- ► Sorting by Transpositions and Reversals (Approx Ratio 1.5)
- ► Substring Parsimony

Recommended Reading

1. Bader, D.A., Moret, B.M.E., Warnow, T., Wyman, S.K., Yan, M.: High-performance algorithm engineering for gene-order phylogenies. In: DIMACS Workshop on Whole Genome Comparison, Rutgers University, Piscataway, NJ (2001)
2. Bader, D.A., Moret, B.M.E., Vawter, L.: Industrial applications of high-performance computing for phylogeny reconstruction. In: Siegel, H.J. (ed.) Proc. SPIE Commercial Applications for High-Performance Computing, vol. 4528, pp. 159–168, Denver, CO (2001)
3. Bader, D.A., Moret, B.M.E., Yan, M.: A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. J. Comp. Biol. **8**(5), 483–491 (2001)
4. Farris, J.S.: The logical basis of phylogenetic analysis. In: Platnick, N.I., Funk, V.A. (eds.) Advances in Cladistics, pp. 1–36. Columbia Univ. Press, New York (1983)
5. Felsenstein, J.: Evolutionary trees from DNA sequences: a maximum likelihood approach. J. Mol. Evol. **17**, 368–376 (1981)
6. Moret, B.M.E., Bader, D.A., Warnow, T., Wyman, S.K., Yan, M.: GRAPPA: a highperformance computational tool for phylogeny reconstruction from gene-order data. In: Proc. Botany, Albuquerque, August 2001
7. Moret, B.M.E., Bader, D.A., Warnow, T.: High-performance algorithm engineering for computational phylogenetics. J. Supercomp. **22**, 99–111 (2002) Special issue on the best papers from ICCS'01
8. Moret, B.M.E., Wyman, S., Bader, D.A., Warnow, T., Yan, M.: A new implementation and detailed study of breakpoint analysis. In: Proc. 6th Pacific Symp. Biocomputing (PSB 2001), pp. 583–594, Hawaii, January 2001
9. Saitou, N., Nei, M.: The neighbor-joining method: A new method for reconstruction of phylogenetic trees. Mol. Biol. Evol. **4**, 406–425 (1987)
10. Sankoff, D., Blanchette, M.: Multiple genome rearrangement and breakpoint phylogeny. J. Comp. Biol. **5**, 555–570 (1998)
11. Yan, M.: High Performance Algorithms for Phylogeny Reconstruction with Maximum Parsimony. Ph. D. thesis, Electrical and Computer Engineering Department, University of New Mexico, Albuquerque, January 2004

Engineering Algorithms for Large Network Applications

2002; Schulz, Wagner, Zaroliagis

CHRISTOS ZAROLIAGIS
Department of Computer Engineering & Informatics, University of Patras, Patras, Greece

Problem Definition

Dealing effectively with applications in large networks, it typically requires the efficient solution of one ore more underlying algorithmic problems. Due to the size of the network, a considerable effort is inevitable in order to achieve the desired efficiency in the algorithm.

One of the primary tasks in large network applications is to answer queries for finding best routes or paths as efficiently as possible. Quite often, the challenge is to process a vast number of such queries on-line: a typical situation encountered in several real-time applications (e. g., traffic information systems, public transportation systems) concerns a query-intensive scenario, where a central server has to answer a huge number of on-line customer queries asking for their best routes (or optimal itineraries). The main goal in such an application is to reduce the (average) response time for a query.

Answering a best route (or optimal itinerary) query translates in computing a minimum cost (shortest) path on a suitably defined directed graph (digraph) with non-negative edge costs. This in turn implies that the core algorithmic problem underlying the efficient answering of queries is the single-source single-target shortest path problem.

Although the straightforward approach of pre-computing and storing shortest paths for all pairs of vertices would enabling the optimal answering of shortest path queries, the quadratic space requirements for digraphs with more than 10^5 vertices makes such an approach prohibitive for large and very large networks. For this reason, the main goal of almost all known approaches is to keep the space requirements as small as possible. This in turn implies that one can afford a heavy (in time) preprocessing, which does not blow up space, in order to speed-up the query time.

The most commonly used approach for answering shortest path queries employs Dijkstra's algorithm and/or variants of it. Consequently, the main challenge is how to reduce the algorithm's *search-space* (number of vertices visited), as this would immediately yield a better query time.

Key Results

All results discussed concern answering of *optimal* (or *exact* or *distance-preserving*) shortest paths under the aforementioned query-intensive scenario, and are all based on the following generic approach. A preprocessing of the input network $G = (V, E)$ takes place that results in a data structure of size $O(|V| + |E|)$ (i. e., linear to the size of G). The data structure contains additional information regarding certain shortest paths that can be used later during querying.

Depending on the pre-computed additional information as well as on the way a shortest path query is answered, two approaches can be distinguished. In the first approach, *graph annotation*, the additional information is attached to vertices or edges of the graph. Then, speed-up techniques to Dijkstra's algorithm are employed that, based on this information, decide quickly which part of the graph does not need to be searched. In the second approach, an *auxiliary graph* G' is constructed hierarchically. A shortest path query is then answered by searching only a small part of G', using Dijkstra's algorithm enhanced with heuristics to further speed-up the query time.

In the following, the key results of the first [3,4,9,11] and the second approach [1,2,5,7,8,10] are discussed, as well as results concerning modeling issues.

First Approach – Graph Annotation

The first work under this approach concerns the study in [9] on large railway networks. In that paper, two new heuristics are introduced: the *angle-restriction* (that tries to reduce the search space by taking advantage of the geometric layout of the vertices) and the *selection of stations* (a subset of vertices is selected among which all pairs shortest paths are pre-computed). These two heuristics along with a combination of the classical *goal-directed* or A^* *search* turned out to be rather efficient. Moreover, they motivated two important generalizations [10,11] that gave further improvements to shortest path query times.

The full exploitation of geometry-based heuristics was investigated in [11], where both street and railway networks are considered. In that paper, it is shown that the search space of Dijkstra's algorithm can be significantly reduced (to 5%–10% of the initial graph size) by extracting geometric information from a given layout of the graph and by encapsulating pre-computed shortest path information in resulted geometric objects, called *containers*. Moreover, the dynamic case of the problem was investigated, where edge costs are subject to change and the geometric containers have to be updated.

A powerful modification to the classical Dijkstra's algorithm, called *reach-based routing*, was presented in [4]. Every vertex is assigned a so-called *reach value* that determines whether a particular vertex will be considered during Dijkstra's algorithm. A vertex is excluded from consideration if its reach value is small; that is, if it does not contribute to any path long enough to be of use for the current query.

A considerable enhancement of the classical A^* *search* algorithm using landmarks (selected vertices like in [9,10]) and the triangle inequality with respect to the shortest path distances was shown in [3]. Landmarks and triangle inequality help to provide better lower bounds and hence boost A^* search.

Second Approach – Auxiliary Graph

The first work under this approach concerns the study in [10], where a new hierarchical decomposition technique is introduced called *multi-level graph*. A multi-level graph $\mathcal{M}$ is a digraph which is determined by a sequence of subsets of V and which extends E by adding multiple levels of edges. This allows to efficiently construct, during querying, a subgraph of $\mathcal{M}$ which is substantially smaller than G and in which the shortest path distance between any of its vertices is equal to the shortest path distance between the same vertices in G. Further improvements of this approach have been presented recently in [1]. A refinement of the above idea was introduced in [5], where the multi-level overlay graphs are introduced. In such a graph, the decomposition hierarchy is not determined by application-specific information as it happens in [9,10].

An alternative hierarchical decomposition technique, called *highway hierarchies*, was presented in [7]. The approach takes advantage of the inherent hierarchy possessed by real-world road networks and computes a hierarchy of coarser views of the input graph. Then, the shortest path query algorithm considers mainly the (much smaller in size) coarser views, thus achieving dramatic speed-ups in query time. A revision and improvement of this method was given in [8]. A powerful combination of the highway hierarchies with the ideas in [3] was reported in [2].

Modeling Issues

The modeling of the original best route (or optimal itinerary) problem on a large network to a shortest path problem in a suitably defined directed graph with appropriate edge costs also plays a significant role in reducing the query time. Modeling issues are thoroughly investigated in [6]. In that paper, the first experimental comparison of two important approaches (time-expanded versus time-dependent) is carried out, along with new extensions of them towards realistic modeling. In addition, several new heuristics are introduced to speed-up query time.

Applications

Answering shortest path queries in large graphs has a multitude of applications, especially in traffic information systems under the aforementioned scenario; that is, a central server has to answer, as fast as possible, a huge number of on-line customer queries asking for their best routes or itineraries. Other applications of the above scenario

involve route planning systems for cars, bikes and hikers, public transport systems for itinerary information of scheduled vehicles (like trains or buses), answering queries in spatial databases, and web searching. All the above applications concern real-time systems in which users continuously enter their requests for finding their best connections or routes. Hence, the main goal is to reduce the (average) response time for answering a query.

Open Problems

Real-world networks increase constantly in size either as a result of accumulation of more and more information on them, or as a result of the digital convergence of media services, communication networks, and devices. This scaling-up of networks makes the scalability of the underlying algorithms questionable. As the networks continue to grow, there will be a constant need for designing faster algorithms to support core algorithmic problems.

Experimental Results

All papers discussed in Sect. "Key Results" contain important experimental studies on the various techniques they investigate.

Data Sets

The data sets used in [6,11] are available from http://lso-compendium.cti.gr/ under problems 26 and 20, respectively.

The data sets used in [1,2] are available from http://www.dis.uniroma1.it/~challenge9/.

URL to Code

The code used in [9] is available from http://doi.acm.org/10.1145/351827.384254.

The code used in [6,11] is available from http://lso-compendium.cti.gr/ under problems 26 and 20, respectively.

The code used in [3] is available from http://www.avglab.com/andrew/soft.html.

Cross References

- ► Implementation Challenge for Shortest Paths
- ► Shortest Paths Approaches for Timetable Information

Recommended Reading

1. Delling, D., Holzer, M., Müller, K., Schulz, F., Wagner, D.: High-Performance Multi-Level Graphs. In: 9th DIMACS Challenge on Shortest Paths, Nov 2006. Rutgers University, USA (2006)
2. Delling, D., Sanders, P., Schultes, D., Wagner, D.: Highway Hierarchies Star. In: 9th DIMACS Challenge on Shortest Paths, Nov 2006 Rutgers University, USA (2006)
3. Goldberg, A.V., Harrelson, C.: Computing the Shortest Path: A^* Search Meets Graph Theory. In: Proc. 16th ACM-SIAM Symposium on Discrete Algorithms – SODA, pp. 156–165. ACM, New York and SIAM, Philadelphia (2005)
4. Gutman, R.: Reach-based Routing: A New Approach to Shortest Path Algorithms Optimized for Road Networks. In: Algorithm Engineering and Experiments – ALENEX (SIAM, 2004), pp. 100–111. SIAM, Philadelphia (2004)
5. Holzer, M., Schulz, F., Wagner, D.: Engineering Multi-Level Overlay Graphs for Shortest-Path Queries. In: Algorithm Engineering and Experiments – ALENEX (SIAM, 2006), pp. 156–170. SIAM, Philadelphia (2006)
6. Pyrga, E., Schulz, F., Wagner, D., Zaroliagis, C.: Efficient Models for Timetable Information in Public Transportation Systems. ACM J. Exp. Algorithmic **12**(2.4), 1–39 (2007)
7. Sanders, P., Schultes, D.: Highway Hierarchies Hasten Exact Shortest Path Queries. In: Algorithms – ESA 2005. Lect. Note Comp. Sci. **3669**, 568–579 (2005)
8. Sanders, P., Schultes, D.: Engineering Highway Hierarchies. In: Algorithms – ESA 2006. Lect. Note Comp. Sci. **4168**, 804–816 (2006)
9. Schulz, F., Wagner, D., Weihe, K.: Dijkstra's Algorithm On-Line: An Empirical Case Study from Public Railroad Transport. ACM J. Exp. Algorithmics **5**(12), 1–23 (2000)
10. Schulz, F., Wagner, D., Zaroliagis, C.: Using Multi-Level Graphs for Timetable Information in Railway Systems. In: Algorithm Engineering and Experiments – ALENEX 2002. Lect. Note Comp. Sci. **2409**, 43–59 (2002)
11. Wagner, D., Willhalm, T., Zaroliagis, C.: Geometric Containers for Efficient Shortest Path Computation. ACM J. Exp. Algorithmics **10**(1.3), 1–30 (2005)

Engineering Geometric Algorithms

2004; Halperin

Dan Halperin
School of Computer Science,
Tel-Aviv University, Tel Aviv, Israel

Keywords and Synonyms

Certified and efficient implementation of geometric algorithms; Geometric computing with certified numerics and topology

Problem Definition

Transforming a theoretical geometric algorithm into an effective computer program abounds with hurdles. Overcoming these difficulties is the concern of *engineering geometric algorithms*, which deals, more generally, with the design and implementation of certified and efficient solutions to algorithmic problems of geometric nature. Typ-