Multi-Agent Simulation of Individual Mobility Behavior in Carpooling using the Janus and JaSim Platforms

Stéphane Galland[a], Ansar-Ul-Haque Yasar[b], Luk Knapen[b], Nicolas Gaud[a], Davy Janssens[b], Olivier Lamotte[a], Geert Wets[b], Abderrafiaa Koukam[a]

[a] *Multiagent Group, IRTES-SET, UTBM, 90010 Belfort Cedex, France*
http://www.multiagent.fr
[b] *Transportation Research Institute (IMOB), Hasselt University, Wetenschapspark 5 bus 6, 3590 Diepenbeek, Belgium*
http://www.uhasselt.be/imob

Abstract

Carpooling is an emerging alternative transportation mode that is eco-friendly and sustainable as it enables commuters to save time, travel resource, reduce emission and traffic congestion. The procedure of carpooling consists of a number of steps namely; (i) create a motive to carpool, (ii) communicate this motive with other agents, (iii) negotiate a plan with the interested agents, (iv) execute the agreed plans, and (v) provide a feedback to all concerned agents. In this paper, we present a conceptual design of an agent-based model (ABM) for the carpooling a that serves as a proof of concept. Our model for the carpooling application is a computational model that is used for simulating the interactions of autonomous agents and to analyze the effects of change in factors related to the infrastructure, behavior and cost. In our carpooling application, we use agent profiles and social networks to initiate our agent communication model and then employ a route matching algorithm, and a utility function to trigger the negotiation process between agents. We developed a prototype of our agent-based carpooling application based on the work presented in this paper and carried out a validation study of our results with real data collected in Flanders, Belgium.

Keywords: Carpooling Problem, Multi-agent simulation, Organizational Model, Janus Platform, JaSim Environment Model, FEATHERS

1. Introduction

Nowadays, carpooling is emerging transportation mode that is eco-friendly and sustainable as it enables commuters not only saves the travel cost, such as fuel, toll and parking costs, of the carpooling participants but also reduces emissions and traffic congestions. Carpooling, known as ride-sharing, is the sharing of a car between people (agents) from a certain origin to a specific destination. Thus, in order to study the carpooling concept, we should take into account the interactions of two or more agents throughout the carpooling process. The procedure of carpooling consists of a number of steps namely (i) create a motive to carpool, (ii) communicate this motive with other agents, (iii) negotiate a plan with the interested agents, (iv) execute the agreed plans, and (v) provide a feedback to all concerned agents. Creating a motive means that a traveler (agent) may choose to carpool because of the availability of travel resources, time, monetary and route cost constraints.

Moreover, change in some socio-economic factors such as the increase in fuel price, in parking costs, or in the implementation of a new traffic policy, may trigger the initiative to carpool. Once to the decision has been made to carpool, the traveler (agent) will try to find one or more potential partners (agents). The carpool initiating agent will send a request to other interested agents in its vicinity. If one or more agents who receive this request are willing to carpool, then they begin the negotiation phase. In this phase, these agents will negotiate about sharing their travel resources and optimizing total costs and daily schedules. After reaching a compromise, these agents can carpool. Meanwhile, an agent can appraise its partners according to their degrees of faithfulness to the carpooling. We call this degree of faithfulness the agent reputation. This reputation factor can serve as a criterion for the selection of a potential partner for carpooling. An agent-based model (ABM) is a class of computational models for simulating the actions and interactions of autonomous agents with a view to assessing their effects on the systems as a whole [9, 20]. ABM is now widely used for modeling increasingly complex systems [18, 5]. Application of ABM is not only limited to the computer science domain. Currently, many research areas such as transportation behavior modeling, need to analyze and model the complex phenomenon of interactions between different entities. While traditional modeling tools cannot catch the complexity, ABM is able to do it through modeling the interaction of autonomous agents and deducing the rules for such a system. We, therefore, in this paper purpose an agent-based interaction model for the carpooling application.

This paper briefly describes a conceptual design of the carpooling application, initially proposed by [4, 1, 13] and using an agent-based model on the JANUS platform[1] [12]. A simulation model needs to be created to support the individual behaviors of the participants. The contribution of this paper is the design and the implementation of an agent-based model upon the JANUS multi-agent platform. This platform permits to individuals to (i) select the best transport mode according to their characteristics; (ii) maintain a social network; (iii) negotiate for carpooling; (iv) carpool the driver and the passengers of a car.

In the rest of this paper, we present some related work on the carpooling concept and ABM in section 2. In section 3, we explain our ABM for the carpooling application with details about the activities and the roles of the agents, and of the environment. We discuss in section 4 several implementation notes. Section 5 is dedicated to our concluding thoughts and ideas for future work.

2. Background

Related research on the carpooling concept is largely separated into two parts: (i) a technical study, and (ii) an empirical study. The first one focuses on the development of carpooling support systems with techniques of travel route matching [8, 19]. In the second part, the overall trend of carpooling — or of interrelationship between willingness-to-carpool and the socio-economic attributes of the carpooling participants — is treated in general [16, 15]. The studies mentioned previously are limited and do not consider the potential agent (participants) interactions to perform carpooling.

Most transportation-related applications of ABM are related to vehicle routing, pedestrian-flow simulation or demand modeling efforts [3]. Among these applications two of the more widely known are the ABM simulation platforms TRANSIMS and MATSIM. TRANSIMS, developed by Los Alamos Lab, is designed to supply transportation planners with more delicate information about traffic impacts, energy consumption, land-use planning and emergency evacuation [22]. MATSIM is also a large-scale agent-based simulator similar to TRANSIMS, but it is different using of XML and quickly

[1] `http://www.janus-project.org`

run simulation, due to a simplified traffic simulator [24]. Those applications only consider the whole effect of each agent's action in a system and cannot handle a detailed agent-to-agent or agent-to-environment coordination, communication and negotiation.

According to James Odell [21], "the environment provides the conditions under which an entity (agent or object) exists." The author distinguishes between the *physical environment* and the *communication environment*. The physical environment provides the laws, rules, constraints and policies that govern and support the physical existence of agents and entities. The communication environment provides (i) the principles and processes that govern and support exchanges of ideas, knowledge and information, and (ii) the functions and structures that are commonly deployed to exchange communication, such as roles, groups and interactions protocols between roles and groups. James Odell [21] defines an agent's social environment as "a communication environment in which the agents interact in a coordinated manner." This approach is shared by Ferber *et al.* [10], Cossentino *et al.* [5], and Galland *et al.* [11] who proposed to integrate the environment with organizational models. The JANUS platform [14, 12] provides an efficient implementation of agent-based and organizational concepts based on the Capacity-Role-Interaction-Organization metamodel [5]. The JASIM model and library[2] [11] is a model of environment based upon the JANUS platform. In the rest of this paper, the graphical notation is inspired from [5].

3. Carpooling Model

TODO: Put here an explanation about: What is really simulated by our model!

An agent is defined as someone who lives in our study area and executes his or her own daily schedule in order to satisfy his or her needs. A schedule is a combination of a number of trips associated with a number of activities. There are two categories of agents. An agent could either belong to one or both categories. The first category is a member of household such as the husband, the wife, the parents or the children. The second one is a member of the society such as a friend, a colleague, a neighbor, an employee (or an employer) or a student. In our model, we consider the socio-economic

[2] http://www.multiagent.fr/Jasim_Platform

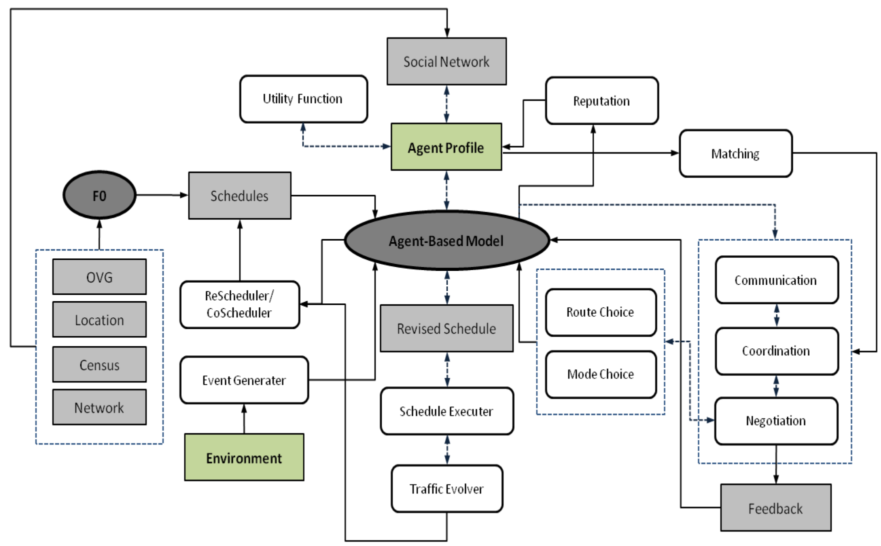

Figure 1: Agent-based Model for the Carpooling Application

attributes, including age, gender, income, education, relationship (within a family), job, vehicle and driving license ownership, as a set of input data.

These schedules and attributes are supplied by FEATHERS [2], an activity-based traffic demand model, which is developed by IMOB - Hasselt University Belgium. The environment is established as the spatio-temporal aggregate where an agent lives and executes its own daily schedule. FEATHERS framework supplies tailored memory structures such as, 'households', 'persons', 'activities', 'trips', 'cars', etc. and at the same time it is also equipped with a database structure that is able to nourish activity-based models being developed, assimilated or modified. FEATHERS inputs & outputs are shown in Figure 2 and Figure 1.

Agents follow a number of steps, including (i) goal setting, (ii) scheduling based on a given resource and environment, and (iii) execution of their schedule. Figure 3 describes the activities of an agent during the simulation.

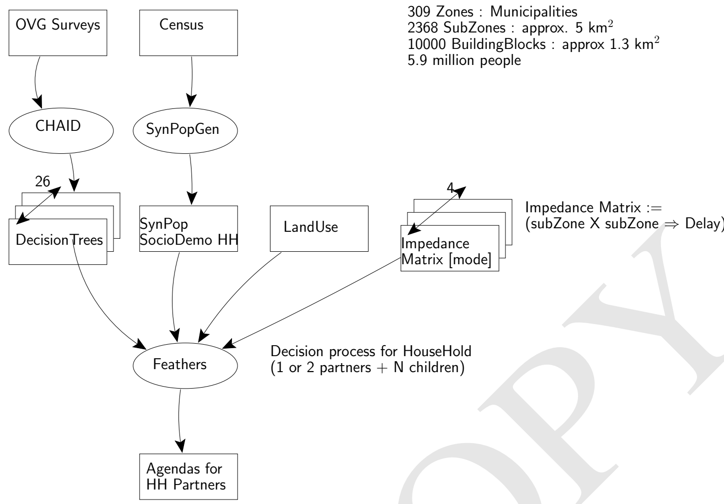

Figure 2: FEATHERS I/O Structure

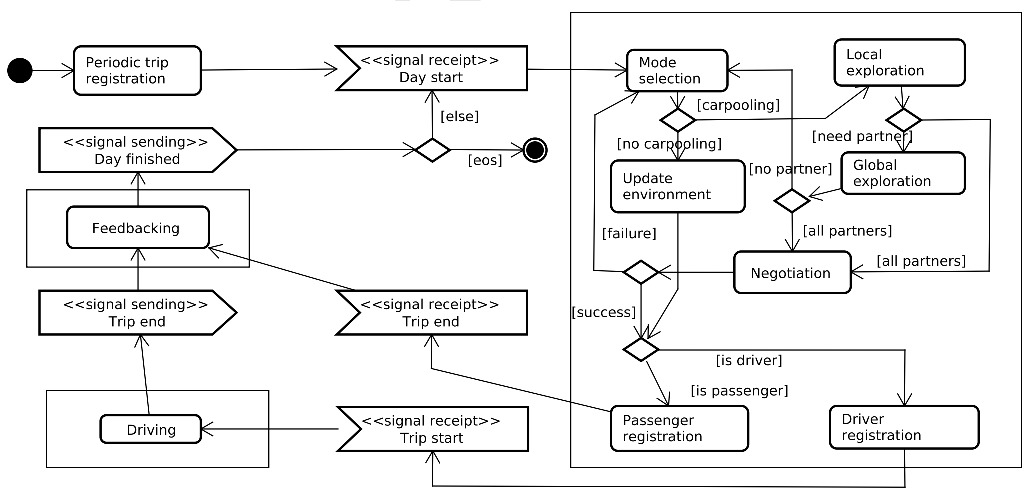

Figure 3: Diagram of the activities of a carpooling agent

Six major activities are considered:

1. "Mode selection:" the agent is selecting its preferred transport mode;
2. "Matching:" the agent is selecting its partners;
3. "Negotiation:" the agent is negotiating with its partners for the details of the carpooling;
4. "Driving:" each driving agent is simulated on a road network;
5. "Feedbacks:" each agent computes the feedback at the end of the day according to the activities of the day;
6. "Environment Updating:" each non-carpooling agent registers its mobility behavior in the environment.

The following sections detail the global behavior of the agent and these major activities.

3.1. Mode Selection Activity

The agent is selecting its preferred transport mode. Here, the FEATHERS data are used to determine the mode choice stochastically. If the agent decides to carpool, he goes in step 2, otherwise he runs its activity in step 6.

3.2. Matching Activity

The matching is applied in both *local* and *global* exploration phases. In both cases, matching precedes the negotiation phase where final decisions to carpool are taken. A person looks for other individuals to cooperate while executing its periodic trip (*periodicTripEx*): this is called *exploration*. *Local* exploration within the private social network (*PrivNet*) is applied before the *global* exploration. *PrivNet* is represented by an organization (Figure 4) in which each agent is playing a role, and has relationship with the other members of the organization. If *carpool candidates* can be found within an agent's *PrivNet*, they will be contacted first (as preferred candidates). *Global* exploration is applied only in a second stage when no suitable *pool* was found in *PrivNet*. In the *Global* exploration phase, the *matcher* provides advice about which *pool*s an individual should negotiate with. This corresponds to the use of an online service by which to explore the set of formerly unknown carpooling candidates. Registration in this service implies first posting some

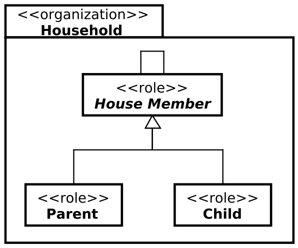

Figure 4: Household

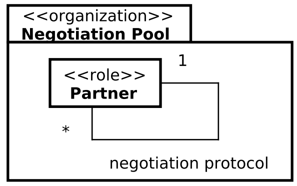

Figure 5: Negotiation Pool

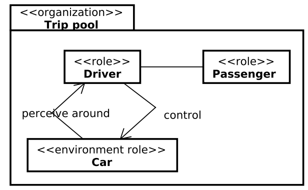

Figure 6: Trip Pool

descriptive characteristics such as age, gender, education level, special interests (e.g. music style preferences), job category, driver license availability, etc. Those qualifiers are used because it is known that continued successful cooperation between people requires a minimal level of similarity.

Two people may carpool together if their $CP = \{L, SR, I, R\}$ are matching. Location (L) is matching the start location of the agents. Spatial Relevance (SR) is the match between the paths from the origin to the destination of all interacting agents. Interests (I) and Requirements (R) are matching the interests and the requirements of each agent in their profiles, respectively. These matchings are based upon the similarity models described in the following sections.

3.2.1. Profile similarity

The candidate carpooler specifies the value for a set of N_A attribute values: those constitute the candidate's *profile*. The model uses the *similarity* between two profiles as a predictor (one independent variable) for the *logit* model. The distance between two attribute tuples a_0 and a_1 having N_{OA} ordinal attributes, is the Euclidean distance divided by a scale factor to normalize the distance (map to interval $[0, 1]$) as described by Equation 1.

$$d(a_0, a_1) = \sqrt{\frac{\sum\limits_{i \in [1, N_{OA}} (a_0[i] - a_1[i])^2}{N_{OA}}} \tag{1}$$

Continuous variables are combined into a single distance value d_C and discrete ordinal values are combined into another one d_D. The range of d_D is a finite subset of $[0, 1]$. The similarity values $s_C = (1 - d_C)$, $s_D = (1 - d_D)$ and $s_E = (1 - d_E)$ are used as independent variables for the *logit* estimator.

3.2.2. Path similarity

The *Global CarPooling Matching Service (GCPMS)* per hypothesis has no information about carpool parkings potentially being used (because that is not specified by the candidates). Therefore, it is assumed that people board and alight at home and work locations only. Periodic trip executions need to be matched, not people. A periodic trip on Wednesday from A to be leaving at about 08:30h needs to be matched with a another one having similar characteristics. Of course, the people involved shall be mutually compatible but they are not the primary subject of matching. A particular individual can periodically carpool with several people for different trips in the week (on Monday with colleague A, on Tuesday with neighbor B who differs from A). The owner of the first *Periodic Trip Execution*, abbreviated by *periodicTripEx*, is the driver.

Let:

- O_i, D_i*:* denote respectively the *origin* and *destination* locations for individual i (e.g. home and work locations);
- $r(a, b, t)$*:* denote the route from a to b when starting at time t that is optimal with respect to some cost function $c(r)$ based on distance and travel time;
- $d(r, t)$*:* denote the duration to travel the route r starting at time t;
- $l(r, t)$*:* denote the length to travel the route r starting at time t;
- $c(r)$*:* denote a cost function based on route length $l(r, t)$ and route travel duration $d(r, t)$;
- $p_{i,solo}(O_i, D_i, t)$*:* denote the optimal path from O_i to D_i when individual i drives alone (solo) and <u>starts</u> at time t;
- $\overline{p}_{i,solo}(O_i, D_i, t)$*:* denote the optimal path from O_i to D_i when individual i drives alone (solo) and <u>ends</u> at time t;
- $p_{i,carpool}(O_i, D_i, t)$*:* denote the optimal path from O_i to D_i when individual i drives the carpool trip via O_j and D_j for $i \neq j$ and <u>starts</u> at time t;

- $\overline{p}_{i,carpool}(O_i, D_i, t)$: denote the optimal path from O_i to D_i when individual i drives the carpool trip via O_j and D_j for $i \neq j$ and ends at time t;
- $pathSim_d()$: denote the path similarity function for the case where the earliest departure is given;
- $pathSim_a()$: denote the path similarity function for the case where the latest arrival is given.

The ratio between the lengths of the optimal routes for the driver is used as a *path similarity function*. For the *given earliest departure* case (starting at t_0) where A is the driver and the trip is $O_A \to O_B \to D_B \to D_A$, this leads to Equation 4.

$$t_1 = t_0 + d(r(O_A, O_B, t_0)) \tag{2}$$

$$t_2 = t_1 + d(r(D_B, D_A, t_1)) \tag{3}$$

$$pathSim_d(pte_A, pte_B, c()) = \frac{c(O_A, D_A, t_0)}{c(O_A, O_B, t_0) + c(O_B, D_B, t_1) + c(D_B, D_A, t_2)} \tag{4}$$

Note that t_1 denotes the time at which the carpool trip leaves O_B and t_2 denotes the time at which the carpool trip leaves D_B. Also note that in general, the Inequalities 5 and 6 hold since the departure times can differ.

$$pathSim_d(pte_A, pte_B, c()) \neq pathSim_a(pte_A, pte_B, c()) \tag{5}$$

$$pathSim_a(pte_A, pte_B, c()) \neq pathSim_a(pte_B, pte_A, c()) \tag{6}$$

The departure time can have a large effect on the trip duration. In the first GCPMS this dependency is ignored due to lack of data. Because of the availability of speed profiles registered using GPS navigators, it will become feasible to take the time dependency into account (which will lead to more accurate negotiation outcome prediction) in the near future although that will require a large amount of data pre-processing and data storage. By ignoring time dependency, Equation 4 is reduced to obtain the Equation 7.

$$pathSim_d(pte_A, pte_B, c()) = \frac{c(O_A, D_A)}{c(O_A, O_B) + c(O_B, D_B) + c(D_B, D_A)} \tag{7}$$

3.2.3. Time Interval Similarity Evaluation for Matching

It is not feasible to ask the individuals to register the piecewise linear preference function mentioned in section 3.3.2. People are assumed to be prepared to register simply a time interval only. Hence the *preference* value is assumed to be a constant f over the time interval specified.

The negotiation outcome is assumed to be positively correlated with the length of the intersection of the intervals associated with the *periodicTripEx*'s to compare. The value is not compared to the constant C mentioned above because this comparison would only imply a linear scaling of an independent variable which has no effect on the *logit* estimator. The time interval similarity tis is given by the equations:TODO: Check if this equation is correct

$$t_0 = \max(t_{i_A,0}, t_{B,0}) \tag{8}$$

$$t_1 = \min(t_{i_A,1}, t_{B,1}) \tag{9}$$

$$tis(i_A, i_B) = t_1 - t_0 \tag{10}$$

For a given pair of *periodicTripEx*'s, tis values are fed into the *logit* estimator as two independent variables; combining them into a single value would cause a loss of information.

3.2.4. Reputation of the agents

Each individual has an *sReputation* value that evolves over time due to qualification by passengers (i.e. individuals who participated in an agreement where the person being evaluated was the driver). Notifications received are registered in a personal *qualifications list* with the individual they apply to; for each issuer, only the most recent qualification is kept. The *sReputation* is calculated as a weighted average of the values posted in the qualification list: the weight decreases with age of the notification and increases with the duration of the cooperation $a.dur(n.ts())$ up to the moment of qualification (the agreement lifetime). Let $\mathcal{Q}_{i_0}$ be the qualifications list for individual i_0. Let $a_j^{i_0}$ denote an agreement in which individuals i_0 and j cooperated. Let $a.iss()$ denote issuer of the qualification. $n.ts()$ denotes the time at which the qualification was issued and $a.dur(t)$ denotes the lifetime of agreement a at time t. Then

$$age = now - n.ts() \tag{11}$$

$$w^{i_0}_{n.iss()} = \exp(-\alpha \cdot age) \cdot a^{i_0}_{n.iss()}.dur(n.ts()) \tag{12}$$

$$sReputation_{i_0} = \frac{\sum\limits_{n \in \mathcal{Q}_{i_0}} n.sRep() \cdot w^{i_0}_{n.iss()}}{\sum\limits_{n \in \mathcal{Q}_{i_0}} w^{i_0}_{n.iss()}} \tag{13}$$

Every agent keeps a list containing a perceived safety reputation value for a limited set of other agents. In the exerciser, a specific agent can be qualified by zero or more safety reputation values (each one of which is owned by a peer). Every agent can determine the reputation of another agent using a method that is not specified in this section and overriding the value already in place (if any). Furthermore, everyone can adjust the reputation of peers based on *gossip* as follows. At a random moment in time, an emitter agent a_e can multicast its reputation value $R^e(q)$ to qualify agent a_q to a subset of agents directly connected to it in the social network. The receiver ar:

1. retransmits the reputation message with a given probability p_r (hence simply drops it with probability $(1 - p_r)$

2. adjusts its own perception of a_q with a given probability p_a (hence simply ignores it with probability $(1 - p_a)$

Consider agents a_e, a_q, ar, a_v that are all pairwise different. a_e emitted a qualification about a_q that reaches a_r via its neighbour a_v. If a_r did not yet have registered an opinion about aq, the value for $R^r(q) = 0$. Reputation update by receiver a_a is done by:

$$\alpha = 2^{-d(e,r)} \tag{14}$$

$$\beta_{r,v} \in [0, 1] \tag{15}$$

$$R'^r_q \leftarrow \frac{R^r(q) + \alpha \cdot \beta \cdot R^e(q)}{1 + \alpha \cdot \beta_{r,v}} \tag{16}$$

where $d(e, r)$ is the distance between emitter and receiver in the network and $\beta_{r,v}$ is the strength of the link between a_r and a_v.

3.3. Negotiation Activity

The negotiation is the process during which the members of a *pool* are negotiating the details of their *periodicTripEx*s (time window, who is the driver and the passengers...)

3.3.1. Overview of the Negotiation Activity

If the negotiation fails, the agent goes back to the step 1 REMARK: check if well written before ENDREMARK ; otherwise to the step 4. The negotiation is based on a specific protocol. According to our organizational approach, an agent, who is negotiating, is a member of the same organization "Negotiation Pool" (Figure 5). The negotiation protocol is described as a sequence of messages exchanged by the different participants, as illustrated by Figure 7. This negotiation protocol is relaunched with different proposed time windows until all the participants were foundTODO: Have it another stopping condition? Explain how the collection of partners is built.

If an individual joins a *pool*, (s)he is added to the *PrivNet* for all other participants in the *pool* (if still required) so that if i_0 and i_1 cooperate in a *pool*, (i_0, i_1) and (i_1, i_0) belong to each others private networks. Because links never are removed from the *PrivNet*, if i_0 and i_1 ever carpooled, $(i_1) \in PrivNet(i_0, 1) \wedge (i_0) \in PrivNet(i_1, 1)$. Candidates register, join and leave *pool*s at random moments in time. As a consequence the main data structures dynamically change due to events external to the matching process.

The negotiation process is a discrete variable with outcome values : *success* (yes) and *failure* (no). A *logit* model is used to predict the negotiation outcome. Negotiation results fed back to the *Global CarPooling Matching Service (GCPMS)* are used to determine the coefficients for the *logit* model by linear regression.

After having found a good match, the matcher conveys its advise to the candidates involved (the owners of the matched *periodicTripEx*); they evaluate the proposal, negotiate about carpooling and possibly agree to cooperate. It is not possible to predict the negotiation phase with certainty; reasons are:

1. Negotiation covers drivers selection, co-route determination and rescheduling (daily planning adaptation) for the cooperators. Schedule adaptation makes use of *VOT* (individual specific *Value Of Time*). An advisory mechanism does not have all required data available nor has any knowledge of the private goals (and in general the *BDI* (Beliefs, Desires, Intentions)) of the individuals (agents) involved in a negotiation.

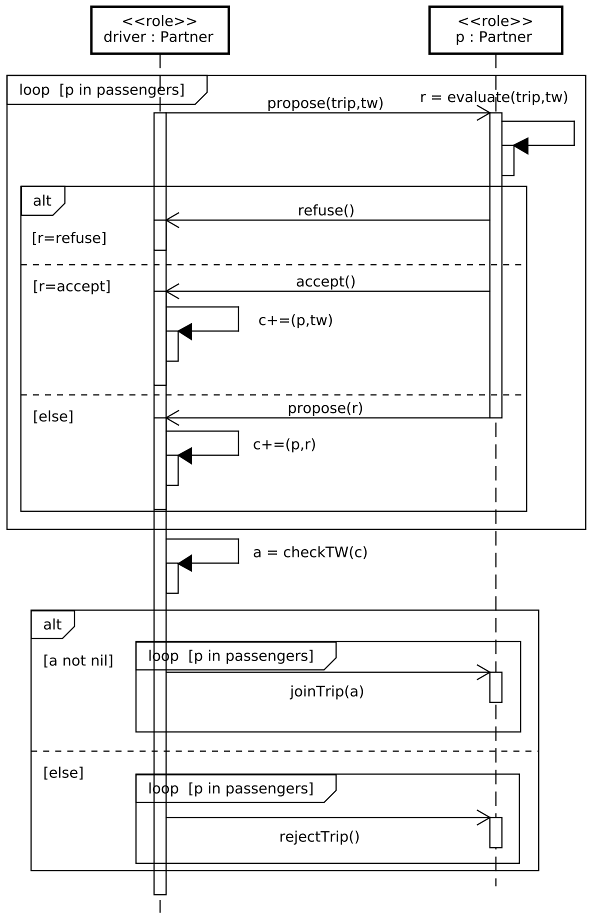

Figure 7: Sequence Diagram of the Negotiation ProtocolTODO: check if the variables are also inside the algorithm and the text

2. The total distance driven cannot be predicted by the *matcher* when carpool parkings are involved because in such cases the co-route can be tree structured. Hence the path similarity function delivers only an approximation of the one involved in negotiation.

3. People are assumed to be prepared posting a minimal amount of data about the time intervals that suit them for departure an arrival respectively; candidates are supposed to specify just the interval boundaries. However, during negotiation, they can make use of *preferences* to state that one of a set of proposed intervals suits better than another one. Hence, the *trip times interval similarity* function available to the matcher is only an approximation for the one used during negotiation.

4. The *Cotravel_Refused* allows individuals to unconditionally avoid any advise to carpool with specific people. For privacy reasons, it is not possible for a refused individual to know the refusing party.

Therefore, the candidates convey the negotiation result back to the matcher service. This paper assumes that sufficient (financial) incentives are in place in order to make this happen. The feedback is used by a learning mechanism incorporated in the matching service. After receiving the feedback, the matching service disposes of the *periodicTripEx* and the individuals characteristics as well as of the negotiation result; those are used to train a predictor.

3.3.2. Time Interval Based Functions for Negotiation

The *departure* (*arrival*) interval for a trip (*periodicTripEx*) is the time interval that suits the traveller to start (end) the trip. Let $pte.i_d()$ and $pte.i_a()$ denote respectively the *departure* and *arrival* intervals of the *periodicTripEx* *pte*.

The individual p_0's *preference* for a given moment in time is given by the function $f_{p_0} : \mathbb{R} \Rightarrow \mathbb{R} : t \mapsto f_{p_0}(t) \in [0, 1]$. The function is not required to be differentiable or continuous but it shall be integrable. For each moment in time belonging to the departure and arrival intervals, the *preference* value needs to be specified.

The *combined preference* function is the product of the preference functions associated with two *periodicTripEx*'s. It is essential to the negotiation process.

The integral of the combined preference over a fixed time interval in $[0, \infty)$ is called the <u>*time interval suitability*</u>. The length of the interval has a pre-specified constant C value; a suitable choice is the expected duration of the trip interruption to get someone on/off board of the vehicle. The *time interval suitability* is denoted by $S(C, i_A, f_A, i_B, f_B)$ where $i_A = [t_{i_A,0}, t_{i_A,1}]$ and $i_B = [t_{i_B,0}, t_{i_B,1}]$ are intervals specified by individuals A and B; f_A and f_B are the associated preference functions. The suitability function is given by the following equations:

$$t_0 = \max(t_{i_A,0}, t_{i_B,0}) \tag{17}$$

$$t_1 = \min(t_{i_A,1}, t_{i_B,1}) \tag{18}$$

$$S(t; C, i_A, i_B, f_A, f_B) = \begin{cases} \int_t^{t+C} f_A(x) \cdot f_B(x) dx & \text{if } t \in [t_0, t_1 - C] \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

where t denotes the start of the boarding/alighting operation. The dimension of the *combined time interval suitability* value is $[prefUnit^2, timeUnit]$. In this context, preference is assumed to be dimensionless hence the suitability dimension reduces to $[timeUnit]$. During negotiation, $S(t; C, i_A, i_B, f_A, f_B)$ is with other functions to find a suitable time to board/alight.

Piecewise linear functions are used because they are flexible, they can easily be specified by the user (in charge for the configuration of the agent-based model) and integration is computationally cheap. The left hand part shows piecewise linear *preference* functions, their product and the associated time interval suitability (crosshatchedTODO: Add the new figure with reference here area under the product function). The right hand part shows case for the same intervals where the preference function is assumed to equal one everywhere: this is the assumption made by the matching service due to lack of information: the user only specifies the boundaries for the departure and arrival intervals.

3.3.3. Agreement Attributes

In this section we will present the three basic attributes important for constituting an agreement between agents.

- *Cohesion of an Agreement:* *Cohesion* is supposed to be a monotonically decreasing function of the time t elapsed since the creation of the *agreement*. *Cohesion* is a monotonically decreasing function of *pool* size s (large *pool*s are more likely to disintegrate). Note that cohesion does not depend on mutual evaluation of carpoolers; cohesion and reputation shall be independent concepts because all of them are fed into a probability estimator. The *cohesion* value is given by:

$$c = e^{\alpha \cdot t} \cdot e^{\beta \cdot (s-1)} \tag{20}$$

In the pairwise case, when considering a specific edge, exactly two *cohesion values* apply (one for each of the vertices (*periodicTripEx*'s)). Each of the cohesion values possibly applies to an *agreement*. For a given *periodicTripEx* pair (pte_0, pte_1), the c_0 and c_1 can relate to either different *agreements* or to a single one. The meaning of the tuple (c_0, c_1) depends on number of *agreement*s involved. Therefore, *cohesion values* are combined into a single *cohesion based indicator* using the function given in equation 23; the second case in equation 23 corresponds to the case where both *periodicTripEx* are members of an *agreement* (but not necessarily to the same one). Let $pte_0, pte_1 \in \mathcal{T}$ the *periodicTripEx*s involved. Let c_0 and c_1 denote the respective corresponding *cohesion values* and $p.T()$ denote the list of *periodicTripEx* involved in pool p. Let $pte.a()$ denote the *agreement* covering *pte* when it belongs to a *pool*. The *cohesion* indicator $\overline{c}(pte_0, pte_1)$ is a measure for the cohesion between two *periodicTripEx*'s when they already form a pair and for the feasibility to get them released when they are bound in pairs with others.

$$pte_x.a() = \begin{cases} nil & \text{if } \nexists p \in \mathcal{P} | pte_x \in p.T() \\ p.a() & \text{if } \exists p \in \mathcal{P} | pte_x \in p.T() \end{cases} \tag{21}$$

$$c_x = \begin{cases} 0 & \text{if } pte_x.a() = nil \\ pte_x.a().c() & \text{else} \end{cases} \tag{22}$$

$$\overline{c} = \begin{cases} (1 - c_0) * (1 - c_1) & \text{if } pte_0.a() \neq pte_1.a() \\ c_0 * c_1 & \text{if } pte_0.a() = pte_1.a() \neq nil \end{cases} \tag{23}$$

In case both *periodicTripEx* belong to the same *agreement*, the cohesion values are taken from that *agreement* and in fact $c_0 = c_1$. In the other case (which also covers the case where at least one of the *periodicTripEx* is not covered by an *agreement*), the complement of the *cohesion* values is used. When neither of the *periodicTripEx* belongs to an *agreement*, $\overline{c} = 1$.

- *Safety Reputation:* While evaluating the success probability for a *pool*, exactly one *sReputation* value applies since only the *sReputation* for the driver is relevant.
- *Timeliness Reputation:* The *tReputation* of an individual i_0 applies to an existing *agreement* and only exists as long as the agreement holds. It can only be affected by the partners in the agreement different from i_0 (in the pairwise case, there is only one such partner). The *tReputation* is an evaluation score assigned by the partners.

3.4. Driving Activity

The driving activity corresponds to the execution of the trip. The driver controls its car (with the carpooled passengers inside) on the roads. The road network is represented by a graph built from geographic data. The Janus platform provides an environment model able to support the displacements of the cars on the roads [11]. Figure 6 presents the organization that is supporting the trip simulation. All the agents in a trip pool must play a role in an instance of this organization. The behavior of the Driver role is composed of two layers: (i) the path planning on the roads; and (ii) the path following and collision avoidance.

The behavior of the agents, who are driving during the simulation, is described by Algorithm 1. Every agent maintains a sequence of connected road segments to follow on the road network, says the *path*:

$$\begin{aligned} path = \Big\langle s_i \Big| \; & i \in [0; n) \wedge \\ & s_i \cap s_{i-1} \neq \emptyset \implies i > 0 \wedge \\ & s_i \cap s_{i+1} \neq \emptyset \implies i < n \Big\rangle \end{aligned} \tag{24}$$

Initially, the *path* is the shortest path between the position of the agent and its goal (given by the function ASTAR in Algorithm 2). At every simulation step, the roads already traveled are removed from the *path* (line 3 of Algorithm 1).

The path planning is dynamic: the driver adapts its path according to its perceptions from the environment jams, roadworks...) with a variant of the A* search algorithm [6, 7], with its principles closed to the D*-Lite algorithm [17]. This family of path-planning algorithms has two advantages: it enables path re-computation during the simulation according to a new state of the environment; and it is suitable for a partial knowledge of the environment's state. Lines 4 to 13 provide the behavior for the dynamic replanning

Algorithm 1 Behavior of the driving agents

```
function DriverBehavior
    repeat
        path ← {⟨s⟩.e | ∀s′ ∈ b, path = b.⟨s⟩.e ∧ position ∈ s ∧ position ∉ s′}
        if ¬isFreePath(path) then
            g ← firstJunctionIn(path)
            if g then
                p ← astar(position, g)
                path ← {⟨s⟩.e | ∀s′ ∈ b, path = b.⟨s⟩.e ∧ g ∈ s ∧ g ∉ s′}
                path ← p.path
            else
                path ← astar(position, goal)
            end if
        end if
        p ← getPerceivedObjects(path, roads)
        updateAttributesAccordingTo(p)
        o ← {a | ∀b ∈ p, distance(position, b) ≥ distance(position, a)}
        if o then
            a ← σ_a(followerDriving(o, acceleration, speed, position))
        else
            a ← σ_a(freeDriving(acceleration, speed, position))
        end if
        speed ← σ_s(speed + (a.Δt)/2)
        position ← position + speed.Δt
        if speed > 0 then
            waitingTime ← 0
        else
            waitingTime ← waitingTime + Δt
        end if
    until position = goal
end function
function isFreePath(path)
    return path ≠ ⟨⟩ ∧ ¬path[1].isBlocked ∧ waitingTime < timeout
end function
function firstJunctionIn(path)
    r ← nil
    for all ⟨a, b⟩ ∈ path ∧ ¬r do
        c ← {i | i ∈ a ∧ i ∈ b}
        if ‖neightbor(c)‖ ≥ 3 then
            r ← c
        end if
    end for
    return r
end function
```

of the *path*. The function isFreePath determines if the next segments of the *path* are traversable. If they are not, the segments to the next junction are replaced by the shortest path from the current position to this junctionTODO: More details + equation. If there is no junction on the *path*, the *path* becomes the the shortest path to goal. The A* search algorithm is used to compute all the shortest paths. It is described in Algorithm 2. A* uses a best-first search and finds a least-cost path from a given initial road to one goal segment. As A* traverses the graph of the roads, it follows a path of the lowest expected total cost or distance, keeping a sorted priority queue of alternate path segments along the way. It uses two base functions: (i) the past path-cost function $g(x)$, which is the known distance from the starting segment to the current segment, and (ii) a future path-cost function $h(x)$, which is an admissible estimation of the distance from x to the goal. The function $h(x)$ must be an admissible heuristic; that is, it must not overestimate the distance to the goal. Thus, for an application like routing, $h(x)$ might represent the straight-line distance to the goal, since that is physically the smallest possible distance between any two points or nodes.

After the driving agent has updated its path, he follows it, and he avoids collisions with the other cars around. The function getPerceivedObjects at line 14 is provided by the JaSim library, and it replies the set of the objects in the field-of-perception of the driver. A standard Intelligent Driver Model [23] is used to adapt the velocity of the car according to the ahead vehicles and to the road signs: the traffic lights and stop signs are assimilated to immobile vehicles until the driver decided to pass through. The other road signs are used to update the variables of the driving model (desired velocity...) On a free road, the vehicle will asymptotically approach its desired velocity. The corresponding acceleration is defined by Equation 25, where a is the current acceleration, v is the current velocity, and v_c is the desired velocity.

$$\text{freeDriving} = a\left(1 - \left(\frac{v}{v_c}\right)^4\right) \tag{25}$$

If there is an ahead object, the acceleration is defined by Equation 26, where Δv is the velocity difference between the agent and the ahead object, Δp is the distance to the ahead object, b is the comfortable braking deceleration, s is the security distance, and w is the desired time headway to the vehicle in front.

$$\textsc{followerDriving} = \begin{cases} -\dfrac{(v\Delta v)^2}{4b\Delta p^2} & \text{if the ahead object is far} \\ -a\dfrac{(s+vw)^2}{\Delta p^2} & \text{if the ahead object is near} \end{cases} \tag{26}$$

TODO: In algorithms: define σ
Msut we add variable types? Yes, only if it help to understand

Algorithm 2 the A* algorithm

```
function ASTAR(start, goal)
    closedset ← ∅
    openset ← {start}
    came_from[start] ← nil
    g[start] ← 0
    f[start] ← g[start] + H(start, goal)
    while openset do
        c ← {s|∀(m, s) ∈ openset², f[s] ≤ f[m]}
        if c = goal then
            p ← ⟨⟩
            while goal do
                p ← ⟨SEGMENT(came_from[goal], goal)⟩.p
                goal ← came_from[goal]
            end while
            return p
        end if
        openset ← openset − {c}
        closeset ← closeset ∪ {c}
        for all n ∈ NEIGHBOR(c) do
            ng ← g[c] + distance(c, n)
            if n ∉ openset ∨ ng < g[n] then
                came_from[n] ← c
                g[n] ← ng
                f[n] ← g[n] + h(n, goal)
                openset ← openset ∪ {n}
            end if
        end for
    end while
    return ⟨⟩
end function
```

After the *pool*'s vehicle has reached its goal, all the agents of the *pool* execute the activity at step 5.

3.5. Feedback Activity

Feedbacks are computed at the end of the day according to the activities of the day, and given to the carpoolers. This activity is the last of the day for a carpooling agent.

3.6. Environment Updating Activity

The environment updating is the optional REMARK: more details, why optional? ENDREMARK activity that permits to a non-carpooling agent to register its mobility behavior in the environment in order to influence the driving simulation of the carpooler agents. Individual vehicles may be generated in the microscopic simulation model used in step 4. This activity is the last of the day for a non-carpooling agent.

4. Experimental Results and Implementation Discussion

Experimentations were done on a population of 1,000 people. The trips are extracted from the data of the FEATHERS project. The considered region is Flanders, Belgium. Figure 8 illustrates the simulator windows: the lower-left window shows the roads, and the upper-left window displays the parameters of the simulation.

The first experimentation is the proportion of drivers and passengers for every quarter of hours during the reference day. Figure 9 gives the histograms for each category. $^2/_3$ of the population are driving; $^1/_3$ of the population is composed of passengers. The results provided by the simulator are close to the original data from FEATHERS. Indeed, during the negotiation process, the pool partners do not change the time window of their trips.

Figure 10 gives the histograms for the distribution among the transport modes for every quarter of hours during the reference day. 65% of the people are traveling in single vehicles; 31% are using the public transports; and 4% are carpooling. The average number of people per carpooling pool is of 2 with standard deviation of 1. The traffic density is still high during the high-activity periods: during the negotiation, the partners are not accepting enough changes to the time window of their trips. Therefore, they prefer to select the single-vehicle or the public transport modes than carpooling.

Figure 11 gives the average computational times for the simulation of a day on an Intel Core i7 CPU 960 at 3.20GHz, four cores, with Windows Seven (64 bits). Because the JaSim environment model is based on the Influence-Reaction model, which permits to handle and solve the conflicts

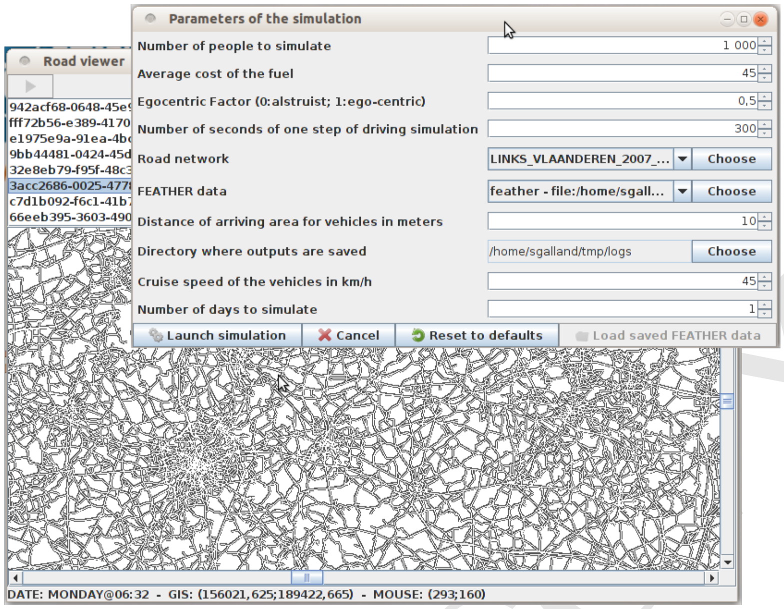

Figure 8: Screenshot of the simulator

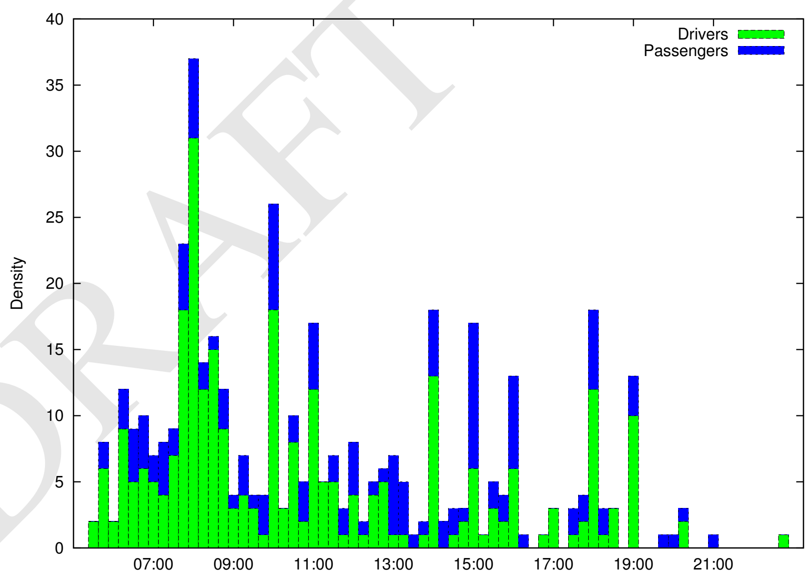

Figure 9: Traffic density for each quarter of hours during the reference day, and per people category

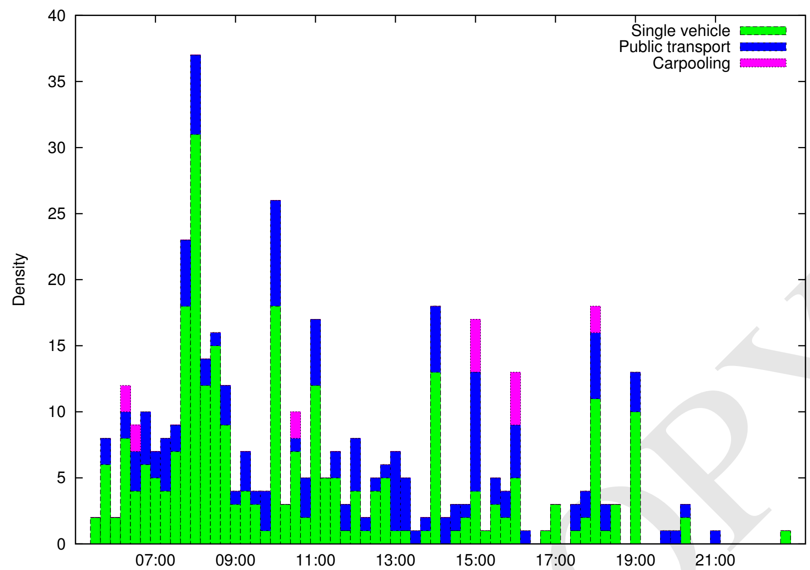

Figure 10: Traffic density for each quarter of hours during the reference day, and per transport mode

among simultaneous actions, the scheduling of the agents is a standard loop (each loop represents two seconds of the day). This approach does not cause a causality problem among the agent actions during the simulation. For the first curve, the people who do not want to carpool are also simulated during the trip execution. For the second curve, only the carpoolers are simulated.

Our simulator may be improved on several points: (i) the quality of the simulation's results; and (ii) the global performance of the simulator. The algorithms deployed in our model are the simplest ones. For example, we may replace the current negotiation protocol by a more complex one in which a complete and sophisticated negotiation between the partners may occur. The global exploration is skipped in this study REMARK: remove or explain ENDREMARK . The microscopic simulation of the vehicles may be replaced by a mesoscopic or macroscopic simulation model to improve the performances of this module. In the case of a microscopic simulation, the agent scheduler used on the JANUS platform may be replaced by an asynchronous scheduler (based on operating-system threads). This solution enables to run groups of agents in parallel in place of the current sequential approach. The current implementation is a proof-of-concept. All the algorithms must be tuned and

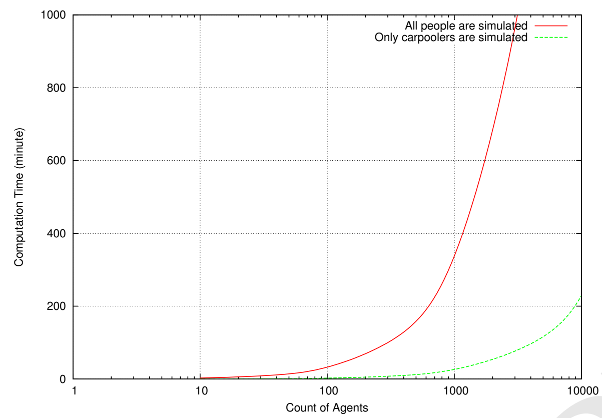

Figure 11: Average Computation Time for a Day

redesigned to improve their performances.

Despite their drawbacks, our model and its implementation reproduces the behavior of a population in a carpooling problem. The microscopic simulation permits to reproduce the trip execution with high level of details (traffic jams...)

5. Conclusion

In comparison to the state-of-the-art research work, the work/study presented in this paper has a number of advantages. Some of these advantages include (i) the ability to analyze various effects of agent interaction with a detailed view on both communication and negotiation aspects, (ii) the ability to simulate learning, adaptation and behavior reproduction of agents through modeling their interactions. Our simulation model on the Janus platform provides a solution to a complex simulation but needs a lot of computing resources (*e.g.* processing time, memory, and data storage) because of the high number of agents to simulate, and the big data processing for each agent. The simulation model proposed in the paper has a few drawbacks: (i) detailed and accurate input data requirement, including agents socio-economic attributes, network information and so on, and (ii) the model must be used in a real-size problem with millions of individuals for checking the scalability of the simulation model.

Acknowledgment

This material is partly based upon the library JaSim, supported by Voxelia SAS. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the Voxelia SAS. Part of this work carried out under the European FP7 Open-FET project Data Science for Simulating the Era of Electric Vehicles (DATASIM) (project number FP7-ICT-270833).

References

[1] Bellemans, T., Bothe, S., Cho, S., Giannotti, F., Janssens, D., Knapen, L., koerner, C., May, M., Nanni, M., Pedreschi, D., Stange, H., Trasarti, R., Yasar, A., Wets, G., 2012. An Agent-Based model to evaluate carpooling at large manufacturing plants, in: Procedia Computer Science, Procedia Computer Science, Elsevier, Niagara Falls.

[2] Bellemans, T., Kochan, B., Janssens, D., Wets, G., Arentze, T., Timmermans, H., 2010. Implementation framework and development trajectory of feathers activity-based simulation platform, Transportation Research Record: Journal of the Transportation Research Board. pp. 111–119.

[3] Bernhardt, Kristen, L., 2007. Agent-based modeling in transportation, Transportation Research E-Circular. pp. 72–80.

[4] Cho, S., Yasar, A.U.H., Knapen, L., Bellemans, T., Janssens, D., Wets, G., 2012. A Conceptual Design of an Agent-based Interaction Model for the Carpooling Application, in: 1st International Workshop on Agent-based Mobility, Traffic and Transportation Models, Methodologies and Applications, Niagara Falls, Canada.

[5] Cossentino, M., Gaud, N., Hilaire, V., Galland, S., Koukam, A., 2010. ASPECS: an agent-oriented software process for engineering complex systems - how to design agent societies under a holonic perspective. Autonomous Agents and Multi-Agent Systems 2, 260–304.

[6] Dechter, R., Pearl, J., 1985. Generalized best-first search strategies and the optimality of a*. J. ACM 32, 505–536.

[7] Delling, D., Sanders, P., Schultes, D., Wagner, D., 2009. Engineering route planning algorithms, in: Lerner, J., Wagner, D., Zweig, K. (Eds.), Algorithmics of Large and Complex Networks. Springer Berlin Heidelberg. volume 5515 of *Lecture Notes in Computer Science*, pp. 117–139.

[8] DeLoach, S.B., Tiemann, T.K., 2012. Not driving alone? American commuting in the twenty-first century. Transportation 39, 521–537.

[9] Ferber, J., 1999. Multiagent Systems: An Introduction to Distributed Artificial Intelligence. Addison-Wesley Professional.

[10] Ferber, J., Michel, F., Baez, J., 2006. AGRE: Integrating environments with organizations, in: Weyns, D., Parunak, H.V.D., Michel, F. (Eds.), Third International Workshop (E4MAS 2006). Springer, Hakodate, Japan. volume 4389 of *Lecture Notes in Artificial Intelligence*, pp. 48–56.

[11] Galland, S., Gaud, N., Demange, J., Koukam, A., 2009. Environment model for multiagent-based simulation of 3D urban systems, in: the 7th European Workshop on Multiagent Systems (EUMAS09), Ayia Napa, Cyprus. Paper 36.

[12] Galland, S., Gaud, N., Rodriguez, S., Hilaire, V., 2010. Janus: Another yet general-purpose multiagent platform, in: the 7th Agent-Oriented Software Engineering Technical Forum (TFGAOSE-10), Agent Technical Fora. Agent Technical Fora, Paris, France.

[13] Galland, S., Gaud, N., Yasar, A.u.h., Knapen, L., Janssens, D., Lamotte, O., 2013. Simulation model of carpooling with the janus multiagent platform, in: Yasar, A.u.h., Knapen, L. (Eds.), 2nd International Workshop on Agent-based Mobility, Traffic and Transportation Models, Methodologies and Applications (ABMTRANS13), Elsevier, Halifax, Nova Scotia, Canada.

[14] Gaud, N., Galland, S., Hilaire, V., Koukam, A., 2008. An organizational platform for holonic and multiagent systems, in: the Sixth International Workshop on Programming Multi-Agent Systems (ProMAS08), of the Seventh International Conference on Autonomous agents and Multiagent Systems (AAMAS), Estoril, Portugal. pp. 111–126.

[15] Horvitz, E., Apacible, J., Sarin, R., Liao, L., 2005. Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service., in: UAI, AUAI Press. pp. 275–283.

[16] Kamar, E., Horvitz, E., 2009. Collaboration and shared plans in the open world: studies of ridesharing, in: Proceedings of the 21st international jont conference on Artifical intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. pp. 187–194.

[17] Koenig, S., Likhachev, M., 2005. Fast replanning for navigation in unknown terrain. Robotics, IEEE Transactions on 21, 354–363.

[18] Macal, C.M., North, M.J., 2005. Tutorial on agent-based modeling and simulation, in: Proceedings of the 37th conference on Winter simulation, Winter Simulation Conference. pp. 2–15.

[19] Massaro, D.W., Chaney, B., Bigler, S., Lancaster, J., Iyer, S., Gawade, M., Eccleston, M., Gurrola, E., Lopez, A., 2009. Carpoolnow - just-in-time carpooling without elaborate preplanning., in: Filipe, J., Cordeiro, J. (Eds.), WEBIST, INSTICC Press. pp. 219–224.

[20] Niazi, M., Hussain, A., 2011. Agent-based computing from multi-agent systems to agent-based models: a visual survey. Scientometrics 89, 479–499.

[21] Odell, J., Parunak, H., Fleisher, M., Brueckner, S., 2002. Modeling Agents and their Environment, in: Giunchiglia, F., Odell, J., Weiss, G. (Eds.), Agent-Oriented Software Engineering III, Springer-Verlag, N.Y. (USA).

[22] Smith, L., Beckman, R., Anson, D., Nagel, K., Williams, M., 1995. TRANSIMS: Transportation analysis and simulation system, in: Conference: 5. National transportation planning methods applications conference, Seattle, WA (United States), 17-21 Apr 1995.

[23] Treiber, M., Hennecke, A., Helbing, D., 2000. Congested traffic states in empirical observations and microscopic simulations. Phys. Rev. E 62, 1805–1824.

[24] Waraich, R., Charypar, D., Balmer, M., Axhausen, K., Waraich, R., Waraich, R., Axhausen, K., Axhausen, K., 2009. Performance improvements for large scale traffic simulation in MATSim. [Arbeitsberichte Verkehrs- und Raumplanung], ETH, Eidgenössische Technische Hochschule Zürich, IVT, Institut für Verkehrsplanung und Transportsysteme.