Jadex BDI Agents Integrated with MATSim for Autonomous Mobility on Demand *

Marcel Mauri[0000−0002−4135−1945], Ömer Ibrahim Erduran[0000−0002−1586−0228], and Mirjam Minor[0000−0002−6592−631X]
{mauri, erduran, minor}@cs.uni-frankfurt.de

Department of Computer Science, Goethe University Frankfurt, Frankfurt am Main, Germany

Abstract. This paper presents our extension for the *BDI-ABM* interface, which provides a connection layer for BDI agents to interact with Agent-based Models (ABM) such as simulation platforms in an integrated Multi-Agent System (MAS). We introduce a new version of the *ABM-Jadex* layer, which provides the possibility to attach *BDI* Agents developed with *Jadex*, an Agent Development Framework, to the *MATSim* traffic simulation environment. We introduce cognitive vehicle agents capable of negotiating among themselves via the contract net protocol. The scalability of the integrated MAS architecture is tested in the first experiments simulating the behavior of a fleet of autonomous e-trikes.

Keywords: BDI Agent, BDI-ABM Framework, Traffic Simulation, Jadex, Agent Development Framework

1 INTRODUCTION

Sustainable mobility is one of the global challenges. Large-scale systems with agent-based technology can contribute to reducing traffic emissions [4]. Simulations of vehicle agents may facilitate better decisions in urban development, usage of multi-modal mobility, or traffic operation. Platforms like *Grab*[1] have recently emerged and launched mobility services with fleets of public and private vehicles. The *Grab* app connects passengers with private hire, taxi, and coach drivers. However, it takes seven minutes according to Grab's web page to be matched with an appropriate vehicle when using its ride-sharing service GrabShare which takes more bookings in one ride than one.[2] Cognitive software agents with negotiation capabilities may provide a more efficient, scalable solution for transport tasks, especially for Autonomous Mobility on Demand (AMoD) scenarios. An AMoD system consists of a fleet of autonomous vehicles that pick

* An earlier version of this paper had been presented at the LWDA 2023 workshop in Marburg, Germany [23]

[1] https://www.grab.com/, last access: 04/16/2024

[2] https://www.grab.com/sg/inside-grab/stories/grabshare-weve-revamped-our-carpooling-service/, last access: 04/16/2024

up passengers and transport them to their destination [40]. Cognitive software agents have been applied in a wide range of real-world domains and scenarios for solving different challenges [16, 4, 22, 15]. BDI agents are based on the human reasoning cycle, translating into beliefs, desires, and intentions [17]. Their ability to deal with multiple goals in parallel predestines them to negotiate, maintain battery power, and steer driving actions simultaneously.

Many agent development platforms supporting BDI agents provide a simulation environment. Frequently, those simulations are rather simplified and closely application-specific. In contrast, stand-alone simulation platforms are mostly limited to simple agent types and do not support BDI agents. This makes it difficult to carry out more complex simulations with BDI agents. Singh et al. [34] open an alternative strand of research. They integrate agent development platforms following the BDI agent architecture [17] with rich, agent-based simulation platforms. Figure. 1 illustrates how the cognitive BDI agents in the upper layer interact with each other and the simulation platform. The spotlights indicate that a cognitive vehicle agent receives sensory inputs from its particular avatar in the simulation and decides its actions.

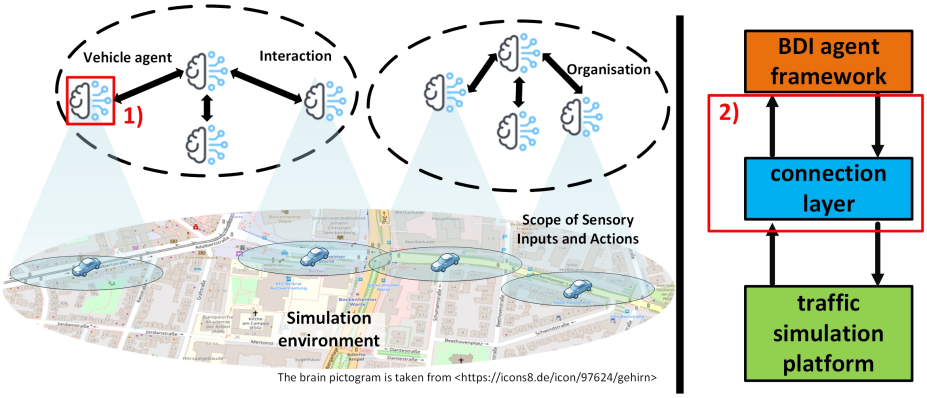


Fig. 1: Paper contributions (left side: following Klügl [20]).

The main contribution of this paper is twofold (cmp. Figure 1):

1. the *design of self-managing vehicle agents* for AMoD applications following the BDI paradigm and using the contract net protocol to negotiate workloads among themselves.
2. the *synchronization between a BDI agent development framework and a traffic simulation platform* building upon the results of Singh et al. [34].

We use mainly open-source tools and framework solutions to implement the integration according to two MAS components. For the agent development framework we choose Jadex [27]. It is a proven framework that supports BDI

agents and is Java-based. A preliminary version of the vehicle agents implemented in Jade [5] has been published in previous work [15]. The richer architectural support of Jadex compared to Jade led us to redesign the cognitive agents in the Jadex framework. MATSim [39] is used as the simulation platform. MATSim is an agent-based traffic simulation platform, widely used and based on Java. To connect these two platforms, we build upon the already existing BDI-ABM framework [34] and add a new interface for integrating Jadex and MATSim which is not yet included. BDI-ABM has been used to train approximately 60 emergency management specialists from 20 different agencies on bushfire evacuation recently [32]. There is already an integration between an outdated version of Jadex and another simulation platform. The intended application scope of the integrated MAS is a broad range of scenarios in AMoD, including ride-hailing, last-mile delivery, or disposal logistics. For the experimental evaluation, we have chosen an AMoD test scenario with a fleet of autonomous e-Trikes.

The remainder of this paper is structured as follows: Section 2 presents the related works. The idea and the design of the BDI-ABM integration in general as well as the design of *Jill-MATSim*[3] integration in evacuation scenario simulation, which is the main basis for the synchronization of Jadex and MATSim, is covered in Section 3. Section 4 contains the concept design and the implementation of the integration layer. The conceptual framework of the Jadex vehicle agent is covered in Section 5. In Section 6, the first experimental results toward scalability of the fleet size for elastic demands are described based on real-world data from a ride-hailing scenario. Section 7 concludes and discusses future work.

2 RELATED WORK

Integrating autonomous software agents into simulation environments has been researched extensively. Software agents have been covered in survey works focusing on the different development frameworks as well as the extensibility of the cognitive architecture [9, 21, 1, 13]. For example, the works of *Timoteo et al.* [38] and *Sadeghi Garyan et al.* integrate software agents, built with Jade in [30], and the traffic simulation SUMO[4]. The application of Contract Net Protocol for transportation scheduling has been investigated by *Fischer et al.* [19] and *Dorer and Calisti* [12]. The development of software agents is a well-researched field with a plethora of Agent development frameworks proposed by different research labs and organizations. *Silva et al.* cover the BDI agent architecture in their survey [31] and point out several research directions. The main contribution of this paper extends the *BDI-ABM* environment[5]. This work is grounded in several research papers mentioned in this section. The concepts and fundamental approach of *BDI-ABM* is presented by *Padgham et al.* in [24]. Here, the authors present multiple layers for integrating different agent development frameworks that especially implement the BDI Agent architecture, with ABMs.

[3] https://github.com/agentsoz/jill (last access: 04/16/2024)
[4] https://sumo.dlr.de/docs/index.html, last access: 04/16/2024)
[5] https://github.com/agentsoz/bdi-abm-integration (last access: 04/16/2024)

In this context, ABMs provide the environment, where the agents can interact. Thus, research has been conducted in the application scenario of emergency evacuation and multi-modal transportation at a city-wide level [32]. One of the layers connects the simulation environment *MATSim* [25] with the agent development framework *Jill*. A Jadex layer is also considered in BDI-ABM. However, it is outdated and does not support the connection to the current versions of Jadex BDI agents, called *BDIv3*. Furthermore, it is developed to connect Jadex agents to the *Repast* simulation[6]. Therefore, a connection to MATSim is not provided. Furthermore, there is also no demonstration or example freely available that demonstrates the interaction of Jadex with those simulation environments.

3 FOUNDATIONS

BDI-ABM is listed as a plugin for *MATSim* providing the connection of BDI agents to MATSim [25, 35]. One application of *BDI-ABM* is in the Emergency Evacuation Scenario (EES) [33, 34], where agents in *Jill* are combined with *MATSim* by using the *BDI-ABM* framework. The integration of Agent development frameworks and traffic simulation has also been conducted by *Soares et al.* by integrating *Jade* platform and the *Sumo* traffic simulation [37]. Developing a simulation environment for software agents represents the same challenge as developing cognitive agents. *Ricci et al.* use an *Artifact-based approach* [29, 28]. *Davoust et al.* consider an *Unmanned aerial vehicle* (UAV) scenario where the agents interact with the simulation environment [10]. Here, the focus is set on the computational performance of executing the framework.

3.1 Traffic simulation

In our considered domain of traffic simulation, we focus on AMoD settings, where the vehicle agents transport customers from a starting position to their desired destination. MATSim is an activity-based, extendable, multi-agent simulation framework implemented in Java, which is open-source [39]. MATSim is developed using the concept of agent-based modeling that is specified for transport simulation. This framework is designed for large-scale scenarios and is usually used to model a single day. With MATSim, it is possible to simulate traffic, taxi fleets, mobility as a service as well as different modes of transportation. By using MATSim, different modes of Mobility on Demand systems can be simulated. In the current version of MATSim[7] the contribution package *DVRP* provides the necessary components for setting up a ride-sharing or ride-hailing simulation. In addition, the contribution *DRT* provides ride-pooling including vehicle agents with additional capacities. It is built on top of the *DVRP* package. Recent work that investigates scenarios on Mobility on Demand is from *Bischoff et al.*, where ride-pooling and shared taxi fleets are simulated on a city-wide scale analyzing

[6] https://repast.github.io/
[7] version 15.0, https://github.com/matsim-org/matsim-libs (last access: 04/16/2024)

the fleet performance [6, 8, 7]. Other mentionable work investigating ride-pooling by using MATSim is from *Zwick et al.* [42] and *Kaddoura & Schlenther* [18]. Our work differentiates from the previously mentioned works since we not only consider MATSim solely but investigate the interaction of external BDI agents with the simulation platform. Therefore, the mentioned MoD and Mobility as a Service (MaaS) components in MATSim are not considered in our work.

On a high level, the *BDI-ABM* framework contains several integration layers for different *Agent Development Frameworks* (ADF) implemented in Java. Especially, the framework contains a generic layer, which represents a connection layer for different ADF and simulation environments. For each ADF and simulation environment, a specific layer is developed, which interacts via *BDI-ABM*. According to *Singh et al.* [34], the mentioned layer provides the possibility to connect other simulation environments as well. Thus, we developed a novel integration layer for the connection of Jadex and MATSim. In *BDI-ABM*, there exists an old integration layer for Jadex. However, the layer is customized for elder versions of the ADF and only works with the simulation environment *Repast* and not with MATSim. The advantage of Jadex over other ADFs is that the contract net protocol [36] for the communication between agents has been integrated and the Jadex application is currently further developed. At the same time, MATSim is a mature and powerful traffic simulator that can be used for large-scale traffic simulations, primarily to assess the likely results of various infrastructure or road network changes.

3.2 Interface for cognitive agents

In the conceptual framework of BDI-ABM [34], some agents in the simulation have a *"brain"* in the BDI system, which is the decision-making component, and a *"body"* in the ABM system which carries out actions. An agent in this integrated framework will be situated in an environment where it can perceive environmental input via percepts, and act, via actions. These activities of perceiving and acting will happen inside the ABM, where the "body" interacts with the physical world of the domain. To be precise, as shown in Fig. 1 with the arrows of action and percept, the perceptions from ABM will be communicated to the "brain" in BDI, and the "brain" will use its decision-making mechanism to select the suitable action based on the input from percepts, the chosen action will be delivered back to ABM to be carried out. It is defined in the conceptual framework that a percept going into BDI from ABM does not have to be identical to the percepts in ABM. The percept in BDI is a high-level percept composed of lower-level observations of the environment, which are the percepts represented in ABM. Similarly, an action going from the BDI agent to its ABM counterpart must typically be decomposed into a sequence of lower-level environment actions that the ABM agent knows how to perform. In terms of data transfer between the BDI and ABM systems, two key optimizations for this integrated framework are defined. The first one is that a single data container is passed between the systems in each simulation cycle. The data container bundles the messages for all agents and delivers them all together to the other system to simplify the

synchronization between the systems. The second one is that not every percept is computed and pushed to the BDI system on every cycle. The reason is the BDI agent processes information contextually and only certain information is useful in certain situations. In case of ad-hoc information requirements, the BDI agent can pull this percept from the ABM environment as needed via the percept queries function. From the technical point of view, the framework consists of three distinct layers. First is a generic layer, which realizes the conceptual model from the previous section. The second layer is the system layer, which provides the code necessary for linking a particular BDI or ABM system into the generic layer. With this layer, built on top of the generic layers, specific BDI systems like Jadex and Jill, as well as ABM systems (i.e. MATSim), can receive and send percepts and actions back and forth. The last layer is the application layer, which provides the application-specific code including agent behavior and reasoning. Overall, the BDI application provides action decisions to the ABM and the ABM provides observations and environmental information of interest to the BDI module.

4 JADEX-MATSIM INTEGRATION LAYER

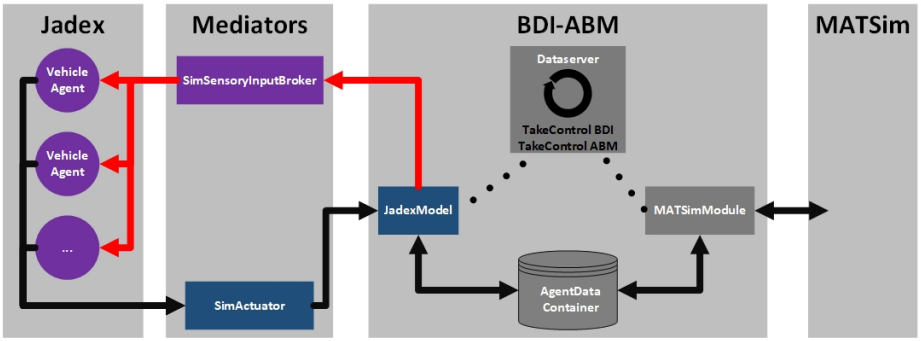


Fig. 2: Simplified illustration of the connection between Jadex and MATSim by BDI-ABM

The Jadex-MATSim integration framework is inspired by the already existing Jill-MATSim integration framework [35]. Figure 2 shows the new integration layers with the existing components depicted in grey color. The BDI-ABM layer synchronizes the mutual control taken by the cognitive side (Jadex agents) and the simulation side (MATSim). The *Dataserver* component controls the access to a shared memory structure called *AgentDataContainer*. The *Dataserver* grants read/write access to the cognitive side via the *TakeControl BDI* command and withdraw it via the command *TakeControl ABM*, which provides the simulation side with read/write access. Intermediate results from the reasoning cycles of

the BDI site are stored in *AgentDataContainer* to be shared with the simulation and the simulation outputs and vice versa.

The *JadexModel* is responsible for initializing the BDI agents and controls the incoming and outgoing data from and to Jadex. To connect the vehicle agents with the BDI-ABM layer, the *SimSensoryInputBroker* and the *SimActuator* play the role of mediators. The mediators are required because Jadex active components like the vehicle agents cannot be accessed directly by external (non-Jadex) components [26]. The *SimActuator* is used by the vehicle agents to write the actions (*drive-to*) into the *AgentDataContainer*. The *SimSensoryInputBroker* distributes the incoming data from the BDI-ABM (MATSim) side. The entries of the *AgentDataContainer* are directly written into the beliefs of the respective vehicle agents. Once the *SimActuator* has collected new *drive-to* commands from the vehicle agents, the *JadexModel* will update the content of the *AgentDataContainer* and notify the *Dataserver* to pass control to the MATSim side again. The *MATSimModule* [24] will then translate the BDI-actions from the *AgentDataContainer* into low-level actions for MATSim. This means that the updates are performed in a non-equidistant manner.

Unlike the Jill-MATSim integration framework, our Jadex-MATSim integration layer allows agents to continue other non-driving related activities when the control is currently at the MATSim side. All other actions such as communication, negotiation, and calculations are carried out independently of the cycle described in Figure 2.

5 BDI VEHICLE AGENTS

The agent framework we have developed consists of different types of agents. The geographical environment is divided into multiple zones, each with a responsible area agent. Vehicle agents are autonomous vehicles that are distributed in the application area. They can check in and out at their area agents when they enter or leave their zone and send an update with their current location after every completed journey. When a customer requests a trip, it is delegated to the area agent whose area of responsibility the starting position of the trip is located.

The area agent sends the request to the vehicle agent which is located closest to the start position. The vehicle agent will then evaluate how well it is suited to fulfill the customer's request considering the amount of already accepted trips, the battery level (etc.). Depending on the outcome, it processes the request or negotiates with other vehicle agents in its area of operation to delegate it to a more suitable one. Therefore, it can request a list of other vehicle agents from its associated area agent. The negotiation between the vehicle agents will be realized by the Contract Net Protocol (CNP) [36]. Thus, the vehicle agents are self-managed. When there are no customers, they drive to safe parking spaces, and when their battery level is low, they drive to a charging station.

Jadex agents implement a BDI architecture using beliefs, goals, and plans. Beliefs represent the current knowledge of the agent. Desires are goals that are desirable for the agent in general while intentions are a subset of the desired

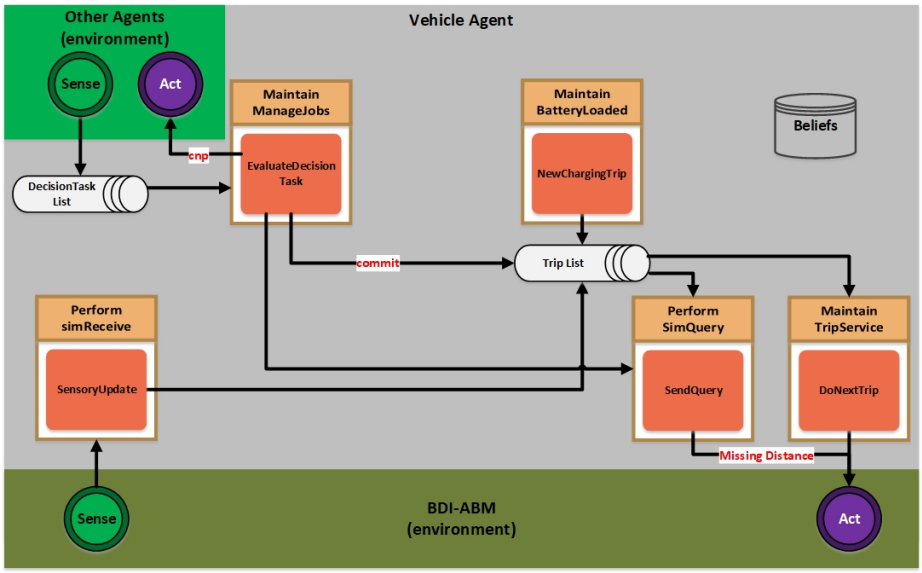


Fig. 3: The components of the vehicle agent architecture

goals for which the agent has committed. Goals in Jadex are used to implement both desires and intentions. Depending on the current state of a goal its theoretical meaning [17] may change between a desire only (inactive goal) or a desire and an intention (active goal). Jadex supports multiple types of goals for different purposes. There are performance goals that will only be executed once and maintain goals that will be triggered by a condition repeatedly. Plans describe the sequence of actions that are executed to achieve a goal.

Figure 3 shows the newly designed architecture of a vehicle agent. The design comprises five goals: *ManageJobs*, *BatteryLoaded*, *TripService*, *SimQuery* and *SimReceive* and their corresponding plans. Vehicle agents have an interface to the BDI-ABM Layer and a second interface for communication with other Jadex agents. Vehicle agents exchange messages with corresponding area agents and other vehicle agents. Incoming jobs are stored inside the *DecisionTaskList*. Every entry in this list is a not yet evaluated *DecisionTask*. A *DecisionTask* contains information about a trip that has been requested by a customer (start time, start position, end position, etc.). As long as this list is not empty the *ManageJobs* goal is active. The corresponding plan *EvaluateDecisionTask* iterates through the entries and determines by their actual progress state the next needed action. We decided against a design in which every *DecisionTask*/ *Trip* will cause the generation of a separate goal/plan that handles its processing. Our approach achieves the same functionality but is easier to handle.

For newly received jobs a utility score is calculated. This score determines how well this vehicle agent is suited to perform this ride. The agent self-assesses

three relevant criteria in its utility function, namely the length of the journey to the customer $u_{distance}$, the battery conditions $u_{battery}$, and the punctuality $u_{punctuality}$. The journey to the customer $journey$ is measured by the estimated distance d the agent has to drive to reach the customer. This calculation is based on the assumed position at which the agent is expected to be before starting the journey to the customer. This can be the final position of the previously planned $journey$, the location of a charging station, or simply the agent's current location. The Euclidean distance approximates the distance between geolocations specified by decimal degrees with a 1-meter variation in every 2,500-meter distance (cmp. the discussion in [16]). $u_{distance}$ is calculated as follows:

$$u_{distance}(trip) = max\{0, 100 - \frac{journey}{dmax}\} \tag{1}$$

where $dmax$ denotes the maximum possible distance between two points at the borders of the territory to normalize the distance values.

The battery conditions are scored as follows. The current battery level from the agent's beliefs is discharged by the estimated battery power consumption for serving the entire trip list committed so far and the job under consideration resulting in an approximate value of $battery$. Assuming a linear decrease of battery during traveling, the battery consumption in terms of several charge units is directly derived from the travel time. The *travel time* between two geolocations l_x, l_y at a constant velocity v is estimated as:

$$travel\_time(l_x, l_y) = \frac{d(l_x, l_y)}{v} \tag{2}$$

We consider a battery factor B_{factor} to rate battery lifetime-friendly thresholds higher:

$$B_{factor} = \begin{cases} 1.0, & battery > 80\% \\ 0.75, & 80\% \geq battery > 30\% \\ 0.1, & battery \leq 30\% \end{cases} \tag{3}$$

These thresholds are also used in other works [2, 41, 11]. An estimated $battery$ beyond the threshold gets a higher score to create incentives for the trip to reach the threshold. The function of the battery utility in light of the agent's current beliefs is defined as follows:

$$u_{battery}(trip) = \begin{cases} -\infty, & battery < 0 \\ B_{factor} * battery, & else \end{cases} \tag{4}$$

The punctuality is scored using the estimated arrival time at the prospective customer and its eventual delay $delay$ (approximated using $travel\_time$, cmp. equation (2)) behind the desired arrival time:

$$u_{punctuality}(trip) = \begin{cases} 100, & delay < \theta \\ 100 - \frac{100(delay-\theta)}{\theta}, & \theta \leq delay \leq 2\theta \\ 0, & else \end{cases} \tag{5}$$

where θ is a threshold the customer is ready to wait without any penalty.

Any distance that has not yet been calculated and is missing in the agent's belief database can be requested from the BDI-ABM environment. The requests are handled by the goal *SimQuery* and its plan *SendQuery*. On the current state of the implementation the goal *SimQuery* is not yet implemented. Currently, all distance calculations on the Jadex side are based on the Euclidean distance, as the agent does not know the actual path their counterparts will take in MATSim.

If the utility score is below a previously defined threshold the agent will start the CNP to delegate the trip. Before starting, it first requests a list of the vehicle agents registered with its associated area agent. The call for proposals (cfp) will be sent to the other vehicle agents in that area. Cfps are treated similarly to DecisionTasks and are stored inside the *DecisionTaskList*. Any further step of the CNP will also be executed by *EvaluateDecisionTask*. The recipients (contractors) then calculate how suitable they are for the job and send their proposals back to the sender (manager). When the manager has received all proposals it will send an accept/ reject to the contractors and delegate the *DecisionTask*. In our CNP implementation, the manager will always take part as a contractor too. By this, we can ensure that always the best-suited vehicle agent will get the job. Especially if the other agents' proposals are even worse than the manager's or if there are no other agents at all. There are currently no unexpected events or the possibility that the CNP will not be completed. In future updates, we will add a robustness component to our agent that will enable it to make correct decisions even in error cases (e.g. connection loss).

If the utility score of a *DecisionTask* is above the threshold or the agent has received an acceptance regarding a completed CNP the agent will commit it and create a corresponding trip. There are different subtypes of trips. Besides the customer trips that contain information about a trip that was requested by a customer, there are charging trips.

In the current state, each vehicle estimates what the charge level of its battery will be when all committed trips are finished. This estimate is refreshed every time a new trip is committed. If it falls below a charge threshold the goal *BatteryLoaded* is triggered. The corresponding plan *NewChargingTrip* will generate a *chargingtrip* that contains information for a *drive-to* charging station. Depending on the type a trip can contain one or more coordinates. While a charging trip just needs the coordinates of the charging station a customer trip needs the start position and the end position of a trip. Trips are stored in the TripList, which contains a sorted list of all committed trips that have not yet been started. Any newly created trip will be sent to a scheduler that will insert the new trips into the TripList and reschedule the entire list if needed. In the current state of our framework, we can only use FIFO scheduling to insert charging trips into the TripList. Future planning will be able to take various criteria into account to ensure that all journeys can still be made and that the remaining battery level is sufficient.

When the *TripList* is not empty the *TripService* goal is activated. The corresponding plan *DoNextTrip* takes the next trip from the *TripList* and sends

a *drive-to* command to the BDI-ABM framework which will cause the MATSim counterpart of the agent to drive to the specified location. When a *drive-to* command is sent the internal progress of this *CurrentTrip* is updated. As long as there is no feedback from the BDI-ABM framework the agent will not perform any other *drive-to* operations. Any information sent from the BDI-ABM framework to the Jadex vehicle agent is handled by the *SimReceive* goal. The plan *SensoryUpdate* processes all incoming information from the simulation site. This information could include, for example, the result of a *drive-to* with the new position of the vehicle or a requested distance. After the vehicle has received the confirmation that the last drive operation on the simulation site is finished *DoNextTrip* can resume its work. If the vehicle should break down every Trip inside the *TripList* will be terminated and will be considered as failed. The vehicle will then continue its journey on a full battery.

6 EXPERIMENTAL EVALUATION

The experimental evaluation investigates the following hypotheses:

H1 The customer satisfaction increases due to the capability to negotiate and delegate trips via CNP.
H2 The average travel distance per satisfied customer remains stable despite negotiation.
H3 Fewer agents are required to do the work due to negotiation.

Two measures are defined as evaluation criteria for the agent's behavior. The *order dropout rate* ODR measures the rate of trip requests that have been dropped. The threshold θ specified for calculating the estimated punctuality (see equation (5)) is also used during driving simulation. If the delay when arriving at the customer is above θ the customer is deemed to be missed. In this case, the vehicle agent drives to the start position of the next trip from the trip list. ODR is a measure of customer satisfaction. The *average travel distance* ATD measures the average travel distance to serve a trip. MATSim records the travel distances when simulating a trip between two geolocations. The ATD is calculated by dividing the overall travel distance from MATSim by the number of successfully served trips for the entire fleet. ATD is a significant factor of the traffic emissions (for a sample calculation see Chapter 18 of the MATSim book of Horni et al. [25]).

6.1 Experimental data

In our experiments, we consider three trip request data sets each containing up to 1000 trip requests from a single day. The dataset has its origin in an open-source bike-sharing data set by Deutsche Bahn[8]. In this historical dataset, we

[8] https://data.deutschebahn.com/dataset/data-call-a-bike.html, last access: 03/05/2024

extracted all trips that have their start and end positions inside the considered network area. Furthermore, we generated valid start and end coordinates for the data set to create a free-floating scenario. Precisely, the original data set contains trip request coordinates from a station-based bike-sharing system and gets new coordinates from a range across the campus map. Thus, we get trip request coordinates starting and ending in a free-float manner [16]. The amount of trip requests for a single day is also increased intentionally since the performance of the MAS can be observed clearly when a high capacity utilization arises during processing. The simulation takes place on a university campus map and simulates a day starting from 0:00 am to 11:59 pm. The campus map is extracted from *OpenStreetMap*[9]. To run our framework, we create a MATSim scenario with the required files [3]. Our MATSim scenario comprises the following data:

- *Population file*: Definition and starting position of the MATSim agents,
- *Network file*: Road network layer with roads of the university campus area,
- *Configuration file*: Configuration file for starting a MATSim simulation.

The sequence of data processing is conducted as follows: First, the trip request dataset is processed by the *Area Agent*. It delegates each trip request to the nearest vehicle agent which in turn processes it by its internal reasoning architecture. The Jadex agents send their *drive-to* operations to the equivalent MATSim agents. After completing the trip, the simulation updates the BDI agents by sending back status information concerning the driven trips including their routes and the travel time. In this manner, the data set is processed in a single simulation run representing a whole day of 24 hours.

6.2 Experimental runs

To show the influence of the total number of agents and the influence of the CNP, we set up ten different experimental configurations and ran each with all three datasets (1000 trip data files described above). The 10 configurations consist of five different numbers of vehicle agents (8, 10, 12, 14, and 16), each once with and once without the ability to use the CNP to delegate trips that received a low utility score (cmp. Tab. 1 and 2).

Each component of our utility function, $u_{distance}$, $u_{battery}$, $u_{punctuality}$ is weighted with 1/3 and will result in a total score between 0 and 100. The commitment threshold is set to 50, any value above 50 will commit the trip. Any value below this will start the CNP, with the other agents' bids based on the same utility function. To ensure that the trip is always delegated to the most suitable agent, the manager agent of the CNP will always be a participant too. It is therefore possible for the CNP to delegate a trip to an agent even if the score is less than 50. θ is set to 10 minutes. If a vehicle arrives at the customer's location later than 10 minutes after booking, the trip is treated as failed. The journey to the end of the trip will not be made in these cases. When the battery has dropped below zero this causes a breakdown of the vehicle. Thus, we estimate

[9] https://openstreetmap.org

the battery level after having served all trips in the trip list regularly for each commitment of a new trip. If the estimated battery level is below a threshold of 40%, we will generate a charging trip.

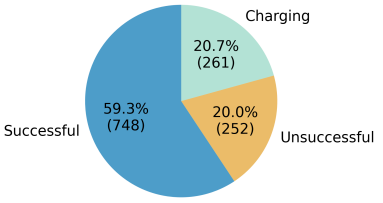
6.3 Experimental results

The experimental results are measured by ODR and ATD as described above. The presented results are the average form of the results gained by simulating the three datasets. Figure 4 depicts the number of served trips, missed trips, and charging trips for the ten configurations of experimental runs. Having a look at the pie charts from up to down, the ODR decreases with an increasing number of agents as expected (see also last column in Table 1). Comparing the pie charts on the left-hand side for the configurations without negotiation(Figure 4a, 4c, 4e, 4g, 4i) versus those including negotiation on the right-hand side (Figure 4b, 4d, 4f, 4h, 4j), the ODR decreases due to the negotiation capabilities for each pair of configurations. For all ten configurations, we can observe that for the same number of agents, the ODR with negotiation is always lower than with negotiation. For larger agent populations, the ODR decreases further. Obviously, at some point, the amount of work per agent becomes so low that the ODR values converge against zero for all configurations. Hypothesis H1 is confirmed by the experimental results.

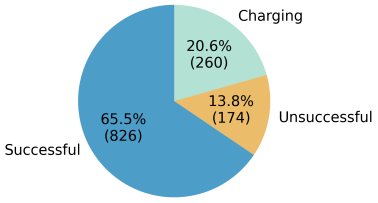
Table 1: Simulation results: Average Order dropout rate (ODR)

config	served trips	missed trips	charging trips	ODR
$nearest_8$	**748**	**252**	**261**	**25.2%**
$with\_neg_8$	**826**	**174**	**260**	**17.4%**
$nearest_{10}$	**804**	**196**	**258**	**19.6%**
$with\_neg_{10}$	**875**	**125**	**253**	**12.5%**
$nearest_{12}$	**852**	**139**	**239**	**13.9%**
$with\_neg_{12}$	**910**	**86**	**227**	**8.6%**
$nearest_{14}$	**881**	**125**	**245**	**12.5%**
$with\_neg_{14}$	**934**	**61**	**234**	**6.1%**
$nearest_{16}$	**893**	**89**	**235**	**8.9%**
$with\_neg_{16}$	**961**	**44**	**228**	**4.4%**

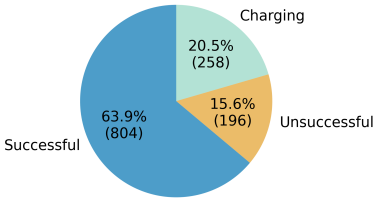
The ATD values are listed in Table 2. Since the configurations with negotiation capabilities serve more trips, they drive longer distances to do this work. Despite the narrow scheduling approach (FIFO), the differences are smaller than expected. This might be due to the fact, that the configurations without negotiation (*nearest*) assign the agents in a simple greedy manner. Only the current position of the agent at the time of assignment is considered. It is completely ignored that the agent travels further when serving the trips committed so far. As a consequence, the journey to the customer under consideration in the nearest configurations might be sub-optimal. Having a look at the ATD values (travel



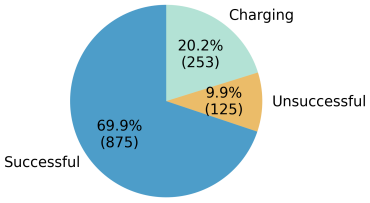
(a) 8 Agents without negotiation



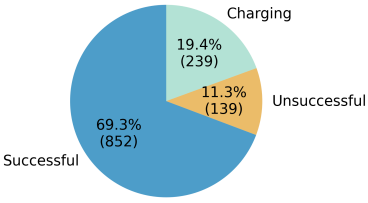
(b) 8 Agents with negotiation



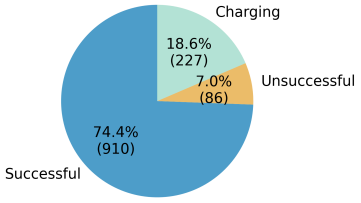
(c) 10 Agents without negotiation



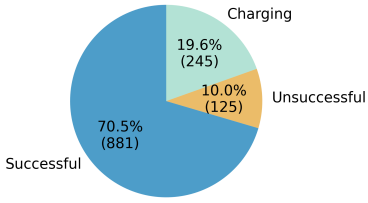
(d) 10 Agents with negotiation



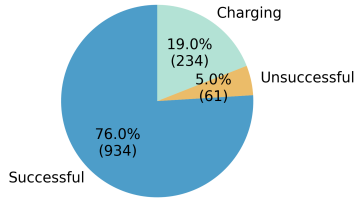
(e) 12 Agents without negotiation



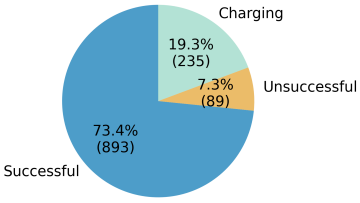
(f) 12 Agents with negotiation



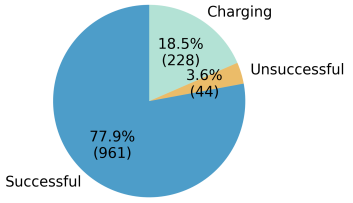
(g) 14 Agents without negotiation



(h) 14 Agents with negotiation



(i) 16 Agents without negotiation



(j) 16 Agents with negotiation

Fig. 4: Average values of 3 data sets of processed trips of each configuration

distance per successfully served trip), Table 2 shows a slight decrease in the travel distances due to negotiating. Hypothesis H2 is confirmed. An improved scheduling method would potentially reduce the total distances further. At the moment, bookings in advance are not yet allowed. Introducing pre-bookings might have an additional positive impact on the total distances driven.

Table 2: Simulation results: Average travel distance (ATD)

config	total distance (meters)	served trips	ATD (meters)
$nearest_8$	**1,442,185**	**748**	**1,928**
$with\_neg_8$	**1,535,585**	**826**	**1,859**
$nearest_{10}$	**1,461,647**	**804**	**1,815**
$with\_neg_{10}$	**1,522,737**	**875**	**1,740**
$nearest_{12}$	**1,467,224**	**852**	**1,722**
$with\_neg_{12}$	**1,500,483**	**910**	**1,649**
$nearest_{14}$	**1,452,130**	**881**	**1,648**
$with\_neg_{14}$	**1,483,613**	**934**	**1,588**
$nearest_{16}$	**1,454,333**	**893**	**1,629**
$with\_neg_{16}$	**1,454,542**	**961**	**1,514**

Hypothesis H3 can be discussed as well using the ODR results. Comparing the pie charts from Figure 4 gives an impression of the savings in terms of the number of agents required to do the work. The results provide a first hint towards the confirmation of H3. Larger experiments with even more data sets, further sizes of agent populations, a more sophisticated utility function, and a scheduling algorithm would hopefully allow us to confirm H3 with higher evidence.

7 DISCUSSION OF RESULTS AND FUTURE WORK

In this paper, we have designed, implemented, and tested two components of a MAS framework for AMoD applications. In detail, we have specified goals and plans for the architecture of BDI vehicle agents. We have defined a utility function for CNP-based communication between vehicle agents considering battery conditions, estimated punctuality, and length of the journey to the customer as decision criteria. We have extended the BDI-ABM integration framework by a Jadex-MATSim connection layer to achieve an integrated MAS framework with reasoning-simulation capabilities. The experiments with artificial data of a simulated e-trike service (based on a real data set from a bike-sharing scenario) show very promising results. A reduction of resources such as vehicles and energy (cmp. H2, H3) becomes apparent. This might have a significant impact on the development of future AMoD services based on MAS technology.

The integration of both ADF and simulation environment is part of a larger project, where other research areas of Multi-agent systems are considered. The project is ongoing work with still some minor open issues. Soon, a more sophisticated scheduling of trips within the vehicle agent's trip list will be implemented

based on distance values received from MATSim. Further, the process of charging and the charging infrastructure will be investigated more in-depth. Larger experiments on the scalability and robustness of the MAS in case of breakdown and connectivity losses will be conducted.

For future work, we plan to extend the cognitive agents with Machine Learning algorithms to investigate *Neuro-symbolic Agents* as well as their explainability [14]. Furthermore, we will design an experimental setup and run ride-hailing scenarios with different configurations. Other application scenarios like ride-pooling and waste collection by a fleet of trucks will be investigated using the presented framework in this paper. We think that our MAS framework with its reasoning-simulation integration contributes some foundational methods to achieve more sustainable solutions for mobility in the future.

Acknowledgements The authors would like to thank Mahkamjon Raupov and Olena Tsvietkova who contributed to the implementation and evaluation of the work.

References

1. Abar, S., Theodoropoulos, G.K., Lemarinier, P., O'Hare, G.M.: Agent based modelling and simulation tools: A review of the state-of-art software. Computer Science Review **24**, 13–33 (2017)
2. Ahadi, R., Ketter, W., Collins, J., Daina, N.: Siting and Sizing of Charging Infrastructure for Shared Autonomous Electric Fleets. AAMAS (2021)
3. Axhausen, K.W., ETH Zürich: The Multi-Agent Transport Simulation MATSim. Ubiquity Press (Aug 2016)
4. Bazzan, A.L., Klügl, F.: A review on agent-based technology for traffic and transportation. The Knowledge Engineering Review **29**(03), 375–403 (2014)
5. Bellifemine, F., Caire, G., Greenwood, D.: Developing multi-agent systems with JADE. Wiley series in agent technology, Chichester, reprint. edn. (2008)
6. Bischoff, J., Kaddoura, I., Maciejewski, M., Nagel, K.: Simulation-based optimization of service areas for pooled ride-hailing operators. Procedia Computer Science **130**, 816–823 (2018)
7. Bischoff, J., Maciejewski, M.: Proactive empty vehicle rebalancing for Demand Responsive Transport services. Procedia Computer Science **170**, 739–744 (2020)
8. Bischoff, J., Maciejewski, M., Nagel, K.: City-wide shared taxis: A simulation study in Berlin. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). pp. 275–280. IEEE, Yokohama (Oct 2017)
9. Cardoso, R.C., Ferrando, A.: A Review of Agent-Based Programming for Multi-Agent Systems. Computers **10**(2), 16 (Jan 2021)
10. Davoust, A., Gavigan, P., Ruiz-Martin, C., Trabes, G., Esfandiari, B., Wainer, G., James, J.: An Architecture for Integrating BDI Agents with a Simulation Environment. In: Dennis, L.A., Bordini, R.H., Lespérance, Y. (eds.) Engineering Multi-Agent Systems, vol. 12058, pp. 67–84. Springer International Publishing (2020)
11. Dlugosch, O., Brandt, T., Neumann, D.: Combining analytics and simulation methods to assess the impact of shared, autonomous electric vehicles on sustainable urban mobility. Information & Management p. 103285 (Feb 2020)

12. Dorer, K., Calisti, M.: An adaptive solution to dynamic transport optimization. p. 45–51. AAMAS '05, Association for Computing Machinery, New York, NY, USA (2005)
13. Dorri, A., Kanhere, S.S., Jurdak, R.: Multi-Agent Systems: A Survey. IEEE Access **6**, 28573–28593 (2018)
14. Erduran, Ö.I.: Machine Learning for Cognitive BDI Agents: A Compact Survey. In: ICAART (1). pp. 257–268 (2023)
15. Erduran, Ö.I., Mauri, M., Minor, M.: Negotiation in ride-hailing between cooperating BDI agents. In: Proceedings of the 14th International Conference on Agents and Artificial Intelligence. vol. Volume X, pp. 425 –432. Scitepress, Online Streaming (Feb 2022)
16. Erduran, Ö.I., Minor, M., Hedrich, L., Tarraf, A., Ruehl, F., Schroth, H.: Multi-agent Learning for Energy-Aware Placement of Autonomous Vehicles. In: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA). pp. 1671–1678. IEEE, Boca Raton, FL, USA (Dec 2019)
17. Georgeff, M., Pell, B., Pollack, M., Tambe, M., Wooldridge, M.: The Belief-Desire-Intention Model of Agency. In: Intelligent Agents V: Agents Theories, Architectures, and Languages, vol. 1555, pp. 1–10. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
18. Kaddoura, I., Schlenther, T.: The impact of trip density on the fleet size and pooling rate of ride-hailing services: A simulation study. Procedia Computer Science **184**, 674–679 (2021)
19. Klaus Fischer, J.P.M., Pischel, M.: Cooperative transportation scheduling: An application domain for dai. Applied Artificial Intelligence **10**(1), 1–34 (1996)
20. Klügl, F.: Multiagentensysteme. In: Handbuch der Künstlichen Intelligenz, pp. 755–781. De Gruyter Oldenbourg, Berlin/Boston, 6. auflage edn. (2021)
21. Kravari, K., Bassiliades, N.: A Survey of Agent Platforms. Journal of Artificial Societies and Social Simulation **18**(1), 11 (2015)
22. Malas, A., Falou, S.E., Falou, M.E., Itmi, M., Cardon, A.: Solving on-demand transport problem through negotiation. In: Proceedings of the Summer Computer Simulation Conference. pp. 1–7 (2016)
23. Mauri, M., Erduran, Ö.I., Anh, T.P.D., Minor, M.: Integrating BDI Agents with the MATSim Traffic Simulation for Autonomous Mobility on Demand. In: Leyer, M., Wichmann, J. (eds.) Lernen, Wissen, Daten, Analysen (LWDA) Conference Proceedings, Marburg, Germany, October 9-11, 2023. CEUR Workshop Proceedings, vol. 3630, pp. 247–258. CEUR-WS.org (2023), https://ceur-ws.org/Vol-3630/LWDA2023-paper23.pdf
24. Padgham, L., Nagel, K., Singh, D., Chen, Q.: Integrating BDI Agents into a MATSim Simulation. ECAI (2014)
25. Padgham, L., Singh, D.: Making MATSim Agents Smarter with the Belief-Desire-Intention Framework. In: ETH Zürich, Horni, A., Nagel, K., TU Berlin (eds.) The Multi-Agent Transport Simulation MATSim, pp. 201–210. Ubiquity Press (Aug 2016)
26. Pokahr, A.: Aktive Komponenten: Ein integrierter Entwicklungsansatz für verteilte Systeme. Ph.D. thesis, Hamburg University (2017)
27. Pokahr, A., Braubach, L., Jander, K.: The Jadex Project: Programming Model. In: Multiagent Systems and Applications: Volume 1:Practice and Experience, pp. 21–53. Springer, Berlin, Heidelberg (2013)
28. Ricci, A., Croatti, A., Bordini, R.H., Hübner, J.F., Boissier, O.: Exploiting Simulation for MAS Programming and Engineering—The JaCaMo-sim Platform. EMAS p. 19 (2020)

29. Ricci, A., Piunti, M., Viroli, M.: Environment programming in multi-agent systems: an artifact-based perspective. Autonomous Agents and Multi-Agent Systems **23**(2), 158–192 (Sep 2011)
30. Sadeghi Garjan, M., Chaanine, T., Pasquale, C., Paolo Pastore, V., Ferrando, A.: Agamas: A new agent-oriented traffic simulation framework for sumo. In: Malvone, V., Murano, A. (eds.) Multi-Agent Systems. pp. 396–405. Springer Nature Switzerland, Cham (2023)
31. Silva, L.d., Meneguzzi, F., Logan, B.: BDI Agent Architectures: A Survey. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. pp. 4914–4921. International Joint Conferences on Artificial Intelligence Organization, Yokohama, Japan (Jul 2020)
32. Singh, D., Ashton, P., Kuligowski, E., Pawan, G.: Bushfire evacuation decision support system use in incident management training. Australian Journal of Emergency Management **37** (2022)
33. Singh, D., Padgham, L.: Emergency Evacuation Simulator (EES) - a Tool for Planning Community Evacuations in Australia. In: Proceedings of the Twenty-Sixth IJCAI. pp. 5249–5251. Melbourne, Australia (Aug 2017)
34. Singh, D., Padgham, L., Logan, B.: Integrating BDI Agents with Agent-Based Simulation Platforms. Autonomous Agents and Multi-Agent Systems **30**(6), 1050–1071 (Nov 2016)
35. Singh, D., Padgham, L., Nagel, K.: Using MATSim as a Component in Dynamic Agent-Based Micro-Simulations. In: Engineering Multi-Agent Systems, vol. 12058, pp. 85–105. Springer International Publishing, Cham (2020)
36. Smith: The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on Computers **C-29**(12), 1104–1113 (1980)
37. Soares, G., Kokkinogenis, Z., Macedo, J.L., Rossetti, R.J.F.: Agent-Based Traffic Simulation Using SUMO and JADE: An Integrated Platform for Artificial Transportation Systems. In: Behrisch, M., Krajzewicz, D., Weber, M. (eds.) Simulation of Urban Mobility, vol. 8594, pp. 44–61. Springer Berlin Heidelberg (2014)
38. Timóteo, I.J., Araújo, M.R., Rossetti, R.J., Oliveira, E.C.: Trasmapi: An api oriented towards multi-agent systems real-time interaction with multiple traffic simulators. In: 13th International IEEE Conference on Intelligent Transportation Systems. pp. 1183–1188 (2010)
39. W Axhausen, K., Horni, A., Nagel, K.: The multi-agent transport simulation MATSim. Ubiquity Press (2016)
40. Zardini, G., Lanzetti, N., Pavone, M., Frazzoli, E.: Analysis and Control of Autonomous Mobility-on-Demand Systems. Annual Review of Control, Robotics, and Autonomous Systems **5** (2021), publisher: Annual Reviews
41. Zhang, H., Sheppard, C.J., Lipman, T.E., Zeng, T., Moura, S.J.: Charging infrastructure demands of shared-use autonomous electric vehicles in urban areas. Transportation Research Part D: Transport and Environment **78** (Jan 2020)
42. Zwick, F., Kuehnel, N., Moeckel, R., Axhausen, K.W.: Agent-based simulation of city-wide autonomous ride-pooling and the impact on traffic noise. Transportation Research Part D: Transport and Environment **90** (Jan 2021)