Multi-agent infrastructure for distributed planning of demand-responsive passenger transportation service

Multi-Agent Infrastructure for Distributed Planning of Demand-Responsive Passenger Transportation Service

Claudio Cubillos
Dip. Automatica e Informatica
Politecnico di Torino
Turin, Italy
claudio.cubillos@polito.it

Franco Guidi-Polanco
Escuela de Ingeniería Industrial
Pontificia U. Católica de Valparaíso
Valparaíso, Chile
franco.guidi@ucv.cl

Claudio Demartini
Dip. Automatica e Informatica
Politecnico di Torino
Turin, Italy
claudio.demartini@polito.it

Abstract - *Demand-responsive transport services are systems that assign users' specific transport requests to different vehicles enabled to fulfill the required service. In order to contain costs, maximize profits or get a higher service level, different planning algorithms have been described in literature. In our work we propose a multi-agent architecture, which adopts a mixed planning model. This model is an integration of centralized and negotiation-based decision-making approaches. The advantage of using this model respect to a centralized approach is that the estimation of the utility function is avoided (the client is involved in the final decision by means of a negotiation process). The model advantages a complete decentralized approach in giving results that are not so far from the optimum for the whole system.*

Keywords: Multi-agent architecture, demand responsive transport system, distributed application, intelligent system.

1 Introduction

Changes in transport requirements in European citizens have brought the opportunity to create new services aimed to fulfill special transportation demand in addition to regular population mobility services. Those systems are known as demand-responsive transport (DRT) services, and their objective is to satisfy personal transportation requests at relatively low costs, thanks to an integrated planification with the use of the different available resources on transport networks.

Earlier planning methodologies developed for DRT systems adopted centralized approaches, where the control and decision-making was done by only one entity that maximized the global utility of the whole system (i.e. the utility for the operator and for his clients). These approaches are usually implemented as heuristic procedures that extend basic graph search algorithms, acting over large data collections that describe the entities of the domain problem (vehicles, service requests, schedules). A key aspect when applying these approaches is the identification of a good estimation of the client's utility function, in order to allow the generation of adequate solutions from the client's point of view. However, the latter is not always feasible because not all the clients share the same desires, nor appreciate them with the same importance. An example of centralized approach can be found in our past research with the Advanced Dial-A-Ride with Time Windows (ADARTW) algorithm [3].

Recent decentralized or market-based approaches, on the other hand, exploit cooperative relationships in communities of agents that perform low-level planning, scheduling, execution, and control tasks. In these cases, negotiation processes among them (e.g. contract-net, auctions, bargain, etc.) tend to maximize the utility of individual participants, leading to Pareto-optimal solutions. As opposite to centralized evaluations, optimization is done with less information and, as consequence, the solution could be far from the best for the whole system.

Thus, existing techniques are not able to find good solutions to DRT problems, especially when it is not only requested the satisfaction of the client's desired service constraints, but also the maximization of his utility perception. This paper addresses a mixed model implementation, which combines the best features of both, the centralized and the decentralized approaches. The model is supported by a multi-agent system architecture.

The rest of the paper is structured as follows: Section 2 presents the general DRT problem; Section 3 describes the proposed multi-agent architecture; Section 4 outlines the distributed optimization model; Section 5 specifies the implemented scheduling algorithm; Section 6 describes our initial tests and their results; Finally in Section 7 some conclusions are given.

2 The DRT system

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 cover-

0-7803-8566-7/04/$20.00 © 2004 IEEE.

age. 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.

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.

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.

3 The MAS architecture

The model is structured as a two-layer architecture (see Figure 1): 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. A common Service ontology uniforms the transport domain concepts used by the different actors.

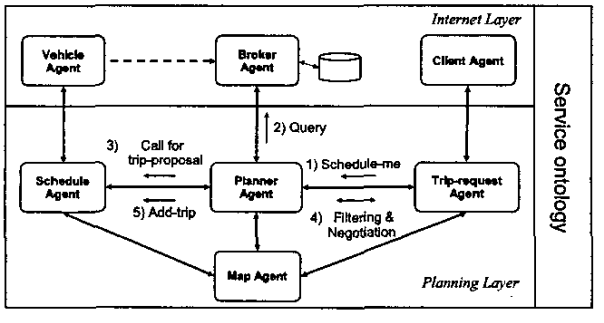

Figure 1. The multi-agent architecture

The resulting multi-agent architecture allows the planning process execution in a heterogeneous network of computer systems [7]. Flexibility is given in this architecture by the possibility of dynamically adding new typologies of services and requests within the service ontology boundaries, as it is used to describe the offered and desired transport services.

The main agents within each layer together with the service ontology shown in Figure 1 are described as follows.

3.1 The internet layer

The Internet layer of the architecture interfaces the system with the external world (vehicles, clients, other systems) and registers the vehicles within the system. This layer consists of: 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.

Each Vehicle agent holds a profile with the properties that define the kind of transport service offered and its utility function. Each vehicle agent generates its own Schedule agent containing the respective service profile, utility function and scheduling algorithm.

The Broker was derived from the Broker design pattern used in the Global Automation Platform [1]. The Broker receives advertisement messages from vehicle agents, containing their service descriptions (service profile). The Broker registers them in its internal database and informs back with a vehicle-registered message to the corresponding vehicle agent. It can answer with a failure message in case of problems with the service profile or if the vehicle is already registered.

The Broker also processes query messages from the planning layer. The received messages contain client's service profiles (descriptions of their desired transport service). The Broker searches within its vehicle's database the ones matching the received profile. It answers with a message containing a list of vehicle's names.

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.) and translating them into a suitable specification using the concepts defined in our Service ontology. As mentioned earlier, the resulting service profile of the client will include both, constraint and utility properties. Hence, Client agents should ask the corresponding human user for the desired properties together with the preferences and level of importance given to each one.

After that, the Client agent will be able to generate its corresponding Trip-request agent containing the resulting service profile description. In addition, a 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. It can also ask him during the process any additional information required by the Trip-request agent for taking a decision, as it depends on the level of autonomy.

Client agents can also be developed externally to the platform but, independently of its origin, they act as factories of the Trip-request agents.

3.2 The planning layer

The Planning layer of the architecture is in charge of processing, assigning and scheduling the received trip requests. It provides four kinds of agents that interact directly in the evaluation and creation of the routing plan: 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 dynamics of the planning process is at follows (see Figure 1): Schedule Agents are created by Vehicle agents when they advertise their availability to the Broker. Users ask for transportation through their Client agents, generating Trip-request agents. (1) Each Trip-request agent asks the Planner to process the request. The Planner processes the request (2) first by obtaining from the Broker the vehicles that match the required profile, and then (3) by making a call for trip-proposals to all the corresponding Schedule agents. They send back their proposals and then the Planner (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 (5) tells the Schedule agent that won the proposal to add the trip to its actual schedule and tells the others their proposal rejection.

The Schedule agent holds a service profile specifying the kind of transport service offered, its characteristics and the corresponding utility function of the vehicle. In this sense, each vehicle (and its Schedule agent) can 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 underlying framework allows Schedule agents to implement different scheduling policies if desired, as each vehicle agent generates its own Schedule agent.

Schedule agents first receive a call for proposal form the Planner for serving a client's trip and make a proposal using the vehicle's utility function and scheduling policy. The proposal is modeled as a profile containing the specification of the offered service, that is, a list of constraint properties describing additional service characteristics (for the Trip-request to use for evaluation) and utility properties with the actual values of the represented variables (for the Planner and Trip-request to use for evaluation)

The Trip-request agent models the client's desires concerning the trip. This is stored into a service profile. This agent also holds certain negotiation capabilities and afterwards during the negotiation process, will store a list of trip proposals from different vehicles and the final chosen alternative. Again in this case, the Trip-request agents can implement different types of negotiation capabilities, provided that the Planner also supports them.

The Planner agent is the one in charge of implementing the assignment through filtering policies and the negotiation process. It holds a list containing the trip requests being processed and a list of filtering policies to apply to the trip solutions.

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 filtering policies allow the underlying framework to provide the possibility of implementing a centralized or mixed optimization approach.

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. The distance values are stored as time (required to reach one node from the other) and as normal distance.

The Map agent processes two types of requests: 1) messages asking for the distance between one (or more) pair of nodes, and 2) messages requiring the complete path of nodes between one (or more) pair of nodes. These requests can be done by Schedule agents, Trip-request agents and the Planner.

3.3 The service ontology

The service ontology provides a support of common knowledge in two areas of the system: the properties definition (for service and proposal profiles) and the messages exchange between agents. This ontology was specified using RDF [7] as language for the description.

In the first case, the ontology supplies a hierarchy of concepts (properties) to use when defining the constraint and utility properties of the proposals. Each constraint property specifies: 1) the name of the involved concept and 2) a value or range of values for that concept.

Examples of ontology concepts used for specifying the constraint part of a service profile are: *PickupTime*, *DeliveryTime*, *PickupAddress*, *DeliveryAddress*, *RequiredPlaces*, *MaximumRideTime*, *PickupServiceTime* and *DeliveryServiceTime*. 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).

Each utility property specifies the concept name to which it refers to, together with its value. When the utility property is used for detailing the coefficients of an utility function, its property value is used to represent the given weight when maximizing. Negative weight values can be used for properties that need to be minimized. Instead when the utility property is used for a proposal profile, the value represents just that, the value of that variable in the proposal.

Examples of ontology concepts used for defining the utility properties in a profile are: *TotalWaitTimeDelivery*, *TotalExcessTravelTime*, *TotalBusSlackTime*, *TotalBusTravelTime*, *ClientExcessTravelTime*, *ClientWaitTimeDe-*

livery, *DeltaBusSlackTime*, *DeltaBusTravelTime*, *DeltaExcessTravelTime*, *DeltaWaitTimeDelivery*, *DeltaVehicleTripCost* and *ScheduledPassengers*. In this case, the first four concepts are used for describing the vehicle's utility function inside its service profile. The *ClientExcessTravelTime* and *ClientWaitTimeDelivery* concepts are used for the client's utility function. The *ScheduledPassengers* is used by the Planner to apply the filtering policy of minimizing the number of used vehicles. Finally, all these concepts are used in the proposal profile sent by the Schedule agents back to the Planner agent.

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 [4]. Examples of ontology concepts defining interaction protocols are: *MADARP-schedule-me*, *MADARP-process-query* and *MADARP-call-for-trip-proposals*.

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*.

4 Optimization and negotiation

The transport planning problem is treated in the planning layer following a two-phased model. The former is an optimization phase, aimed to the identification of trip solutions that maximize the operator's utility, subject to the (client) request profile restrictions. This involves the Planner and Schedule agents. The latter is a negotiation phase that pursues an agreement between the transportation proposals and the client's interests. This is done by the Planner and the Trip-request agent.

This two-phased model offers a major difference: traditional DRT systems use a unique objective function that includes the client's utility (level of service). This raises the problem of adequately weighting the client's and the operator's coefficients within the objective function, but whose solution is not always realistic or easy to find.

Our model instead, uses a negotiation process to solve the tradeoffs between clients and the operator, avoiding the assumption of wrong estimations about the client's desires, by simply incorporating the client in the final decision-making. In addition, the framework allows the use of different types of negotiation processes.

In particular, we implemented a simple negotiation mechanism that was used for processing all the trip-requests. In practice, the Planner gives the client (through his Trip-request agent) a list of filtered proposals. The client ranks and selects from the list of alternatives the most suitable trip solution offered by the Planner.

For doing the evaluation, the Trip-request needs from each trip-proposal the values of the variables used by the client's utility function. The required variables are modeled as a list of utility properties inside the client's service profile. The list arrives to the Planner inside the request and then to the Schedule agents inside the call-for-trip-proposal message. In this way, each Schedule agent can provide back, inside the proposal, the list with the variables' values for that trip insertion evaluation.

A similar situation happens with the Planner when applying the filtering policies. It also needs some variables' values from the trip proposals. So it simply attaches additional utility properties to the call for trip-proposal message (see Figure 1, 3)) and gets back their respective values in the vehicle's proposals.

5 The scheduling algorithm

The Schedule agents do the main planning work, evaluating the insertion of a trip. As mentioned earlier, the underlying MAS framework allows the implementation of different scheduling policies. In our preliminary tests, the system used all the schedule agents with the same scheduling policy. The adopted policy was the ADARTW algorithm, a constructive greedy heuristic [4]. ADARTW finds all the feasible ways in which a new customer can be inserted into the actual vehicle's work schedule, choosing the one that offers the maximum additional utility according to a certain objective function.

While serving, a vehicle can be: 1) in transit, serving one or more requests or 2) idle, available for serving the next customer. Therefore, a complete vehicle's work-schedule will have periods of vehicle's utilization (schedule blocks) and the so-called slacks times, periods in which the vehicle is available and waiting. A schedule block always begins with the vehicle starting on its way to pick-up a customer and ends when the last on-board customer is discharged. Figure 2 shows a schedule block that serves 3 customers (h,i,j) while evaluating the insertion of a fourth one (customer x). Each of them has their pick-up (+) and delivery (-) stops respectively.

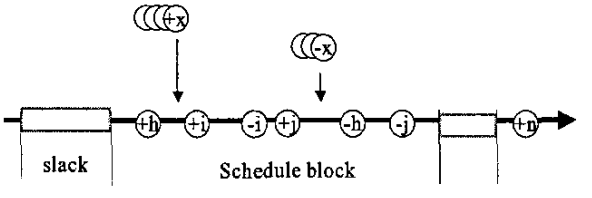

Figure 2. Vehicle's work-schedule

The search must include all the vehicle's schedule blocks. In a block with already d stops (2 per customer) there are (d+1)(d+2)/2 possible insertions, considering that the customer's pickup must precede his delivery.

The objective function Max z = ΔVc measures the additional utility for the vehicle due to the insertion of a new customer. For the term (ΔVc) some variables that can be considered are: driving time, traveled distance, capacity used, among others. Notice that the variables and their weights are not fixed on advance, nor the same for the whole system. They are different from vehicle to vehicle according to their service profile description.

6 Preliminary tests

The proposed MAS architecture has been developed in a prototype application using the JADE [1] agent platform. Using this architecture have been implemented and compared both, the centralized and the mixed approaches.

The centralized model was based on the original ADARTW heuristic for the scheduling and assignment of requests, as documented in [5]. On the other hand, the following operational decisions were adopted for the mixed model implementation: (1) the same utility function and scheduling algorithm have been used for all the vehicles; and (2) all the clients share the same utility function and implement the same negotiation model.

The preliminary tests considered 20 demand scenarios, each with 50 trip requests, distributed uniformly in a two-hours horizon. For each demand scenario 25 runs were done. The objective function for the centralized model considered the DeltaExcessTravelTime, the DeltaWaitTimeAtDelivery, the DeltaBusSlackTime, and the DeltaBusTravelTime. In the case of the mixed model, the utility function of the vehicles was composed by the DeltaBusSlackTime and the DeltaBusTravelTime; the utility function of the clients was composed by the ClientExcessTravelTime, and the ClientWaitTimeAtDelivery. In all cases the variables were weighted with the same value.

The preliminary results (see Table 1) have shown that the mixed approach gives better results for the clients, in terms of both, excess travel time and waiting time. On the other hand the centralized model performs better from the vehicles' perspective.

Table 1. Simulation results

	Centralized	Mixed	Variation
Wait time [min]	23,31	22,80	2,19%
Excess time [min]	8,65	7,20	16,72%
Travel time [min]	1655,42	1715,35	-3,62%

7 Further work

A next work is the use of an XML-based language (e.g. DAML+OIL) for describing the service profiles and trip proposals instead of the used Profile objects. It also involves implementing the corresponding parser.

Another future direction is to enable the Broker with semantic matchmaking capabilities, in order to allow client's and vehicle's service profiles to be semantically more different and still match.

Finally, other further improvements are: the test with different utility functions for clients and vehicles, the implementation of other scheduling algorithms and the use of more complex negotiation schemes between Planner and clients.

8 Conclusions

In this paper we have described an architecture for the implementation of DRT systems. The proposed mixed model assigns in a more natural way the roles to the different actors (vehicles, operator, and clients) participating in the optimization and decision making process. In this way, the estimation of coefficients to weight the actors' utility functions is avoided, and the idea of a more concealed solution is adopted.

Moreover, the multi-agent architecture allows the planning process execution on a heterogeneous network of computer systems and represents a flexible approach for a DRT system: the integration of a Broker agent together with the specification of a service ontology allows the dynamic registration of new vehicle profiles with new typologies of services.

References

[1] F. Bellifemine et al., "JADE - A FIPA Compliant Agent Framework". CSELT Internal Technical Report, 1999.

[2] D. Brugali, and G.Menga, "Architectural models for global automation systems". In IEEE Transactions on Robotics and Automation, Vol. 18, No. 4,2002, pp 487-493.

[3] C. Cubillos, et al, "On user requirements and operator purposes in Dial-a-Ride services". Proceedings of the 9th Meeting of the EURO Working Group on Transportation, 2002, pp. 677-684.

[4] Foundation of Intelligent Physical Agents (FIPA), "FIPA Abstract Architecture Specification 2002". Available at: http://www.fipa.org/.

[5] J.Jaw, et al, "A heuristic algorithm for the multiple-vehicle advance request dial-a-ride problem with time windows". Transportation Research. Vol. 20B, No 3, 1986, pp. 243-257.

[6] N.H. Psaraftis, "Dynamic vehicle routing: Status and prospects". Annals of Operation Research. No 61, 1995, pp. 143-164.

[7] W3C, "Resource Description Framework (RDF): Concepts and Abstract Syntax". Proposed recommendation, December 2003. Available at: http://www.w3.org/.

[8] G. Weiss, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Massachusetts, USA. 1999.