Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Faster shortest paths in dense distance graphs, with applications ☆

Shay Mozes [a], Yahav Nussbaum [b], Oren Weimann [b,*]

[a] IDC Herzliya, Israel
[b] University of Haifa, Israel

ARTICLE INFO

Article history:
Received 23 April 2017
Received in revised form 12 October 2017
Accepted 31 October 2017
Available online xxxx
Communicated by A. Marchetti-Spaccamela

Keywords:
Planar graphs
Shortest paths
Recursive r-divisions
Dynamic range minimum queries
Monge heaps

ABSTRACT

We show how to combine two techniques for efficiently computing shortest paths in directed planar graphs. The first is the linear-time shortest-path algorithm of Henzinger, Klein, Subramanian, and Rao [STOC'94]. The second is Fakcharoenphol and Rao's algorithm [FOCS'01] for emulating Dijkstra's algorithm on the *dense distance graph* (DDG). A DDG is defined for a decomposition of a planar graph G into regions of at most r vertices each, for some parameter $r < n$. The vertex set of the DDG is the set of $\Theta(nr^{-1/2})$ vertices of G that belong to more than one region (boundary vertices). The DDG has $\Theta(n)$ arcs, such that distances in the DDG are equal to the distances in G. Fakcharoenphol and Rao's implementation of Dijkstra's algorithm on the DDG (nicknamed *FR-Dijkstra*) runs in $O(n\log(n)r^{-1/2}\log r)$ time, and is a key component in many state-of-the-art planar graph algorithms for shortest paths, minimum cuts, and maximum flows. By combining these two techniques we remove the $\log n$ dependency in the running time of the shortest-path algorithm at the price of an additional $\log r$ factor, making it $O(nr^{-1/2}\log^2 r)$.
This work is part of a research agenda that aims to develop new techniques that would lead to faster, possibly linear-time, algorithms for problems in planar graphs such as minimum-cut, maximum-flow, and shortest paths with negative arc lengths. As immediate applications, we show how to compute maximum flow in directed weighted planar graphs in $O(n\log p)$ time, and minimum st-cut in undirected weighted planar graphs in $O(n\log\log p)$ time, where p is the minimum number of edges on any path from the source to the sink. We also show how to compute any part of the DDG that corresponds to a region with r vertices and k boundary vertices in $O(r\log k)$ time, which is faster than has been previously known for small values of k.

1. Introduction

Finding shortest paths and maximum flows are among the most fundamental optimization problems in graph theory. On planar graphs, these problems are intimately related and can all be solved in linear or near-linear time. Obtaining strictly-linear time algorithms for these problems is one of the main current goals of the planar graphs community [11, 22]. Some of these problems are known to be solvable in linear time (minimum spanning tree [33], shortest-paths with

☆ The research was supported in part by Israel Science Foundation grants 794/13 and 592/17.
* Corresponding author.
E-mail addresses: smozes@idc.ac.il (S. Mozes), yahav.nussbaum@cs.tau.ac.il (Y. Nussbaum), oren@cs.haifa.ac.il (O. Weimann).

https://doi.org/10.1016/j.tcs.2017.10.034
0304-3975/

non-negative lengths [20], maximum flow with unit capacities [11], undirected unweighted global min-cut [9]), but for many others, only nearly-linear time algorithms are known. These include shortest-paths with negative lengths [13,31,36], multiple-source shortest-paths [6,7,28], max-flow/min-st-cut [3–5,12,22], and global min-cut [8,32,34].

Many of the results mentioned above were achieved fairly recently, along with the development of more sophisticated shortest paths techniques in planar graphs. In this paper we show how to combine two of these techniques: the technique of Henzinger, Klein, Rao, and Subramanian for computing shortest paths with non-negative lengths in linear time [20], and the technique of Fakcharoenphol and Rao to compute shortest paths on dense distance graphs in nearly linear time in the number of vertices [13].

The *dense distance graph* (DDG) is an important element in many of the algorithms mentioned above. It is a non-planar graph that represents (exactly) distances among a subset of the vertices of the underlying planar graph G. More precisely, an r-division [14] of an n-vertex planar graph G, for some $r < n$, is a division of G (i.e., a partition of G's edge-set) into $O(n/r)$ subgraphs (called *regions*) $\{G_i\}$, where each region has at most r vertices and $O(\sqrt{r})$ *boundary* vertices (vertices that the region shares with other regions). There exist r-divisions of planar graphs with the additional property that the boundary vertices in each region lie on a constant number of faces (called *holes*) [39,13,1,30]. Another property that we can obtain is that each boundary vertex belongs to a constant number of regions. This can be achieved by first ensuring that each vertex of G has constant degree (say by triangulating the dual graph with zero-length edges) and then using the algorithm of [30].

Consider an r-division of G for some $r < n$. Let K_i be the complete graph on the $O(\sqrt{r})$ boundary vertices of the region G_i, where the length of arc uv is the u-to-v distance in G_i. The graph K_i is called the DDG of G_i. The union $\bigcup_i K_i$ is called the DDG of G (or more precisely, the DDG of the given r-division of G).[1]

DDGs are useful for three main reasons. First, distances in the DDG of G are the same as distances in G. Second, it is possible (FR-Dijkstra [13]) to compute shortest paths in DDGs in time that is nearly linear in the number of vertices of the DDG (i.e., in sublinear time in the number of vertices of G). Finally, DDGs can be computed in nearly linear time either by invoking FR-Dijkstra recursively, or by using a multiple source shortest-path algorithm [7,28] (MSSP). Until the current work, the latter method was faster in all cases.

Since it was introduced in 2001, FR-Dijkstra has been used creatively as a subroutine in many algorithms. These include algorithms for shortest paths with negative lengths [13], maximum flows and minimum cuts [4,5,22,32], distance oracles [6,13,24,28,35,37], and DDG constructions [13]. Improving FR-Dijkstra is therefore an important task with implications to all these problems. For example, consider the minimum st-cut problem in undirected planar graphs. Italiano et al. [22] gave an $O(n \log\log n)$ algorithm for the problem, improving the $O(n \log n)$ algorithms of Frederickson [14] and of Reif [38] (using [20]). Three of the techniques used by Italiano et al. are: (i) constructing an r-division with $r = \text{polylog}(n)$ in $O(n \log r)$ time, (ii) FR-Dijkstra, and (iii) constructing the DDG in $O(n \log r)$ time. In a step towards a linear time algorithm for this fundamental problem, Arge et al. [1], and independently Klein et al. [30] gave an $O(n)$ algorithm for constructing an r-division with few holes.[2] This leaves the construction of the DDG as the only current bottleneck for obtaining min-st-cut in $O(n)$ time. The work described in the current paper is motivated by the desire to obtain such a linear time algorithm, and partially addresses techniques (ii) and (iii). It improves the running time of FR-Dijkstra, and, as a consequence, implies faster DDG construction, although for a limited case which is not the one required in [22].

The linear time algorithm for shortest-paths with nonnegative lengths in planar graphs of Henzinger, Klein, Rao, and Subramanian [20] (HKRS) is another landmark result that has been used in many subsequent algorithms. HKRS [20] differs from Dijkstra's algorithm in that the order in which it relaxes the arcs is not determined by the vertex with the current globally minimum label. Instead, it works with a recursive division of the input graph into regions. It handles a region for a limited time period, and then skips to other regions. Within this time period it determines the order of relaxations according to the vertex with minimum label in the *current* region, thus avoiding the need to perform many operations on a large heap. Planarity, or more precisely the existence of small recursive separators, guarantees that local relaxations have limited global effects. Therefore, even though some arcs are relaxed by HKRS more than once, the overall running time can be shown (by a fairly complicated argument) to be $O(n)$. Even though HKRS has been introduced roughly 20 years ago and has been used in many other algorithms, to the best of our knowledge, and unlike other important planarity exploiting techniques, it has always been used as a black box, and was not modified or extended prior to the current work.[3]

1.1. Our results

By combining the framework of Henzinger et al. (HKRS) with a modification of the internal building blocks of Fakcharoenphol and Rao's Dijkstra implementation (FR-Dijkstra), we obtain a faster algorithm for computing shortest paths on dense distance graphs. On the conceptual level, this work is the first to suggest and achieve a combination of these two powerful

[1] DDGs are similarly defined for decompositions which are not r-divisions, but our algorithm does not necessarily apply in such cases.

[2] The algorithm in [30] produces a recursive r-division with few holes (i.e., one in which each region is further divided recursively with an r–division until constant sized regions are achieved) in linear time, whereas the one in [1] does not. A recursive r-division is used in this paper and may be essential for obtaining linear time r–division based algorithms.

[3] Tazari and Müller-Hannemann [40] extended [20] to minor-closed graph classes, but that extension uses the algorithm of [20] without change. It deals with the issue of achieving constant degree in minor-closed classes of graphs, which was overlooked in [20].

Please cite this article in press as: S. Mozes et al., Faster shortest paths in dense distance graphs, with applications, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2017.10.034