Engineering Applications of Artificial Intelligence 71 (2018) 155–165

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Road map partitioning for routing by using a micro steady state evolutionary algorithm

Andrés Camero, Javier Arellano-Verdejo *, Enrique Alba

Department of Computer Science, University of Malaga, Spain

ARTICLE INFO

Keywords:
Evolutionary algorithm
Shortest path problem
Map partitioning
Real world application

ABSTRACT

The constantly increasing number of vehicles and the immense size and complexity of road maps set a tough scenario for real world routing. In spite of the tremendous efforts done up to date to tackle this problem, finding the shortest-path in practice is still a challenge due to memory and time constrains. Moreover, most of the efforts have been focused on algorithmics, setting aside the management of the data. However, memory and time constraints are very important to actually construct real world applications, advising a new global approach. In this study we propose a holistic strategy for finding the shortest-path based on efficiently managing the road map data. Our proposal is based on the *tile map* partitioning, a logic geographical partition strategy.

We have developed a routing system highly scalable based on a *micro steady state evolutionary algorithm* to find the optimal tile map partitioning. We show the actual efficiency and scalability by using the road maps of Malaga, Spain, and Mexico City, making it clear the significant reductions in the time needed to compute the shortest-path (in a real application), what is a key issue that can be freely exploited in future open software for maps.

1. Introduction

Going from home to work is a daily task, but choosing the *best* route every day may not be trivial (shortest path or SP). Thanks to computer systems, like GPS navigation systems[1] or web applications,[2] we are able to find the route to wherever we want to go. Furthermore, the routes may be customized based on our preferences (e.g. road type, tolls) and enhanced by live information (e.g. present traffic, incidents). However, offering this kind of service in a *real world* scenario presents a technical challenge: processing constantly changing and increasing size road maps to satisfy the requests of millions users who demand for real-time and up-to-date information even at the same time.

According to the United Nations, the World population by 2050 will exceed 9 billion people, of whom more than the 66% will be living in urbanized areas (United Nations and Affairs, 2015). Moreover, it is estimated that the 60% of the urbanized areas needed by 2050 have not yet been built (Swilling et al., 2013). Thus the road map scenario is expected to change dramatically in the next decades. For example, Open Street Map (OSM) reports that more than 41 million intersections are added every month.[3]

On the other hand, by 2030, it is estimated that the number of cars will rise up to 2 billions (Dargay et al., 2007), aggravating the traffic congestion. Besides, according to Gartner,[4] by 2020 there will be more than 20 billions of *connected* devices, a 3 times increase over the actual number. Consequently, the demand for real-time information will grow, as well as the amount of data produced.

As the database systems have had to evolve in order to be able to manage the amount of data produced these days, the information systems will have to adapt their paradigm to satisfy the future demand. Taking into account this, we think that the navigation systems of the future must be seen in a holistic way, in which the computation of the route and the management of the data is seen as a whole.

Many ideas have been proposed to compute the shortest path in real world scenarios, from exploiting the topological features of a road map (Efentakis et al., 2011), to top technological solutions that rely on parallelism (Aridhi et al., 2015).

In this study we propose a novel strategy for managing real world road map data specifically designed for routing purposes. This strategy is an extension to the *Tile Map Partitioning*, originally proposed by Camero

* Corresponding author.
E-mail addresses: andrescamero@uma.es (A. Camero), javerdejo@lcc.uma.es (J. Arellano-Verdejo), eat@lcc.uma.es (E. Alba).

[1] http://www.garmin.com/, https://www.tomtom.com/.

[2] https://www.google.com/maps.

[3] http://wiki.openstreetmap.org/wiki/Stats(30/1/2017).

[4] http://www.gartner.com/newsroom/id/3165317 (4/6/2017).

https://doi.org/10.1016/j.engappai.2018.02.016
Received 31 May 2017; Received in revised form 7 February 2018; Accepted 19 February 2018

et al. (2017). By dividing a road map into heterogeneous (i.e. of different size) logic geographic sectors, we propose to improve the adaptability of the original strategy to real world cities (i.e. non-uniform geographical road distributions), implicitly finding the trade-off between time and resources usage. The *Optimal Heterogeneous Tile Size* (OHTS) (i.e. the heterogeneous partitioning that minimizes the SP computing time) is computed by a *micro steady state evolutionary* (mSSEA) algorithm, due to the suitability of these algorithms to solve complex problems with large search spaces (Alba et al., 2009). The strategy is tested by using the road maps of the *Province of Malaga, Spain*, and *Mexico City*, showing that significant performance improvements are made in comparison to the original proposal, as well as we include here strong evidence of beating state-of-the-art techniques (Aridhi et al., 2015).

The rest of this work is presented as follow: Section 2 summarizes the state-of-the-art and introduces the basic concepts used in this study. Section 3 introduces the OHTS problem. Section 4 describes the mSSEA proposed to solve the OHTS problem. Section 5 briefly introduce the competitors and their results. Section 6 presents the experimental study. Section 7 discusses the main conclusions, and Section 8 proposes future work.

2. Related work

Everyday we deal with the problem of finding the *best* route to a place. The size of road networks makes this problem hard to handle, while the immense number of *road users* worsen the situation by dynamically modifying the status of the roads.

There are many algorithms proposed to find the shortest route (Dijkstra, 1959; Hart et al., 1968; Floyd, 1962; Potamias et al., 2009; Breugem et al., 2017; Sedeno-Noda and Raith, 2015; Li et al., 2015), however there is not much work related to efficiently managing the data used to calculate that route (Aridhi et al., 2015; Efentakis et al., 2011). Furthermore, most of the algorithms just assume that the data is available and/or require precalculations, but without taking care of how the data is accessed (Santos, 2009). However, due to the rapid increment of the amount of data observed in the past decade, efficiently managing the data imposes a challenge we cannot ignore.

In the rest of this section we overview the state-of-the-art of road map data management for efficiently solving the SP problem, and present the basic concepts of this study.

2.1. Shortest-path problem

The shortest-path (SP) problem consists in finding a path (or route) between two vertices of a graph, such as the *distance* between these vertices is minimum. This problem is found in many diverse areas, from social networks analysis (Liben-Nowell and Kleinberg, 2007) to robotics (Canny and Reif, 1987). Particularly in the road map context, the SP problem is usually tackled by modeling the map as a directed weighted graph $G = (V, E, \omega)$, $V, E \neq \emptyset$, whose vertices $v \in V$ represent the intersections of streets, edges $e \in E$ correspond to the streets (between two intersections), and $\omega(e)$ is the function that returns the weight (a positive value) of an edge.

Then, a route on the road map is denoted by the sequence $P = \{e_1, \ldots, e_n\}$ $(e_i, e_j \in V, e_i \neq e_j, 1 \leq i, j \leq n)$, where all edges of P connect a sequence of vertices, and the *length* of the route P is defined as:

$$L(P) = \sum_{i=1}^{n} \omega(e_i). \quad (1)$$

Thus, the minimum length route from v_i to v_j $(v_i, v_j \in V)$, denoted by $\delta(v_i, v_j)$, corresponds to the SP and is commonly mentioned as the distance from v_i to v_j.

Note that the weight of an edge may be interpreted not only as the geographical length, but also as a combination of several measurement, for example the air pollution, the noise pollution, among others.

The SP problem is usually divided into three sub-problems: 1. *point-to-point shortest-path* (P2PSP), finding the SP between two given vertices, 2. *single source shortest-path* (SSSP), finding the distance from one vertex to all other vertices, and 3. *all pairs shortest-path* (APSP), finding the distance between all pairs of vertices.

The algorithm of *Dijkstra* (1959) is the canonical method to solve SP and theoretically the most efficient algorithm for solving SSSP (Ahuja et al., 1990). If an heuristic distance function is available, A^* (A-Star) (Hart et al., 1968, 1972) is considered to be the P2PSP algorithm of reference, and the *Algorithm 97*, proposed by Floyd (1962), is commonly used as a benchmark for APSP.

The above mentioned algorithms have been studied in depth and many proposals that improve the performance in terms of time have been made. The improvements can be grouped into three main groups:

(1) algorithms that exploit domain specific information, e.g. use the road network topology to prune the graph (Gutman, 2004; Jing et al., 1998);
(2) algorithms that use specialized data structures, e.g. the *Fibonacci heap* (Fredman and Tarjan, 1987) or the *radix heap* (Ahuja et al., 1990); and
(3) algorithms that rely on the precalculation of variables, e.g. landmarks (Potamias et al., 2009) or distances (Djidjev, 1996).

Despite the great results shown in the literature, in a real world application most of the improvements may not perform as well as expected or even not work at all in practice, because they focus mainly on improving the computation of the SP, while the data availability is took for granted. Thus, efficiently managing the data to comply with memory restrictions is a fundamental task (Miler et al., 2014).

2.2. Road map data management

There are several proposals for efficiently managing the data to solve the SP problem. For example, in 2011, Efentakis et al. (2011) proposed to partition a map by the topological characteristics, particularly the road hierarchies. By implementing a system that relies on database (DB) storage they showed that the time for computing P2PSP is improved when a map is partitioned.

The emergence of social networks and the need for analyzing the relationships among people led to the specification of specialized DB systems for managing graphs: the *Graph Database* System (GDB). GDB systems implement customized data structures to store graphs, as well as optimized implementations of SP algorithms. Two of the most popular GDB implementations are PgRouting,[5] an open source implementation that runs on top of PostreSQL, and Teradata Aster SQL-GR,[6] a world-class business implementation. Despite the specificity of GDB systems, Miler et al. (2014) showed that "graph database management systems [...] are not suitable for full graph traversal purposes", and proved that in-memory SP computation outperforms GDB routing, but with the handicap of the memory constraint.

Using a different paradigm, Aridhi et al. (2015) proposed in 2015 a strategy for computing P2PSP based on *MapReduce*. The strategy "provides high quality solutions in acceptable computational time", but does not guarantee optimality.

More recently, in 2017 Camero et al. (2017) proposed the *Tile Map Partitioning*, a logical geographic-based partitioning of a road map. By testing the proposal over a real world map, they showed performance improvements in terms of time, without imposing memory restrictions.

Considering the advances made up to this date, and the need for a strategy that minimizes the processing time (computing the SP) in a constrained scenario, where is not feasible to load the whole road map into main memory, we propose to extend the Tile Map Partitioning strategy (Camero et al., 2017) by adding heterogeneous (different size) tiles in order to improve the adaptation of the partitioning to the density of the road map (i.e. the number of intersections in an area).

[5] http://pgrouting.org/.
[6] http://www.teradata.com.es/SQL-GR-Engine.