SALT. A Unified Framework for All Shortest-Path Query Variants on Road Networks

Alexandros Efentakis\(^1\(\odot\)), Dieter Pfoser\(^2\), and Yannis Vassiliou\(^3\)

\(^1\) Research Center “Athena”, Marousi, Greece  
efentakis@imis.athena-innovation.gr  
\(^2\) Department of Geography and GeoInformation Science,  
George Mason University, Fairfax, US  
dpfoser@gmu.edu  
\(^3\) National Technical University of Athens, Zografou, Greece  
yv@cs.ntua.gr

Abstract. Although recent scientific literature focuses on multiple shortest-path (SP) problem definitions for road networks, none of the existing solutions can efficiently answer all the different SP query variations. This work proposes SALT, a novel framework that not only efficiently answers most SP queries but also \(k\)-nearest neighbor queries not tackled by previous methods. Our solution offers excellent query performance and very short preprocessing times, thus making it also a viable option for dynamic, live-traffic road networks and all types of practical use-cases. The proposed SALT framework is a deployable software solution capturing a range of graph-related query problems under one “algorithmic hood”.

Keywords: Shortest-paths \· \(k\)-nearest neighbors \· \(k\)NN \· Salt framework

1 Introduction

During the last decades, recent scientific literature has produced efficient methods for shortest-path (SP) queries on road networks (cf. [1] for the latest overview). Unfortunately, most aforementioned algorithms are tuned to solving a specific problem efficiently, but are rather inefficient when used in a different context. Contrarily, engineering a framework that efficiently solves multiple shortest-path problems, would be the first step towards the direction of a grand unified SP toolkit. To this end, the GRASP algorithms [11], solve most variants of the single-source shortest-path problems on road networks, including one-to-all (finding SP distances from a source vertex \(s\) to all other vertices), one-to-many (computing the SP distances between the source vertex \(s\) and a set of target vertices \(T\)) and range queries (find all vertices reachable from \(s\) within a given timespan). GRASP requires minimal preprocessing and provides excellent query performance needed in the context of practical and commercial applications.
Another fundamental problem frequently encountered in location-based services is the \(k\)NN query, i.e., given a query location and a set of objects on the road network, the \(k\)NN search finds the \(k\)-nearest objects to the query location. Unfortunately, even the latest work of [21] is not scalable with the network size, since it requires several hours for preprocessing continental road networks. In addition, for a large number of randomly distributed objects, an efficient Dijkstra implementation could answer \(k\)NN queries by settling a few hundreds nodes and requiring \(< 1ms\). Moreover, most previous methods require a target-selection phase, i.e., they need to mark the objects location within the underlying index. This phase requires a few seconds, hence having limited appeal for applications involving moving objects (e.g., vehicles). Therefore, it only makes sense to use a complex (non-Dijkstra) \(k\)NN processing framework in cases of either rather “small” numbers of objects or objects following skewed distributions (e.g., POIs located near the city center), i.e., for cases in which Dijkstra does not perform well.

The contribution of this work is to provide a unified algorithmic solution that may be used in a dynamic road network context, while covering a wide range of shortest-path problems, such as (i) single-pair, (ii) one-to-all, (iii) one-to-many, (iv) range and (v) \(k\)NN queries. Specifically, we aim at combining the fragmented approaches related to the various shortest-path problem definitions and instead propose a unified framework that tackles all of them. Our proposed SALT (graph Separators + ALT) framework requires seconds for preprocessing continental road networks and provides excellent query performance for a wide range of problems. We will show that SALT is (i) \(3-4\times\) faster for point-to-point queries when compared to existing methods of similar preprocessing times, (ii) it answers one-to-all, one-to-many and range queries with comparable performance to state-of-the-art approaches, and most importantly, (iii) it may also answer \(k\)NN queries in \(<1ms\), for both, static or moving objects. As such, our SALT framework could be a swiss-army-knife for tackling all shortest-path problem variants, making it a serious contender for use in commercial applications.

The outline of this work is as follows. Section 2 describes previous related work. Section 3 describes our novel SALT framework and algorithms. Experiments establishing SALT’s benefits are provided in Section 4 and Section 5 concludes the paper.

2 Related Work

Throughout this work, we use directed weighted graphs \(G(V, E, w)\), where \(V\) is the set of vertices, \(E \subseteq V \times V\) is the set the arcs and \(w\) is a positive weight function \(E \rightarrow R^+\). The reverse graph \(\overline{G} = (V, E)\) is the graph obtained from \(G\) by substituting each arc \((u, v) \in E\) by \((v, u)\). A partition of \(V\) is a family of sets \(C = \{c_0, c_1, \ldots, c_M\}\), such that each node \(u \in V\) is contained in exactly one set \(c_i\). An element of a partition is called a cell. A multilevel partition of \(V\) is a family of partitions \(\{C^0, C^1, \ldots, C^L\}\) where \(\ell\) denotes the level of a partition \(C^\ell\). Similar to [4], level 0 refers to the original graph, \(L\) is the highest partition