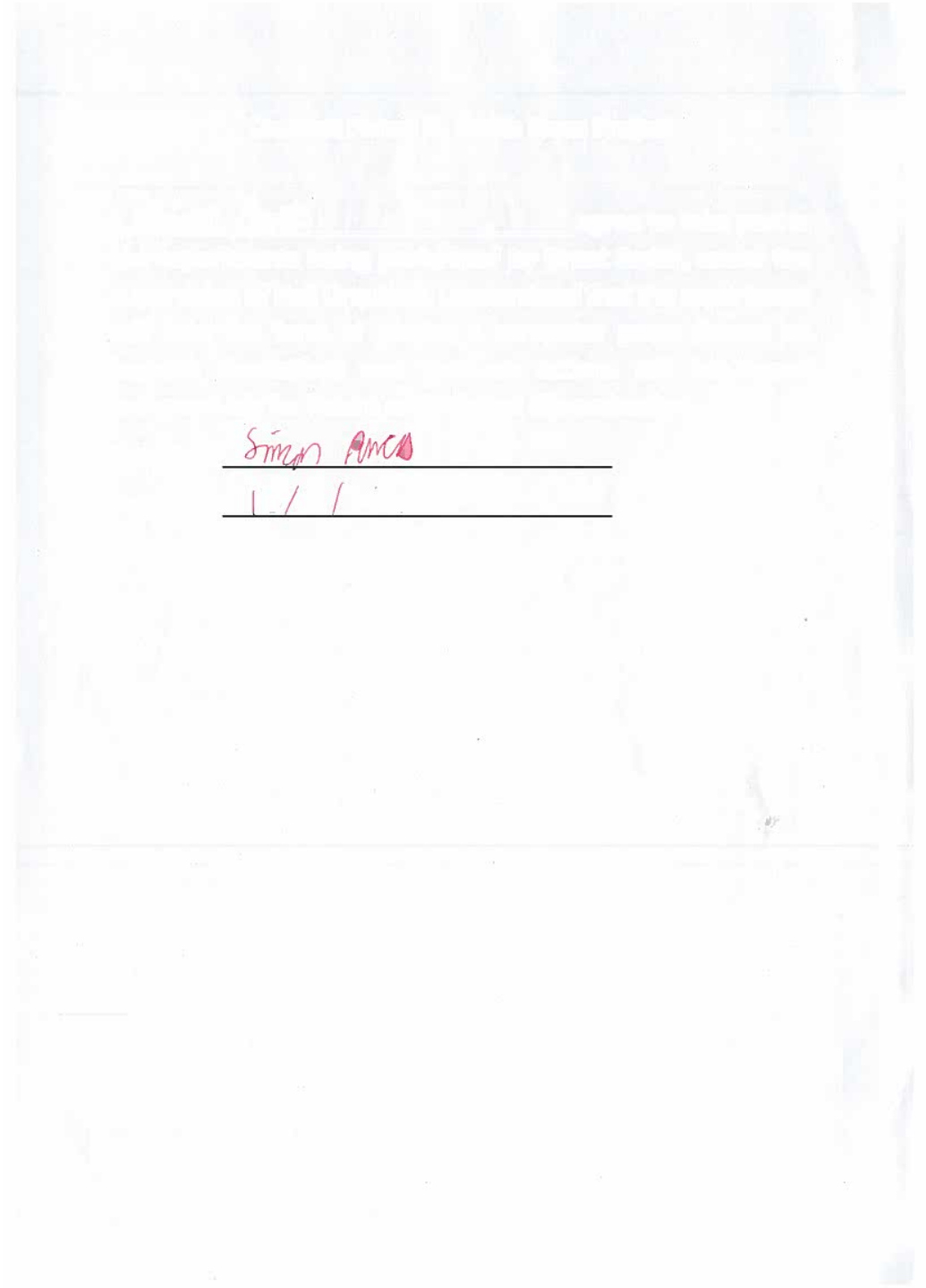
10/4/17

- Further work on dissertation:
  - Evaluation
  - Future Work
  - Conclusion
  - Abstract

D Class Diagram



E Ethics Consent Forms



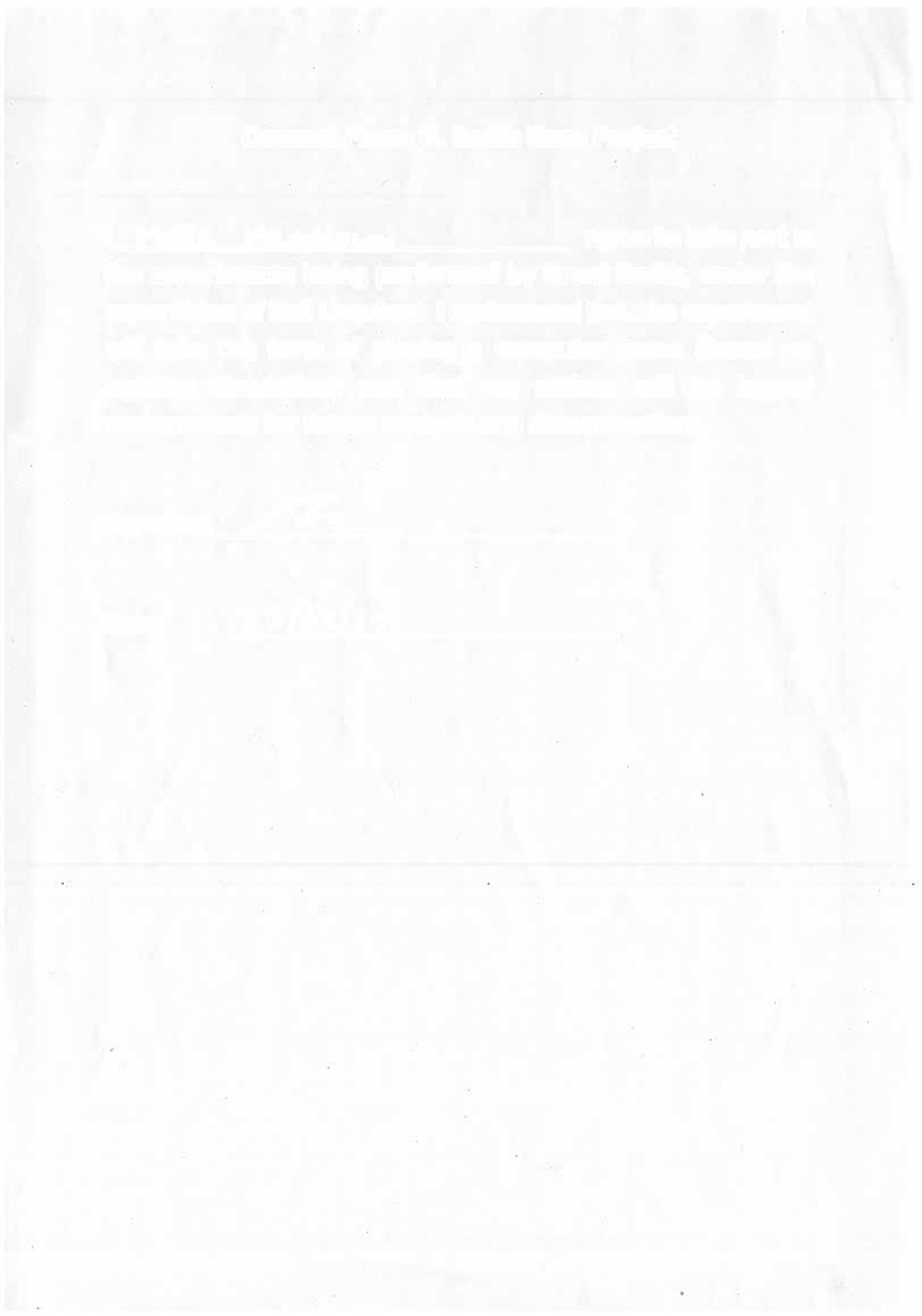
N L UROUCHART

N Unit

5/17

c

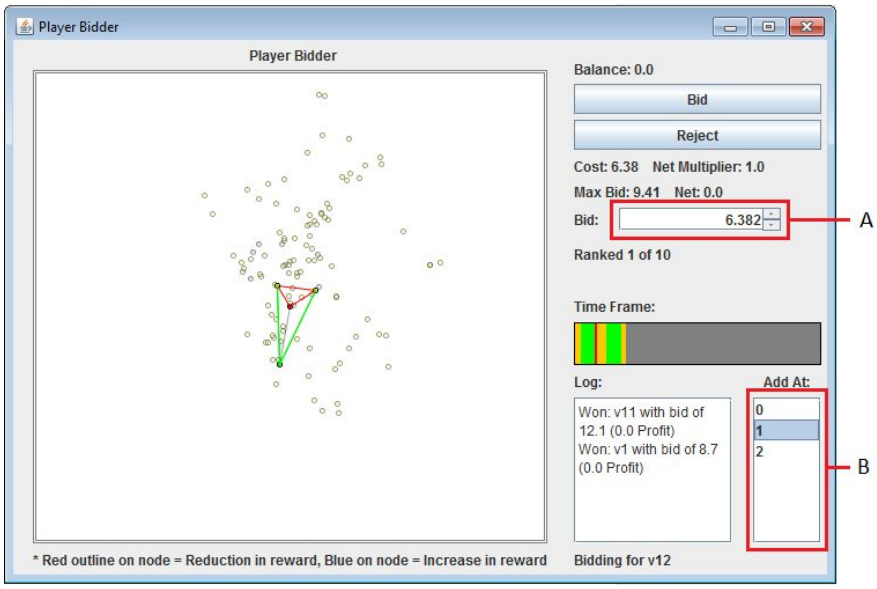
o 17



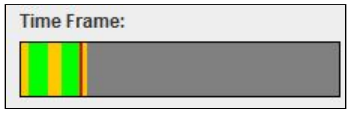
F Experiments Help Sheet

Bidding

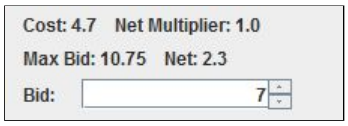
There are two options with each visit you are presented ôBidôand ôRejectô If you reject a visit, the values ôBidôand ôAdd Atôwill have no effect. If you want to bid for a visit, you set the value you want paid for the visit by modifying the value indicated by A in the image below. The lower this value is, the higher chance you have to win the bid but you will likely come out with a loss rather than a profit (this may be worth it in the long run).



By selecting from the list ôAdd Atô(marked as B in the image above), you can change where in your route the visit will be added if you win the visit. Changing this value will update the map on the left of the screen. Red lines represent your current route, Green show the connections for if you win the visit, and Grey lines are the other connections if you were to select a different item in the ôAdd Atôlist.



The element labelled ôTime Frameôindicates how much time you have left in your route. Grey is unused time, green is time spent at a visit, and yellow is travel time. The red line shows where the new visit will be added if it is won.



The aim is to obtain the highest balance out of all bidders. To do this, you want to increase your bid to gain a ÔNetôvalue greater than 0. The ÔNetôis calculated as :

Net = (Bid - Cost) * Net Multiplier

In most problems the Net Multiplier will be 1.0 but in some cases it will be higher or lower. In these cases, the Net multiplier will be indicated on the map using a blue or red outline. Blue outlines are those with a net multiplier greater than 1, this can give you a boost to your profit/loss (better to go for these with a high bid). Red outlines have a net multiplier less than 1, decreasing the profit/loss. Going for these can be beneficial if you want to bid at a loss to obtain a visit near a cluster. All other visits have a net multiplier of 1.