A Multiagent Traffic Management Framework for Cooperative Intelligent Transportation Systems

Pedro Oscar Pérez Murueta, César Cárdenas, David Muñoz
Telecommunications and Networking Research Group
Tecnológico de Monterrey
{pperezm, ccardena, dmunoz}@itesm.mx

***Abstract*—The rapid growth experienced by many cities has created huge pressures on their traffic and transport systems. One common solution consists of increasing the number of roads available, but is not always the most appropriate. The application of Information and Communications Technologies enabling more efficient and effective use of the existing transport infrastructure is necessary (also known as Intelligent Transportation Systems). Furthermore, cooperation between vehicles and infrastructure is currently studied under the name of Cooperative Intelligent Transportation Systems. This paper proposes a traffic management framework based on agents that enable users to make better use of traffic and transport infrastructure by cooperating with infrastructure. We also propose a parallel algorithm to determine the K-routes for the drivers.**

Keywords—Multiagent; Traffic Management; Cooperative Intelligent Transportation Systems.

I. INTRODUCTION

The traffic generated by public and private transport is the major cause of problems such as air pollution, pernicious noise levels, economic losses caused by the time spent on transfers and global warming to name a few. For example, it is estimated that each of the inhabitants of Mexico City loses two to four hours in transit, which means 3.3 million hours in traffic daily and generates a loss of 33 billion pesos per year [1]. In order to mitigate the effects of traffic congestion, the governments of many cities held every year significant investments in transportation infrastructure. Mexico City, for example, conducted an investment of 2.5 billion pesos in 2013 in the construction or improvement of new roads [2]. Nevertheless, the construction of new streets and avenues is not an appropriate solution. Braess [3] showed that adding a new path can lead to an increase in the total travel time. Alternative and affordable solutions apply new technologies to enable more efficient and effective use of the existing infrastructure.

One of the technologies that promise a solution to the problem of congestion are Intelligent Transportation Systems (ITS). The ITS integrate information and communication systems, computers and sensor technology and that enables effective and efficient management of traffic systems [4]. In fact, there are many proposals that show how the use of these technologies can reduce travel time for travelers who are still unfamiliar with a given route [5]. The idea of ITS consist not only to reduce the travel time, but to assist the driver to take into account all the security parameters to maintain a proper speed and distance to the vehicle ahead, avoiding passing in critical situations, among others.

But what if this individual information is shared with vehicles that are around us and transportation infrastructure? In theory, the benefits would be amplified. This is the concept behind a Cooperative ITS (C-ITS). The goal of C-ITS is to use wireless communication means for vehicles to communicate with other vehicles (V2V) and transport infrastructure (V2I) to improve safety and mobility in the streets. C-ITS can significantly increase the quality and reliability of the information available about the vehicles, their location and the road environment. They improve existing and lead to new services for the road users, which, in turn, will bring major social and economic benefits and lead to greater transport efficiency, increased safety, reduce congestion; mainly through collision avoidance, improve traffic management and drivers' decision making. For example, a vehicle with C-ITS approaching an intersection, and no visibility of oncoming traffic, broadcasts its location, speed and direction signal. At the same time it receives the same signal from another vehicle with C-ITS which is approaching from a different direction. If the vehicles' trajectories indicate an imminent collision, both drivers would receive warnings from their in-vehicle systems. This would allow the drivers to respond to these warning systems and reduce speed to avoid a collision, even before they can see the other vehicle coming.

C-ITS are still in development and are expected to be integrated in new vehicles over the following years. Several EU-funded projects have delivered promising results [1] : Coopers, CVIS, Safespot, COMeSafety, EuroFOT, Drive 2X, etc. Most of the previous projects were focused on the communication architecture, mainly on the physical layer. To the authors' knowledge, there is no previous work presenting innovations in traffic management for C-ITS. In this paper, we are proposing a framework for traffic management in C-ITS. The framework not only take into account information from other vehicles and infrastructure, but also consider the information generated from social networks and people, lets us know real-time traffic in the city. Furthermore, we also propose a parallel algorithm for the calculation of the k routes.

The paper is organized as follows. In section II, we present the multi-agent traffic management framework. In section III,

[1]http://ec.europa.eu/transport/themes/its/road/action_plan/cooperative_systems_en.htm

we present how gather offline and online information from the users and infrastructure. In section IV, we present a parallel algorithm to calculate the K-routes. Finally, in section IV we provide conclusions and future work.

II. Multiagent Traffic Management Framework

The National ITS Architecture provides a common framework for planning, defining, and integrating ITS. It is a well worked product reflecting contributions of the ITS community. The architecture defines the functions, the physical entities or subsystems where the functions reside and the information and data flows that connect functions with entities in an integrated platform[2]. The architecture also contains 33 services organized in 8 bundles. One of these bundles is dedicated to travel and traffic management. The services of these categories are: pre-trip travel information, en-route driver information, route guidance, ride matching and reservation, traveler service information, traffic control, incident management, travel demand management, emissions testing and mitigation and highway rail intersection. As we can see, most of these services are related to route guidance. Therefore, in the architecture, there is a module dedicated to travel and traffic management.

In this paper we propose a multi-agent traffic management framework for C-ITS. The framework is composed of three layers. The first layer is an agent platform; in the next layer we find the multi-agent architecture and finally the traffic information system. See Figure 1.

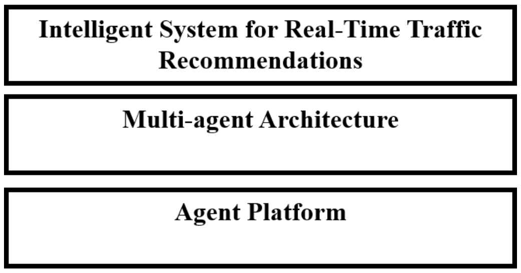

Figure 1. System Agent Framework

In the following we explain each of the layers of the multi-agent traffic management framework.

A. Agent Platform

For the agent platform we use JADE. JADE (Java Agent Development Framework) is a framework based on Java for the development of distributed multi-agent applications. It is actually a middleware that provides a set of easy-to-use services and complies with the FIPA standard. Moreover, JADE is a very flexible framework and can be used on devices with limited resources such as smart phones [6].

B. Multiagent Architecture

In the multi-agent architecture layer we find agents and structures that are necessary for implementing the system. There are four types of agents (See Figure 2).

[2] http://www.iteris.com/itsarch/

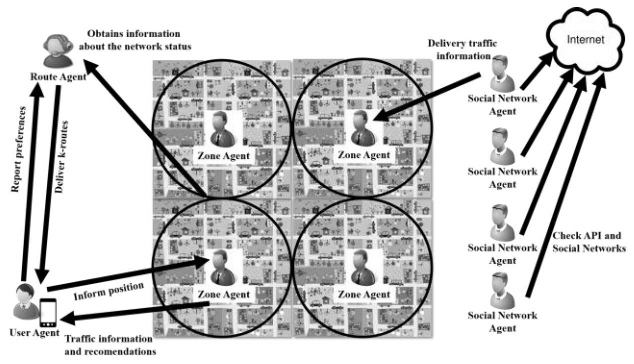

Figure 2. Interactions between agents

User Agent. This agent represents the user in their interactions with the system. It is responsible for reporting the geographical position of the user and provides communication and interoperability between the user and the system. It is also responsible for learning the way that the user drives in order to better understand his/her preferences and needs. Based on this information presented to the user the routes that best adhere to his/her profile. Finally is responsible for storing the best routes for the user evaluated for later use.

Zone Agent. The zone agent gets information from user agents to determine the density of traffic flow, updating this information in an Origen-Destination (OD) matrix. It informs to the user agents inside its area about possible congestion on the routes that the agents are using. When congestion on its area reaches a threshold, communicate this information to other zone agents.

Route Agent. The route agent is responsible for generating the K routes between a given source and destination. This agent gets the flow traffic from the zone agent and user preferences of the user agent. Use this information to select the routes with better traffic flow to and comply with the restrictions imposed by the user. This agent uses a parallelized algorithm k routes [7] based on the algorithm of Yen [8].

Social Network Agent. The social network agent gets information about traffic incidents and issues, such as construction sites and traffic congestion from open traffic APIs. Also analyzes traffic-related information that may exist in social networks. The information obtained is sent to the zone agents.

III. Data Gathering for Real-Time Traffic Recommendations

One of the biggest challenges we find is how to obtain the information in a discrete and continuous way on a citywide scale. To do this, we have in mind the use of a set of heterogeneous sources that allow us to approximate at any time the flow of traffic in whatever area of the city. The sources used and the challenges that each one presents are discussed below. (See Figure 3)

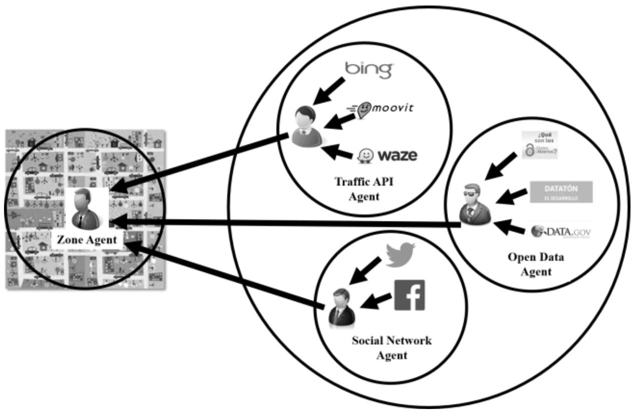

Figure 3. Agents with specialization by source

A. *Traffic APIs*

Currently there is a lot of open APIs such as API Bing Traffic, Traffic MapQuest API, Regional Traffic Data System (RTDS) API that allows us to obtain data related to a specific area or country. The data that we can obtain are the speed of the cars in a street, traffic incidents and issues, such as construction sites and traffic congestion. The agent in charge of monitoring the kind of source is not complex, since that all information obtained is perfectly structured.

B. *Open Data*

Open data is data that can be used, reused and redistributed freely by anyone, and are subject, at most, to the requirement of attribution and shared in the same way they appear [9]. Many governments have begun to place online and with freely download datasets on open standards. The idea is that these data can be used by civil society to propose solutions to the different problems that can exist in a city or nation. We can mention as examples the proposal of the government of Mexico City [10] or the competition held by the Mexican Federal Government [11]. The data are perfectly structured and can be easily processed. The challenge is to perform an analysis to forecasting the behavior of the traffic in a given area.

C. *Social Network Mining*

There is a set of hashtag Twitter and Facebook accounts you continually show reports of incidents that may affect traffic on a certain area. Monitoring these accounts would be possible to get updated information on the behavior of data traffic in the city. The main challenge of this kind of source is to build agent able to learn and recognize the language used and filter messages that contain information that can be really useful. An example of data mining for this purpose is the application developed by SAP called AUS Traffic. The application automatically searches in Twitter for relevant traffic updates and displays them on a Google city map [12].

IV. A Parallel Algorithm for Multiagent-based Traffic Management

A parallel algorithm for the real-time traffic architecture based on multi-agents, inexpensive and able to generate recommendations for alternate routes with less traffic flow in real time considering user preferences was proposed. This system will reduce congestion through more efficient and informed use of existing streets and avenues. The main task of an agent route is to calculate the best route and present it to the user. This path must satisfy all preferences and needs. But what if the best route is not available? Then it would be necessary to calculate the second best. If this also cannot be used then we need to calculate the third. The generation of this set of K is known as the K Shortest Paths problem (KSP).

A. *Definitions*

In the K Shortest Paths problem, we have a directed graph $G = (V, E)$ where V is a finite set of vertices and E is a finite set of edges connecting the vertices. Each edge $e \in E$ has associated a nonnegative cost. A route (or path), $R = \{v_s, v_1, \ldots, v_{n-1}, v_t\}$, such that each consecutive pair (v_i, v_{i+1}) is connected by an edge in G and need to be followed alternately in order to move from an origin node s to a destination node t [14]. A prefix-path, $P_i = \{v_s, v_1, \ldots, v_{i-1}, v_i\}$, is the subset of vertices that spanning from s to an intermediate vertex i. A suffix -path, $S_i = \{v_i, v_{i+1}, \ldots, v_t\}$, is the subset of vertices that spanning from i to the destination node t. The total cost of a path is the sum of the cost of all edges in it. The shortest path between s and t is the path that connects s to t, assuming it exists, which has the minimum cost [14]. To solve the problem KSP is necessary that an algorithm that calculates the shortest route (as that proposed by Dijkstra [15]) is forced to choose different paths through the graph. This is accomplished by marking vertices and edges ineligible or removed for further implementation of shortest path algorithm [16].

B. *Implementation*

This problem has been extensively studied since the early fifties and several implementations have been proposed, including works by Yen [17], Lawler [18], Katoh [19], Hoffman [20], Ahuja [21], Eppstein [22] and Hershberger [14] among others. All of them extend the shortest path algorithm defined by Dijkstra [15] and their efficiency is based on how they are constructed each of the K paths. For this implementation we opted for the Yen algorithm, mainly for two reasons. The first is that it is considered to date, which has better results when implemented with modern data structure ($O(kn\ (m + n \log n))$) [14]. Secondly, it is an easily parallelizable algorithm.

We will comment a little and informally how our parallelized algorithm works. Initially generate the shortest route using Dijkstra algorithm, which is stored a minimum heap. Then we get a route from the heap. In parallel and for each of the arcs of the path, marked as ineligible arc i connecting the pair of vertices (v_i, v_{i+1}) in both graph and the path. By doing this on the route we segmented it in a prefix-path, P_i, from the initial node to the vertex v_i and suffix -path, S_i, leaving the vertices v_{i+1} and reaches the end node. We execute once again the Dijkstra algorithm, but using (v_i, v_{i+1}) as an origin-destination pair. With the route obtained from the previous step, R_j, and the segments P_i and S_i we create a new route, which is stored in the heap. This process is repeated until the K routes have requested.

C. KSP Parallelized Algorithm

The pseudo-code of our algorithm is shown in the following:

```
============================================
• Initially generate the shortest path using Dijkstra's
  algorithm, which receives as parameters a pair of origin-
  destination points and a vector containing the weighs used
  to evaluate each edge. This first route generated is placed
  in a minimum heap.
• The following steps are repeated k times or until there are
  no more routes to process:
    1. Obtaining the shortest route R = {vs, vi, …, vt}.
    2. In parallel, for each pair of vertices (vi, vi+1):
        a. We marked as ineligible the edge connecting
           them. So that we obtain a prefix-path, Pi, and
           suffix-path, Si.
        b. Execute the Dijkstra algorithm using (vi, vi+1) as
           origin-destination pair and the weight vector V
           getting the route Rj.
        c. Create a new route, Rk, from the union of Pi, Rj
           and Si.
        d. If this route has not been previously generated, it
           is placed in the minimum heap.
============================================
```

V. CONCLUSIONS AND FUTURE WORK

Traffic problems of big cities can be very complex and dynamic. However combining different technologies such as multi-agent systems, mobile devices, distributed computing, data mining, machine learning can create solutions that are able to help people in these cities to improve their quality of life. Furthermore, to better manage the traffic, vehicle and infrastructure information can be shared for cooperation between vehicles and infrastructure. In this paper we present an architecture for cooperative multi-agent traffic management system able to know the status of the network traffic and make recommendations, in real time, to the users of it. We also propose a parallel algorithm for the calculation of the K routes in the framework. Future work will consist on implement the architecture and make exhaustive evaluations of it.

REFERENCES

[1] Atraccion360, "Causas y Problemas Tráfico Ciudad de México," [Online]. Available: http://www.atraccion360.com/caos-vial-frena-competitividad-en-el-df. [Accessed September 3, 2014].

[2] A. L. M. Morales, "Presenta GDF a través de SOBSE informe sobre inversión en remoción y mejoramiento vial," [Online]. Available: http://www.elpuntocritico.com/noticias-metropoli/noticias-df/64286-presenta-gdf-a-trav%C3%A9s-de-sobse-informe-sobre-inversi%C3%B3n-en-remoci%C3%B3n-y-mejoramiento-vial.html. [Accessed September 3, 2014].

[3] [D. Braess, A. Nagurney, and T. Wakolbinger, "On a Paradox of Traffic Planning," Transp. Sci., vol. 39, no. 4, pp. 446–450, Nov. 2005..

[4] Y. Huang, N. Jing, and E. A. Rundensteiner, "Route Guidance Support in Intelligent Transportation Systems: An Encoded Path View Approach." 1995.

[5] J. L. Adler, "Investigating the learning effects of route guidance and traffic advisories on route choice behavior," Transp. Res. Part C Emerg. Technol., vol. 9, no. 1, pp. 1–14, 2001.

[6] R. H. Bordini, L. Braubach, M. Dastani, A. El Fallah-Seghrouchni, J. J. Gomez-Sanz, J. Leite, G. M. P. O'Hare, A. Pokahr, and A. Ricci, "A survey of programming languages and platforms for multi-agent systems," Inform., vol. 30, no. 1, pp. 33–44, 2006.

[7] P. Pérez, E. García and M. Junco, "Finding in Multimodal Networks without Timetable," in VEHICULAR 2014, The Third International Conference on Advances in Vehicular Systems, Technologies and Applications, Seville, Spain, June 2014, IARIA XPS Press, pp. 29-32, 2014.

[8] J. Y. Yen, "Finding the K Shortest Loopless Paths in a Network," Manage. Sci., vol. 17, no. 11, pp. 712–716, 1971.

[9] Open Knowledge Foundation, "¿Qué son los datos abiertos?," [Online]. Available: http://datosabiertos.df.gob.mx/. [Accessed July 31, 2014].

[10] Dirección de Sistemas de Comunicación para la Atención Ciudadana de la Coordinación General de Modernización Administrativa, "Datos abiertos," [Online]. Available: http://opendatahandbook.org/es/what-is-open-data/index.html. [Accessed July 31, 2014].

[11] Datatón, "Datatón," [Online]. Available: http://mxabierto.github.io/dataton/acerca.html. [Accessed July 31, 2014].

[12] SAP TV, "SAP TV: Surf Through Traffic with SAP's iPhone App." [Online]. Available: https://www.youtube.com/watch?v=qRKjEpQAQQM&feature=player_embedded#. [Accessed July 31, 2014].

[13] C. Kaur, D. Krishnamurthy, and B. H. Far, "Using Web Mining to Support Low Cost Historical Vehicle Traffic Analytics," [Online]. Available: http://ksiresearchorg.ipage.com/seke/seke14paper/seke14paper_228.pdf. [Accessed July 31, 2014].

[14] Hershberger, J., Maxel, M., & Suri, S. (2007). Finding the k Shortest Simple Paths: A New Algorithm and its Implementation. ACM Transactions on Algorithms, 3(4), 45. doi:10.1137/S0097539795290477.

[15] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1(1), 269–271. doi:10.1007/bf01386390.

[16] Brander, A., & Sinclair, M. (1996). A comparative study of k-shortest path algorithms. Proc. of 11th UK Performance Engineering Workshop (pp. 370–379). doi:10.1007/978-1-4471-1007-1_25.

[17] Yen, J. Y. (1971). Finding the K Shortest Loopless Paths in a Network. Management Sci-ence, 17(11), 712–716. doi:10.1287/mnsc.17.11.712.

[18] Lawler, E. L. (1972). A Procedure for Computing the K Best Solutions to Discrete Optimi-zation Problems and Its Application to the Shortest Path Problem. Management Science, 18(7), 401–405. doi:10.1287/mnsc.18.7.401.

[19] Katoh, N., Ibaraki, T., & Mine, H. (1982). An efficient algorithm for K shortest simple paths. Networks, 12(4), 411–427. doi:10.1002/net.3230120406.

[20] Hoffman, W., & Pavley, R. (1959). A Method for the Solution of the Nth Best Path Problem. J. ACM, 6(4), 506–514. doi:10.1145/320998.321004.

[21] Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). Network Flows: Theory, Algorithms, and Applications. Upper Saddle River, NJ, USA: Prentice-Hall, Inc..

[22] Eppstein, D. (1999). Finding the K Shortest Paths. SIAM J. Comput., 28(2), 652–673. doi:10.1137/S0097539795290477.