Transportation Research Part C 73 (2016) 1–29

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Algorithms to find shortest and alternative paths in free flow and congested traffic regimes

Alberto Faro *, Daniela Giordano

Department of Electrical, Electronics and Computer Engineering, University of Catania, Italy

ARTICLE INFO

Article history:
Received 14 October 2015
Received in revised form 1 July 2016
Accepted 22 September 2016
Available online 18 October 2016

Keywords:
All Pair Shortest Paths (APSP)
Single Source Shortest Paths (SSSP)
Ubiquitous information systems
Traffic emergency assistance

ABSTRACT

Location-based systems can be very helpful to mobile users if they are able to suggest shortest paths to destination taking into account the actual traffic conditions. This would allow to inform the drivers not only about the current shortest paths to destination but also about alternative, timely computed paths to avoid being trapped in the traffic jams signaled by cyber-physical-social systems. To this aim, the paper proposes a set of algorithms that solve very fast the All Pair Shortest Paths problem in both the free flow and congested traffic regimes, for road networks of medium-large size, thus enabling location-based systems to deal with emergencies and critical traffic conditions in city and metropolitan areas, whose transport networks typically range from some hundreds to many thousands of nodes, respectively. The paths to avoid being trapped in the traffic jams are computed by using a simulation of the shockwave propagation, instead of historical data. A parallel version of the algorithms is also proposed to solve the All Pair Shortest Paths problem for metropolitan areas with very large road networks. A time performance analysis of the proposed algorithms for transport networks of various size is carried out.

1. Introduction

The determination of the shortest path in terms of travel times connecting two intersections of a transport network is a classic problem in graph theory and effective solutions are available by using either procedural or logic programming. Procedural shortest path algorithms, such as the one proposed by Dijkstra and subsequent optimized versions (Rodríguez-Puente and Lazo-Cortés, 2013), aim at reducing computation complexity and execution time; shortest path algorithms based on logic programming, e.g., the one proposed in (Faro et al., 2011a), are recommended if they should be integrated in a rule based decision support system.

The available shortest path algorithms have a satisfactory time performance, but one main shortcoming when they are applied to transport networks is the need to be re-executed for every user request, unless one reuses the most recently computed paths, under the assumption of slowly time-varying traffic. Of course, it is not possible to reuse such paths if the traffic conditions are modified by a traffic collision or sudden congestion.

The key idea of the approach underlying the new, fast algorithms proposed in the paper is the one of keeping continuously updated the matrices dealing, respectively, with the Shortest Path (SP) and related minimum Travel time (Tmin) between every relevant intersection pair of a transport network, to allow the Location based Service (LBS) of a city information system to find the path and travel time that meets the user query very fast. This improved responsiveness will allow any

* Corresponding author.
E-mail address: albfaro@gmail.com (A. Faro).

http://dx.doi.org/10.1016/j.trc.2016.09.009
0968-090X/

ubiquitous city information system to deal with emergencies and critical traffic conditions by informing timely the drivers about:

(a) the shortest path to destination (where the roads are weighted by travel times instead of distances, also known as fastest paths) to ensure a timely rescue of the people involved in accidents and
(b) the alternative routes to prevent the drivers from being trapped in the traffic jams signaled by cyber-social systems, i.e., by either electronic sensing devices or by alerts coming from other drivers.

In this way, ubiquitous city information systems may evolve towards cyber-social transportation systems (CSTS), able to merge social, computational and transportation elements to optimize relevant aspects of the city mobility dealing with travel information, traffic control and accident management, as envisaged in (Murakami, 2013; Sadek and Qiao, 2014; Xiong et al., 2015).

Indeed, current community-based traffic and navigation Apps, e.g., Waze (https://www.waze.com/) and irezQ (http://irezq.com/) don't take into account sensed data, and the navigation systems that are embedded on specialized devices (e.g., Tom Tom and Garmin) don't use in a systematic way traffic data sensed from the field. Also, adopting formal and verifiable algorithms is mandatory if the validity of the suggested paths to destination is a critical requirement, as is the case for emergency situations. This requirement cannot be satisfied by proprietary systems (e.g., Google Maps) that are not able to guarantee formally the matching between the travel suggestions and the current traffic conditions.

The intended use of the proposed algorithms is for adoption by city information systems that aim at suggesting alternative paths in case of sudden events signaled by social networks or monitoring systems likely managed by third party organizations, and may be the basis for better navigation systems that can be formally verified.

The methodology of the proposed approach is illustrated in Sections 2 and 3. In particular, Section 2 first provides the reasons why a novel computational framework for the future navigation systems is needed, then presents two algorithms and the related data structures to compute the source-destination tables containing the fastest path and related minimum travel time from any intersection of a transport network to all the others. The conditions under which the proposed algorithms may be implemented on the available computational systems are analyzed, and it is pointed out how they satisfy the time performance requirements for transport networks of various sizes.

Section 3 shows how to update very fast the above mentioned tables at every significant change of the travel times of the roads. This section illustrates also how to compute rapidly the fastest alternative path to destination in case of traffic jams signaled by either the alert systems in place. In Section 4 we discuss the parallelization of the proposed algorithms to improve their time performance so as they can scale to large road networks. Section 5 presents the time performance analysis of the proposed solutions, whereas Section 6 describes how the proposed algorithms work in practice, pointing out the validity of the suggested shortest paths to destination in typical urban traffic scenarios.

2. Methodology

There are many scenarios for which the available shortest path algorithms allow the navigation systems to satisfy the driver queries. For example, if the service has to meet the user requests about the shortest distance path between any node pair of a traffic network, it is enough, as pointed out in (Zeng and Church, 2009), to execute for each query an algorithm based on A*, i.e., an algorithm that speeds up the Dijkstra's algorithm by including a "forward-looking" component that estimates the length to complete the paths to the destination, helping to decide what paths to test with priority from a specific node. See (Stout, 1997) for a tutorial and (Cui and Shi, 2011; Ansari et al., 2015) for more recent optimized versions.

However, A* is a heuristic algorithm that has been proved to work effectively to find the shortest distance path to destination since in this case the lower bound of the distance from any node to the destination may be reasonably estimated using the Euclidean distance. On the contrary, this estimate may be inadequate if the aim is finding the fastest path because in this case the lower bound of the time needed to go from any node to the destination is not known a priori and may be set only with a certain approximation, e.g., as the Euclidean distance divided by the maximum allowed car speed.

This explains why A* is not suitable to find the *one to one fastest paths,* as confirmed experimentally by the tests reported in (Kim and Jeong, 2009), where it is also shown that A* and similar algorithms, e.g., ALT (Goldberg and Harrelson, 2004), are not the best choice to find the *one to all* shortest paths.

For this reason, to meet the queries issued from drivers interested in finding the fastest paths to destination, navigation systems have adopted more suitable algorithms. In particular, the tests carried out by (Kim and Jeong, 2009) pointed out that, accordingly to the analysis previously developed in (Zhan and Noon, 1998), the top three candidates for shortest/fastest paths applications on real road networks are two optimized versions of the Dijkstra's algorithm, known as DIKBA and DIKDB (Cherkassky et al., 1996), and the algorithm TWO-Q proposed in (Pallottino, 1984). These algorithms have comparable time performance to find the *one to all* paths for traffic networks from few thousands to a hundred thousands of nodes, whereas DIKBA and DIKDB outperform TWO-Q for the largest networks.

The above discussion clarifies why one of the earliest free online web mapping service, i.e., Mapquest (www.mapquest.com), finds the shortest distance paths between any node pair using A* in combination with an algorithm that finds the