Data Structures and Dynamic Algorithms for Planar Graphs

Summary of the PhD Dissertation

Adam Karczmarz

Institute of Informatics, University of Warsaw

1 Introduction

Obtaining provably efficient algorithms for the most basic graph problems like finding (shortest) paths or computing maximum matchings, fast enough to handle real-world-scale graphs (i.e., consisting of millions of vertices and edges), is a very challenging task. For example, in a very general regime of strongly-polynomial algorithms (see, e.g., [65]), we still do not know how to compute shortest paths in a real-weighted sparse directed graph significantly faster than in quadratic time, using the classical, but somewhat simple-minded, Bellman-Ford method.

One way to circumvent this problem is to consider more restricted computation models for graph algorithms. If, for example, we restrict ourselves to graphs with integral edge weights, we can improve upon the Bellman-Ford algorithm [14, 31]. Although these results are very deep algorithmically, their theoretical efficiency is still very far from the only known trivial linear lower bound on the actual time complexity of the negatively-weighted shortest path problem.

Another approach is to develop algorithms specialized for certain graph classes that appear in practice. Planar graphs constitute one of the most important and well-studied such classes. Many of the real-world networks can be drawn on a plane with no or few edge crossings. The examples include not very complex road networks and graphs considered in the domain of VLSI design. Complex road networks, although far from being planar, share with planar graphs some useful properties, like the existence of small separators [20]. Special cases of planar graphs, such as grids, appear often in the area of image processing (e.g., [7]).

And indeed, if we restrict ourselves to planar graphs, many of the classical polynomial-time graph problems, in particular computing shortest paths [35, 58] and maximum flows [4, 5, 21] in real-weighted graphs, can be solved either optimally or in nearly-linear time. The very rich combinatorial structure of planar graphs often allows breaking barriers that appear in the respective problems for general graphs by using techniques from computational geometry (e.g., [27]), or by applying sophisticated data structures, such as dynamic trees [4, 10, 21, 66].

In this thesis, we focus on the data-structural aspect of planar graph algorithmics. By this, we mean that rather than concentrating on particular planar graph problems, we study more abstract, "low-level" problems. Efficient algorithms for these problems can be used in a black-box manner to design algorithms for multiple specific problems at once. Such an approach allows us to improve upon many known complexity upper bounds for different planar graph problems simultaneously, without going into the specifics of these problems.

We also study *dynamic algorithms* for planar graphs, i.e., algorithms that maintain certain information about a dynamically changing graph (such as "is the graph connected?") much more efficiently than by recomputing this information from scratch after each update. We consider the *edge-update* model where the input graph can be modified only by adding or removing

single edges. A graph algorithm is called *fully-dynamic* if it supports both edge insertions and edge deletions, and *partially dynamic* if it supports either only edge insertions (then we call it *incremental*) or only edge deletions (then it is called *decremental*).

When designing dynamic graph algorithms, we care about the *update time*, i.e., the time needed by the algorithm to adapt to an elementary change of the graph, and *query time*, i.e., the time needed by the algorithm to recompute the requested portion of the maintained information. Sometimes, especially in partially dynamic settings, it is more convenient to measure the *total update time*, i.e., the total time needed by the algorithm to process any possible sequence of updates. For some dynamic problems, it is worth focusing on a more restricted *explicit maintenance* model where the entire maintained information is explicitly updated (so that the user is notified about the update) after each change. In this model the query procedure is trivial and thus we only care about the update time.

Note that there is actually no clear distinction between dynamic graph algorithms and graph data structures, since dynamic algorithms are often used as black-boxes to obtain efficient *static* algorithms (e.g., [26]). For example, the *incremental connectivity problem*, where one needs to process queries about the existence of a path between given vertices, while the input undirected graph undergoes edge insertions, is actually equivalent to the *disjoint-set data structure* problem, also called the *union-find data structure* problem (see, e.g., [15]).

We concentrate mostly on the decremental model and obtain very efficient decremental algorithms for problems on unweighted planar graphs related to reachability and connectivity. We also apply our dynamic algorithms to static problems, thus confirming once again the data-structural character of these results.

In the following, let $G = (V, E)$ denote the input planar graph with n vertices. For clarity of this summary, assume G is a simple graph. Then, by planarity, it has $O(n)$ edges. When we talk about general graphs, we denote by m the number of edges of the graph.

2 Contracting a Planar Graph

The first part of the thesis is devoted to the data-structural aspect of contracting edges in planar graphs. Edge contraction is one of the fundamental graph operations. Given an undirected graph and its edge e, contracting the edge e consists in removing it from the graph and merging its endpoints. The notion of contraction has been used to describe a number of prominent graph algorithms, including Edmonds' algorithm for computing maximum matchings [19], or Karger's minimum cut algorithm [44].

Edge contractions are of particular interest in planar graphs, as a number of planar graph properties can be described using contractions. For example, it is well-known that a graph is planar precisely when it cannot be transformed into K_5 or $K_{3,3}$ by contracting edges, or removing vertices or edges (see e.g., [17]). Moreover, contracting an edge preserves planarity.

We would like to have at our disposal a data structure that performs contractions on the input planar graph and still provides access to the most basic information about our graph, such as the sizes of neighbors sets of individual vertices and the adjacency relation. While contraction operation is conceptually very simple, its efficient implementation is challenging. This is because it is not clear how to represent individual vertices' adjacency lists so that adjacency list merges, adjacency queries, and neighborhood size queries are all efficient. By using standard data structures (e.g., balanced binary search trees), one can maintain adjacency lists of a graph subject to contractions in polylogarithmic amortized time. However, in many planar graph algorithms this becomes a bottleneck.

As an example, consider the problem of computing a 5-coloring of a planar graph. There exists a very simple algorithm based on contractions [53] that only relies on a folklore fact that

a planar graph has a vertex of degree no more than 5. However, linear-time algorithms solving this problem use some more involved planar graph properties [23, 53, 60]. For example, the algorithm by Matula et al. [53] uses the fact that every planar graph has either a vertex of degree at most 4 or a vertex of degree 5 adjacent to at least four vertices, each having degree at most 11. Similarly, although there exists a very simple algorithm for computing a minimum spanning tree of a planar graph based on edge contractions, various different methods have been used to implement it efficiently [23, 51, 52].

The problem of maintaining a planar graph under contractions has been studied before. In their book, Klein and Mozes [46] showed that there exists a (a bit more general) data structure maintaining a planar graph under edge contractions and deletions, and answering adjacency queries in $O(1)$ worst-case time. The update time is $O(\log n)$. This result is based on the work of Brodal and Fagerberg [8], who showed how to maintain a bounded-outdegree orientation of a dynamic planar graph so that the edge set updates are supported in $O(\log n)$ amortized time.

Gustedt [32] showed an optimal solution to the union-find problem in the case when at any time the actual subsets form disjoint and connected subgraphs of a given planar graph G. In other words, in this problem the allowed unions correspond to the edges of a planar graph and the execution of a union operation can be seen as a contraction of the respective edge.

Our Results

We show a data structure that can efficiently maintain a planar graph subject to edge contractions in linear total time, assuming the standard word-RAM model with word size $\Omega(\log n)$. It can report groups of parallel edges and self-loops that emerge. It also supports constant-time adjacency queries and maintains the neighbor lists and degrees explicitly. The data structure can be used as a black-box to implement planar graph algorithms that use contractions.

As an example, our data structure can be used to give clean and conceptually simple linear-time implementations of algorithms for computing 5-coloring or minimum spanning tree.

More importantly, by using our data structure, we give improved algorithms for a few problems in planar graphs. In particular, we obtain optimal algorithms for decremental 2-edge-connectivity (see, e.g., [30]), finding a unique perfect matching [26], and computing maximal 3-edge-connected subgraphs [12].

In order to obtain our result, we first partition the graph into small pieces of roughly logarithmic size (using so-called *r-divisions* [24]). Then we solve our problem recursively for each of the pieces, and separately using a simple-minded approach for the subgraph induced by $o(n)$ vertices contained in multiple pieces (the so-called boundary vertices). Such an approach proved successful in obtaining optimal data structures for the planar union-find problem [32] and decremental connectivity [50]. In fact, our data-structural problem can be seen as a generalization of the former problem. However, maintaining the status of each edge e of the initial graph G (i.e., whether e has become a self-loop or a parallel edge) subject to edge contractions, and supporting constant-time adjacency queries without resorting to randomization, turn out to be serious technical challenges. Overcoming these difficulties is our main contribution of this part of the thesis.

3 Decremental Reachability

The second part of this thesis is devoted to dynamic reachability problems in planar graphs. In the *dynamic reachability* problem we are given a (directed) graph G subject to edge updates and the goal is to design a data structure that would allow answering queries about the existence of a path between a pair of query vertices $u, v \in V$.

Two variants of dynamic reachability are studied most often. In the *all-pairs* variant, our data structure has to support queries between arbitrary pairs of vertices. This variant is also called the *dynamic transitive closure* problem, since a path $u \to v$ exists in G if uv is an edge of the transitive closure of G.

In the *single-source reachability* problem, a source vertex $s \in V$ is fixed from the very beginning and the only allowed queries are about the existence of a path $s \to v$, where $v \in V$.

If we work with undirected graphs, the dynamic reachability problem is called the *dynamic connectivity* problem. Note that in the undirected case a path $u \to v$ exists in G if and only if a path $v \to u$ exists in G.

State of the Art

Dynamic reachability in general directed graphs turns out to be a very challenging problem. First of all, it is computationally much more demanding than its undirected counterpart. For undirected graphs, fully-dynamic all-pairs algorithms with polylogarithmic amortized update and query bounds are known [36, 38, 71]. For directed graphs, on the other hand, in most settings (either single-source or all-pairs, either incremental, decremental or fully-dynamic) the best known algorithm has either polynomial update time or polynomial query time. The only exception is the incremental single-source reachability problem, for which a trivial extension of depth-first search [68] achieves $O(1)$ amortized update time.

One of the possible reasons behind such a big gap between the undirected and directed settings is that one needs only linear time to compute the connected components of an undirected graph, and thus there exists a $O(n)$-space *static* data structure that can answer connectivity queries in undirected graphs in $O(1)$ time. On the other hand, the best known algorithm for computing the transitive closure runs in $\widetilde{O}(\min(n^\omega, nm)) = \widetilde{O}(n^2)$[1] time [11, 59].

So far, the best known bounds for fully-dynamic reachability are as follows. For dynamic transitive closure, there exist a number of algorithms with $O(n^2)$ update time and $O(1)$ query time [16, 61, 64]. These algorithms, in fact, maintain the transitive closure explicitly. There also exist a few fully-dynamic algorithms that are better for sparse graphs, each of which has $\Omega(n)$ amortized update time and query time which is $o(n)$ but still polynomial in n [62, 63, 64]. For the single-source variant, the only known non-trivial (i.e., other than recompute-from-scratch) algorithm has $O(n^{1.53})$ update time and $O(1)$ query time [64].

Algorithms with $O(nm)$ total update time are known for both incremental [39] and decremental [48, 62] transitive closure. Note that for sparse graphs this bound is only poly-logarithmic factors away from the best known static transitive closure upper bound [11].

All the known partially-dynamic single-source reachability algorithms work in the explicit maintenance model. As mentioned before, for incremental single-source reachability, an optimal (in the amortized sense) algorithm is known. Interestingly, the first algorithms with $O(mn^{1-\epsilon})$ total update time (where $\epsilon > 0$) have been obtained only recently [33, 34]. The best known algorithm to date has $\widetilde{O}(m\sqrt{n})$ total update time and is due to Chechik et al. [13].

Dynamic reachability has also been previously studied for planar graphs. Diks and Sankowski [18] showed a fully-dynamic transitive closure algorithm with $\widetilde{O}(\sqrt{n})$ update and query times, which works under the assumption that the graph is plane embedded and the inserted edges can only connect vertices sharing some adjacent face. Łącki [48] showed that one can maintain the strongly connected components of a planar graph under edge deletions in $O(n\sqrt{n})$ total time. By known reductions, it follows that there exists a decremental single-source reachability algorithm for planar graphs with $O(n\sqrt{n})$ total update time. Note that this bound matches the recent best known bound for general graphs [13] up to polylogarithmic factors.

[1] We denote by $\widetilde{O}(f(n))$ the order $O(f(n)\operatorname{polylog} n)$.

Our Results

We show new decremental reachability algorithms for planar digraphs.

For decremental single-source reachability, we obtain an almost optimal (up to polylogarithmic factors) algorithm explicitly maintaining the set of vertices reachable from the source. Our algorithm processes any online sequence of edge deletions in $O(n \log^2 n \log\log n)$ total time. This seems to be the first dynamic algorithm with polylogarithmic amortized update time on general directed planar graphs that solves a non-trivial reachability-type problem for which only polynomial bounds are known in general graphs.

For decremental transitive closure, we obtain a randomized trade-off algorithm. For any chosen $t \in [1, n]$, our algorithm has $\widetilde{O}(n^2/t)$ total update time and $\widetilde{O}(\sqrt{t})$ query time. In particular, for $t = n$, our algorithm has polylogarithmic amortized update time and $\widetilde{O}(\sqrt{n})$ time. For $t = n^{2/3}$, we obtain an algorithm with $\widetilde{O}(n^{1/3})$ update and query time. To the best of our knowledge, this is the first dynamic algorithm for general planar digraphs answering arbitrary point-to-point queries with $O(n^{1/2-\epsilon})$ update and query bounds.

As a byproduct, we also obtain nearly-linear time planar algorithms for a few static and decremental reachability-related problems that have been previously studied only for general graphs, and for which no nearly-linear algorithms have been known. These include computing maximal 2-edge-connected subgraphs of a directed graph [12] and decremental maintenance of the set of so-called strong bridges of the graph [29].

Technical Overview

In all our reachability-related algorithms we first construct a *recursive decomposition* of the initial graph G. A recursive decomposition is a tree-like hierarchy of subgraphs of a graph G (pieces) built by recursively partitioning G with $O(\sqrt{n})$-size cycle separators [54]. Such decompositions proved very useful in obtaining nearly-linear time static algorithms for planar graphs (e.g., [3, 6, 49]), as well as dynamic planar graph algorithms (e.g., [18]).

Subsequently, for each piece H of the decomposition, we explicitly maintain only a certain part of the transitive closure of subgraph H, induced by its *boundary vertices* ∂H. These are defined as the vertices that H shares with the remaining part $G - H$ of G. This way, the total amount of information we store is nearly-linear in n. We prove that these parts of transitive closures can be computed inductively based on the transitive closures stored in the child pieces of H. Therefore, the needed information can be *maintained* inductively using a decremental transitive closure algorithm run on the corresponding data accompanying the child pieces of H.

In order to obtain an efficient algorithm maintaining the needed reachability information dynamically, as described above, we analyze and exploit the structural properties of a reachability matrix of a set of vertices[2] that, roughly speaking, lie on a constant number of faces of a plane graph (which is a property satisfied by vertices ∂H of each piece H in our decomposition). The matrix viewpoint allows us to obtain properties that are algorithmically useful and otherwise not easy to capture using the previously used *separating path* approach to reachability in planar digraphs [18, 67, 69]. Subsequently, we show that these properties can be used to simulate the randomized decremental transitive closure algorithm of Bernstein [2] very efficiently, so that the total update time of our recursive data structure is still nearly-linear in n.

Finally, we show that the maintained reachability information is sufficient to support reachability queries in $\widetilde{O}(\sqrt{n})$ time and explicitly maintain the strongly-connected components of G under edge deletions. By known black-box reductions, this is sufficient to solve decremental

[2] A reachability matrix of a set S is a submatrix of the transitive closure matrix consisting of rows and columns corresponding to vertices belonging to S.

single-source reachability in nearly-linear time. Using a technique of [57], we can reduce the point-to-point query time to $\widetilde{O}(\sqrt{t})$ at the cost of increasing the total update time to $\widetilde{O}(n^2/t)$.

Note that this way we obtain randomized algorithms. We also show that by using plane duality we can, in a sense, turn the decremental strongly-connected components problem into a certain incremental reachability problem, where the edges of a graph (fixed from the beginning) are switched-on in an online manner. This problem is solved using an analogous recursive data structure, but now the reachability information between the boundary vertices of individual pieces has to be maintained incrementally. As a result, we can replace Bernstein's decremental transitive closure algorithm with a conceptually simpler, "folklore" incremental algorithm which is more efficient and deterministic at the same time. Consequently, we obtain a deterministic decremental single-source reachability algorithm with $O(n \log^2 n \log\log n)$ total update time.

4 Shortest Paths in Dense Distance Graphs

Finally, we consider the problem of computing shortest paths in so-called dense distance graphs, a basic building block for designing efficient planar graph algorithms.

In their breakthrough paper [22], Fakcharoenphol and Rao introduced the general concept of a *dense distance graph*. Let G be a non-negatively-weighted plane digraph and let U denote some subset of its "boundary" vertices lying on some $O(1)$ faces of G. Such graphs with a topologically nice boundary typically emerge after decomposing a plane graph using a cycle separator. For example, by using a cycle separator of Miller [54], one can decompose any n-vertex triangulated plane graph H into two subgraphs H_{in} and H_{out} such that (i) $H_{\text{in}} \cup H_{\text{out}} = H$; (ii) H_{in} and H_{out} are smaller than H by a constant factor; (iii) the set $U = V(H_{\text{in}}) \cap V(H_{\text{out}})$ has size $O(\sqrt{n})$, and lies both on a single face of H_{in} and on a single face of H_{out}.

We define a *distance clique* of G, denoted $\text{DC}(G)$, to be a complete graph on U such that the weight of an edge uv is equal to the length of the shortest path from u to v in G. A dense distance graph is a union of possibly many unrelated distance cliques $\text{DC}(G_1), \ldots, \text{DC}(G_q)$.

Fakcharoenphol and Rao [22] proposed an efficient implementation of Dijkstra's algorithm (later called *FR-Dijkstra*) computing single-source shortest paths in a dense distance graph. Their algorithm spends $O(b \log^2 n)$ time per distance clique with b vertices, even though a clique has b^2 edges. Here, n is the total number of vertices of the dense distance graph. Whereas Dijkstra's algorithm uses a priority queue to maintain its distance labels and extract a non-visited vertex with minimum label, a much more sophisticated data structure is used in FR-Dijkstra. This data structure is capable of relaxing many edges in a single step by leveraging the fact that certain submatrices of the weight matrix of a distance clique constitute so-called *Monge matrices* [70].

Fakcharoenphol and Rao originally employed FR-Dijkstra to construct their dense distance graph recursively, and consequently solve the real-weighted single-source shortest paths problem on planar graphs in nearly-linear time. However, the applications of FR-Dijkstra proved much broader. As a result, it has become an important planar graph primitive used to obtain numerous breakthrough results in recent years. We briefly cover the most important of these results below.

The dense distance graphs and FR-Dijksta have been used to break the long-standing $O(n \log n)$ barrier for computing minimal s,t-cuts [41] in undirected planar graphs and global min-cuts in both undirected [47] and directed [55] planar graphs. Borradaile et al. [6] developed an oracle answering arbitrary min s,t-cut queries in a weighted undirected planar graph after only nearly-linear preprocessing. This result has been later generalized to bounded-genus graphs [3], thus proving the usefulness of FR-Dijkstra in more general graph classes.

The most sophisticated applications of FR-Dijkstra to date are probably those related to

computing maximum flow in directed planar graphs. Borradaile et al. [5] gave a nearly-linear time max-flow algorithm for the case of multiple sources and multiple sinks, and consequently, a nearly-linear algorithm for maximum bipartite matching. Later, Łącki et al. [49] gave a nearly-linear time algorithm computing the maximum flow values between a specified source and all possible sinks.

Most recently, Asathulla et al. [1] used FR-Dijkstra to break through the $O(n^{3/2})$ barrier for the planar assignment problem with integer weights. Cabello [9] showed the first truly subquadratic algorithm for computing a diameter of a weighted planar graph. Even though it mainly builds on a new concept of additively-weighted Voronoi diagrams for planar graphs, dense distance graphs and FR-Dijkstra are still used extensively in his work.

Last but not least, FR-Dijkstra has been instrumental in obtaining virtually all sublinear update/query time *exact* dynamic algorithms for shortest paths, maximum flows and minimum cuts in planar graphs [22, 41, 42, 45, 47].

Dense distance graphs are pivotal in designing efficient planar graph algorithms, and therefore obtaining fine-grained bounds for computing and manipulating them is an important research direction. Although a better algorithm (in comparison to the recursive method of [22]), running in $O((|V| + |U|^2) \log n)$ time, has been proposed for computing a distance clique [45], improving FR-Dijkstra itself proved very challenging and no progress over [22] has been made in the most general setting so far.[3]

Our Results

We show an algorithm for computing single-source shortest paths in a dense distance graph with $O\left(b\frac{\log^2 n}{\log^2 \log n}\right)$ time overhead per distance clique with b vertices.

Our result implies an immediate improvement by a factor of $O(\log^2 \log n)$ in the time complexity for a number of planar digraph problems such as multiple-source multiple-sink maximum flows, maximum bipartite matching [5], single-source all-sinks maximum flows [49] for which the best known time bounds were $O(n \log^3 n)$, i.e., already nearly-linear. It also yields polylog-logarithmic speed-ups to both preprocessing and query/update algorithms of dynamic algorithms for shortest paths and max-flows [41, 42, 45]. More generally, we make polylog-logarithmic improvements to all previous results (such as [1]), for which the bottleneck of the best known algorithm is computing shortest paths in a dense distance graph.

Technical Overview

We treat the problem of computing shortest paths in a dense distance graph from a purely data-structural perspective. At a high level, instead of developing an entirely new shortest paths algorithm, we propose a new data structure for maintaining distance labels and extracting minimum labeled vertices in amortized $O\left(\frac{\log^2 b}{\log^2 \log b}\right)$ time, as opposed to $O(\log^2 b)$ time in [22].

In [22], a distance clique is first partitioned into *square* Monge matrices, each handling a subset of the clique's edges. For any such matrix, a separate data structure is used for relaxing the corresponding edges and extracting the labels possibly induced by these edge relaxations. Recall that in the case of Dijkstra's algorithm, the improvement from $O(m \log n)$ to $O(m + n \log n)$ time is obtained by noticing that relaxing edges is cheaper than extracting minimum labeled vertices. Consequently, one can use a Fibonacci heap [25] in place of a binary heap. We show that in the case of the data structure originally used in [22] for handling Monge matrices the situation is in a sense the opposite: label extractions can be made cheaper than

[3] A slightly better time bound has been showed only for a certain special case [56].

edge relaxations. We make use of this fact by proposing a different than in [22], biased scheme of partitioning distance cliques into *rectangular* (as opposed to square) Monge matrices. Whereas in [22] the partition follows from a very natural idea of splitting a face boundary into halves, our partition is tailored to exploit this asymmetry between the cost of processing a row and the cost of processing a column.

5 Articles Comprising This Thesis

The preliminary versions of the contents of this thesis have been included in the following conference papers.

- *Contracting a Planar Graph Efficiently*, joint work with Jacob Holm, Giuseppe F. Italiano, Jakub Łącki, Eva Rotenberg, and Piotr Sankowski, published at ESA 2017 [37].
- *Decremental Single-Source Reachability in Planar Digraphs*, joint work with Giuseppe F. Italiano, Jakub Łącki, and Piotr Sankowski, published at STOC 2017 [40].
- *Decremental Transitive Closure and Shortest Paths for Planar Digraphs and Beyond*, published at SODA 2018 [43].
- *Improved Bounds for Shortest Paths in Dense Distance Graphs*, joint work with Paweł Gawrychowski, published at ICALP 2018 [28].

References

[1] Mudabir Kabir Asathulla, Sanjeev Khanna, Nathaniel Lahn, and Sharath Raghvendra. A faster algorithm for minimum-cost bipartite perfect matching in planar graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 457–476, 2018.

[2] Aaron Bernstein. Maintaining shortest paths under deletions in weighted directed graphs. *SIAM J. Comput.*, 45(2):548–574, 2016.

[3] Glencora Borradaile, David Eppstein, Amir Nayyeri, and Christian Wulff-Nilsen. All-pairs minimum cuts in near-linear time for surface-embedded graphs. In *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*, pages 22:1–22:16, 2016.

[4] Glencora Borradaile and Philip N. Klein. An $O(n \log n)$ algorithm for maximum st-flow in a directed planar graph. *J. ACM*, 56(2):9:1–9:30, 2009.

[5] Glencora Borradaile, Philip N. Klein, Shay Mozes, Yahav Nussbaum, and Christian Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. *SIAM J. Comput.*, 46(4):1280–1303, 2017.

[6] Glencora Borradaile, Piotr Sankowski, and Christian Wulff-Nilsen. Min st-cut oracle for planar graphs with near-linear preprocessing time. *ACM Trans. Algorithms*, 11(3):16:1–16:29, 2015.

[7] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, 2004.

[8] Gerth Stølting Brodal and Rolf Fagerberg. Dynamic representation of sparse graphs. In *Algorithms and Data Structures, 6th International Workshop, WADS '99, Vancouver, British Columbia, Canada, August 11-14, 1999, Proceedings*, pages 342–351, 1999.

[9] Sergio Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2143–2152, 2017.

[10] Sergio Cabello, Erin W. Chambers, and Jeff Erickson. Multiple-source shortest paths in embedded graphs. *SIAM J. Comput.*, 42(4):1542–1571, 2013.

[11] Timothy M. Chan. All-pairs shortest paths with real weights in $O\ (n\ ^3/\log\ n)$ time. *Algorithmica*, 50(2):236–243, 2008.

[12] Shiri Chechik, Thomas Dueholm Hansen, Giuseppe F. Italiano, Veronika Loitzenbauer, and Nikos Parotsidis. Faster algorithms for computing maximal 2-connected subgraphs in sparse directed graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1900–1918, 2017.

[13] Shiri Chechik, Thomas Dueholm Hansen, Giuseppe F. Italiano, Jakub Łącki, and Nikos Parotsidis. Decremental single-source reachability and strongly connected components in $\widetilde{O}(m\sqrt{n}\,)$ total update time. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 315–324, 2016.

[14] Michael B. Cohen, Aleksander Madry, Piotr Sankowski, and Adrian Vladu. Negative-weight shortest paths and unit capacity minimum cost flow in $\tilde{o}\ (m^{10/7}\ \log\ W)$ time (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 752–771, 2017.

[15] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

[16] Camil Demetrescu and Giuseppe F. Italiano. Mantaining dynamic matrices for fully dynamic transitive closure. *Algorithmica*, 51(4):387–427, 2008.

[17] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[18] Krzysztof Diks and Piotr Sankowski. Dynamic plane transitive closure. In *Algorithms - ESA 2007, 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007, Proceedings*, pages 594–604, 2007.

[19] Jack Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, pages 449–467, 1965.

[20] David Eppstein and Michael T. Goodrich. Studying (non-planar) road networks through an algorithmic lens. In *16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2008, November 5-7, 2008, Irvine, California, USA, Proceedings*, page 16, 2008.

[21] Jeff Erickson. Maximum flows and parametric shortest paths in planar graphs. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 794–804, 2010.

[22] Jittat Fakcharoenphol and Satish Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *J. Comput. Syst. Sci.*, 72(5):868–889, 2006.

[23] Greg N. Frederickson. On linear-time algorithms for five-coloring planar graphs. *Inf. Process. Lett.*, 19(5):219–224, 1984.

[24] Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM J. Comput.*, 16(6):1004–1022, 1987.

[25] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987.

[26] Harold N. Gabow, Haim Kaplan, and Robert Endre Tarjan. Unique maximum matching algorithms. *J. Algorithms*, 40(2):159–183, 2001.

[27] Pawel Gawrychowski, Haim Kaplan, Shay Mozes, Micha Sharir, and Oren Weimann. Voronoi diagrams on planar graphs, and computing the diameter in deterministic $\tilde{O}(n^{5/3})$ time. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 495–514, 2018.

[28] Pawel Gawrychowski and Adam Karczmarz. Improved bounds for shortest paths in dense distance graphs. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 61:1–61:15, 2018.

[29] Loukas Georgiadis, Thomas Dueholm Hansen, Giuseppe F. Italiano, Sebastian Krinninger, and Nikos Parotsidis. Decremental data structures for connectivity and dominators in directed graphs. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 42:1–42:15, 2017.

[30] Dora Giammarresi and Giuseppe F. Italiano. Decremental 2- and 3-connectivity on planar graphs. *Algorithmica*, 16(3):263–287, 1996.

[31] Andrew V. Goldberg. Scaling algorithms for the shortest paths problem. *SIAM J. Comput.*, 24(3):494–504, 1995.

[32] Jens Gustedt. Efficient union-find for planar graphs and other sparse graph classes. *Theor. Comput. Sci.*, 203(1):123–141, 1998.

[33] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Sublinear-time decremental algorithms for single-source reachability and shortest paths on directed graphs. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 674–683, 2014.

[34] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Improved algorithms for decremental single-source reachability on directed graphs. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, Proceedings, Part I*, pages 725–736, 2015.

[35] Monika Rauch Henzinger, Philip N. Klein, Satish Rao, and Sairam Subramanian. Faster shortest-path algorithms for planar graphs. *J. Comput. Syst. Sci.*, 55(1):3–23, 1997.

[36] Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM*, 48(4):723–760, 2001.

[37] Jacob Holm, Giuseppe F. Italiano, Adam Karczmarz, Jakub Łącki, Eva Rotenberg, and Piotr Sankowski. Contracting a planar graph efficiently. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pages 50:1–50:15, 2017.

[38] Shang-En Huang, Dawei Huang, Tsvi Kopelowitz, and Seth Pettie. Fully dynamic connectivity in $O(\log n(\log\log n)^2)$ amortized expected time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete 16-19*, pages 510–520, 2017.

[39] Giuseppe F. Italiano. Amortized efficiency of a path retrieval data structure. *Theor. Comput. Sci.*, 48(3):273–281, 1986.

[40] Giuseppe F. Italiano, Adam Karczmarz, Jakub Łącki, and Piotr Sankowski. Decremental single-source reachability in planar digraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1108–1121, 2017.

[41] Giuseppe F. Italiano, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Improved algorithms for min cut and max flow in undirected planar graphs. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 313–322, 2011.

[42] Haim Kaplan, Shay Mozes, Yahav Nussbaum, and Micha Sharir. Submatrix maximum queries in monge matrices and partial monge matrices, and their applications. *ACM Trans. Algorithms*, 13(2):26:1–26:42, 2017.

[43] Adam Karczmarz. Decremental transitive closure and shortest paths for planar digraphs and beyond. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 73–92, 2018.

[44] David R. Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas, USA.*, pages 21–30, 1993.

[45] Philip N. Klein. Multiple-source shortest paths in planar graphs. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 146–155, 2005.

[46] Philip N. Klein and Shay Mozes. Optimization algorithms for planar graphs, 2017.

[47] Jakub Łącki, and Piotr Sankowski and. Min-cuts and shortest cycles in planar graphs in $O(n \log \log n)$ time. In *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, pages 155–166, 2011.

[48] Jakub Łącki. Improved deterministic algorithms for decremental reachability and strongly connected components. *ACM Trans. Alg.*, 9(3):27:1–27:15, 2013.

[49] Jakub Łącki and Yahav Nussbaum and Piotr Sankowski and Christian Wulff-Nilsen. Single source - all sinks max flows in planar digraphs. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 599–608, 2012.

[50] Jakub Łącki and Piotr Sankowski. Optimal decremental connectivity in planar graphs. *Theory Comput. Syst.*, 61(4):1037–1053, 2017.

[51] Martin Mareš. Two linear time algorithms for mst on minor closed graph classes. *Archivum mathematicum*, 40(3):315–320, 2002.

[52] Tomomi Matsui. The minimum spanning tree problem on a planar graph. *Discrete Applied Mathematics*, 58(1):91–94, 1995.

[53] David W. Matula, Yossi Shiloach, and Robert E. Tarjan. Two linear-time algorithms for five-coloring a planar graph. Technical report, Stanford University, Stanford, CA, USA, 1980.

[54] Gary L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *J. Comput. Syst. Sci.*, 32(3):265–279, 1986.

[55] Shay Mozes, Kirill Nikolaev, Yahav Nussbaum, and Oren Weimann. Minimum cut of directed planar graphs in $O(n \log \log n)$ time. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 477–494, 2018.

[56] Shay Mozes, Yahav Nussbaum, and Oren Weimann. Faster shortest paths in dense distance graphs, with applications. *Theor. Comput. Sci.*, 711:11–35, 2018.

[57] Shay Mozes and Christian Sommer. Exact distance oracles for planar graphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 209–222, 2012.

[58] Shay Mozes and Christian Wulff-Nilsen. Shortest paths in planar graphs with real lengths in $O(n\log^2 n/\log\log n)$ time. In *Algorithms - ESA 2010, 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part II*, pages 206–217, 2010.

[59] J. Ian Munro. Efficient determination of the transitive closure of a directed graph. *Inf. Process. Lett.*, 1(2):56–58, 1971.

[60] Neil Robertson, Daniel P. Sanders, Paul D. Seymour, and Robin Thomas. Efficiently four-coloring planar graphs. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 571–575, 1996.

[61] Liam Roditty. A faster and simpler fully dynamic transitive closure. *ACM Trans. Algorithms*, 4(1):6:1–6:16, 2008.

[62] Liam Roditty and Uri Zwick. Improved dynamic reachability algorithms for directed graphs. *SIAM J. Comput.*, 37(5):1455–1471, 2008.

[63] Liam Roditty and Uri Zwick. A fully dynamic reachability algorithm for directed graphs with an almost linear update time. *SIAM J. Comput.*, 45(3):712–733, 2016.

[64] Piotr Sankowski. Dynamic transitive closure via dynamic matrix inverse (extended abstract). In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 509–517, 2004.

[65] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Number t. 1 in Algorithms and Combinatorics. Springer, 2003.

[66] Daniel Dominic Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26(3):362–391, 1983.

[67] Sairam Subramanian. A fully dynamic data structure for reachability in planar digraphs. In *Algorithms - ESA '93, First Annual European Symposium, Bad Honnef, Germany, September 30 - October 2, 1993, Proceedings*, pages 372–383, 1993.

[68] Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[69] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004.

[70] Wikipedia. Monge array — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Monge_array`, 2018. [online; accessed 02-11-2018].

[71] Christian Wulff-Nilsen. Faster deterministic fully-dynamic graph connectivity. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1757–1769, 2013.