A Multi-agent Intelligent System for on Demand Transport Problem Solving

Mohamad EL Falou and Mhamed Itmi
National Institute of Applied Sciences, Rouen, France

Keywords: On Demand Transport, Multi-agent System, Intelligent Transport.

Abstract: In recent years, urban traffic congestion and air pollution have become huge problems in many cities in the world. A possible investment in order to reduce congestion is to increase the number of passengers in vehicles, and to decrease the number of vehicles on streets. This problem is defined as on demand transport (ODT) problem. The ODT problem environment is defined by three components: the infrastructure of the city, the vehicles and the drivers. Clients formulate requests for transportation from a pickup, to a drop off places. These requests are received and must be served on real time by the set of vehicles, which require real time environment updates. This paper is a first step to model the ODT problem as a multi-agent distributed planning problem. Our model relaxes some definitions and reduces the complexity of the ODT problem to allow better optimization.

1 INTRODUCTION

Research on new traffic information control and traffic guidance strategies are particularly necessary and important to reduce traffic congestion and air pollution. The application of information technologies such as multi-agent approaches to urban traffic information control has made it possible to create and deploy more intelligent traffic management systems such as "On Demand Transport" ODT system's.

A multi-agent system (Weiss, 1999) is an aggregation of agents, with the objective of decomposing the resolution of a large problem into agents in which they communicate and cooperate with one other. Multi-agent simulation has been looked as an efficient tool for urban dynamic traffic services. However, the main problem is how to build an agent-based model for such problem.

In this paper, we propose a multi-agent based multi-layer distributed planning model for the real-time ODT problem solving. In the proposed model, a client submits a request to transport passengers from a pick up to a drop off places. A pathfinder agent is defined to find a path between places. One vehicle (resp. driver) agent is associated per vehicle (resp. driver). Agents collaborate to find vehicles to transport passengers and drivers to drive the vehicles.

The paper is organized as follows: next section describes related works. Section 3 describes the formal framework of the ODT problem. Section 4 describes the multi-agent modeling of the ODT problem. In section 5 we define the agents for our problem domain and their cooperation strategy. Finally, section 6 concludes the paper and discusses the perspectives and the future work.

2 RELATED WORK

The on demand transport problem, as defined in the literature, has motivated a lot of studies of network logistic problems in the last decade.

A set of algorithms for the vehicle routing and scheduling with time window constraints are presented in (Marius, 1987). These algorithms use approximation and heuristics methods to offer solutions for practical size problems.

(Savelsbergh and Sol, 1998) studied the problem of finding the optimal way of assigning a set of transportation requests to a fleet of vehicles, by minimizing a specific purpose objective function, subject to a variety of constraints. Due to the complexity of the problem, approximation and incomplete optimization techniques as well as a sophisticated column management scheme have been employed to create the right balance between solution speed and solution quality.

(Lu and Dessouky, 2004) solved the pickup and delivery problem with time windows using a branch-

and-cut technique, based on new valid inequalities proposed by the authors. More recently, (Dumitrescu, 2005) provides several valid inequalities for solving the traveling salesman problem with pickups and deliveries (TSPPD), establishing those among all known inequalities that define facets of the TSPPD polytope.

Although many exact methods have been developed for solving variants of the on demand transport (so called, pickup and delivery problem also in the literature), none of them actually avoid the complexity of the problem, limiting their solution power to small size problems. This evident drawback motivates the development of good heuristics and/or the decomposition of the problem, to solve medium and large scale systems.

(Xu, Abdulrab and Itmi 2004) present a multi-agent based multi-layer distributed hybrid planning model for demand responsive transportation system. This approach doesn't model the problem in a decentralized manner, but uses the A-globe agent platform which provides a distributed environment to execute its algorithms.

Finally, in (Bertelle, Nabaa, Olivier and Tranouez 2009) develop a decentralized approach based on the optimization and negotiation between vehicles to resolve the ODT problem. Their goal is to face the lack of service in certain zones or the over-concentration of vehicles in certain other zones in a changing environment.

To the best of our knowledge, the multi-agent approach modeling the ODT problem in the literature does not give solution in real time and doesn't separate the driver from the vehicle. These drawbacks reduce the optimality performance of their approaches.

In this paper, we propose a new architecture to solve the ODT problem. This architecture is a first step to achieve its full optimization.

3 FORMAL FRAMEWORK

In this section, we give the formal framework of the on demand transport problem.

Definition 1. ***Infrastructure:*** *The infrastructure of our system is defined by a graph* $G = (P,R)$ *comprised by a set of nodes* $P = \{p_1,\ldots,p_n\}$ *designating the set of place in the city, and a set of links* $R = \{(r_1,\ldots,r_n\}$ *designating the roads of the city. Each road* $r_i = p_i \xrightarrow{(t_i,d_i)} p_j$ *relates two places* p_i, p_j *where* t_i *is the estimated time to cross* r_i *and* d_i *is the distance between* p_i *and* p_j.

Definition 2. ***Request:*** *The client request is defined by*

$R = ((init, t_{init}) \xrightarrow{n_{passengers}} (goal, t_{goal}), n_{vehicle}, n_{driver})$
where :

- *init and goal are the pickup and the drop off places,*
- t_{init} *and* t_{goal} *are the pickup and drop off time,*
- $n_{passengers}$*: the number of passengers to be transported,*
- $n_{vehicle}$*: the maximum number of times passengers agree changing vehicle,*
- n_{driver}*: the maximum number of times passengers agree changing driver.*

In the ODT system, a set of vehicles is defined. In what follows we give the vehicle definition and its associated information.

Definition 3. ***Vehicle:*** *A vehicle is defined by (name, capacity, place, cost, planning table) where :*

- *name: is the name of the vehicle,*
- *capacity : is the number of places in the vehicle,*
- *garage: is the garage place of the vehicle,*
- *cost: is the cost of using the vehicle per kilometer,*
- *planning table: is the planning table of occupation of the vehicle per day. Each line in the table is defined by* $(t, n_{places}, travelers, drivers)\}$ *where:*
  - t *is the beginning time of a crossing a link* l_i *between two places during day,*
  - n_{places} *is the occupied places in the vehicle during* l_i,
  - $n_{travelers}$ *is the number of travelers during* l_i,
  - *driver is the driver of the vehicle during* l_i.

  By defining the table planning for a vehicle, we know at each moment the place of the vehicle (which can be a node or a link), the number of passengers and the vehicle driver.

Definition 4. ***Driver:*** *a driver is defined by: (name, cost, working place, working table) :*

- *name : is the name of the driver,*
- *cost : is the cost of the driver per hour of driving,*
- *working place is the working place of the driver,*
- *working table : contains the working time and the occupation of the driver during a day. Each line in the table is defined by* $(t, work, place, vehicle)$*:*
  - t *is the beginning time of a crossing a link* l_i *on day,*
  - *work indicates if the driver works at time* t*,*
  - *place indicates the place of the driver at* t*,*
  - *vehicle is the vehicle conducted by the driver during* l_i.

By defining the table planning for a driver, we know at each moment its place (which can be a node or a link), and the vehicle he conducts.
The table contains also the information if during her/his working time, the driver has a vehicle to conduct or is free.

Definition 5. ***Trajectory:*** *a trajectory in the ODT system is defined by*
$\Big((init, start_1) \xrightarrow{v_1, dr_1} \dots (x_i, start_i) \xrightarrow{v_i, dr_i} \dots goal)\Big)$
where:

- *$path = init \to ...x_i... \to goal$ is the list of places crossed by the vehicles to reach drop off place (goal) from the pick up place init,*
- *$start_i$ is the time of each transition in the path*
- *v_i is the vehicle that takes passengers for the transition $x_i \to x_{i+1}$,*
- *dr_i is the driver who drives the vehicle for the transition $x_i \to x_{i+1}$.*

After giving the preliminary definitions of the ODT system, we now focus on the ODT problem.

Definition 6. ***On Demand Transport Problem:*** *Let us give an operational on demand transport system defined by: an infrastructure, a set of drivers, a set of vehicles and a set of trajectories. The ODT problem is defined by the client request and its solution consists in finding the optimal trajectory to respond to the client request.*

3.1 Optimality in the ODT System

In this section, we discuss our vision of optimality in the ODT system. Two optimality criteria can be distinguished for a trajectory: the time and the cost.

Let $Traj = \{traj_1, \dots, traj_n\}$ to be a set of trajectories and $Req = \{r_1, \dots, r_n\}$ to be a set of requests. We said Req satisfies $Traj$ iff $\forall req \in Req, \exists traj \in Traj / traj$ is the response to req. We note by $time(traj_i)$ (resp. $cost(traj_i)$) the time (resp. cost) execution of the trajectory $traj_i$.

Definition 7. ***ODT Optimality***
let $Traj = \{traj_1, \dots, traj_n\}$ satisfying $Req = \{r_1, \dots, r_n\}$.
ODT is timely optimal *iff : $\forall$ $Traj'$ satisfying Req* [1] *, $\sum time(traj_i) \leq \sum time(traj'_i)$ where $traj_i \in Traj$ and $traj'_i \in Traj'$.*
ODT is costly optimal *iff : $\forall$ $Traj'$ satisfying Req, $\sum cost(traj_i) \leq \sum cost(traj'_i)$ where $traj_i \in Traj$ and $traj'_i \in Traj'$.*

[1] $Traj$ satisfies Req iff $\forall r \in Req, \exists traj \in Traj$ which is a unique solution to r in $Traj$

Optimality Assumption
The development scenario of our ODT system is as follows:

Firstly, the infrastructure of the system is defined, the vehicles and the drivers are added. Then, the system is activated to be requested by its clients. The system receives and resolves the requests one by one. When the system receives the first request, it responds by the optimal trajectory. When another client submits a second request, the system can search the optimal trajectory with (or without) allowing the change of the first one.

Our position is to allow the change because our objective is to have an ODT system that is optimal (see definition 7) continuously.

We note that change in the ODT system is not only due to the reception of new requests, but change can be due to other factors such as:

- infrastructure: an accident or a work on a road may change the infrastructure,
- vehicles: a vehicle breakdown can change its own planning table.
- drivers: a driver can get sick which causes changing in its working table.

In these cases, the system must adapt its trajectories to remain operational and optimal.

As we said in the introduction, this work is a first step to model the ODT problem as a multi-agent planning problem. So studying the optimality is out of scope of this paper. All what we want to do here is to open a new door towards a real-time ODT system.

4 MODELING AN ODT PROBLEM AS A MULTI-AGENT PLANNING PROBLEM

4.1 General Architecture

In this section, we model the ODT problem as a multi-agent distributed planning problem. The architecture of the multi-agent system is illustrated in Figure 1. In the middle of the figure, we show the space of dialogue. The agents discussions are centered around their own information (i.e. planning table), the infrastructure and the trajectories of the system.

In parallel of explaining each type of agents, we show the life cycle of the client demand (from it's appearance till the trajectory is calculated), and the role of each type of agents to set the life cycle state.

In the multi-agent architecture, four types of agents are distinguished:

- **Client agent:** the client agent acts as an interface between the clients and the ODT system. IT sends the client request (Definition 2) to the system, and receives -if exists- the trajectory response (Definition 5).

  The life cycle of the client demand starts with the request produced by a client.
- **Pathfinder agent:** this agent is responsible of finding for a request, a path between its pickup and drop off places in the infrastructure of the system.
- **Vehicle agent:** a vehicle agent is associated with each vehicle in the system. Its role is to respond if it can transport - partially or totally- passengers through the path.
- **Driver agent:** a driver agent is associated with each driver in the system. Its role is to respond if it can drive -partially or totally-the vehicles associated to a path (called *vehicles-path* in what follow).

The details of how agents work and coordinate are explained below.

4.2 Agents Coordination Strategy

The coordination strategy between agents is illustrated in Figure 1. Firstly, the client agent sends the request to the pathfinder agent (1) which searches the path between the pickup and the drop off places in the infrastructure (2). Then the pathfinder submits the path to the vehicle agents (3).

Each vehicle agent computes its contribution to execute -partially or totally- the path, then the vehicles agents collaborate to produce the vehicles-path (4) or fail (5).

In success situation, the drivers agents take the hand (4). Each driver agent computes its contribution to execute -partially or totally- the vehicles-path, then the drivers agents collaborate to produce the trajectory (6) or fail (7).

In the next sections, we define the details of each kind of agents and their cooperation strategy.

5 AGENTS ROLES AND COOPERATION

In this section we define the role of each agent in our multi-agent model, and show how it coordinates with other agents.

Before starting, we give some preliminary path notations and properties.

The path produced by the pathfinder can be represented by:
$path = \Big((init, start_1) \xrightarrow{t_1} \dots (x_i, start_i) \xrightarrow{t_i} \dots goal)\Big)$,

where t_i is the estimated time to cross the transition $x_i \rightarrow x_{i+1}$ and $start_i$ is its starting crossing time by a vehicle.

We define $path_k$ by $\Big((init, start_1) \xrightarrow{t_1} \dots (x_k, start_k)\Big)$. It is obvious that the minimal time execution of path is $t_{path} = \sum_{i=1}^{n} t_i$. $start_i$ is introduced to express the possible delay between crossing two successive transitions when agents don't have solution to execute the path without wait.

There are multiple constraints to have a valid path which are :

- $t_{path} \leq (t_{goal} - t_{init})$. The request delay must be greater than the minimal time path execution.
- $t_{init} \leq start_1$. The first transition crossing must be after the pickup time.
- $start_i \geq start_{i-1} + t_{i-1}$. The starting time for the $i-th$ transition must be greater than achieving the $(i-1)-th$ transition.
- $t_{path} + delay_{path} \leq t_{goal} - t_{init}$, when $delay_{path} = \sum_{i=1}^{n} (start_i - (start_{i-1} + t_{i-1}))$. The time to cross all the transitions with the possible delay must be less than the sum of delays between the drop off and the pick up places in the client request.

5.1 Pathfinder Agent

The pathfinder role is to find the path between the pickup and the drop off places of the request (Definition 2). A number of classical graph search algorithms have been developed to resolve the shortest path problem on a weighted graph. Studying these algorithms is out of scope of this paper. Readers interested can be refereed to the two popular ones such as: Dijkstras algorithm (Dijkstra, 1959) and A* (Hart, Nilsson, and Raphael (1986)).

5.2 Vehicles Agents Coordination Strategy

Now we explain how vehicles agents coordinate their works to produce their best vehicles-path.

A central vehicle agent $\mathcal{VA}_c$ is added to the vehicles agents. It plays the role of an interface between the vehicles agents and the other components of the ODT system (pathfinder and driver agents), and manages the coordination between the vehicles agents.

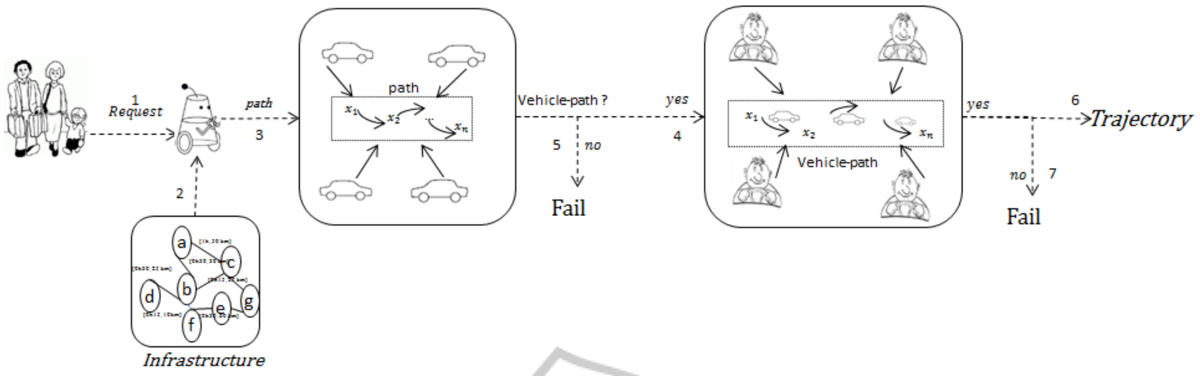

Figure 1: Agents coordination strategy.

When $\mathcal{VA}_c$ receive the $path = (init \cdots \rightarrow \dots goal)$ from the pathfinder, it diffuses it to all the vehicles agents $< \mathcal{VA}_1, \dots, \mathcal{VA}_n >$. Each one $\mathcal{VA}_i$ responds with its best partial *vehicle-path$_i$* [2].

When $\mathcal{VA}_c$ receives all agents responses, it chooses the best one. Let *vehicle-path*$_{i_1} = (init \dots \xrightarrow{v_j} \dots x_{j_1})$ be the best partial vehicle-path executed by $\mathcal{VA}_j$.

Then, the central agent $\mathcal{VA}_c$ notifies $\mathcal{VA}_j$ to update its vehicle planning table and diffuses the remained part of the path $(x_{j_1} \cdots \rightarrow \dots goal)$ to the agents. This iteration is repeated until :

- reaching a step with empty remaining part of the path. This means that vehicle agents successfully found a total *vehicle-path* which is equal to the concatenation of intermediate partial *vehicles-paths*. In this case, the vehicles-path is sent to the drivers agents.
- vehicles agents doesn't succeed to execute the path totally means there is no solution for the ODT system.

Finally, the central vehicle agent check that the number of vehicles executing the path is less than $n_{vehicle}$ of the request.

In the next section, we detail how the vehicle agent responses to a *path* by a partial or total *vehicle-path*.

5.3 Vehicle Agent Work

Let us recall that the agent receives the $path = \Big((init, start_1?) \xrightarrow{t_1} \dots (x_i, start_i?) \xrightarrow{t_i} \dots goal)\Big)$ that is the response to the request $R = ((init, t_{init}) \xrightarrow{n_{passengers}} (goal, t_{goal}), n_{vehicle}, n_{driver})$. Taking into account its planning table (see definition 3), the vehicle must respond with the sub-path $path_k$ it can execute and at which time $start_i$ $(1 \leq i \leq k)$ it crosses each transition. The agent tries to execute the maximum part of the path.

At the first step, the vehicle computes its free time-slot $Ts = \{ts_1, \dots, ts_m\}$ between t_{init} and t_{goal}. Let $ts_i = [a_i, b_i]$ and $|t_i| = b_i - a_i$. From Ts, we delete intervals ts_i having the number of free places in the vehicle during ts_i less than $n_{passengers}$.

We should recall here that the vehicle is occupied before each free time-slot. The vehicle searches for the interval ts_i that maximizes its proposed *vehicle-path*.

$ts_i = \underset{ts_i \in Ts}{\operatorname{argmax}} \Big((b_i - a_i) - t_{path_{x_i \rightarrow init}}\Big)$ where :

- x_i be the last place of the vehicle before a_i,
- $path_{x_i \rightarrow init}$ the shortest path between x_i and $init$ and $t_{path_{x_i \rightarrow init}}$ its associated execution time.

In selecting the maximal free time-slot, we must take into account the time $(t_{path_{x_i \rightarrow init}})$ the vehicle needs to reach the *init* place to take passengers.

The agent *vehicle-path* response is $path_k$ where $k = \underset{k \in [1..n]}{\operatorname{argmax}} \left((\sum_{j=0}^{k} t_j \leq |b_i - a_i - t_{path_{x_i \rightarrow init}}| \right)$. This means the computation of the maximum number of transitions that can be crossed in the free interval, taking into account the time to reach *init*.

In this case, $start_1 = a_i + t_{path_{x_i \rightarrow init}}$ and $start_j = start_{j-1} + t_i$ where $1 < j < k$.

5.4 Driver Agent Work

The drivers agents receive from the vehicles agents the *vehicles-paths*. They work and coordinate similarly to vehicles agents to produce the trajectory solution.

A central driver agent $\mathcal{DA}_c$ is added to the drivers agents. It plays the role of an interface between the drivers agents $\mathcal{DA}_i$ and the

[2]The best *vehicle-path* is the one that includes the maximum number of transitions.

other vehicles agents and manages the coordination between the drivers agents.The vehicle path $\Big((init, start_1) \xrightarrow{v_1} \ldots \xrightarrow{v_m} goal)\Big)$ received from the vehicles agents is diffused to the drivers agents. Each one responds with its best partial *trajectory$_i$* and the best one is selected. The associated driver agent is notified to update its driver planning table and the remained part of the *vehicle-path* is diffused to the agents, until finding the trajectory solution or fail.

Finally, the central driver agent checks whether the number of driver executing the path is less than n_{driver} of the request.

5.5 Driver Agent

When a driver agent receives a *vehicles-path* equal to $\Big((init, start_1) \xrightarrow{v_1} \ldots (x_i, start_i) \xrightarrow{v_i} \ldots goal)\Big)$, it responds by the maximum number of vehicles that can conducts.

Like the vehicle agents, the free time-slots are computed and the one maximizing the number of vehicles transitions is selected. Then, these vehicles transitions are associated to the drivers.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a new multi-agent architecture to solve the ”On demand transport problem”. Our model defines the problem as general as possible to produce the best solutions to a real operational ODT system. This distribution of the solution computation reduces its complexity and permits to have a real time system. Due to the lack of space, we cannot detail the complexity study of our model.

This work is a first step to solve the ODT problem as a muli-agent problem. In the future, we must implement our model and compare it to existing works.

Our architecture will be extended to ensure the completeness and the optimality of the algorithm. In our actual model, a solution may exist without being found by the agents, and even if it is found, it is not necessarily optimal. This is due to the lack of communication between the agents.

The multicriteria optimization notion must be introduced in the system. We also must study how to ensure the optimality of the ODT system continuously. This means that when the system searches a new trajectory, it allows to modify existing trajectories towards a global optimization of the system.

Finally, we have to study how the ODT system remains operational and optimal in real time, following a change in an ODT component like: accident, roads works, etc.

7 EXTENSION

This work deals with the transport of passengers. However it may also serve for goods transportation.

An improved and adapted version of the proposed approach can be the core of an intelligent transportation system. It involves users from transport companies, farmers' associations, supermarkets, public authorities, etc. in a system dedicated to Fruits and Vegetables transportation for example. Each user will be implemented as an electronic application (such as web services). Each one will be connected to the TOD system by an adapter to map between different electronic business standards (UN/CEFACT, UBL, NIEM, GS1, etc.) and the system.

ACKNOWLEDGEMENTS

This research is supported by the SEETFEL project The Trans-Mediterranean Electronic Exchange System for Fruits and Vegetables. SEEFTEL is financed by the French Government in the framework of ”Investissements d'Avenir” (Investments in the Future).

REFERENCES

Weiss, G., 1999. Multiagent systems: a modern approach to distributed artificial intelligence. *MIT Press*.

Dijkstra, E, 1959. A Note on Two Problems in Connexion with Graphs. *Numerische mathematik*, 269-271.

Hart, P. and Nilsson, N. and B. Raphael, 1986. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science, and Cybernetics*, 100–107.

Marius, M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 254–265.

Savelsbergh, M. and Sol, M. 1998. Drive: Dynamic Routing of Independent Vehicles. *Institute for Operations Research and the Management Sciences INFORMS*, 474–490.

Lu, Q. and Dessouky, M. 2004. An Exact Algorithm for the Multiple Vehicle Pickup and Delivery Problem. *Transportation Science*, 503–514.

Dumitrescu, I., 2005. Polyhedral results for the pickup and delivery travelling salesman problem. *Centre de Recherche Sur Les Transports, Université de Montreal*.

Xu, I. and Abdulrab and H., Itmi, M., 2008. A multi-agent based model for urban demand-responsive passenger transport services. *International Joint Conference on Neural Networks*, 3668-3675.

Bertelle, C. and Nabaa, M. and Olivier, D. and Tranouez, P., 2009. A Decentralized Approach for the Transportation On Demand Problem. *From System Complexity to Emergent Properties*, 281–289.