ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-681

Integrating BDI Agents into a MATSim Simulation

Lin Padgham[1] and **Kai Nagel**[2] and **Dhirendra Singh**[3] and **Qingyu Chen** [4]

Abstract.

MATSim is a mature and powerful traffic simulator, used for large scale traffic simulations, primarily to assess likely results of various infrastructure or road network changes. More recently there has been work to extend MATSim to allow its use in applications requiring what has been referred to as "within day replanning". In the work described here we have coupled MATSim with a BDI (Belief Desire Intention) system to allow both more extensive modelling of the agent's decision making, as well as reactivity to environmental situations. The approach used allows for all agents to be "intelligent" or for some to be "intelligent"/reactive, while others operate according to plans that are static within a single day. The former is appropriate for simulations such as a bushfire evacuation, where all agents will be reacting to the changing environment. The latter is suited to introducing agents such as taxis into a standard MATSim simulation, as they cannot realistically have a predetermined plan, but must constantly respond to the current situation. We have prototype applications for both bushfire evacuation and taxis. By extending the capabilities of MATSim to allow agents to respond intelligently to changes in the environment, we facilitate the use of MATSim in a wide range of simulation applications. The work also opens the way for MATSim to be used alongside other simulation components, in a simulation integrating multiple components.

1 Introduction

MATSim (Multi-Agent Transport Simulation, [4, 5, 19]) is a traffic simulation software framework built around two principles: (1) Behavioral entities, such as travellers or taxi drivers, but also, say, traffic signals, are resolved individually. (2) The system should be fast enough to run on large scenarios with several millions of travellers [5]. The framework uses a co-evolutionary iterative approach to move the simulated system towards a user equilibrium [17]: Each synthetic traveller has several daily plans, one of them "selected"; all selected plans are executed in a synthetic reality and then scored based on their performance; plans with low scores are removed while plans with high scores are duplicated and mutated (cf. [1]). The result is that each synthetic traveller individually improves his/her plans, conditional on the environment generated by the other synthetic travellers. Mutation can include various choice dimensions, such as route choice [16], departure time choice [4], mode choice [10], activity location choice [11], etc.

A shortcoming of the original approach is that the synthetic travellers can only replan between iterations [17]; so-called "within-day" or "en-route" replanning [2] is not possible. However, in some applications it is difficult, or impossible, to model appropriately without the ability to generate or modify plans reactively during the within-day execution. For example, to understand possible emergency evacuation scenarios, it does not make sense to evolve a stable equilibrium. Rather one wants to model individuals dynamically adjusting their plans depending on the unfolding situation. Similarly, within a transport simulation, it is difficult to model taxis in the standard manner. It is far preferable to have them dynamically determine their plans throughout the day.

Two approaches have currently been followed to address this shortcoming. One approach [7] leaves the plan-following agents intact, but from time to time selects certain agents and modifies their plans in reaction to external conditions. For example, the routes of agents within a certain radius of a disturbance are recomputed, say, every 60 simulation time steps. Another approach [13] replaces the original MATSim DriverAgent object, which follows the precomputed plan, by a new object, which does not know its plan beforehand but instead decides on every next step when it is necessary.

A problem with both approaches is that they are somewhat ad-hoc: They have grown out of the specific MATSim environment. In both of the two approaches, the programmer has to work deeply within the MATSim environment, in order to either be able to replace existing plans while the agent is under way, or in order to be able to answer the specific requests that the simulation may have towards the agent. This is not always desirable, and requires the developer to have an in depth understanding of MATSim internals.

This paper presents an attempt at a more general approach, grounded in the general computer science principles of modularity. We use the very general and standard view of situated agents in an environment as taking *percepts* as input from the environment and producing *actions* which take effect in the environment [20]. We also use a well understood paradigm for modelling agent decision making, the Belief, Desire, Intention (BDI) approach, supporting a powerful yet intuitive model of human decision making, which balances commitment to specific goals, with constant ability to react to environmental changes. There are some BDI platforms (e.g. Jack [22] and Gorite [12]) which are very fast, and are capable of running tens or hundreds of thousands of agents. Plans are underway to extend this to even larger numbers as required by some MATSim applications, but already many applications can benefit from the addition of tens of thousands of BDI agents, within MATSim, possibly alongside a larger number of standard agents.

The main contributions of this work are:

- it allows for selected MATSim agents to have a complex decision making capability, that is both goal oriented, and responsive to changes in the environment, using the well established BDI paradigm. This is done using a general purpose interface of percepts and actions, for each agent, as described in section 2.1.

[1] RMIT University, Australia, email: lin.padgham@rmit.edu.au
[2] TU Berlin, Germany, email: kai.nagel@tu-berlin.de
[3] RMIT University, Australia, email: dhirendra.singh@rmit.edu.au
[4] RMIT University, Australia, email: s3257935@student.rmit.edu.au@rmit.edu.au

- it provides a modular approach to increasing the complexity of some of the agents in MATSim, with minimal impact on the current system. We show how the BDI reasoning can be incorporated into the existing MATSim infrastructure, using existing experimental MATSim packages. We also sketch an alternative possible approach within MATSim, to this integration (section 2.2).
- it supports representation of high level reasoning at an appropriate level of abstraction, separated out from simulation of the physical aspects. We describe in section 2.3, a generic infrastructure that can be incorporated into a BDI platform, to allow the percept/action interface to work seamlessly.
- it also allows for the decision making to receive information from a part of the simulation not modelled within MATSim, but affecting what happens in MATSim, such as fire spread data. This is explored further in our bushfire evacuation example, based on an application developed with the Country Fire Authority (section 3).

2 Integration Architecture

The integrated BDI/MATSim architecture allows selected MATSim agents to register with the BDI system, creating a decision making agent for each registered MATSim agent, as shown in figure 1. These decision making agents can then inform their MATSim counterparts regarding what they should be doing (actions), and can receive relevant environmental information from their MATSim counterparts (percepts), which can in turn affect the decision making. The BDI component can be seen as the agent "brain" while MATSim manages the agent "body" within the environment. Some MATSim agents may not require BDI counterparts, but simply follow their evolved daily plans. Depending on the application, there may also be BDI agents which are not represented in MATSim, but do have a role within the BDI component. The taxi job controller is such an agent in our application incorporating taxis into MATSim.

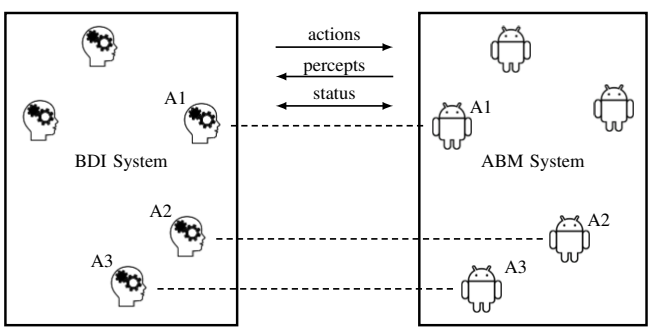

Figure 1. Some agents have a BDI brain and a MATSim body

2.1 The conceptual interface

The communication interface between the BDI representation of an agent and its MATSim counterpart is defined in terms of the standard agent concepts, *actions* and *percepts*. We have provided an infrastructure which is similar to that described in [15] which packages up all agent and percept information as a single message, with the appropriate actions/percepts collected from and distributed to the relevant agents by this infrastructure. To distinguish the high level actions managed via this infrastructure from related but different low level actions that agents perform within MATSim, we refer to the actions as *BDI-actions*.

- **BDI actions** are basically the level at which the reasoning agent decides to do something which can be carried out by the MATSim agent. For example in all the applications where we have used MATSim, `drive-to(loc)` is one of the BDI-actions. The MATSim counterpart executes this high level BDI-action by planning the route from the current position to the required location and inserting the relevant legs into the MATSim plan. Standard MATSim behaviour will then cause the agent to follow this route to the destination. The MATSim counterpart reports when the action succeeds, or when it fails.
  BDI actions are initiated by the BDI agent, which can also suspend or abort them. The BDI agent thus has control at all times over what action is being pursued, but the actual detailed carrying out is done by the MATSim agent. BDI-actions will generally take more than a single MATSim timestep to complete.
- **Percepts** are inputs "pushed" from the MATSim system, to the BDI system, providing information which conceptually should be *perceived* by each reasoning (BDI) agent. These percepts contain information that should automatically be *noticed*, and may trigger some response from the BDI agent. A traffic blockage may well be such a percept.

Additionally, the infrastructure provides a query mechanism to allow information to be "pulled" from the MATSim system, on demand, in order to reason about a particular situation. For example when deciding whether to drive-to location A or location B, the BDI agent may want to know the distance to each of these from its current location. This is done via a BDI *sensing action*, which sends a *query*, and results in a *response* percept being provided. Queries may be sent at any time, and may require some calculations, but are responded to immediately, and are guaranteed not to cause any changes to the MATSim environment. Their results may be stored in BDI agent beliefs, but the developer should be aware that such beliefs will not be updated until there is a new query.

Figure 2 shows the details of these interface communications.

BDI Actions	< *action-id, parameters, status*>
Percepts	< *percept-type, value*> (value may be a complex object)
Queries	< *query, response* >

Messages provided to specific agents via the interface.

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

Action states

Figure 2. The data that is passed between BDI and MATSim

2.2 Integration into MATSim

The MATSim package is made up of 5 separate components [3]: `Demand Modelling`, which sets up the scenario of the network and agents with plans, `Mobility Simulation` which runs the simulation for one day, `Scoring` which gives a score to each plan depending on how well the actual execution matched what was expected with regard to the plan, `Replanning` which takes some

number of agents with poorly performing plans, and generates new plans for them to try on the next day (using genetic algorithms) and `Result Analysis` which provides statistics and analysis at the end of the simulation. In this work we are primarily concerned with the `Mobility Simulation`, although the other components may still be used.

Our aim is to allow at least some of the agents (possibly all) to modify their plans during the execution of the `Mobility Simulation`, depending on factors in the environment. Figure 3 shows how MATsim (Mobility Simulation) has been extended with our *MATSim Module* containing:

- the *ModuleInterface* which communicates about actions and percepts with the BDI reasoning component, as described in section 2.1;
- the BDIMATSimWithinDayEngine which translates BDI Actions into updates to MATSim plans, as well as notifying percepts and action updates to the *ModuleInterface*; and
- the AgentActivityEventHandler which implements callbacks from the relevant MATSim events and provides percepts to the WithinDayEngine.

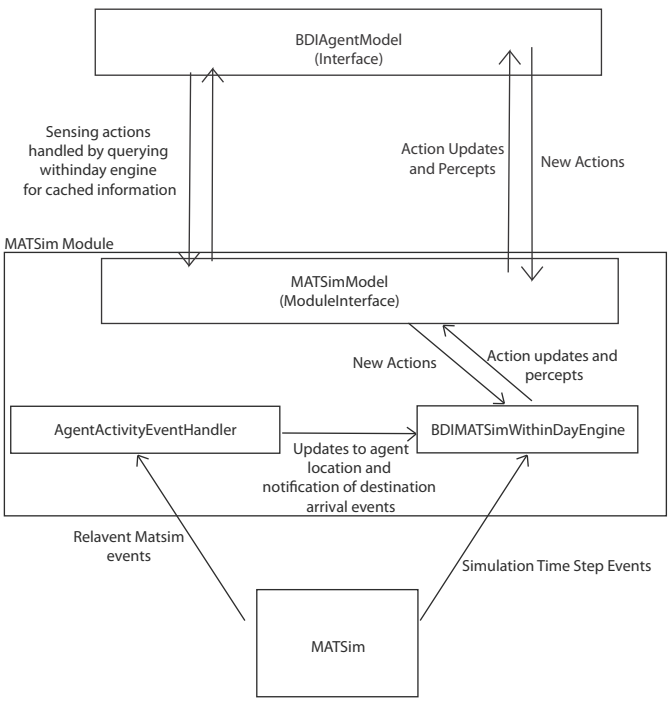

Figure 3. The enhanced framework which integrates the BDI reasoning with the MATSim simulation

Since MATSim (Mobility Simulation) progresses in time steps, we use a straightforward approach to managing the integration with the BDI system as follows:

Algorithm 1:

```
1 for each time step do
2     send percepts and action status to BDI module ;
3     possibly respond to queries from BDI module;
4     receive agent actions from BDI module ;
5     provide BDI agent actions to MATsim agents ;
6     move MATsim agents ;
```

This was achieved by starting with components developed by Dobler [7], but re-writing most of it for the purposes here. In particular, rather than collecting and centrally determining which agents will have changes made to their plans, the decisions about what each plan should contain for the next period, is made by the BDI reasoning counterpart. Typically a *BDI action* may involve several entries in the MATSim plan. Existing MATSim functionality for such things as route planning is used to translate a *BDI action* to drive from X to Y, into a series of legs in the MATSIM network, to be inserted into the MATSim agent plan. This plan is then followed, until such time as it is again changed by the BDI decision making.

The percepts that can be obtained must be linked in some way to the events which are available within MATSim, the full list of which is available from the documentation on the MATsim website. These include things such as vehicles entering or leaving a road segment, or agents arriving at a location. For example, the taxi application uses a percept to indicate when it is approaching the destination. This is used to trigger a plan to start monitoring for a suitable new job (if the destination is a drop-off) or to notify the operator so the passenger can be sent a text message (of a pick-up). To obtain this information (i.e. to add the monitoring for this percept) we needed to subscribe to the event of entering a road segment, and check if this road segment was regarded as "close to destination". This is an example of a percept which builds on the available information but requires some additional calculations. The other percept used in the taxi application was "at destination" which was simply a matter of subscribing to and passing on the event of an agent arriving at a location.

By defining and providing the percepts needed for the reasoning, the BDI agents are reactive to what is happening in the simulation. Queries, or percepts which are pulled rather than pushed, rely on accessing information stored in some attributes, such as, for example "current location" of an agent.

Discussion of implementation choices

An alternative could have been to use the approach of [13]. That approach replaces the original MATSim `PersonDriverAgentImpl`, which follows the pre-computed plan, by an agent which computes answers to requests by other means. For example, a driver agent needs to implement the method `Id chooseNextLinkId()`, which needs to return the id of the next link downstream every time a driver approaches an intersection. An advantage of this approach is that it does nothing until the relevant MATSim event occurs, whereas the approach used does have the communication overhead of interacting with the BDI infrastructure periodically, regardless of whether any percepts of interest have been observed. This is however a necessary price to pay if one wishes to allow, as we do, for percepts generated from some other component. These percepts (such as fire or weather information in a bushfire evacuation) may require the BDI agent to rethink what it is doing, and communicate this to its MATSim counterpart. This cannot be accomplished if control is relinquished only on generation of MATSim events.

Also, the approach by [13] necessitates that the complete agent logic is replaced. This goes against the requirements for the present study, which operates on a higher level: Percepts which trigger actions are relatively rare, and a typical action is the determination of a new destination, rather than a new turn at an intersection. Thus, it is advantageous to keep the MATSim plan-following structure intact, and to modify or add to the future parts of a plan when the need arises. Computational efficiency is achieved by filtering the informa-

tion on the MATSim side, to send only percepts which are specified to be of interest to the BDI module. In the case where there is no relevant information (in the form of percepts, or action status changes) from MATSim to cause any agent to consider what they are doing, or should do next, and also no information from other sources to initiate new reasoning in any agent, then only the infrastructure call will be executed. No BDI agents will actually run, making the computational cost minimal.

2.3 BDI Infrastructure support

The BDI frameworks that we are targetting in this work are those in the AgentSpeak [18] tradition, such as PRS [9], Jack [22] and Jason [6]. We have currently used Gorite and Jack, and will in future be using an open source BDI system for ongoing work. Any Java based system[5] in the AgentSpeak family can readily be used, by simply providing a small amount of platform specific infrastructure which we describe below. The *BDIAgentModel* of Figure 3 is the communicating interface of an encompassing *BDIModule* (not shown for brevity), and is responsible for unpacking incoming action updates and percepts, as well as collecting and sending new actions from BDI agents at the end of the reasoning cycle.

BDI Goals and Plans

BDI agent programs are essentially a set of plan rules of the form $G : \psi \leftarrow P$, meaning that plan P is a reasonable plan for achieving the goal (or responding to the percept) G when (context) condition ψ is believed true. P (the plan body) is then made up of sub-goals, which have associated plans, and actions. A plan rule can be chosen for instantiation, and its plan body executed, if the context condition ψ is True according to the agent's beliefs. The plan trigger G may be an internally generated goal (e.g. taking a lunch break at a certain time), a percept which is externally notified to the agent from the environment, or a message from another agent.

Given a set of plan rules, (the plan library), this collection of goals and plans can be represented as a goal-plan tree as shown in Figure 4. This is basically an AND/OR tree where goals have links to some number of plans, one of which must be chosen (OR), and plans have links to some number of (sub)goals/actions, all of which must be accomplished (AND) for the goal to succeed.

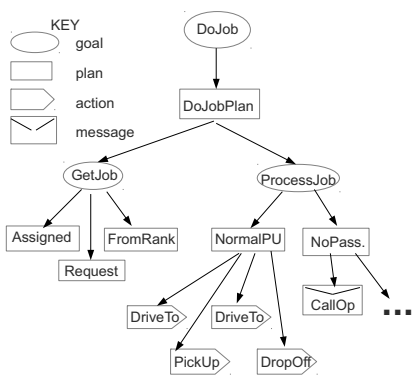

Figure 4. Partial Goal-Plan tree for a BDI Taxi agent,

[5] A non-java based BDI system could also be used, but a java system is more efficient given that MATSim is in java, so they can run in the same process.

So in Figure 4 the two subgoals *GetJob* and *ProcessJob* must both be processed successfully for the *DoJob* plan to succeed. *GetJob* has three possible plans: *Assigned* for the case where the taxi has already been assigned a job by the operator, *Request* for where he will select a job from the job board to request, and *FromRank* for where he will go to a rank to look for a passenger. Note that *Request* may have multiple possible instances, depending on how many suitable jobs are listed, and if one *Request* plan fails, the BDI execution will automatically try another plan for that goal, if available.

When the BDI agent reaches an Action node, such as *DriveTo* or *PickUp*, this is communicated to the MATSim agent counterpart, via the interface previously described. The BDI agent then waits until this action has completed, before continuing its decision making and acting. Thus the agent does not continue to the *PickUp* action until the *DriveTo* action is complete, which may take several MATSim timesteps. In order to complete the *NormalPU* plan, the passenger must be at the location for the *PickUp* action to succeed. If they are not, the action will fail. This can be achieved by having the BDI agent tell MATSim to *PickUp*, and add code in our *MATSim Module* to record a failure if the passenger is not there, which will then be passed back to the BDI agent. Alternatively the BDI agent can use a sensing action to ascertain if the passenger is there, and if not, fail the action. Once the action fails, its containing plan (*NormalPU*) will fail. This causes an alternative plan (perhaps involving contacting the operator to call the customer, or waiting 5 minutes) to be chosen if available and appropriate. Otherwise the failure will propagate up the Goal Plan tree, causing that *DoJob* goal instance to fail, in which case a new *DoJob* goal will be posted.

Responding to Environmental Events

The trigger for a BDI plan may be a goal (generated internally by the agent, or requested by some other agent), or an environmental event (or percept). Figure 5 shows how the specified percept, *CloseToDest* may trigger some plans, depending on the contextual information as to whether the destination is a pickup or dropoff location. The appropriate plan when close to destination, if dropping off, may be to source a new job (*SourceNext*) from the operator, whereas if picking up, may be to notify the waiting passenger *NotifyApproaching* with a text message.

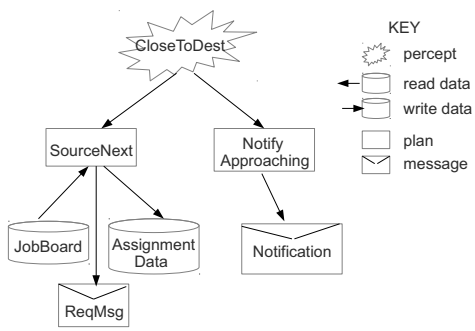

Figure 5. Event/Percept CloseToDest Triggers Plans

Generic Action Goal and Plan

In order to support the management of actions executed externally to the BDI system (i.e. in MATSim), we have defined and implemented a generic goal and plan that manages the communication with the

interface, and the waiting for the action to terminate. The structure can be seen in figure 6. All actions (such as *DriveTo* or *PickUp*) result in posting of an ActivateAction subgoal, containing the name of the specific action and its parameters (e.g. location for *DriveTo*). This in turn triggers a generic ActionPlan. This plan uses the parameters of the action name and any other parameters relevant to the action, to populate the communication data structures of the ModuleInterface which will be provided to the MATSim agent, as discussed in section 2.1. It also sets a belief regarding the status of the action, and then waits for a change in that status which will cause it to progress. In this way all infrastructure for managing the integration between the two systems is hidden from the application developer, who simply needs to specify that it is an action being executed.

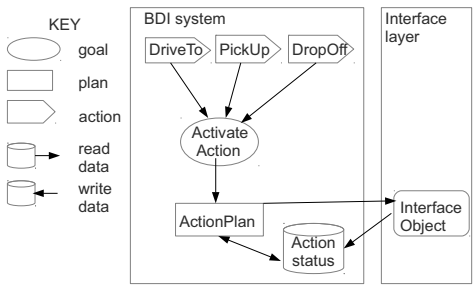

Figure 6. Generic Action Goal and Plan structure

Note that a plan executing a generic action is "blocked" until a success or failure is received from MATSim. This does not prevent the agent from excecuting other intentions. For instance, when dropping off (executing *DriveTo*), a taxi agent is still able to respond to the *CloseToDest* event and initiate the search for a new job. Moreover, the infrastructure supports the ability to abort executing plans by signalling DROPPED to the MATSim side (Figure 2).

3 Example Applications

We have developed two prototype applications using a BDI system integrated with MATSim in the manner described. The first is the integration of a simple taxi system into MATSim, as a demonstrator. The second is a bushfire evacuation system, developed initially in collaboration with the local Country Fire Authority, to help them understand potential scenarios [21]. We are now extending this for use as an interactive tool in training and with community groups.

Taxi application

In the Taxi application we used the Berlin road network as provided on the MATSim website. We have run the application with 15,963 standard MATSim agents, taken from the provided sample population, and 1000 of our BDI (GORITE) taxi agents. We have also run the full 15,963 agents as BDI taxi agents, for stress testing purposes. Profiling of the code reveals that a very small portion of time is spent in the BDI reasoning. The majority of execution time is spent in route planning. The need to plan routes during the single day execution would seem to be inevitable if agents are to have flexibility to decide their actions during the simulation execution – a pre-requisite for any approach to reactive agents. It may be possible in future work to investigate additional efficiencies in route planning. We note also that in our implementation both the BDI reasoning and MATSim run in a single java process. If separate processes were needed then communications could be expected to add to computational load.

As mentioned earlier the key percepts for this application were *arrival at destination*, and *close to destination*. The BDI actions were *DriveTo*, *Pick-up* and *DropOff* and the query required was *CurrentLocation*. In our prototype demonstrator, the taxi module is very simple, with a constant number of taxis and a single operator which generates jobs with a fixed frequency and random distribution. However, it would be straightforward to model a range of taxi specific policies and configurations, such as multiple competing companies, density of taxis at various times, time related fares, etc. that could impact on and be impacted by the rest of the traffic simulation. The ability to keep this separate from the MATSim component supports and facilitates modelling modularity.

Bushfire evacuation application

The role of MATSim in the bushfire evacuation simulation is control of the traffic simulation: a crucial aspect of any evacuation scenario. MATSim was chosen initially because of its wide usage, robustness and ability to model road speeds on segments, and traffic behaviour causing traffic jams. A specialised fire simulator, Phoenix RapidFire [8] provides fire data for a specific location under certain configured weather conditions. Use of this fire simulator was mandated by the Country Fire Authority, for whom we developed the simulation, as it is used in all their simulation work, and is highly trusted and respected.

The specific location which was used was the small coastal town of Breamlea in Victoria, Australia, as this was what was requested by the Country Fire Authority. The population at the latest census date (2011) was 444, and the number of dwellings was 279. In running the simulation, we placed only one agent per dwelling, assuming that this was roughly appropriate for an evacuation scenario where probably the household would evacuate in a single vehicle. In order to build the integrated simulation we needed to first obtain road network maps via OpenStreetMap[6] and then convert these to MATSim format using the support software provided by MATSim. Data on placement of buildings was obtained in the form of a shapefile from DataSearch Victoria[7] and population statistics were obtained from the Australian Bureau of Statistics.

Currently the simulation models only residents and their evacuation behaviour as fire warnings or other relevant instructions are issued. This behaviour includes activities such as checking on family or neighbours, either by phone, or driving, packing possessions, picking up family members, getting the car, arranging and waiting for a lift, etc. Currently all BDI actions affect MATSim only by driving to a location (as in the taxi application), or doing activities that simply take time, spent at a specified location.

The key percepts for this application currently are the percepts coming from other subsystems, in particular the fire simulator, which can alert when the fire is within a specified distance, and can provide data on speed and direction if queried. Although not currently used, percepts providing information about very slow travel speeds (indicating traffic jams), would make sense so that agents could use this as a trigger to reconsider either their route or destination. However, this was not required initially by the user organisation, and so has not yet been added.

[6] www.openstreetmap.org/a
[7] http://services.land.vic.gov.au/SpatialDatamart (account needed).

In the current extension work for use in training and with community groups, we are expecting to model also emergency management services who may need to take actions such as closing off roads. Currently we have facilitated only the dynamic change to the behaviour of agents. However the next phase will require looking at appropriate interfaces and mechanisms to allow within day modification of aspects of the road network.

4 Discussion and Conclusion

In this work we have successfully integrated BDI agents with MATSim allowing a separation of the reasoning (brain) from the physical simulation (the body). The integrated agents can dynamically instantiate goals and act to achieve them, choosing and modifying their behaviour depending on aspects of the environment. BDI platforms provide a powerful programming paradigm for developing intelligent agents. The basic concepts of goals and plans to achieve them are intuitive and easy for end users or domain experts to understand. In our work in the emergency management domain, we have found that emergency services workers and other personnel can readily discuss and refine the BDI graphical representations of people's behaviours.

This approach also realises many aspects of the architecture described in [14, p.178-79] with the agent brain(s) decoupled from the physical simulation. However, the approach is much more efficient than envisaged there, as each agent brain, although fully autonomous, runs within a single thread of a single process.

This decoupling of the detailed representation and reasoning of a component, from how that plays out within the traffic simulation facilitates integration of MATSim with other components. For example, in the emergency management scenario, the traffic component is clearly central. However the components that model the decision making of residents, and also potentially the decision making of emergency services personnel, are equally important. Some of the information needed for that decision making (and particularly how those results play out), is central to MATSim. But some may well arise from other sources. The obvious additional source is the fire itself, but also weather services, or other components may well be important. In addition to providing an approach to modelling reactive reasoning agents within MATSim, this work is a first step in allowing MATSim to be used as one component, within a larger whole.

ACKNOWLEDGEMENTS

This work is partly funded by ARC Discovery grant DP1093290, ARC Linkage grant LP130100008, and NCCARF grant EM1105. We would like to acknowledge the work of RMIT University students Arie Wilsher, Daniel Kidney, Faraz Muhammad, Megha Dhillion, and previous staff David Scerri and Sarah Hickmott, for assistance in development of some aspects of this work.

REFERENCES

[1] B. Arthur, 'Inductive reasoning, bounded rationality, and the bar problem', *American Economic Review (Papers and Proceedings)*, **84**, 406–411, (1994). 1

[2] N.C. Balijepalli, D.P. Watling, and R. Liu, 'Doubly dynamic traffic assignment – Simulation modeling framework and experimental results', *Transportation Research Record*, **2029**, 39–48, (2007). 1

[3] M. Balmer, *Travel demand modeling for multi-agent transport simulations: Algorithms and systems*, Ph.D. dissertation, Swiss Federal Institute of Technology (ETH) Zürich, Switzerland, 2007. 2

[4] M. Balmer, B. Raney, and K. Nagel, 'Adjustment of activity timing and duration in an agent-based traffic flow simulation', in *Progress in activity-based analysis*, ed., H.J.P. Timmermans, 91–114, Elsevier, Oxford, UK, (2005). 1

[5] M. Balmer, M. Rieser, K. Meister, D. Charypar, N. Lefebvre, K. Nagel, and K.W. Axhausen, 'MATSim-T: Architecture and simulation times', in *Multi-Agent Systems for Traffic and Transportation*, eds., A.L.C. Bazzan and F. Klügl, 57–78, IGI Global, (2009). 1

[6] Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge, *Programming Multi-agent Systems in AgentSpeak Using Jason*, Wiley, 2007. Wiley Series in Agent Technology, ISBN: 0470029005. 4

[7] C. Dobler, 'Implementations of within day replanning in MATSim-T', IVT Working paper 598, Institute for Transport Planning and Systems, ETH Zurich, Zurich, Switzerland, (2009). 1, 3

[8] TJ Duff, D Chong, and KG Tolhurst, 'Quantifying spatio-temporal differences between fire shapes: Estimating fire travel paths for the improvement of dynamic spread models', *Environmental Modelling and Software*, **46**, 33–43, (2013). 5

[9] M.P. Georgeff and F.F. Ingrand, 'Decision Making in an Embedded Reasoning System', in *IJCAI*, pp. 972–978, (1989). 4

[10] D. Grether, Y. Chen, M. Rieser, and K. Nagel, 'Effects of a simple mode choice model in a large-scale agent-based transport simulation', in *Complexity and Spatial Networks. In Search of Simplicity*, eds., A. Reggiani and P. Nijkamp, Advances in Spatial Science, chapter 13, 167–186, Springer, (2009). 1

[11] A. Horni, K. Nagel, and K. Axhausen, 'High-resolution destination choice in agent-based demand models', IVT Working paper 682, Institute for Transport Planning and Systems, ETH Zurich, Zurich, Switzerland, (2011). 1

[12] Dennis Jarvis, Jacqueline Jarvis, Ralph Rnnquist, and Lakhmi C. Jain, *Multiagent Systems and Applications - Volume 2: Development Using the GORITE BDI Framework*, volume 46 of *Intelligent Systems Reference Library*, Springer, 2013. 1

[13] M. Maciejewski and K. Nagel, 'Simulation and dynamic optimization of taxi services in MATSim', VSP working paper, TU Berlin, Transport Systems Planning and Transport Telematics, (2013). see www.vsp.tu-berlin.de/publications. 1, 3

[14] K. Nagel and F. Marchal, 'Computational methods for multi-agent simulations of travel behaviour', in *Moving through nets: The physical and social dimensions of travel*, ed., K.W. Axhausen, Elsevier, (2007). 6

[15] Lin Padgham, David Scerri, Gaya Buddhinath Jayatilleke, and Sarah Hickmott, 'Integrating BDI reasoning into agent based modelling and simulation', in *Winter Simulation Conference (WSC)*, pp. 345–356, Pheonix, Arizona, USA, (December 2011). 2

[16] B. Raney and K. Nagel, 'Iterative route planning for large-scale modular transportation simulations', *Future Generation Computer Systems*, **20**(7), 1101–1118, (2004). 1

[17] B. Raney and K. Nagel, 'An improved framework for large-scale multi-agent simulations of travel behaviour', in *Towards better performing European Transportation Systems*, eds., P. Rietveld, B. Jourquin, and K. Westin, 305–347, Routledge, London, (2006). 1

[18] Anand S. Rao, 'AgentSpeak(L): BDI agents speak out in a logical computable language', in *Agents Breaking Away: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96)*, eds., Walter Van de Velde and John Perrame, pp. 42–55. Springer Verlag, (January 1996). LNAI, Volume 1038. 4

[19] Marcel Rieser, Christoph Dobler, Thibaut Dubernet, Dominik Grether, Andreas Horni, Gregor Lämmel, Rashid Waraich, Michael Zilske, Kay W. Axhausen, and Kai Nagel. MATSim user guide, 2013. Accessed 2013. 1

[20] Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edn., 2009. 1

[21] David Scerri, Ferdinand Gouw, Sarah L. Hickmott, Isaac Yehuda, Fabio Zambetta, and Lin Padgham, 'Bushfire BLOCKS: a modular agent-based simulation', in *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 1643–1644, (2010). 5

[22] Michael Winikoff, 'Jack intelligent agents: An industrial strength platform', in *Multi-Agent Programming: Languages, Platforms and Applications*, eds., Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah-Seghrouchni, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, 175–193, Springer, (2005). 1, 4