MADARP: An Agent-Based Passenger Transport Framework

CLAUDIO CUBILLOS AND CLAUDIO DEMARTINI
Dipartimento di Automatica e Informatica
Politecnico di Torino
C.so Duca degli Abruzzi 24, Turin, 10129
ITALY

Abstract: - In this work we present our MADARP framework for implementing passenger transport systems using the agent paradigm. MADARP is constructed over an external agent platform that provides an environment with the basic communication and management services for the agents. The framework provides a set of base agents together with an architecture and an ontology for building a customized passenger transport system. In addition, three transport models implemented using the framework and their results are presented.

Key-Words: - Multi-Agent System, Intelligent Transport, Distributed Planning.

1 Introduction
The field of passenger transport systems has received an increasing attention during last years due to different factors: contamination, high traffic and congestion problems. In addition, citizens require more flexible transportation alternatives in the cities according to their needs. As response, new alternatives to satisfy de transport demands of the citizens are being conceived [1].

From a technological perspective, the recent advances in network systems together with the low cost of the processing power have move us to the era of distributed systems and ubiquity.

Following these tendencies, the concept of Intelligent Transport Systems (ITS) can be seen in many research work in the transport field. In this sense, the work presented here corresponds to piece in the puzzle of a more bigger ITS.

The MADARP framework explained in this article allows the implementation of final passenger transport systems. The framework adopts the agent paradigm as a way to tackle the inherent distributed problem in a flexible and reliable way.

Furthermore, the agent use allows the concrete system to provide different ways of interoperation with the different actors (pervasiveness) and hence more flexibility to customers.

Together with the agents, the framework uses an ontology specification. This provides the system openness to integrate new actors and typologies of transport services.

The rest of the paper is structured as follows: Section 2 presents related work in the field; §3 describes the passenger transport domain considered in the work; §4 outlines the MADARP framework for then in §5 give details on the multi-agent architecture; §6 shows three concrete transport system examples; §7 shows the tests and their results; Finally §8 states the conclusions.

2 Related Work
In literature, the passenger transportation problem can be found under different names depending on the type of transport service tacked. Traditionally, this problem was seen as a sub-type of Travel Salesman Problem (TSP) and more specifically, the Pickup and Delivery Problem (PDP) devoted to goods transport. Under the name of dial-a-ride problem (DARP) were developed centralized passenger transport systems, usually based on greedy insertion heuristics (see [4], [8], [16] and [20]).

Newer techniques try to improve the quality of the obtained solutions. In [11], Li presented a meta-heuristic for the PDPTW. A tabu-search solution can be found in [14], implemented for real-life problems including time-window constraints. Soft computing has also been applied to the transport domain: [14] and [17] used genetic algorithms for optimization and in [13] an ant-colony based system is described.


Finally, in [5], [10] and [15] are presented agent-based systems for the planning and scheduling of client’s rides.
3 The Passenger Transport Domain

The DRT system we are treating consists of transport requests coming from a set of clients which should be satisfied by a heterogeneous fleet (e.g. composed by busses, minivans, vehicles for disabled people, etc.). Vehicles are characterized by different properties, but in general, they have a limited capacity, periods of time during the day in which they are available, and an area of geographic coverage. As well, vehicles have additional characteristics (e.g. types of seats, WC, air conditioning, etc) or complementary services (e.g. Bar, bicycle transport, etc). These usually affect the client’s comfort and hence their perception of the transport system.

On the other hand, transport requests commonly specify a pick-up and delivery place. They also give a time interval within which the client has to be picked-up at the origin place and another time interval for his delivery at the destination. Moreover, the requests can include further descriptions of the desired service, e.g. wheel-chair places, number of seats, shared or exclusive use of the vehicle, etc.

Finally, service requests have to be assigned to vehicles and scheduled according to the above restrictions, considering not exceeding each vehicle’s capacity. The model considers the possibility of a multi-depot scenario, that is, the vehicles can start from and arrive to different depots.

4 The MADARP Framework

We conceive the MADARP framework as part of a layered model that allows the implementation of different kinds of passenger transport systems.

As Fig. 1 shows, the model has three layers: first, an agent platform (Jade specifically), on top of it the MADARP framework and finally, the concrete passenger transport system.

In the lower layer, the Jade agent platform [2] provides a full environment for agents to work: an agent management system (AMS), the possibility of agent containers in different hosts (distribution), a directory facilitator (DF) providing yellow-page services, a message transport system (MTS) for supporting communication between agents and mobility services between containers.

On top of that is built the MADARP framework, providing the base agents and architecture for implementing a passenger transport system. MADARP provides a set of agents for the planning of trip requests and another group of agents for the interaction with the different involved actors (clients and vehicles).

Over the above two, by extending and implementing the base agents provided by MADARP a concrete transport system is obtained.

In the following sections the framework’s architecture is explained in detail and concrete implementations are given.

5 The MA Architecture

In this section is detailed the underlying multi-agent architecture of the MADARP framework. The agents are grouped in two layers (see Fig. 2): the Internet layer, in charge of the communication with the external world (vehicles, clients, other systems), and the Planning layer, that encapsulates the assignment and scheduling services.

Service profiles are used for the description of both, the clients’ desired transport services and the vehicles’ offered services. A profile is modeled as a list of properties that can be of two types: constraint and utility properties. Constraint-properties are characteristics that must be ensured in order to be accepted (e.g. must have WC or air conditioning). Utility-properties are not mandatory characteristics that have a positive contribution in the utility function of the client or vehicle. This properties-list that constitutes a profile specification is made using a common ontology for the whole system: the Service Ontology. It uniformizes the transport domain
concepts and interactions used by the different actors. Flexibility is given by the possibility of adding dynamically new typologies of services and requests within the service ontology boundaries, as their concepts are used to describe the offered and desired transport services. The resulting multi-agent architecture allows the planning process execution in a heterogeneous network of computer systems [19].

The main agents within each layer together with the Service ontology shown in Fig. 2 are described as follows.

5.1 The internet layer
The Internet layer of the architecture interfaces the system with the external world (vehicles, clients and other systems) and registers the vehicles within the system. Fig. 3 shows the base agents corresponding to this layer: 1) Vehicle agents, that wrap real world transportation vehicles; 2) the Broker agent, that registers the available vehicles and its profiles; and 3) Client agents, that interface users with the system.

Client agents are interface agents in charge of capturing the final user’s trip requirements (pick-up place and time, delivery place and time, seat type, services offered, etc). They should ask the corresponding human user for the desired properties together with the level of importance given to each one. In addition, the Client agent can give its Trip-request agent different negotiation capabilities and degree of autonomy for making trip decisions. The Client agent is also responsible for communicating the user the final result.

In a similar way, Vehicle agents are interface agents in charge of capturing the real vehicle properties. For this, the Vehicle agent holds a service profile that defines the kind of transport service offered. Each Vehicle agent generates its own Schedule agent when they advertise their availability to the Broker.

The Broker receives advertisement messages from vehicle agents, containing their service descriptions (service profile) and registers them in its internal database.

5.2 The planning layer
In this layer, the requests are assigned to vehicles and scheduled in their route plans. The layer is composed by four types of base agents (see Fig. 4): 1) Schedule agents, representing the route plan and the scheduling policy of single vehicles; 2) Trip-request agents, modeling single client request specifications; 3) the Planner agent, implementing the assignment policy; and 4) the Map agent, in charge of providing the geographical data (streets, distances, etc.).

The Schedule agent holds a service profile specifying the kind of transport service offered, its characteristics and the corresponding utility function of the vehicle. The framework allows each vehicle (and its Schedule agent) to specify a different utility function by simply defining a different set of utility-properties within the service profile description.

Schedule agents also contain a scheduling policy, that is, the scheduling algorithm for processing trip requests. The framework allows customizing for each vehicle a different algorithm if desired.

The Trip-request agent models the client’s desires concerning the trip. This is stored into a service profile. This agent can implement different types of negotiation capabilities. Again in this case, the framework allows to implement negotiation mechanisms provided that the Planner also supports them.

The Planner can implement different filtering policies that will be applied to the trip-proposals received from the Schedule agents (e.g. minimize the number of used vehicles). The existence or absence of filtering policies allows the MADARP framework to provide different optimization approaches (i.e. market-based, centralized or a mixture of both).

The Map agent models the underlying geographical region under coverage. This is implemented as a distance matrix containing a graph representation. In addition, the Map agent manages another matrix with the minimum distances between each pair of nodes.
5.2.1 The generic planning sequence
The dynamics of the generic planning sequence is as follows (see Fig. 4): Users ask for transportation through their Client agents, generating Trip-request agents. In step 1 (Schedule-me), each Trip-request agent asks the Planner to process the request. The Planner processes the request (step 2) first by obtaining from the Broker the vehicles that match the required profile, and then (step 3) by making a call for trip-proposals to all the corresponding Schedule agents. They send back their proposals and then the Planner (step 4) selects the most suitable alternatives among the received trip proposals by applying filtering policies and starts a negotiation process with the client (through its Trip-request agent). After arriving to agreement the Planner (step 5) tells the Schedule agent that won the proposal to add the trip to its actual schedule and tells the others their proposal rejection.

5.3 The service ontology
The service ontology provides a support of common knowledge in two areas of the system: the properties definition of service and proposal profiles (Content Ontology), and the messages exchange between agents (Interaction Ontology). This ontology was specified using RDF [18].

In the first case, the ontology supplies a hierarchy of concepts (properties) to use when defining the constraint and utility properties of the proposals.

Examples of ontology concepts used for specifying the constraint part of a Service Profile are shown in Fig. 5.a. The first two concepts are used with a range for indicating the possible values (e.g PickupTime: [10:30,10:35]), while the rest of the concepts are used together with only a single value (e.g. RequiredPlaces: 3).

Examples of ontology concepts for defining the utility properties in a profile are shown in Fig. 5.b. In this case, the first four concepts are usually used for describing the vehicle’s utility function (inside its service profile) under a centralized model. The ClientExcessTravelTime and ClientWaitTimeDelivery concepts are used for the client’s utility function.

The second ontology use is for supporting the messages exchange, specifically for detailing the interaction protocols and the message’s content. The interaction protocols define the sequence of messages exchanged during the communication of two or more agents. The interaction protocols used in the architecture are similar to the ones defined by the FIPA standard [7]. Examples of ontology concepts defining interaction protocols are: MADARP-schedule-me, MADARP-process-query and MADARP-call-for-trip-proposals.

6 Concrete Transportation Systems
In order to test the MADARP framework, were implemented three models of transport system: a centralized, a market-based (decentralized) and a mediated one. For each model, the framework’s base agents were extended and modified. These are explained in the following sub-sections making reference to the generic planning sequence explained at §5.2.1.

6.1 The centralized model
The centralized model considers the optimization of the global utility for the system. To implement this, first in the step 1) Schedule-me, the trip-request agent has to include the client’s utility function in the request message. In practical terms, it means to include a profile with its utility properties.

Then in step 3) Call for trip-proposals, the Planner agent has to forward Schedule agents the client’s utility function received from the trip-request at step 1). The Schedule agents have to add the client’s utility function in their evaluation. This can be done by simply overwriting the EvaluateInsertion() method of the Schedule and add the utility properties.

In step 4) filtering & negotiation, the planner has no filtering policies. In the negotiation the trip-request takes the list of proposals and selects the one with best global utility (DeltaVehicleTripCost in the ontology). This is done by overwriting its ChooseAlternative() method.

<table>
<thead>
<tr>
<th>PickupTime</th>
<th>DeliveryTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>PickupAddress</td>
<td>DeliveryAddress</td>
</tr>
<tr>
<td>RequiredPlaces</td>
<td>MaximumRideTime</td>
</tr>
<tr>
<td>PickupServiceTime</td>
<td>DeliveryServiceTime</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TotalWaitTimeDelivery</th>
<th>TotalExcessTravelTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>TotalBusSlackTime</td>
<td>TotalBusTravelTime</td>
</tr>
<tr>
<td>ClientExcessTravelTime</td>
<td>ClientWaitTimeDelivery</td>
</tr>
<tr>
<td>DeltaExcessTravelTime</td>
<td>DeltaBusTravelTime</td>
</tr>
<tr>
<td>DeltaWaitTimeDelivery</td>
<td>DeltaWaitTimeDelivery</td>
</tr>
<tr>
<td>DeltaVehicleTripCost</td>
<td>ScheduledPassengers</td>
</tr>
</tbody>
</table>

Fig. 5. Ontology concepts
Messages between agents usually have serialized profile objects as content. In some cases the answer messages contain single concepts as content. Examples of ontology concepts used in these cases are: No-Proposals-Available, No-Matching-Vehicle-Found, Proposal-Accepted and Proposal-Rejected.
Finally, the objective function for the centralized model considered: the \textit{DeltaExcessTravelTime}, the \textit{DeltaWaitTimeAtDelivery}, the \textit{DeltaBusSlackTime}, and the \textit{DeltaBusTravelTime}.

### 6.2 The decentralized model

The implemented decentralized model corresponds to a market-based approach based in the CNP. In step 1) the request goes without the client’s utility function. It only has the profile with the description of the desired service. Then, in step 3) each Schedule agent evaluates to make a proposal considering only its own utility function.

In step 4) the Planner does not apply any filter to the proposals and in the negotiation; the trip-request evaluates the received proposals and chooses according to its own (client’s) utility function.

Finally, the utility functions of the agents were the following: the Vehicle agents had the \textit{DeltaBusSlackTime} and the \textit{DeltaBusTravelTime}, the Client agents had the \textit{ClientExcessTravelTime}, and the \textit{ClientWaitTimeAtDelivery}.

### 6.3 The mediated model

This model takes advantage of the mediation role of the Planner by applying filtering policies. In practical terms the steps are implemented as in the previous approach, the decentralized one, with exception of step 4). In this step the Planner agent does apply a filtering policy to the list of received proposals.

The utility properties used by the Planner agent when realizing the filtering were: \textit{DeltaExcessTravelTime}, \textit{DeltaWaitTimeAtDelivery}, and the \textit{DeltaBusTravelTime}.

### 7 Results

In this section are presented the results obtained when running the 3 concrete models explained in the previous section.

The following operational decisions were adopted in all the models: 1) the same utility function and scheduling algorithm have been used for all the vehicles, 2) all the clients share the same utility function, 3) the available fleet is of 30 identical vehicles with capacity 20, 4) one depot is used for all the vehicles and 5) in all cases the effectiveness measures (utility variables) were weighted with the same value.

All the tests considered the same geographical net and 20 demand scenarios with 50 trip requests each, distributed uniformly in a two-hour horizon. For each demand scenario 25 runs were done.

#### 7.1 Centralized model v/s ADARTW

The centralized model was confronted with ADARTW [4], a traditional centralized greedy insertion heuristic. ADARTW was set with the same parameters of the centralized model and both systems minimize also the number of used vehicles.

In the following table a summary of the runs is shown.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Centralized</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>50</td>
<td>27</td>
<td>1.687</td>
<td>23</td>
<td>9</td>
<td>1.655</td>
</tr>
<tr>
<td>Desv.</td>
<td>0</td>
<td>2</td>
<td>115</td>
<td>6</td>
<td>5</td>
<td>117</td>
</tr>
<tr>
<td>Min</td>
<td>47</td>
<td>21</td>
<td>1.412</td>
<td>6</td>
<td>0</td>
<td>1.383</td>
</tr>
<tr>
<td>Max</td>
<td>50</td>
<td>30</td>
<td>2.008</td>
<td>41</td>
<td>22</td>
<td>1.984</td>
</tr>
<tr>
<td><strong>ADARTW</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>50</td>
<td>27</td>
<td>1.688</td>
<td>23</td>
<td>9</td>
<td>1.585</td>
</tr>
<tr>
<td>Desv.</td>
<td>0</td>
<td>2</td>
<td>113</td>
<td>6</td>
<td>6</td>
<td>119</td>
</tr>
<tr>
<td>Min</td>
<td>48</td>
<td>20</td>
<td>1.394</td>
<td>4</td>
<td>0</td>
<td>1.280</td>
</tr>
<tr>
<td>Max</td>
<td>50</td>
<td>30</td>
<td>2.001</td>
<td>41</td>
<td>28</td>
<td>1.912</td>
</tr>
</tbody>
</table>

It is possible to appreciate that our centralized model extended from the MADARP framework performs in a similar way to ADARTW.

The main difference between both systems is in the processing time. ADARTW takes 1 second in processing the run with 50 requests, while our agent-based model takes 20 seconds running on 1 machine. This allows to have an idea of the overhead introduced by two things: 1) the agent platform (Jade), and 2) the MADARP framework.

#### 7.2 The three models

The tests on the three models (see Table 2) have shown that the Mediated model is able to provide results in the gap between the Centralized and Decentralized models.

The Mediated model gives better results for the clients, in terms of both, \textit{Waiting Time} and \textit{Excess Ride Time} when compared with the Centralized model. On the other hand, when compared with the Decentralized model, it gives better results from the vehicle’s (or operator’s) viewpoint, this in terms of \textit{Travel Time} and \textit{Slack Time}.

### 8 Conclusions

The MADARP framework for implementing agent-based passenger transportation systems was described. The framework stands over the Jade platform as agent environment. By extending the base agents is possible to obtain an ad-hoc concrete
system, appropriate for a specific passenger transport domain. The framework has been tested by implementing three transport models and results show that comparable results are obtained. The idea is to continue testing the framework with: different utility functions, other scheduling algorithms and more complex negotiation schemes.

References:


Table 2. Three models’ results

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Decentralized</td>
<td>Mean</td>
<td>47 30</td>
<td>1.650</td>
<td>18 2 1.629</td>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>St. Dev.</td>
<td>2 0</td>
<td>114 5</td>
<td>3 113 29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>42 30</td>
<td>1.344</td>
<td>2 0 1.336</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Max</td>
<td>50 30</td>
<td>2.034</td>
<td>36 13 2.004</td>
<td>174</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mediated</td>
<td>Mean</td>
<td>47 30</td>
<td>1.627</td>
<td>20 4 1.603</td>
<td>62</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>St. Dev.</td>
<td>2 0</td>
<td>115 5</td>
<td>4 115 24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>42 30</td>
<td>1.353</td>
<td>6 0 1.336</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Max</td>
<td>50 30</td>
<td>2.005</td>
<td>35 19 1.971</td>
<td>141</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Centralized</td>
<td>Mean</td>
<td>50 29</td>
<td>1.627</td>
<td>22 9 1.596</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>St. Dev.</td>
<td>1 1</td>
<td>110 5</td>
<td>6 110 19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>46 24</td>
<td>1.327</td>
<td>7 0 1.300</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Max</td>
<td>50 30</td>
<td>1.954</td>
<td>36 27 1.919</td>
<td>145</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>