Introducing reactive BDI agents into MATSim
Honours Thesis

Student: Mr Qingyu Chen
Supervisors: Professor Lin Padgham and Dr Dhirendra Singh

November 2013

Abstract

Most agent based traffic simulation platforms use simplistic agent modelling approaches. The agents under these approaches cannot make any decisions or perform goal directed behaviour in a dynamic environment, which makes these simulation software unable to model complex traffic scenarios that involve making decisions to respond to the situated environment, such as an evacuation. This thesis uses Belief Desire Intention (BDI), a popular cognitive agent model and integrates it into an agent based traffic simulation platform in order to resolve the above problem. In particular, we embedded it into Multi-Agent Transport Simulation (MATSim), a popular agent based traffic simulation platform using a simple agent model. This provides an enhanced BDI-MATSim framework to support the functionality to model goal oriented behaviour and decision making in MATSim. BDI agent systems maintain a balance between goal directed behaviour and reacting to the environment. We have modified MASTim to allow ongoing reactivity, based on BDI decision making. We have illustrated this by successfully implementing a BDI taxi service embedded within MATSim.

Contents

1 Introduction

Simulating traffic scenarios involving complex decision making agents is becoming increasingly important. Nevertheless, most existing agent-based traffic simulation platforms use simplistic agents. This thesis integrates the popular Belief Desire Intention (BDI) cognitive agent model [36] using GORITE platform [38] into Multi-Agent Transport Simulation (MATSim) [34], a well-known transportation simulation platform. This integration will add the capability to model rich goal directed as well as reactive behaviours (such as evacuation and taxi services) in MATSim, which it currently does not have. It also allows MATSim to be leveraged for the analysis of human decision making in a fast changing environment.

MATSim is an agent based simulation tool which models traffic scenarios [1]. It is designed for finding stable traffic patterns over many iterations/days. In order to achieve this, its framework contains four important and successive processes [45]: *Demand modelling*, i.e. taking travellers (agents)' daily activities (plans) such as "at work" and "at home" attributes as the input to set up the simulation, *Simulating* each agent independently during each one day iteration, *Scoring* plans based on their quality, giving low mark for things such as arriving at the destination too late, and *Replanning* by gathering all of the lowly-scored plans and modifying them at the end of the iteration using genetic algorithms. It terminates on upon reaching a stable state, that is when all agents are no longer finding any improved plans.

Based on its current framework, MATSim is very good for analysing traffic patterns for large-scale scenarios. For instance, it was used for the analysis of the placement of a new Berlin airport[1]. However, it cannot be used for exploring problems such as evacuating from a flood, which requires immediate response or replanning during an iteration, because the current MATSim replanning process is conducted at the end of the iteration. The capability to explore these problems (called within-day replanning in MATSim) would allow MATSim to be used to address a wider range of issues. For instance, an earth quake that occurred at WenChuan, China in 2008, resulted in more than 68,000 deaths and the loss of hundreds of billions of RMB [44]. Simulation tools that support modelling of such scenarios would facilitate developing processes and infrastructure that could potentially save many lives. Likewise, within-day replanning can be used to simulate a taxi service where decisions need to be made directly on the fly based on the situated environment. For example, taxi drivers need to request jobs immediately when new jobs arrive. Hence, the ability to model taxis improves MATSim for use in scenarios where taxis could play a significant role in people's travel habits.

[1] http://matsim.org/scenario/berlin

A few attempts have been made previously to try to realise within-day replanning in MATSim but these have distinct shortcomings. Most of these attempts adopt the approach that collect agents and modify their plans centrally [16]. This approach has the limitation that it is not possible to model agents autonomously making decisions to change their plans as they go. In another approach, the agents are able to make their own decisions, as to decide whether they need to replan, but these agents are very simplistic and can only change their current route [22].

In order to support within-day replanning and overcome the shortcomings mentioned above, we bring the idea of BDI agent modelling [36] to MATSim. This mature behaviour model keeps the balance of pro-activeness (pursuing goals over time) and re-activeness (perceiving changes in the environment and responding to them), which are typically required in within-day replanning. Also, there are many platforms to support the BDI agent model, such as GORITE [38], JACK [4], and Jadex [33], which makes implementing the BDI agent model relatively easy. The potential of implementing a BDI agent model in MATSim has also been recognised by MATSim developers [16]. The authors suggest that agents can prioritise their desires (what to achieve) and make intentions (what they plan to do) based on the changing environment efficiently with a BDI agent model implemented. This advantage plays a key role in modelling large-scale within-day replanning scenarios, such as major flooding where the residents need to evacuate from the affected areas [6].

Work has previously been done on integrating the BDI agent model into the Repast agent based simulation [30] framework [31]. It proposes a general purpose mechanism for the integration between the BDI agent model and the agent based simulation, so we can use this mechanism for the current project to provide an enhanced MATSim framework incorporating the BDI agent model to support within-day replanning. This framework is generic so it can be adapted to different types of applications. In addition, we have implemented a concrete example of a taxi service using the enhanced framework for demonstration. This example illustrates the successful integration of BDI into MATSim and it shows that an agent embedded with the BDI agent model can make more complex decisions than simply changing the route which was made by the previous work [22]. For example, the taxi agent in our example can decide to request new jobs, reason about which job to take next, pick up passengers and drive them to destinations. Our implemented taxi model has been tested to run in MATSim with about 16,000 total agents, consisting of about 1,000 taxi agents.

In summary, we have made the following contributions:

1. Developed an enhanced MATSim framework through the use of the BDI agent model. To our knowledge, this is the first attempt to integrate an advanced behaviour model into MATSim.

2. Modified MATSim to allow the agents to make autonomous decisions during execution based on the situated environment.

3. The integration of BDI into MATSim is generic. Hence it can be used for other applications. For instance, the enhanced framework can be used to implement an evacuation simulation.

As explained above, the BDI agent model is adopted to support within-day replanning in MATSim. This involves the integration of these two different paradigms. They use unique ways to represent the agent – the BDI agent model uses Goals and Plans to describe what agents want to achieve and how to achieve these respectively, whereas MATSim uses only Plans which consist of activities to represent a similar concept. Hence, it is important to map the representation of BDI agents to the current MATSim architecture during the integration. In addition, both BDI and MATSim have their own execution cycles. Thus the integration must ensure synchronisation between these two different execution cycles. Furthermore, we want to implement a taxi model in MATSim integrated with the BDI system to demonstrate that it can realise within-day replanning, so we need to define what specific functionality is required in a taxi application. All of these lead to the following three research questions:

1. How to map the representation of BDI GORITE agents into the current MATSim architecture?

2. How to integrate the execution of BDI GORITE agents with the execution of MATSim?

3. What specific functionality is needed for the taxi application?

The remainder of the thesis is organised as follows. In Section 2, we review related work, including introducing the existing traffic simulation models and associated applications, describing the background of MATSim (especially the state of art in within-day replanning and taxi agent modelling) and outlining different approaches to model agents with the focus of introducing the BDI agent model. In Section 3, we describe the method we have explored to integrate BDI into MATSim and the enhanced BDI-MATSim framework. In Section 4, we describe how the system successfully realises the within-day replanning functionality. We also summarise our findings and indicate areas of future work.

2 Literature review

In this section we review related work. Firstly we provides a high level view of the traffic simulation models with well-known tools that implement them. In particular, we provide detailed information for the MATSim agent-based

traffic simulation platform. After that, we survey the approaches to modelling individual agents (simple and cognitive) in agent-based traffic simulation, and introduce the BDI cognitive agent model. Finally, we describe the existing work on integrating cognitive agent models into agent-based traffic simulations.

2.1 Existing traffic simulation models and tools

Traffic simulation models are usually categorised into two groups: Agent based models and Macroscopic models [28, 5]. These two groups have distinct focuses and thus their associated applications are used to address different problems in traffic simulations. The descriptions for both groups are as follows.

Agent-based models, also called Microscopic or Individual-based models, focuses on modelling entities (such as travellers and vehicles) in traffic simulations individually [24]. They use agents to model entities that can make decisions autonomously based on a specific situation [26]. In contrast, Macroscopic models emphasise simulating the entire traffic flow rather than individuals [24]. They focus on describing the characteristics of the whole traffic flow such as spatial vehicle density and average velocity [21].

Due to these unique focuses, both models have strengths and weaknesses. Agent-based model is advantageous in modelling the traffic scenarios where the capability to model each individual is necessary [3], such as evacuation(For instance, some people may leave these houses and drive to safe areas immediately, whereas others may stay at the houses and use sandbags to prevent their houses from being flooded) and taxi service (For instance, taxi drivers may decide whether to take jobs based on their specific situations, such as whether they are free, or close to passengers' destinations.). Well-know transport simulators which adopt this model include PARAMICS [7] (a powerful tool for analysing driver behaviour control on a freeway and car parking in urban networks[2]), AIMSUN[3] (a popular tool for analysing things such as dynamic traffic assignment and traffic management strategy decision-making) and MATSim [1] (a popular tool for analysing traffic patterns and supporting large-scale traffic scenarios[4]). However, the disadvantage of Agent-based models cannot be overlooked. As they model each agent individually, and it is common that a traffic scenario involves a large number of agents, this consumes more time during the simulation, which has negative effects on the performance [3]. On the other hand, Macroscopic models do not model entities individually so simulations are typically faster to run in comparison [20]. As they concentrate on the overall traffic without taking each individual entity into consideration, this architecture leads itself to overall traffic flow management, such as PTV Group[5] and

[2]http://www.paramics-online.com/
[3]http://www.aimsun.com/wp/
[4]http://MATSim.org
[5]http://vision-traffic.ptvgroup.com/

Emme[6], which are software used for analysing the entire traffic flow. However, this simple architecture has the limitation that it is not suitable for modelling traffic scenarios where individual behaviour is important.

As mentioned above, the major drawback of Agent-based models is a high computational cost. MATSim overcomes this difficulty by using a simple agent architecture with pre-defined journeys for each agent and conducting a central replanning process at the end of the iteration [27]. It is able to simulate approximately 7,000,000 agents on a road network which contains around 28,000 links in 87 seconds for one iteration on 64 dual core Intel Itanium 2 processors with 1.6GHz and 256GB of RAM [9]. It has a substantial history of being used for major real world applications such as analysis of the placement of a new Berlin airport[7] and it also has a spin-off company Senozon AG for MATSim products and services[8]. For these reasons, we chose MATSim as the experimental transport simulation platform for this research. The following section describes MATSim in detail.

2.2 MATSim background

Agents and Networks are the two essential elements for setting up a simulation in MATSim. A MATSim agent takes a pre-defined plan that it follows for the duration of the simulation [27]. The plan outlines the daily schedule of the agent, e.g. "home"-"work"-"home". A sample plan is shown in Figure 1 taken from a MATSim code repository[9]. In MATSim, "home" or "work" is called Activity and the path between the two activities is called Leg [1]. Both Activity and Leg must be clearly defined in the plan of the agent. Hence, agents have two main states when following the plan during the simulation: ACT (performing an activity) and LEG (travelling from one activity to another). Agents cannot decide to change the plan autonomously. The plan can only be changed in the Replanning process if it is scored low in the Scoring process [10], which will be explained in the following paragraphs. In terms of network in MATSim, the physical network is mapped to a conceptual network of nodes connected by links. The conceptual network contains information about the nodes (such as node coordinates) and the links (such as length and capacity). Figure 1 shows an example of an agent with a pre-defined plan defined in Extensible Markup Language (XML). It describes an agent travelling from home to the work place and then going back home. It can be seen from this example that both ACT and LEG have descriptive information such as transport mode, finish time and the route which is a sequence of road link IDs.

[6] http://www.inro.ca/en/products/emme/index.php
[7] http://matsim.org/scenario/berlin
[8] http://www.senozon.com/
[9] https://svn.code.sf.net/p/matsim/code

```
<person id="66128" sex="m" age="20" license="yes" car_avail="always" employed="yes">
  <plan age="0" selected="yes">
    <act type="home" link="1921" x="4590220.710354274" y="5822706.101701171"
    start_time="00:01" dur="08:00" end_time="08:00" />
    <leg mode="car" dep_time="08:00" trav_time="00:33" arr_time="08:33">
    <route trav_time="00:33">
    1880 1881 1884 9322 1887 1886 1968 1966 9333 9332 9336 9359 9355 9344 9346 9350
    9375 9379 9365 9369 650087 650084 650204 650374 650075 650172 650171 630414 630056
    630055 630054 630053 630153
    </route>
    </leg>
    <act type="work" link="13816" x="4562224.935893458" y="5831830.1171124475"
    start_time="08:00" dur="08:00" end_time="24:00" />
    <leg mode="car" dep_time="16:33" trav_time="00:33" arr_time="17:07">
    <route trav_time="00:33">
    630053 630054 630055 630056 630414 650171 650172 650075 650374 650204 650084 650087
    9370 9368 9380 9376 9351 9347 9345 9354 9358 9337 9330 9204 1950 9333 1966 1968
    1886 1887 9322 1884 1881 1880 1879
    </route>
    </leg>
    <act type="home" link="1921" x="4590220.710354274" y="5822706.101701171"
    start_time="11:33" dur="08:00" end_time="24:00" />
  </plan>
 </person>
```

Figure 1: A sample of agent plan from a MATSim code repository defined in XML. Different plans may have different attributes and these attributes are defined in the MATSim official website

The MATSim framework consists of four important processes [45]: *Demand Modeling* (generating the initial plans of agents), *Mobility Simulation* (simulating the agents over one day), *Scoring* (giving a score to each plan), and *Replanning* (collecting low-scored plans and updating them), which is shown in Figure 2. These four key processes are described as follows.

Demand Modelling generates the agents' initial plans. The main methodology for creating initial plans is applying an Iterative Proportional Fitting (IPF) algorithm, which generates a population of agents, using disaggregated census data [19]. A few variations have been made recently to improve the performance of *Demand Modelling* such as using Hierarchical IPF [29] and using IPF integrated with Resampling [18]. In this project, we use the existing initial plans from a MATSim code repository [10] in our experiments, which consists of about 16,000 agents in total.

Mobility Simulation conducts a traffic flow simulation based on the input of initial plans generated in Demand Modelling, a road network of the place to simulate (such as Switzerland), and a configuration file which contains the parameters to set up in the simulation, such as the number of iterations and the directory of output files. There are three available simulation modules for the

[10] https://svn.code.sf.net/p/matsim/code

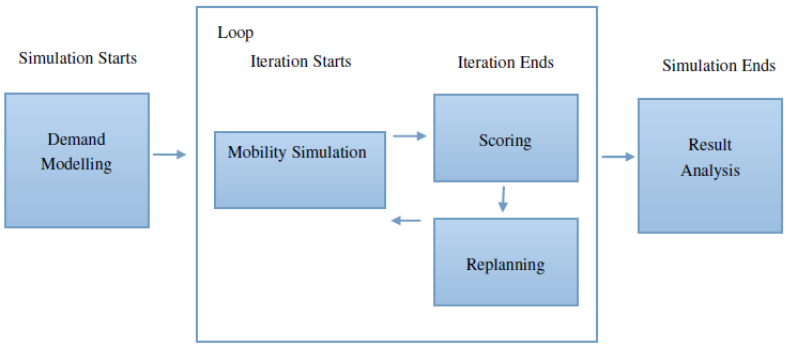

Figure 2: MATSim current framework (note that the Replanning cannot happen during the Mobility Simulation)

user to select in MATSim: Event-based Queue-Simulation (DEQSim), simulating based on specific events, such as processing the vehicles under the event when they enter a road [8]; JDEQSim, an similar version of DEQSim but implemented in Java (DEQSim was written in C++) [45], and Queue Simulation (QSim), a time step based simulation module, simulating the entities sequentially [15].

For our project, we use QSim as the simulation module since it ensures synchronisation. As mentioned in Section 1, synchronisation must be guaranteed when integrating BDI into MATSim. Using QSim ensures that the modules in MATSim execute sequentially, which makes the integration easier. It is also the latest simulation module and highly recommended by the MATSim developers.

Scoring gives the actual score of plans using utility functions. The quality of the plan is determined by the utility scores of its activities. The utility of each activity of a plan is estimated using five criteria: the distance of travelling from the last location to the next location, the real time spent on the activities, the cost of waiting (e.g. waiting on the road or waiting for the school to open), the penalty of arriving too late, and the penalty of leaving too early [17]. The calculated scores are used to select low scoring plans for replanning.

Replanning happens at the end of the iteration using genetic algorithms to generate better quality plans [10]. It randomly selects two plans as the parents from the population (a set contains multiple iteration plans of a traveller) at the start. Then an offspring plan is generated through crossover between selected parent plans and self-mutation. For crossover, it randomly selects a parent plan and places its activity in the same position of the offspring plan until all the unique activities are processed. In addition, the offspring plan can perform self-mutation using randomisation to change its information such as exchanging the order of its activities, exchanging the duration of its activities, and changing a start time of an activity by adding a random time [35]. After crossover,

mutation and a further day mobility simulation, the utility function mentioned above scores the offspring plan. The offspring plan will replace the worst-scored plan in the plan population if its score is higher. Otherwise it will be discarded. As a consequence, the plan population will only keep track of well-scored day plans and eventually a stable (no significant changes in plan scores) pattern of traffic is produced.

2.3 Within-day replanning in MATSim

This subsection describes the state of the art in within-day replanning in MATSim. Firstly, it explains why the current MATSim framework does not support within-day replanning. Then it outlines two major approaches that have been taken so far to realise within-day replanning. It explains the strengths and weaknesses of each approach and briefly mention what our project does to maintain the advantages and avoid the disadvantages of these approaches.

As mentioned in Section 2.2, the replanning process is conducted at the end of each iteration. The agent is made to be very simple and cannot make any decisions during the iteration and can only follow the pre-defined plan. This iteration-based approach satisfies the focus of MATSim (finding traffic patterns with relatively fast computational speed), but does not support within-day replanning where agents replan during the iteration based on their situated environment.

An experiment conducted in 2012 shows that in the current MATSim framework, trying to achieve within-day replanning is not possible and produces inappropriate results [16], as shown in Figure 3 of Dober etc [16]. The experiment starts with a sample network and an agent travels from the start node to the end node with a pre-calculated path (specified in the agent's plan). The agent's plan defines the expected arrival time at each node in the path. For example the agent is expected to start at 13:45 pm and finish at 14:40 pm. Then a road block suddenly occurs at a link that the agent needs to travel on, and it lasts for two hours. Figure 3(c) shows that the agent was stuck at the link where the road block occurred and arrived at the destination at 16:40 pm, two hours later than the expected of 14:40 pm. This is because the agent could not replan during the iteration under the current MATSim framework. Figure 3(d) shows that the route that would be found by replanning - although of would not be desirable to permanently replan on the basis of a temporary roadblock. The above experiment points out that the agent under the current MATSim framework cannot make the decision to change its plan if an unforeseen event happens (from the experiment, the agent was 2 hours late because it simply waited at the road block instead of immediately replanning).

A few attempts have been made to address the above problem and they can be classified in two groups. The major group uses an approach that collects the

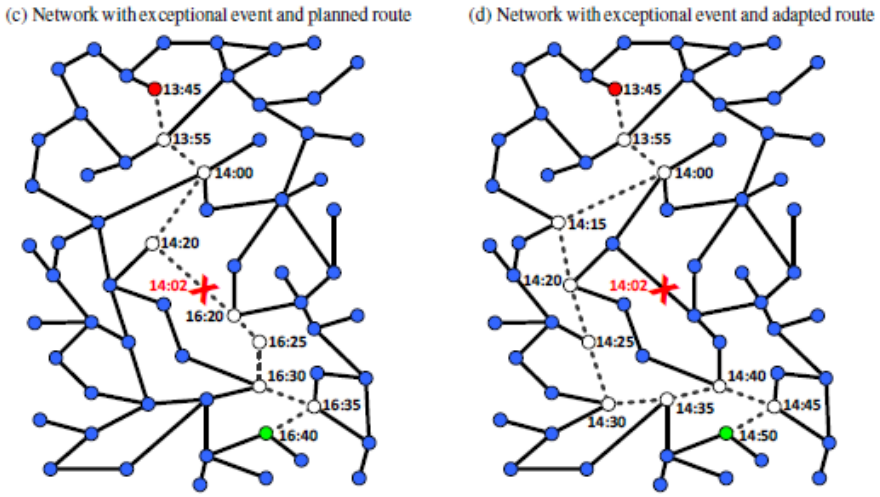

Figure 3: An experiment proves that current MATSim framework cannot realise within-day replanning [16]

agents and replans them every time step during the iteration [16, 15] whereas the other group allows agents to decide whether to replan autonomously and they can change their current routes [22]. Both approaches realise within-day replanning in MATSim, but have distinct limitations.

The first approach inserts the following two steps before each time step within the iteration, which is illustrated in Figure 4 from [16]:

Step 1 Using the *identifiers* to collect a subset of the agents

Step 2 Using the *replanners* to adapt the plans of collected agents

An identifier refers to a characteristic or a state that agents share, e.g. all of the agents that are performing an activity (agents in ACT state, as mentioned in Section 2.2). There are four identifiers implemented so far: (1) agents in ACT state; (2) agents in ACT state and will finish the activity at the current time step; (3) agents in LEG state (as explained before, LEG state means that agents are travelling from one activity to another); (4) agents in LEG state that leave the current link and enter the next link. By using identifiers, the agents with the same characteristics or states can be collected easily.

A replanner refers to using a specific replan strategy to adapt an agent plan. For instance, using a shortest path algorithm to adapt agents' pre-calculated paths. Based on the descriptions of the most recent paper related with this mechanism, the replanner can only adapt the agent's current route [16].

The travel time is used to identify whether an agent needs to replan. When an agent finishes travelling a road link, it will collect its travel time and calculate

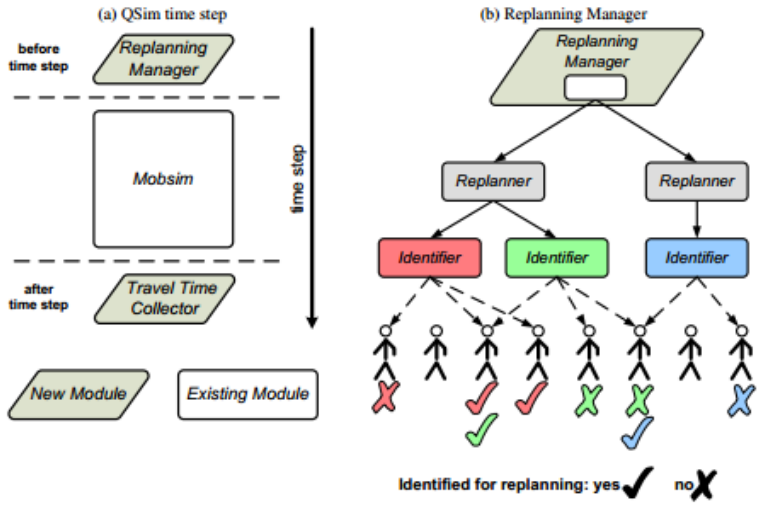

Figure 4: An framework for realising within-day replanning based on the existing framework, from [16]

the average time for agents to pass this link. The average time of passing a link is updated for a specific duration, such as every 15 minutes. All the travel times of links are public to all the agents. The agents want to minimise the travel time and they use the average travel time of the link to see whether replanning is required.

This approach has clear advantages and disadvantages. The major advantage of this approach is that it enforces synchronisation by stopping the simulation to collect all of the agents for replanning. Additionally, this mechanism is modularized so that it can be integrated with other behaviour models. For instance, Step 1 can be replaced with a BDI agent model to collect the agents who decide to change and other steps can remain the same. However, the limitation of this approach is that only the travel time is used to decide whether an agent needs to replan. We would want agents to consider their own situations to make a decision based on any relevant information rather than purely the travel time. The authors mention that it is necessary to integrate a more complex behaviour model to make the agents make decisions based on their own situations independently and they consider this as future work [16, 6]. In addition, the replanner can only change the current route of the agent. A more complex replanner needs to be implemented, for instance to allow a new activity to be added and adapt a future route to an activity.

The second approach tries to add simple reasoning to the MATSim agent, which can be used to decide whether to modify its plan [22]. The reasoning consists of a contentment module and a link cost perception. The content-

ment module returns a score calculated by the utility function mentioned previously [10], which is used to determine whether the agent needs to replan. The link cost perception represents the agent knowledge of the experienced cost of the current link and the estimation cost of the nearby links. These costs are used in the utility function to calculate the score. During the simulation, the agents make decisions to replan using the knowledge of the costs and the plans are modified if a replan is necessary. This approach can only adapt the current route of the agent [22]. This approach has the advantage of making the agent decide to replan autonomously and independently compared to the first approach. There is no need to collect the agents using identifiers. However, the disadvantages cannot be overlooked either. Firstly there is no clear mechanism to ensure synchronisation (for the first approach, it pauses and resumes the simulation to achieve the synchronisation), which makes it more difficult to integrate other advanced agent models to control the agents (synchronisation is essential for integration between an agent model and a traffic simulation platform). Also, similar to the first approach, the replanning is very simple. It only adapts the current route of the agent and does not support more complex replanning strategies.

Based on this, it is still necessary to propose an enhanced approach to support within-day replanning in MATSim to overcome the limitations of these two existing approaches. We will explain our approach in detail in Section 3.

2.4 Taxis in MATSim

Modelling taxis is still under development in MATSim. A taxi must be made to be reactive because it needs to respond to different scenarios dynamically, such as it needing to request a job when it is available. The first attempt was made in 2009 which provides the preliminary results of implementing taxi but it has significant limitations [11]. As shown in Figure 2, an agent can take different transport modes (defined in MODE tag in the plan xml file) from one activity to another. With this approach, an agent can take a taxi immediately if the MODE is defined as "taxi" in its plan. The differences between taxi and other transport modes are only made in the characteristic level rather than the reactiveness, such as the speed of Public Transport being defined as 33 kilometers per hour while the speed of taxi is defined as 50 kilometers per hour. In other words, the taxi implementation is made nearly identical to other transport tools in this approach. Additionally, this approach assumes that there is no waiting time, i.e. an agent can take a taxi directly if it is in LEG state and the MODE is defined as "taxi" in its pre-defined plan. This assumption is not realistic because typically passengers need to wait when they call for taxis. As a result, it might affect the accuracy of the experiment results. Therefore, this approach is not considered ideal to implement taxis and indicates a need

for further development using individual agents to represent taxis so that each taxi agent can perform dynamically [11].

An enhanced attempt was made in 2013, which models taxis by using agents and incorporating the Dynamic Vehicle Routing Problem Optimiser (DVRP Optimiser) [27]. That has made distinct progress over the first attempt, but still has drawbacks that need to be resolved. In this approach, the taxi agent has three sequential tasks: WaitingTask (wait for DVRP Optimiser to assign a job), ServeTask (pick up a passenger at a specific location), and DriveTask (drive to the destination). DVRP Optimiser is used to assign a job to a taxi agent based on two different strategies – No-scheduling strategy (assign the job to the closest available taxi) and One-time-scheduling strategy (append the job to the closest taxi who is doing a job). It also uses different ways to estimate the distance, such as Straight Line (calculate the bee-line distance between two locations) and Travel Distance (the shortest distance between two locations based on the real road network). Therefore, when a passenger calls for a taxi, DVRP Optimiser will choose a taxi and that taxi will pick up the passenger and drive to the destination. Compared with the above approach, it has two distinct advantages: (1) Using agents to model taxis and these agents can perform the tasks dynamically; (2) It realises an entire taxi service which can evaluate the performance of different strategies adopted by DVRP Optimiser in taxi service. However, the agent is still very simple in this approach. They cannot make any decisions based on their own situations. In real life, taxi drivers can request jobs independently (they may request jobs when they are free or near the passengers' destinations) whereas this approach makes the agent wait for DVRP Optimiser to assign the jobs. Additionally, integrating DVRP Optimiser into MATSim can only model the scenarios which require calculating the distance and route, it still cannot model complex scenarios where agents need to make decisions from available plans, such as needing to decide whether to drive to the safe area directly or go home to take family members first, in the case of a bush fire. Therefore, adding an advanced behaviour model to MATSim that supports agents making decisions reactively is important. In fact, similar to MATSim, most of traffic simulation tools use a simplistic approach to model agents, which is described in the following section. Our project uses the BDI agent model to address this shortcoming. In terms of taxis, we integrate the BDI agent model into MATSim which allows taxi agents to make decisions independently and reactively compared with this latest approach.

2.5 Modelling of individual agents

Most agent-based traffic simulation platforms use a simple approach to model agents. This can be classified into two groups in terms of decision-making capability: (1) The agent cannot make decisions and it relies on other modules

to realise decision-making; (2) The agent can make decisions but it is based on simple strategies, such as choosing which path to take based on calculated shortest straight line distance. The well-known traffic simulation tools of the first group are PARAMICS [7] (the agent cannot decide to replan by itself. The replanning happens centrally by collecting and ranking) and MATSim [1] (as mentioned previously, the agent cannot decide to adapt its plan by itself. This is done by a central evolutionary process). The latter group consists of popular traffic simulation platforms such as AIMSUN [2] (the agent can calculate the shortest routes and use this to make decisions) and MOSAIIC [43] (the agent makes decision based on calculated best path).

Much fewer traffic simulators use a cognitive approach to model agents compared to the simplistic approach. A well-known example is TACAIR-SOAR [41]. It adopts the State, Operator And Result (SOAR) architecture, a cognitive agent model using problem space (contains states and operators) to represent the system and finding a solution from the initial state (the state that agent was in initially) to the final state (the state that the agent wants to achieve) [25], to simulate pilot behaviours in a battlefield environment. It has been used in the Synthetic Theater of War 1997 and demonstrates the strong capability to model pilot behaviours such as identifying enemy planes and telling its position to fellow pilots [23]. Additionally, quite a few traffic simulation platform contributors point out the potential benefits of integrating the Belief, Desire, Intention (BDI) agent model into traffic simulators and they consider this as future work [6, 14, 43]. For instance, the developers of MATSim wants to use BDI to model human decision making in reactive traffic scenarios such as evacuation [6], and the developers of MOSAIIC also wants to use BDI to represent the agent decision process in unforeseeable events such as accidents [43].

Both simplistic and cognitive agent modelling has advantages and disadvantages. As mentioned in Section 2.1, one of the drawbacks of agent based modelling is the high computational cost. Using simplistic approach can reduce the computational cost because there is no complex decision-making for each agent and even the agent cannot make any decisions in some of the platforms using this approach. However, this also brings the disadvantage that it cannot model traffic scenarios that require agents to make decisions reactively in the environment, such as evacuation and taxi service. Using cognitive agent modelling supports human decision making but it potentially increases the computational cost because of complex decision making in every agent. However, some BDI platforms are extremely efficient and are used for soft real time control systems, so it is hoped that the additional cost is not too great.

2.6 BDI Agent Model

The BDI agent model provides a computational framework for modeling rational agents [37]. It is based on work in human psychology to describe intentional behaviours and practical reasoning [12]. It uses the following concepts:

- Beliefs are the personal opinions about the world. In the BDI agent model, beliefs are used to describe the agents perception of how the environment is.
- Desire relates to what one wants to achieve. In the BDI agent model, goals are used to capture desires. An agent has at least one goal and it tries its best to achieve its goals by executing appropriate plans. A plan consists of a sequence of abstract steps (which could be the sub-goals) that achieve a goal.
- Intentions are things that one has committed to do. In the BDI agent model, intentions are instantiated plans.

A BDI architecture is shown in Figure 5 from [39]. The agent observes the environment for new events. If an event occurs, it selects a matched plan from the plan library (a set of pre-defined plans) based on its belief. Once it confirms the selected plan from the list of applicable plans, it will add the selected plan to the intention stack (a set of the plans that need to be executed) for execution in the future. It then picks an intention to execute and progresses it partially. These steps are repeated as an execution cycle in a BDI agent model. This above architecture leads to the following characteristics which are useful for the project. Firstly, as mentioned above, the agent is able to perform the behaviour directly if the environment changes, which makes it reactive. For instance, if a taxi agent sees a traveller that wants to take a taxi, it can respond to that event and take the traveller rather than stick to a pre-defined plan.

In addition, as the agent has a plan library from which to select plans, it gives the agent flexibility in achieving its goals. If a plan fails, it can still achieve the goal by following an alternative plan. This flexibility is also required in the project. For instance, if an agent finds that its planned road to the destination is blocked, it should be able to follow alternative paths to the destination rather than solely focus on that road.

Table 1 highlights the key advantages of BDI enhanced MATSim agents. From Table 1, it is obvious that the BDI agent architecture is more appropriate for modelling traffic scenarios which involve reactive decision making. However, the BDI agent model also has drawbacks, such as the lack of a learning ability to improve decision making from past experiences [32]. The important drawback

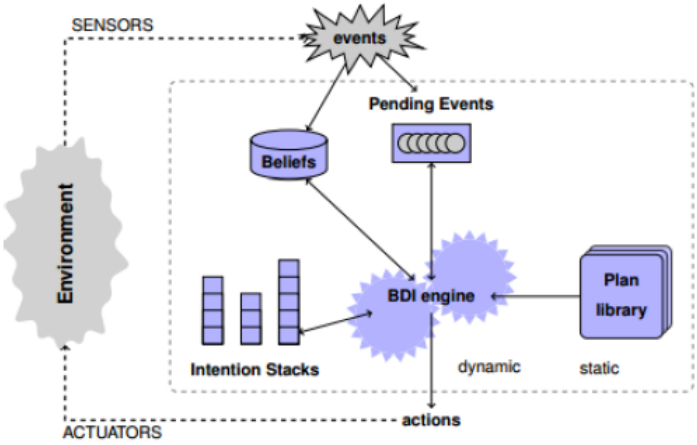

Figure 5: A standard BDI framework from [39]

Characteristics	MATSim Agent	BDI enhanced MATSim Agent
Goal	Does NOT have goals.	Has a set of goals which are used to describe what wants to achieve. The goals can also be defined in a hierarchy (top goals with sub-goals)
Plan	Normally has only one plan, which is the daily schedule.	Has multiple plans to achieve a goal
Decision Making	Cannot make decisions.	Can make decisions based on its knowledge
Reactiveness	Cannot change its behaviour if an unpredictable issue happens.	Can change its behaviour to adapt to the situated environment
Robustness	Will get stuck if the daily schedule cannot be achieved in the expected time	Can choose alternative plans if one plan fails.

Table 1: Comparison between simple and BDI-enhanced MATSim agents

is that its standard architecture does not have the learning capability[11]– agents cannot learn from the past situations. This limitation does not affect our project because our focus is on modelling reactive traffic scenarios where learning from previous experiences is of little value.

As mentioned before, there are many platforms that implement BDI agent model, such as GORITE [38], JACK [4], GOAL [13], and CANPLAN [39]. GORITE is used to implement BDI agent model in this project because it is Java-based like MATSim, and therefore does not have cross platform issues. Also, it is light-weight, fast and allows us to instantiate a large number of agents, which is beneficial in traffic scenarios.

2.7 Integrating cognitive agents into simulations

There is limited work in terms of integrating cognitive agents into agent-based simulators using simple agents such as those found in most agent based traffic simulation. An attempt was made in 2011 which incorporates cognitive BDI agents into the Repast simulation platform [31]. Repast is a widely used agent based simulation software which supports simple rule based agents [30]. This work demonstrates Repast with integrated BDI agent can support modelling human behaviour and it proposes a framework to integrate a BDI agent model into agent based simulation, which provides valuable resources for our project.

The proposed framework consists of three components: agent-based simulation, a BDI agent system, and the communication interface between these two modules. Each BDI agent inside the BDI agent system is mapped to an agent inside the agent-based simulation. The BDI agent performs the reasoning which uses the knowledge gained from its mapped agent (BDI percepts) to do reasoning and make decisions (BDI actions). The agent in the agent-based simulation follows the decision made by its mapped BDI agent. For instance, if a BDI agent makes the decision to drive to a specific location, its mapped agent will drive to that location in the simulation. It also uses BDI sensing actions for BDI agents to acquire the required information in a specific time step. For instance, when the BDI agent needs to know the location of its mapped agent in the current step, it will ask the mapped agent inside the agent-based simulation for its location within that time step.

In addition, it uses message passing protocols to define BDI percepts, BDI actions, and BDI sensing actions during the interaction between the BDI agent model and the agent-based simulation. These defined protocols specify the format of the message and make the framework generic to any relevant applications (different applications might have different contents but the format of the protocol is consistent and both agent based simulation and BDI agent model manage

[11]However various research projects have added learning to BDI frameworks, such as [32] and [40]

State	Description
INITIATE	Initiated by BDI agent and to be executed
RUNNING	Being executed by the simulation agent
PASS	Completed as expected
FAIL	Aborted by the simulation agent
DROPPED	Aborted by the BDI agent
SUSPENDED	Suspended by the BDI agent

Table 2: A table shows a sample protocol to define different states of goals from [31]

the same protocol). Table 2 shows the protocol to define BDI goal states [31].

Furthermore, as there are two modules – agent-based simulation and a BDI agent system involved, synchronisation must be ensured to make the whole system consistent. It provides useful rules to achieve this, such as that the BDI agent model must be idle when the agent-based simulation is performing.

This existing work is very relevant to our project. It is the first attempt at integrating BDI agents into an agent based simulation platform and it demonstrates the integrated platform can support modelling human behaviours. In addition, the proposed framework clearly illustrates how to integrate BDI agents into agent based simulation, which can also be applied to our project. Thirdly, in this project we need to ensure synchronisation since the BDI agent system and MATSim have different execution cycles. We can combine much of this work with the existing MATSim within-day replanning mechanism (mentioned in Section 2.2) to meet our requirement.

3 Integration of BDI into MATSim

This section describes the method we have explored in detail. Firstly we explain the enhanced MATSim framework incorporating the BDI agent model, which is used to model traffic scenarios that include reactive decision making. Then we describe the implementation of a taxi service which applies this enhanced framework, as a demonstration system.

We have added a BDI agent system to decide what to do as the situation unfolds and modified MATSim so that the decisions made by the BDI agent system can change the agents' behaviours in MATSim during the execution. The BDI agent system receives the information (called percepts) such as whether it is close to the destination, from the environment (MATSim simulation) which is necessary for it to do the reasoning. Once the BDI side finishes reasoning, it

will tell MATSim what the agents should do (called actions). As a result, the MATSim agents' plans will be adapted based on these decisions. The information of percepts and actions transferred between these two systems is defined in message protocols as mentioned before. To use this framework in an application, the percepts required in the BDI side and the actions modelled in the MATSim side should be identified. To obtain the percepts and to execute the actions, we may need to add some functionality. In our developed taxi service example, we have implemented the action "DriveTo" for taxi drivers to obtain a route and drive to a given location for picking up or dropping off passengers. The percept it needs was whether a taxi is close to the drop off location to know whether the taxi has nearly finished the job. The BDI agent can use this percept to determine whether new jobs should be requested. The descriptions of the enhanced framework, the functions that we added on MATSim side, and the taxi service example are described in the following sections.

3.1 Enhanced MATSim framework

The BDI agent model is embedded into *Mobility Simulation* process of the current MATSim framework and an interface is also integrated to support the communication between the BDI agent model and the MATSim simulation module. The rest of the MATSim framework remains the same. Figure 6 shows the enhanced MATSim framework with the BDI agent model embedded.

From this enhanced framework, the BDI agent model communicates with the MATSim simulation module via BDI-MATSim interface during the simulation. The BDI agent model is responsible for reasoning (making the decisions based on the knowledge) and the MATSim simulation module is responsible for performing the decision made by the BDI agent model. When the simulation starts (before the commencement of the iteration), the BDI agents will be generated inside the BDI agent model to correspond to the target agents inside MATSim. For example, a BDI agent with ID 1 is mapped to a MATSim agent with the same ID. When an iteration begins, the associated information of MATSim agents will be collected, such as whether taxis are close to their destinations and placed into the communication interface. Then the interface will notify the mapped BDI agents to update their knowledge. For example, the MATSim agent with ID 2 is found to be close to the destination in the current step, the interface will notify the mapped BDI agent to update this knowledge and it can make decisions based on the updated knowledge. Once the BDI agent model finishes processing all the updated information in the interface, the BDI agents will make decisions based on their knowledge. If an action needs to be made for a particular agent, it will place the associated information into the interface, such as agent with ID 3 needs to drive to location with the coordinates (112, 150). After that, the interface notifies the corresponding MATSim

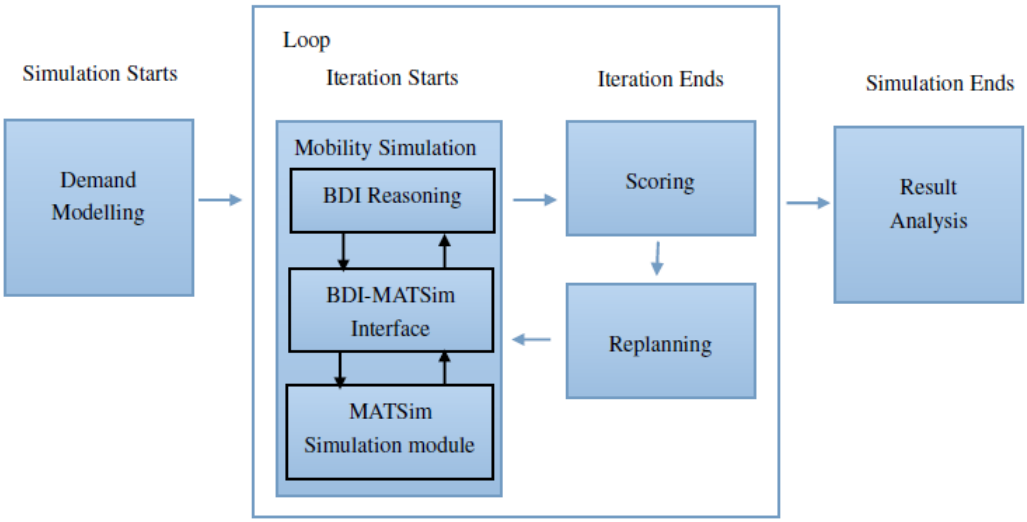

Figure 6: The enhanced framework which integrates the BDI reasoning during the MATSim simulation

agents to perform the specific actions decided by the BDI agents. This happens repeatedly until the completion of the simulation. The sequence diagram in Figure 7 illustrates the above description.

As can be seen from the sequence diagram, the BDI agent model will not be executed until the MATSim simulation module pauses, and vice versa. This makes the execution cycles of these two different paradigms synchronised. Also, we use the specific protocol format mentioned in Section 2.7 for message passing so that these two different modules can understand the message content. We have made this framework general so that any applications which require decision making during the execution in MATSim can apply it.

As mentioned before, BDI agents need to use percepts to do reasoning. In the taxi service application, they need to know whether they are close to their destinations. We have developed the following functionality to collect this information and provide to the BDI side.

3.2 Estimating agent distance to destination

We introduced a way to approximate whether a given agent is near the destination. This provides the information needed for BDI agents to make their decisions in a realistic way. An option could have been to only provide the information on arrival, but for a taxi service, this would then be less accurate because in reality taxi drivers request a job when they are about to finish their

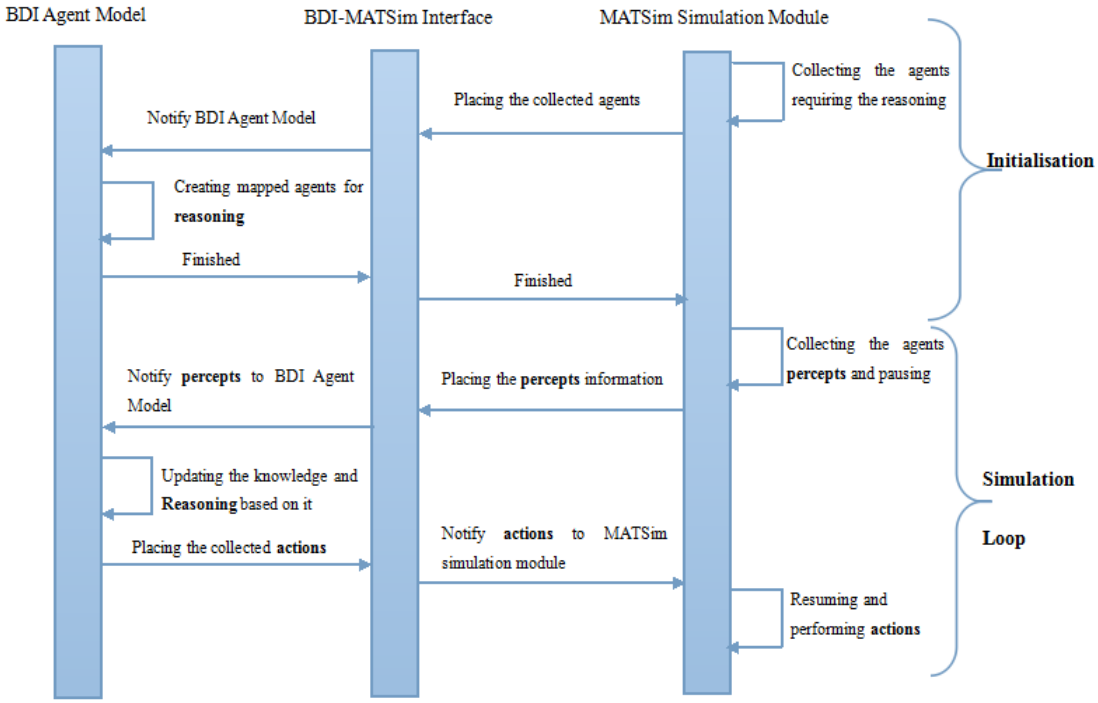

Figure 7: A sequence diagram shows the interaction between BDI and MATSim under the enhanced framework. Note that the processes in Simulation Loop bracket are executed repeatedly in each turn

jobs. This function also can be applied for any applications that need to know how soon an agent will finish the LEG state in MATSim.

In order to achieve this, we have implemented an event handler using MATSim Application Programming Interface (API). It handles the events when an agent starts driving from one activity to another (the commence of LEG state) and when an agent finishes driving from one activity to another (the completion of LEG state). We have also created a simulation parameter called DISTANCETHRESHOLD for users to customise, e.g. when DISTANCETHRESHOLD is defined as 2 kilometers, an agent will be claimed as "close to destination" if the distance between its current location to the destination is within 2 kilometers.

When an agent starts driving to the next activity, the event handler will calculate a route (consisting of an ordered list of links) between the agent's current position and the destination only once, using existing routing functions in MATSim. From this ordered list of links, we can calculate the real distance from the agent's current location to the destination by summing the length of the links at any time (as mentioned in Section 2.2, link in MATSim has its length information). Hence we can know how close the agent is to the destination.

This approach identifies and provides information when an agent is nearly at its destination, and the BDI agent can use this as the percept to make decisions.

3.3 Advanced MATSim within-day replanners

The decision made by the BDI agents during the reasoning is likely to involve changes to the future LEG and ACT in agents' plans which the existing work does not support at the moment. We have implemented the following two replanners to overcome this problem.

The first replanner we have made allows an agent in ACT state to stop its current activity and add a new activity to its plan. This is particularly useful in reactive scenarios such as evacuating from a bush fire. When the agent is at home (ACT state) and receives notifications that the bush fire is close to its house, it needs to leave home immediately (stop the current activity) and drive to the safe area (add a new activity).

The second replanner allows an agent in LEG state to add a new activity to its existing plan, and plans the route to the newly created activity. This is also appropriate to model traffic scenarios where agents make decisions dynamically. Suppose an agent is driving from home to work (LEG state) and suddenly gets a phone call that its parent at home needs to go to hospital, so it decides to drive back home to take his parent to hospital (add a new activity).

In our enhanced framework, these two replanners will change the agents' plans if the BDI sides make such decisions during the reasoning. They can also be used in other applications which require changes to the future ACT and LEG

of an agent.

3.4 Implementation of a taxi service

We have used the enhanced framework to implement a taxi service model. The MATSim simulation module contains taxi agents and normal agents. The BDI agent model contains the BDI taxi agents which are mapped with MATSim taxi agents and an operator agent which generates jobs and confirms whether a taxi agent can take a job that it has requested.

When the simulation starts, the MATSim simulation module collects all of the taxi agent IDs and places them in the interface. The interface then notifies the BDI agent model to create the operator agent and mapped BDI taxi agents. It does not collect other MATSim agents because there is no reasoning required for them.

During each turn, the MATSim side checks whether each taxi agent is close enough to the destination. If so, it places this percept to the interface using the pre-defined message protocol and pauses. The interface then notifies the BDI side to update the specific mapped BDI agents' knowledge. The operator generates new jobs and notifies all of the BDI taxi agents. The BDI taxi agents decide whether to request a job based on its knowledge (whether it is free or it is close to its destination). The operator will confirm whether the job can be taken by the agent who requested it. Once confirmed, the BDI taxi agent will place the message in the interface to tell its associated MATSim taxi agent to drive to the specific location to pick up the passenger. The BDI agent model will be paused when all of the BDI agents finish reasoning. Then the interface notifies the MATSim simulation module. The MATSim taxi agents will perform the actions of its mapped BDI agent(using the replanners mentioned above), by adding the new destinations to their plan.

We have used the Prometheus design tool [42] and GORITE programming language [38] to design and implement the BDI agents respectively. The BDI agents use a hierarchy of goals and a set of alternative plans. Figure 8 shows a sample of Goal-Plan tree diagram of a BDI taxi agent, which illustrates only a part of the complete Goal-Plan tree due to its size. From this diagram, a BDI taxi needs to accomplish the DoJob goal which is handled by the DoJob plan. The DoJob plan has three subgoals: GetJob (getting a job), PickUp (picking up the customer), and DropOff (dropping off the customer). For the GetJob subgoal, it has three alternative plans to realise: HaveJobInQueue (have jobs which have been confirmed by the operator), GetJobFromOperator (request available jobs from the operator), and Wait (wait until notification from the operator). If any one of these plans fails, it can select other alternatives. The PickUp and DropOff goals will be achieved by raising a DriveTo action to notify its MATSim counterpart to drive to a specific location, either to pick up or drop

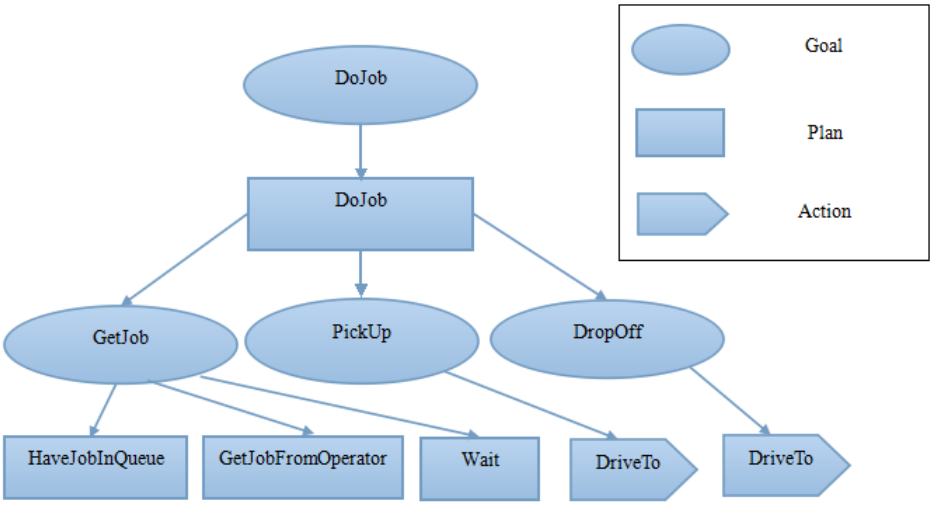

Figure 8: A partial design of an BDI agent

off passengers.

4 Discussion and Conclusion

We have run the implemented taxi service and it clearly shows the agents can make decisions reactively under the enhanced MATSim framework. It is the first attempt to make the complex decision making under the dynamic environment among agents in MATSim possible.

Our enhanced framework successfully achieves within-day replanning in MATSim when comparing agent's original agents' original and adapted plans – the agents are able to make decisions and adapt their plans during the execution. We have run the system with up to 1,000 BDI enhanced MATSim agents on a laptop (Window 7 32-bit, 4 CPU cores, and 1GB RAM).

We can observe that an agent under the enhanced framework can adapt its behaviour based on the reasoning from Figure 9, which compares the original plan (before the simulation) with the adapted plan (after the simulation) of an agent from the output of the simulation. The adapted plan is demonstrated only partially due to its size. The complete adapted plan has 12 ACTs and 11 LEGs. We observe that the taxi agent originally started at a road link with ID 14,443 and its default plan is to go to the workplace (road link ID 14,286), which is shown in its original plan. Once it gets confirmed with a job which has the road link ID 8,701 for picking up and the road link ID 3,933 for dropping off, it immediately adapts its original plan. It firstly removes the pre-defined

route (the route from its start place to work place in the original plan) and removes the original activity (the activity located at road link ID 14,286). Then it adds a route calculated from its current location (road link ID 14,443) to the pick up location (road link ID 8,701) and adds a new activity located at road link ID 8,701 for picking up the passengers. When it finishes the added activity for picking up the passengers, it will add a route calculated from the pick up location (road link ID 8,701) to drop off location (road link ID 3,933) and add a new activity located at road link ID 3,933 for dropping off the passengers. These scenarios such as adding updated routes and adding new activities will occur repeatedly when the taxi agent gets a job confirmed during the simulation. From this observation, it can be clearly seen that the agents can successfully adapt their plans, such as adding new activities, removing activities, and replanning the routes for the future activities. These are more complex than the existing framework (the most recent publication related with within-day replanning claims that it can only change the current route of an agent [16]). More importantly, all of these decisions such as adding or removing activities are based on the reasoning by the BDI agents. This means every agent under the enhanced framework can make its own decision based on its specific situation. As mentioned in Section 2.2, neither the most commonly-used existing within-day replanning framework (the first approach described in Section 2.2) nor the most recent MATSim taxi service implementation realises the complex decision making of agents. The other approach to support within-day replanning described earlier is only able to perform simplistic decision making which cannot model rich goal oriented behaviours. Therefore, our enhanced framework takes advantage in terms of the functionality to model decision making in the dynamic environment. It also has the following advantages.

The taxi model that we implemented so far is the first attempt to add cognitive agent reasoning to MATSim and it has the potential to model complex taxi service such as fluctuating demand and multiple taxi companies. As the taxi model is a decoupled component, there is no change needed to the MATSim side. This allows the development of a taxi application without detailed understanding of the internals of MATSim. Also, the enhanced framework can support more than a taxi model. It has the flexibility to swap the taxi model with any other application requiring decision making during the execution, such as evacuation or ambulance service. As described in Section 3, as long as the percepts (the knowledge from MATSim side for BDI agents to do reasoning) and the actions (the decision made by BDI agents to adapt MATSim agents' plans) are identified and implemented for the specific application, the enhanced framework will then satisfy the requirement.

Moreover, the enhanced framework also has the flexibility to be used either as a part of multiple day simulation, using it with the current MATSim framework

<act type="home" link="14443"
x="4522719.523630672"
y="5830135.521943586"
start_time="00:00" dur="08:00"
end_time="08:00" ref_id="921" />

<leg num="0" mode="car" dep_time="08:00"
trav_time="00:14" arr_time="08:14">
<route trav_time="00:14">
630351 630278 630277 630284
630240 630246 630239 630260 630344
</route>
</leg>

<act type="work" link="14286"
x="4517395.673524921"
y="5839333.605215912"
start_time="08:00" dur="08:00"
end_time="24:00" ref_id="921" />

Before the simulation (complete agent plan)

<act type="home" link="14443" x="4522719.523630672"
y="5830135.521943586"start_time="00:00:00"
max_dur="08:00:00" end_time="08:00:00" />
<leg mode="car" trav_time="00:00:00">
<route type="links">14443 14444 14449 13754 13749 13751
14461 20427 21585 20426 21590 20423 8488 8486 8526
8483 8527 8476 8535 8474 8481 8513 8510 8511 20421
21296 21989 21327 21317 21316 21314 21720 21721 21723
21729 21797 21821 25748 25601 25600 25941 25939 25328
25326 11564 11562 11559 12117 11557 18804 18805 18808
19098 19099 18809 18812 18339 19004 19341 18738 8755
8702 8701</route>
</leg>
<act type="work" link="8701" x="4667630.0"
y="5802211.25" max_dur="00:00:10" />
<leg mode="car" trav_time="00:00:00">
<route type="links">8701 8756 18737 19342 19003 18340
18811 18810 19100 19097 18807 18806 18803 11558 12118
11560 11561 11581 12056 11583 8270 8278 6924 5865 5867
5869 5968 5954 5938 7241 7587 7586 7583 7582 7589 28854
7468 8099 7906 7895 28907 28904 7889 28881 28879 28892
28884 4611 27971 27979 27978 5016 5013 5024 5019 5027
5031 28687 3462 3467 3483 28669 3476 3375 3504 3379
3390 3388 3385 3405 3417 3420 3391 3401 3403 3964 3960
28253 3916 3913 7342 6980 3920 3837 3845 7004 3851 3815
7302 3855 3867 3865 28210 3933</route>
</leg>
<act type="work" link="3933" x="4594727.75"
y="5822195.0" max_dur="00:00:10" />

After the simulation with the enhanced framework (partial agent plan)

Figure 9: A comparison between the original and adapted plan of an agent under the enhanced framework

(shown in Figure 1) to simulate the traffic scenarios over many iterations or as a separate component to run a single day simulation to simulate exceptional events. When it is used as the former, MATSim can assess the overall traffic patterns with more intelligent components since the cognitive agent modelling is adopted. When it is used as the latter, it runs the mobility simulation module to model traffic scenarios for a single day, which does not need *Scoring* and *Replanning* of the current MATSim framework.

Now we can use the BDI agent model to realise reactive and goal oriented behaviours and MATSim to reflect the effect of decisions of BDI agents on the road network under this enhanced framework. This provides the preliminary results for integrating a cognitive agent model into traffic simulation platforms. It also provides MATSim with richer capabilities and allows analysis of human decision making in the situated environment. For traffic scenarios which are likely to feed back into decisions, such as in an evacuation simulation, when there is road block or congested traffic, MATSim will provide this information to the BDI side and the BDI agents will change the decisions based on the updated information. All of this is now possible in our enhanced framework.

We have answered all the research questions described in Section 1. We used a uniform message protocol to define the percepts and actions for two different paradigms (BDI and MATSim) to transfer. These two systems are able to communicate via the pre-defined message protocols although they have different representations of agents and plans. Hence, it solves the question of how to map the representation of BDI agents to MATSim (Research Question 1). We also proposed an enhanced framework that makes BDI and MATSim execute sequentially during the simulation (illustrated in Figure 6 and Figure 7). This framework ensures the synchronisation between two different systems, so that BDI agents will not perform the reasoning until receiving the percepts collected by MATSim, which solves the question of integrating the execution of BDI agents with the execution of MATSim (Research Question 2). We also defined the functionality of taxi application, such as requesting jobs and picking up passengers (explained in Section 4), which answers the third research question.

Apart from the advantages described above, our enhanced MATSim framework has some weaknesses and the future work is listed to address these. We have not evaluated it for time efficiency nor optimised it. There is a lot of room for improving its efficiency. GORITE, the BDI agent platform we used, is light weight and fast. As a result, the reasoning time of BDI agents do not add significant overhead to the current system. This should make it possible to maintain the time efficiency of the current MATSim architecture. It is currently implemented using socket. This is because sockets offer flexibility so we can swap GORITE for a non-Java BDI system. However, this slows down the speed. We can remove the sockets to improve the speed specifically for MATSim

and GORITE as they are both in Java.

The BDI sensing behaviour is not implemented yet. For instance, the taxi agent cannot get information regarding which job is closest from the MATSim environment. Hence, the future work also includes implementing functionality such as allowing BDI agents to query specific percepts from MATSim during its execution. For instance, when a BDI taxi agent needs to decide which job to take, it can query to find which job is closest to it. This would allow more precise reasoning and make the system more realistic.

In summary, in this thesis we realise modelling rich goal directed and reactive behaviour in MATSim by incorporating a BDI agent system. Firstly we describe how the existing MATSim architecture does not support this functionality and how the existing approaches have distinct limitations. In addition, we use a BDI agent system, a standard cognitive agent modelling approach and integrate it to MATSim in order to resolve this problem. We show that the integration successfully achieves this functionality. This is the first work that uses a cognitive agent model in MATSim and it makes it possible to simulate the human decision making in dynamic environment such as flood and earthquake. Furthermore, during the review of existing agent based traffic simulation platforms, we have found a few of them claim that they want to use BDI agent model for the agents in the future. This work provides preliminary results for these agent based traffic simulation platforms to achieve this.

5 Acknowledgments

I want to express my special thanks to my supervisors Lin Padgham and Dhirendra Singh for their patience and advice. Without their support, I could not finish this project. I also want to thank to Ralph Ronnquist (the leader of the GORITE team) and Christoph Dobler (the developer of MATSim) for their help during my implementation. In addition, I would like to appreciate Sebastian Sardina and Andy Song for reviewing my honours proposal. Moreover, I want to thank to Timos Sellis and Justin Zobel for their invaluable guidance and help. Last, I want to thank to my great friends Halil Ali and Robert Mcquillan, and my parents Xi Chen and Jun Qing for their consistent support.

References

[1] M Balmer, M Rieser, K Meister, D Charypar, N Lefebvre, K Nagel, and K Axhausen. Matsim-t: Architecture and simulation times. *Multi-agent systems for traffic and transportation engineering*, pages 57–78, 2009.

[2] J Barceló, E Codina, J Casas, JL Ferrer, and D Garcia. Microscopic traffic simulation: A tool for the design, analysis and evaluation of intelligent

transport systems. *Journal of Intelligent and Robotic Systems*, 41(2-3):173–203, 2005.

[3] E Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America*, 99(Suppl 3):7280–7287, 2002.

[4] P Busetta, R Rönnquist, A Hodgson, and A Lucas. Jack intelligent agents-components for intelligent agents in java. *AgentLink News Letter*, 2(1):2–5, 1999.

[5] A Byrne, K Courage, and C Wallace. *Handbook of computer models for traffic operations analysis*. 1982.

[6] G Lmmel C Dobler. A framework for large-scale multi-modal microscopic evacuation simulations. In *In proceeding of: International Conference on Evacuation Modeling and Management*.

[7] GDB Cameron and GID Duncan. Paramicsparallel microscopic simulation of road traffic. *The Journal of Supercomputing*, 10(1):25–53, 1996.

[8] D Charypar, K Axhausen, and K Nagel. *An event-driven parallel queue-based microsimulation for large scale traffic scenarios*. 2007.

[9] D Charypar, M Balmer, and K Axhausen. High-performance traffic flow microsimulation for large problems. In *Transportation Research Board 88th Annual Meeting*, number 09-1492, 2009.

[10] D Charypar and K Nagel. Generating complete all-day activity plans with genetic algorithms. *Transportation*, 32(4):369–397, 2005.

[11] F Ciari, M Balmer, and K Axhausen. Large scale use of collective taxis. 2009.

[12] P Cohen and H Levesque. Intention is choice with commitment. *Artificial intelligence*, 42(2):213–261, 1990.

[13] F de Boer, K Hindriks, W van der Hoek, and J Meyer. A verification framework for agent programming with declarative goals. *Journal of Applied Logic*, 5(2):277–302, 2007.

[14] H Dia. An agent-based approach to modelling driver route choice behaviour under the influence of real-time information. *Transportation Research Part C: Emerging Technologies*, 10(5):331–349, 2002.

[15] C Dobler. Implementation of a time step based parallel queue simulation in matsim. In *10th Swiss Transport Research Conference*, 2010.

[16] C Dobler, M Kowald, N Rieser-Schüssler, and K Axhausen. Within-day replanning of exceptional events. *Transportation Research Record: Journal of the Transportation Research Board*, 2302(1):138–147, 2012.

[17] M Feil, M Balmer, and K Axhausen. Enhancement and empirical estimation of matsims utility function.

[18] T Frazier and A Alfons. Generating a close-to-reality synthetic population of ghana. *Available at SSRN 2086345*, 2012.

[19] M Frick and K Axhausen. *Generating synthetic populations using IPF and monte carlo techniques: Some new results*. ETH, Eidgenössische Technische Hochschule Zürich, Institut für Verkehrsplanung, Transporttechnik, Strassen-und Eisenbahnbau, 2004.

[20] J HÃ1/4per, G Dervisoglu, A Muralidharan, G Gomes, R Horowitz, and PP Varaiya. Macroscopic modeling and simulation of freeway traffic flow. In *Control in Transportation Systems*, pages 112–116, 2009.

[21] D Helbing, A Hennecke, V Shvetsov, and M Treiber. Micro-and macrosimulation of freeway traffic. *Mathematical and computer modelling*, 35(5):517–547, 2002.

[22] J Illenberger, G Flotterod, and K Nagel. Enhancing matsim with capabilities of within-day re-planning. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 94–99. IEEE, 2007.

[23] RM Jones, JE Laird, PE Nielsen, KJ Coulter, P Kenny, and FV Koss. Automated intelligent pilots for combat flight simulation. *AI magazine*, 20(1):27, 1999.

[24] A Kesting, M Treiber, and D Helbing. Agents for traffic simulation. *arXiv preprint arXiv:0805.0300*, 2008.

[25] JE Laird, A Newell, and PS Rosenbloom. Soar: An architecture for general intelligence. *Artificial intelligence*, 33(1):1–64, 1987.

[26] CM Macal and MJ North. Tutorial on agent-based modeling and simulation. In *Proceedings of the 37th conference on Winter simulation*, pages 2–15. Winter Simulation Conference, 2005.

[27] M Maciejewski and K Nagel. Simulation and dynamic optimization of taxi services in matsim. Technical report, VSP Working Paper 13-05. TU Berlin, Transport Systems Planning and Transport Telematics, 2013. URL: www. vsp. tu-berlin. de/publications, 2013.

[28] A May. *Traffic flow fundamentals*. 1990.

[29] K Müller and Axhausen. *Hierarchical IPF: Generating a synthetic population for Switzerland*. Eidgenössische Technische Hochschule Zürich, IVT, 2011.

[30] M North, T Howe, N Collier, and J Vos. A declarative model assembly infrastructure for verification and validation. In *Advancing social simulation: the first world congress*, pages 129–140. Springer, 2007.

[31] L Padgham, D Scerri, G Jayatilleke, and S Hickmott. Integrating bdi reasoning into agent based modeling and simulation. In *Proceedings of the Winter Simulation Conference*, pages 345–356. Winter Simulation Conference, 2011.

[32] T Phung, M Winikoff, and L Padgham. Learning within the bdi framework: An empirical analysis. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 282–288. Springer, 2005.

[33] A Pokahr, L Braubach, and W Lamersdorf. Jadex: A bdi reasoning engine. In *Multi-agent programming*, pages 149–174. Springer, 2005.

[34] B Raney, N Cetin, A Völlmy, Mi Vrtic, K Axhausen, and K Nagel. An agent-based microsimulation model of swiss travel: First results. *Networks and Spatial Economics*, 3(1):23–41, 2003.

[35] B Raney and K Nagel. An improved framework for large-scale multi-agent simulations of travel behaviour. *Towards Better Performing Transport Networks*, page 305, 2006.

[36] A Rao and M Georgeff. Modeling rational agents within a bdi-architecture. *KR*, 91:473–484, 1991.

[37] A Rao and M Georgeff. Bdi agents: From theory to practice. In *ICMAS*, volume 95, pages 312–319, 1995.

[38] R Rönnquist. The goal oriented teams (gorite) framework. In *Programming Multi-Agent Systems*, pages 27–41. Springer, 2008.

[39] S Sardina, L de Silva, and L Padgham. Hierarchical planning in bdi agent programming languages: A formal approach. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1001–1008. ACM, 2006.

[40] D Singh, S Sardina, L Padgham, and G James. Integrating learning into a bdi agent for environments with changing dynamics. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 2525–2530. AAAI Press, 2011.

[41] M Tambe, WL Johnson, RM Jones, F Koss, JE Laird, PS Rosenbloom, and K Schwamb. Intelligent agents for interactive simulation environments. *AI magazine*, 16(1):15, 1995.

[42] J Thangarajah, L Padgham, and M Winikoff. Prometheus design tool. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 127–128. ACM, 2005.

[43] P Tranouez, É Daudé, and P Langlois. A multiagent urban traffic simulation. *arXiv preprint arXiv:1201.5472*, 2012.

[44] Z Wang. A preliminary report on the great wenchuan earthquake. *Earthquake Engineering and Engineering Vibration*, 7(2):225–234, 2008.

[45] R Waraich, D Charypar, M Balmer, K Axhausen, R Waraich, R Waraich, K Axhausen, and K Axhausen. *Performance improvements for large scale traffic simulation in matsim*. ETH, Eidgenössische Technische Hochschule Zürich, IVT, Institut für Verkehrsplanung und Transportsysteme, 2009.