Abstract of " Toward Practical Planar Graph Algorithms " by David Elliot Eisenstat, Ph.D., Brown University, May 2014

Many optimization problems involving graphs naturally feature road networks, grids, or other large planar graphs. By exploiting the special structure of planar graphs, it is often possible to design algorithms that are faster or give better results than their counterparts for general graphs. We present an efficient polynomial-time approximation scheme for the Steiner forest problem in planar graphs, a bicriteria polynomial-time approximation scheme for the ball cover problem in planar graphs, a linear-time algorithm for multiple-source shortest paths problem (MSSP) in planar graphs with integer weights that are small on average, and algorithms that use MSSP to compute closeness and betweenness centrality in planar graphs. We report on our implementation of the latter.

Toward Practical Planar Graph Algorithms

by

David Elliot Eisenstat

BS, University of Rochester; Rochester, NY, 2006

BA, University of Rochester; Rochester, NY, 2006

ScM, Brown University; Providence, RI, 2011

A dissertation submitted in partial fulfillment of the

requirements for the degree of Doctor of Philosophy

in The Department of Computer Science at Brown University

PROVIDENCE, RHODE ISLAND

May 2014

This dissertation by David Elliot Eisenstat is accepted in its present form by The Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date_____ _____

Claire Mathieu, PhD, Coadvisor

Date_____ _____

Philip N. Klein, PhD, Coadvisor

Recommended to the Graduate Council

Date_____ _____

Renato F. Werneck, PhD, Reader

Approved by the Graduate Council

Date_____ _____

Peter M. Weber, Dean of the Graduate School

Curriculum Vitae

The latest version of this document is available at `http://www.davideisenstat.com/cv/`.

David Eisenstat was born on November 12, 1983 in New Haven, CT.

Publications

- Dana Angluin, James Aspnes, Rida A. Bazzi, Jiang Chen, David Eisenstat, and Goran Konjevod. Effective storage capacity of labeled graphs. *Information and Computation*, 234:44–56, February 2014.

- David Eisenstat, Philip N. Klein, and Claire Mathieu. Approximating k-center in planar graphs. In *Proceedings of the Twenty-Fifth Symposium on Discrete Algorithms (SODA)*, pages 617–627, January 2014.

- David Eisenstat and Philip N. Klein. Linear-time algorithms for max flow and multiple-source shortest paths in unit-weight planar graphs. In *Proceedings of the Forty-Fifth Symposium on Theory of Computing (STOC)*, pages 735–744, June 2013.

- James Aspnes, David Eisenstat, and Yitong Yin. Low-contention data structures. *Journal of Parallel and Distributed Computing*, 72(5):705–715, May 2012.

- David Eisenstat, Philip N. Klein, and Claire Mathieu. An efficient polynomial-time approximation scheme for Steiner forest in planar graphs. arXiv:1110.1320v2 [cs.DS], October 2011.

- David Eisenstat, Philip N. Klein, and Claire Mathieu. An efficient polynomial-time approximation scheme for Steiner forest in planar graphs. In *Proceedings of the Twenty-Third Symposium on Discrete Algorithms (SODA)*, pages 626–638, January 2012.

- David Eisenstat. Random road networks: the quadtree model. In *Proceedings of the Eighth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 76–84, January 2011.

- Dana Angluin, David Eisenstat, Leonid (Aryeh) Kontorovich, and Lev Reyzin. Lower bounds on learning random structures with statistical queries. In *Proceedings of the Twenty-First International Conference on Algorithmic Learning Theory (ALT)*, pages 194–208, October 2010.

- Dana Angluin, James Aspnes, Rida A. Bazzi, Jiang Chen, David Eisenstat, and Goran Konjevod. Storage capacity of labeled graphs. In *Proceedings of the Twelfth International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 573–587, September 2010. Journal article: Effective storage capacity of labeled graphs.

- James Aspnes, David Eisenstat, and Yitong Yin. Low-contention data structures. In *Proceedings of the Twenty-Second Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 345–354, June 2010.

- David Eisenstat. k-fold unions of low-dimensional concept classes. *Information Processing Letters*, 109(23-24):1232–1234, November 2009.

- Dana Angluin, James Aspnes, Jiang Chen, David Eisenstat, and Lev Reyzin. Learning acyclic probabilistic circuits using test paths. *Journal of Machine Learning Research*, 10(Aug):1881–1911, August 2009.

- David Eisenstat. Two-enqueuer queue in Common2. arXiv:0805.0444v2 [cs.DC], April 2009.

- Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, September 2008.

- Dana Angluin, James Aspnes, Jiang Chen, David Eisenstat, and Lev Reyzin. Learning acyclic probabilistic circuits using test paths. In *Proceedings of the Twenty-First Conference on Learning Theory (COLT)*, pages 169–180, July 2008.

- Dana Angluin, James Aspnes, and David Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21(2):87–102, July 2008.

- David Eisenstat, Jennifer Feder, Greg Francos, Gary Gordon, and Amanda Redlich. Expected rank and randomness in rooted graphs. *Discrete Applied Mathematics*, 156(5):746–756, March 2008.

- David Eisenstat, Gary Gordon, and Amanda Redlich. Combinatorial properties of a rooted graph polynomial. *SIAM Journal on Discrete Mathematics*, 22(2):776–785, March 2008.

- Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, November 2007.

- Dana Angluin, James Aspnes, and David Eisenstat. A simple population protocol for fast robust approximate majority. In *Proceedings of the Twenty-First International Symposium on Distributed Computing (DISC)*, pages 20–32, September 2007.

- Stephen Soltesz, Soner Sevinc, David Eisenstat, Marc Fiuczynski, and Larry Peterson. On the design and evolution of an architecture for federation. In *Proceedings of the Second International Workshop on Real Overlays and Distributed Systems (ROADS)*, July 2007.

- David Eisenstat and Dana Angluin. The VC dimension of k-fold union. *Information Processing Letters*, 101(5):181–184, March 2007.

- Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. In *Proceedings of the Twentieth International Symposium on Distributed Computing (DISC)*, pages 61–75, September 2006.

- Dana Angluin, James Aspnes, and David Eisenstat. Stably computable predicates are semilinear. In *Proceedings of the Twenty-Fifth Symposium on Principles of Distributed Computing (PODC)*, pages 292–299, July 2006. Journal article: The computational power of population protocols.

- Virendra J. Marathe, Michael F. Spear, Christopher Heriot, Athul Acharya, David Eisenstat, William N. Scherer III, and Michael L. Scott. Lowering the overhead of nonblocking software transactional memory. In *Proceedings of the First Workshop on Languages, Compilers, and Hardware Support for Transactional Computing (TRANSACT)*, June 2006.

- Arrvindh Shriraman, Virendra J. Marathe, Sandhya Dwarkadas, Michael L. Scott, David Eisenstat, Christopher Heriot, William N. Scherer III, and

Michael F. Spear. Hardware acceleration of software transactional memory. In *Proceedings of the First Workshop on Languages, Compilers, and Hardware Support for Transactional Computing (TRANSACT)*, June 2006.

- Virendra J. Marathe, Michael F. Spear, Christopher Heriot, Athul Acharya, David Eisenstat, William N. Scherer III, and Michael L. Scott. Lowering the overhead of nonblocking software transactional memory. Technical Report 893, Computer Science Department, University of Rochester, May 2006.

- David Eisenstat and Gary Gordon. Non-isomorphic caterpillars with identical subtree data. *Discrete Mathematics*, 306(8-9):827–830, May 2006.

- Arrvindh Shriraman, Virendra J. Marathe, Sandhya Dwarkadas, Michael L. Scott, David Eisenstat, Christopher Heriot, William N. Scherer III, and Michael F. Spear. Hardware acceleration of software transactional memory. Technical Report 887, Computer Science Department, University of Rochester, March 2006.

- Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. On the power of anonymous one-way communication. In *Proceedings of the Ninth International Conference on Principles of Distributed Systems (OPODIS)*, pages 396–411, December 2005. Journal article: The computational power of population protocols.

- David Eisenstat. Simpler proofs of the power of one query to a p-selective set. Technical Report 883, Computer Science Department, University of Rochester, October 2005.

Education

- PhD 2014 (expected), Computer Science, Brown University. Dissertation: *Toward practical planar graph algorithms.* Advisors: Philip N. Klein and Claire Mathieu.
- ScM 2011, Computer Science, Brown University.
- Graduate studies 2006–2007, Computer Science, Princeton University.
- BS 2006, Computer Science, University of Rochester (summa cum laude, highest distinction and highest honors in research in Computer Science).
- BA 2006, Mathematics, University of Rochester (summa cum laude, highest distinction in Mathematics).

Awards and honors

- van Dam Fellow, Computer Science Department, Brown University, Summer 2011.
- Best Student Paper, SSS 2010.
- Honorable Mention, National Science Foundation (NSF) Graduate Research Fellowship Program (GRFP), 2006 and 2010.
- Best Student Paper, DISC 2007.
- National Defense Science and Engineering Graduate (NDSEG) Fellow, 2006–2007.
- Gordon Y.S. Wu Fellow, School of Engineering and Applied Science (SEAS), Princeton University, 2006–2007.
- Outstanding Undergraduate Award, Computing Research Association (CRA), 2006.

- Phi Beta Kappa (ΦBK), University of Rochester, 2005 (junior year).
- Stoddard Prize, Mathematics Department, University of Rochester, 2004.

Teaching experience

- TA, Coding the Matrix: Linear Algebra through Computer Science Applications, Brown University, Fall 2012.
- TA, Approximation Algorithms (CSCI2510), Brown University, Fall 2011.
- TA, Introduction to Computational Geometry (CSCI1950J), Brown University, Spring 2011.
- TA, Operating Systems (COS 318), Princeton University, Fall 2007.

Acknowledgments

Much of the work described in this dissertation was done jointly with my advisors, Philip N. Klein and Claire Mathieu, and supported in part by their NSF grant, CCF-0964037. I would also like to thank my reader, Renato F. Werneck, for his suggestion of using MSSP to compute centrality measures; my friends, for moral support as well as many enjoyable hours of board games; Lauren Clarke, for minimizing the accidental complexity of being a PhD student; and my parents, for everything.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER **One**

Introduction

Consider two problems of the sort that a city planner might face. First, given some service requirements for a new streetcar system, which road segments should receive tracks? Let's suppose that the requirements are a list of pairs of stops that must be connected to one another. Second, where should fire stations be built so that, for every inflammable structure, some fire engine can drive to it in ten minutes or less? In both cases, the planner would like to conserve limited city resources.

The first problem is an example of the **Steiner forest problem**, and the second problem is an example of the **ball cover problem**. Both problems are well studied, and it is known that, under plausible assumptions, there exists a constant $c > 1$ such that neither Steiner forest [Bern and Plassmann, 1989] nor ball cover [Feder and Greene, 1988] has a polynomial-time approximation algorithm whose ratio is better than c. Fortunately, these hardness results pertain to general graphs only.

Road networks are embedded on or near the Earth's surface, and we model them as undirected embedded planar graphs where each edge has a nonnegative length. This model, as models often are, is a compromise; while we do not account for one-way roads and overpasses, we can assume the rich structure of a graph metric derived from a planar graph. Another abundant source of planar graphs is grids, often arising from optimization problems in computer vision.

By exploiting this structure, we obtain an efficient polynomial-time approximation scheme for Steiner forest and an inefficient polynomial-time bicriteria approximation scheme for ball cover. Our scheme for Steiner forest improves on an inefficient scheme due to Bateni, Hajiaghayi, and Marx [2011].

Like many algorithms for planar graphs, these schemes compute a lot of shortest paths. In particular, computing all-pairs distances between $O(\sqrt{n})$ vertices incident to the infinite face of a planar graph is a staple of planar graph algorithm design dating back to the work of Lipton, Rose, and Tarjan [1979]. The current fastest algorithm for this task uses the multiple-source shortest path (MSSP) algorithm of Klein [2005], who showed that the changes between two shortest path trees rooted at adjacent vertices can be computed in time $O(c \log n)$, where c is the number of changes, and that the total number of changes around the boundary is $O(n)$. Cabello, Chambers, and Erickson [2013] simplified Klein's algorithm and extended it to graphs of genus g, obtaining bounds of $O\big((g+c)\log n\big)$ time to compute changes and $O(gn)$ changes in total.

We refine Klein's bound on the number of changes and replace the core data structure of the algorithm of Cabello et al. to obtain a linear-time algorithm for MSSP in planar graphs with unit-length edges. This refinement additionally allows us to use a practically faster data structure to implement the original algorithm of

Finally, we present algorithms and experimental results for using MSSP to compute centrality measures in planar graphs. By avoiding the substantial memory traffic associated with computing the full shortest path tree for each root vertex, we improve significantly on the previously fastest method, PHAST [Delling, Goldberg, Nowatzyk, and Werneck, 2013], which retains the advantage of handling nonplanar graphs.

CHAPTER **Two**

Preliminaries

In this chapter, we review, with proofs, some well known results about planar embedded graphs. We don't claim novelty on anything herein.

2.1 Notation

The **integers** comprise the set

$$\mathbb{Z} \triangleq \{\ldots, -1, 0, 1, \ldots\}.$$

The **nonnegative** integers comprise the set

$$\mathbb{N} \triangleq \{0, 1, 2, \ldots\}.$$

The **positive** integers comprise the set

$$\mathbb{Z}_+ \triangleq \{1, 2, 3, \ldots\}.$$

Let X and Y be sets. The **Cartesian product** of X and Y is

$$X \times Y \triangleq \big\{(x, y) : x \in X,\ y \in Y\big\}.$$

The **relative complement** of Y in X is

$$X \setminus Y \triangleq \{x : x \in X,\ x \notin Y\}.$$

The collection of maps from X to Y is Y^X.

2.2 Embedded graphs

Let E be a finite set of **edges**. A **dart** is formally a member of the set

$$\operatorname{Darts} E \triangleq E \times \{1, -1\}.$$

Intuitively, given an edge $e \in E$, the darts $(e, 1)$ and $(e, -1)$ are interpreted as its two possible orientations. We define two maps on $\operatorname{Darts} E$.

$$\mathsf{edge}\,(e, \sigma) \triangleq e \qquad\qquad \mathsf{rev}\,(e, \sigma) \triangleq (e, -\sigma).$$

An **embedded graph** $G = (E, \pi)$ specifies in addition to E an **embedding** permutation π on $\operatorname{Darts} E$. We say that G is **nonempty** if and only if $E \neq \varnothing$.

With respect to G, every dart has a **head** and a **tail**.

$$\mathsf{head}_G\, a \triangleq \left\{\pi^k(a) : k \in \mathbb{Z}\right\} \qquad\qquad \mathsf{tail}_G \triangleq \mathsf{head}_G \circ \mathsf{rev}.$$

We sometimes omit the subscripts when the choice of G is clear. Intuitively, π maps each dart to the next dart in counterclockwise order with the same head. The **vertices** of G comprise the collection of sets

$$\operatorname{Vertices} G \triangleq \{\mathsf{head}_G\, a : a \in \operatorname{Darts} E\}.$$

This collection is a partition by the following argument. Suppose that $b \in \mathsf{head}_G\, a_1 \cap \mathsf{head}_G\, a_2$. There exist $k_1, k_2 \in \mathbb{Z}$ such that $\pi^{k_1}(a_1) = b = \pi^{k_2}(a_2)$. For every $k \in \mathbb{Z}$,

$$\pi^k(a_1) = \pi^k\big(\pi^{-k_1}(b)\big) = \pi^k\Big(\pi^{-k_1}\big(\pi^{k_2}(a_2)\big)\Big) = \pi^{k-k_1+k_2}(a_2),$$

so $\mathsf{head}_G\, a_1 \subseteq \mathsf{head}_G\, a_2$ and thus $\mathsf{head}_G\, a_1 = \mathsf{head}_G\, a_2$.

With respect to G, every dart has a **right** and a **left**.

$$\mathsf{right}_G\, a \triangleq \left\{(\mathsf{rev} \circ \pi)^k(a) : k \in \mathbb{Z}\right\} \qquad\qquad \mathsf{left}_G \triangleq \mathsf{right}_G \circ \mathsf{rev}.$$

We sometimes omit the subscripts when the choice of G is clear. Intuitively, $\mathsf{rev} \circ \pi$ maps each dart to the next dart in clockwise order with the same right. The **faces** of G comprise the partition

$$\operatorname{Faces} G \triangleq \{\mathsf{right}_G\, a : a \in \operatorname{Darts} E\}.$$

A vertex v is **incident** to a face f if and only if $v \cap f \neq \varnothing$. A vertex v is **incident**

to an edge e if and only if $v \cap \operatorname{Darts}\{e\} \neq \varnothing$. A face f is **incident** to an edge e if and only if $f \cap \operatorname{Darts}\{e\} \neq \varnothing$. A dart is **incident** to a vertex or face if and only if its edge is.

The **dual** of G is the embedded graph $G^* = (E, \pi^*)$, where $\pi^* = \mathsf{rev} \circ \pi$. When studying G^*, we refer to G as the **primal**. Since $\mathsf{rev} \circ \mathsf{rev} \circ \pi = \pi$, the dual of the dual of G is, as expected, G itself. Observe the following correspondences between primal and dual.

$$\operatorname{Vertices} G = \operatorname{Faces} G^* \qquad \operatorname{Faces} G = \operatorname{Vertices} G^*$$

$$\mathsf{head}_G = \mathsf{right}_{G^*} \qquad \mathsf{right}_G = \mathsf{head}_{G^*}$$

$$\mathsf{tail}_G = \mathsf{left}_{G^*} \qquad \mathsf{left}_G = \mathsf{tail}_{G^*}$$

2.3 Paths

Let $G = (E, \pi)$ be an embedded graph. A v_0v_ℓ-**path** is a sequence

$$P = \langle v_0, a_1, v_1, a_2, v_2, \ldots, v_{\ell-1}, a_\ell, v_\ell \rangle$$

of alternating vertices and darts such that, for every $i \in \{1, 2, \ldots, \ell\}$, we have $\mathsf{tail}_G\, a_i = v_{i-1}$ and $\mathsf{head}_G\, a_i = v_i$. The **tail** of P is v_0, and the **head** of P is v_ℓ. We define

$$\operatorname{Edges} P \triangleq \{\mathsf{edge}\, a_1,\ \mathsf{edge}\, a_2,\ \ldots,\ \mathsf{edge}\, a_\ell\}.$$

A **subpath** of P is a subsequence of P that is also a path.

We define the **reversal** $\mathsf{rev}\, P$, a $v_\ell v_0$-path, as follows.

$$\mathsf{rev}\, P \triangleq \langle v_m,\ \mathsf{rev}\, a_m,\ v_{m-1},\ \mathsf{rev}\, a_{m-1},\ v_{m-2},\ \ldots,\ v_1,\ \mathsf{rev}\, a_1,\ v_0 \rangle$$

Given a $v_\ell v_m$-path

$$Q = \langle v_\ell, a_{\ell+1}, v_{\ell+1}, a_{\ell+2}, v_{\ell+2}, \ldots, v_{m-1}, a_m, v_m \rangle,$$

we define the **concatenation** $P * Q$, a $v_0 v_m$-path, as follows.

$$P * Q \triangleq \langle v_0, a_1, v_1, \ldots, v_{\ell-1}, a_\ell, v_\ell, a_{\ell+1}, v_{\ell+1}, \ldots, v_{m-1}, a_m, v_m \rangle$$

The path P is **simple** if and only if $v_0, v_1, \ldots, v_\ell$ are pairwise distinct. The path $\mathsf{rev}\, P$ is simple if and only if P is simple. The following lemma implies that, if there exists an $v_0 v_\ell$-path, then there exists a simple $v_0 v_\ell$-path.

Lemma 1. *Every $v_0 v_\ell$-path P has a simple $v_0 v_\ell$-subpath.*

Proof. We induct on ℓ, the number of darts in P. If P is not simple, then there exist $i, j \in \{0, 1, \ldots, \ell\}$ such that $i < j$ and $v_i = v_j$. We define a proper $v_0 v_\ell$-subpath P' of P.

$$P' = \langle v_0, \ldots, a_i, v_i \rangle * \langle v_j, a_{j+1}, \ldots, v_\ell \rangle$$

By the inductive hypothesis, P' has a simple $v_0 v_\ell$-subpath, which is a subpath of P also. □

2.4 Connectivity

An embedded graph $G = (E, \pi)$ is **connected** if and only if, for every pair of vertices s, t, there exists an st-path. The following lemma implies that G is connected if and only if its dual is connected.

Lemma 2. *Let $G = (E, \pi)$ be an embedded graph. For every pair of darts b, c, if there exists a $(\mathsf{head}_G\, b)(\mathsf{head}_G\, c)$-path P, then there exists a $(\mathsf{right}_G\, b)(\mathsf{right}_G\, c)$-path Q in the dual of G such that, for every face f or dart a in Q, there exists a vertex v in P incident to f or a respectively.*

Proof. Let $s = \mathsf{right}_G\, b$ and $t = \mathsf{right}_G\, c$. We prove first the special case where $\mathsf{head}_G\, b = \mathsf{head}_G\, c = v$. Let $k \in \mathbb{Z}$ satisfy $\pi^k(b) = c$. If $k \geq 0$, then

$$\left\langle \mathsf{right}_G\, b,\ \pi(b),\ \mathsf{right}_G\, \pi(b),\ \pi^2(b),\ \mathsf{right}_G\, \pi^2(b),\ \ldots,\ \mathsf{right}_G\, \pi^k(b) \right\rangle$$

is a suitable st-path. If $k < 0$, then

$$\left\langle \mathsf{right}_G\, b,\ \mathsf{rev}\, b,\ \mathsf{right}_G\, \pi^{-1}(b),\ \mathsf{rev}\, \pi^{-1}(b),\ \mathsf{right}_G\, \pi^{-2}(b),\ \ldots,\ \mathsf{right}_G\, \pi^k(b) \right\rangle$$

is a suitable st-path.

For every vertex v, let

$$F(v) = \{\mathsf{right}_G\, a : a \in v\}$$

be the set of faces incident to v. We show next for every dart a that $\mathsf{right}_G\, a \in$

$F(\mathsf{tail}_G\, a) \cap F(\mathsf{head}_G\, a)$.

$$\mathsf{right}_G\, a = \mathsf{right}_G\, \pi^{-1}(\mathsf{rev}\, a) \in F\big(\mathsf{head}_G\, \pi^{-1}(\mathsf{rev}\, a)\big) = F(\mathsf{head}_G\, \mathsf{rev}\, a) = F(\mathsf{tail}_G\, a)$$

That $\mathsf{right}_G\, a \in F(\mathsf{head}_G\, a)$ is clear.

Finally, we consider arbitrary b, c. Let

$$\langle v_0, a_1, v_1, \ldots, v_{\ell-1}, a_\ell, v_\ell \rangle$$

be a $(\mathsf{head}_G\, b)(\mathsf{head}_G\, c)$-path. We construct the st-path Q by concatenating suitable paths between each adjacent pair of faces in the sequence

$$\langle s,\ \mathsf{right}_G\, a_1,\ \mathsf{right}_G\, a_2,\ \ldots,\ \mathsf{right}_G\, a_\ell,\ t \rangle.$$

These paths exist because $s \in F(v_0)$ and $t \in F(v_\ell)$ and, for every $i \in \{1, 2, \ldots, \ell\}$, we have $\mathsf{right}_G\, a_i \in F(v_{i-1}) \cap F(v_i)$. □

2.5 Interiors, exteriors, and frontiers

Let $G = (E, \pi)$ be an embedded graph. For every set of faces F, define

$$\begin{aligned}
\text{DartInterior}\, F &\triangleq \Big\{ a : a \in \bigcup_{f \in F} f,\ \mathsf{rev}\, a \in \bigcup_{f \in F} f \Big\} \\
\text{Interior}_G\, F &\triangleq \{ v : v \in \text{Vertices}\, G,\ v \subseteq \text{DartInterior}\, F \} \\
\text{Exterior}_G\, F &\triangleq \text{Interior}\, (\text{Faces}\, G \setminus F) \\
\text{Frontier}_G\, F &\triangleq \text{Vertices}\, G \setminus (\text{Interior}_G\, F \cup \text{Exterior}_G\, F).
\end{aligned}$$

The vertices comprising $\text{Interior}_G F$ are those whose incident darts have left face and right face belonging to F, i.e., the vertices incident only to faces in F. Observe that $\text{Interior}_G F \cap \text{Exterior}_G F = \varnothing$, and hence interior, exterior, and frontier comprise a partition. Every interior-exterior path crosses the frontier.

Lemma 3. *Let $G = (E, \pi)$ be an embedded graph and F be a set of faces. Let $P = \langle v_0, a_1, v_1, \ldots, v_\ell \rangle$ be a path such that $v_0 \in \text{Interior}_G F$ and $v_\ell \in \text{Exterior}_G F$. There exists $i \in \{1, 2, \ldots, \ell - 1\}$ such that $v_i \in \text{Frontier}_G F$.*

Proof. Let $s = \mathsf{left}_G a_1$ and $t = \mathsf{right}_G a_\ell$. Observe that $s \in F$ and $t \notin F$. By Lemma 2, there exists an st-path in the dual of G, every one of whose darts is incident to some vertex in P. This path contains a dart b such that $\mathsf{left}_G b \in F$ and $\mathsf{right}_G b \notin F$. We conclude that $b \notin \text{DartInterior}\, F \cup \text{DartInterior}\, (\text{Faces}\, G \setminus F)$, so $\{\mathsf{head}_G b, \mathsf{tail}_G b\} \subseteq \text{Frontier}_G F$, and P contains one of these vertices. □

The following proposition describes the frontier of Boolean combinations of sets.

Proposition 4. *Let $G = (E, \pi)$ be an embedded graph and F_1, F_2 be sets of faces. Then*

$$\text{Frontier}_G\, (\text{Faces}\, G \setminus F_1) = \text{Frontier}_G F_1$$

$$\text{Frontier}_G\, (F_1 \cup F_2) \subseteq \text{Frontier}_G F_1 \cup \text{Frontier}_G F_2.$$

2.6 Arborescences

Let $G = (E, \pi)$ be an embedded graph. An **arborescence** with **root** vertex r is a map $p \in (\text{Darts}\, G)^{\text{Vertices}\, G \setminus \{r\}}$ having the following properties.

- For every vertex $v \neq r$, we have $\mathsf{head}_G\, p(v) = v$.
- There exists a map $\mathsf{depth}_p \in \mathbb{N}^{\mathrm{Vertices}\, G}$ such that, for every vertex v, we have $(\mathsf{tail}_G \circ p)^{\mathsf{depth}_p\, v}(v) = r$.

The map depth_p is unique. We define

$$\mathrm{Edges}\, p \triangleq \big\{\mathsf{edge}\, p(v) : v \in \mathrm{Vertices}\, G \setminus \{r\}\big\}$$
$$\mathrm{Descendants}_p\, v \triangleq \big\{w : k \in \mathbb{N},\ (\mathsf{tail}_G \circ p)^k(w) = v\big\}.$$

Observe that $v \in \mathrm{Descendants}_p\, v$ and that $\mathrm{Descendants}_p\, r = \mathrm{Vertices}\, G$.

Arborescences encode families of simple paths. For every vertex v and arborescence p with root r, define the simple rv-path

$$\mathsf{path}_p\, v \triangleq \big\langle r,\ \ldots,\ (p \circ \mathsf{tail}_G \circ p)(v),\ (\mathsf{tail}_g \circ p)(v),\ p(v),\ v\big\rangle.$$

If G has an arborescence p, then G is nonempty, with the root as a witness, and connected because, for every pair of vertices s, t, the concatenation $\mathsf{rev}\, \mathsf{path}_p\, s * \mathsf{path}_p\, t$ is an st-path. The following lemma implies the converse.

Lemma 5. *Let $G = (E, \pi)$ be an embedded graph and r be a vertex. Let $\mathcal{P}$ be a collection of paths having the following properties.*

- *For every vertex v, there exists an rv-path belonging to $\mathcal{P}$.*
- *Every subpath of a path belonging to $\mathcal{P}$ itself belongs to $\mathcal{P}$.*
- *For every pair of rv-paths P_1, P_1' and every vw-path P_2, if $\{P_1 * P_2,\ P_1'\} \subseteq \mathcal{P}$, then $P_1' * P_2 \in \mathcal{P}$.*

There exists an arborescence p with root r such that, for every vertex v, we have $\mathsf{path}_p\, v \in \mathcal{P}$.

Proof. Define $\mathsf{depth}_p\, r = 0$. We construct p and the rest of depth_p incrementally, maintaining several invariants that hold for every vertex v where depth_p is defined.

- If $v \neq r$, then $\mathsf{head}_G\, p(v) = v$.
- $(\mathsf{tail}_G \circ p)^{\mathsf{depth}_p\, v}(v) = r$.
- $\mathsf{path}_p\, v \in \mathcal{P}$.

When depth_p is defined for every vertex, p is an arborescence with the desired property.

Initially, $\mathcal{P}$ contains some rr-path, of which $\langle r \rangle$ is a subpath, so $\mathsf{path}_p\, r = \langle r \rangle \in \mathcal{P}$, and all of the invariants hold. While there exists a vertex v for which $\mathsf{depth}_p\, v$ is undefined, let $P' \in \mathcal{P}$ be an rv-path with simple rv-subpath

$$P = \langle v_0, a_1, v_1, a_2, v_2, \ldots, v_\ell \rangle,$$

whose existence follows from Lemma 1. Since P is a subpath of $P' \in \mathcal{P}$, we have $P \in \mathcal{P}$. Let $i \in \{0, 1, \ldots, \ell\}$ have its maximum value such that $\mathsf{depth}_p\, v_i$ is defined and, for every $j \in \{i+1,\ i+2,\ \ldots,\ \ell\}$, define

$$p(v_j) = a_j \qquad\qquad \mathsf{depth}_p\, v_j = \mathsf{depth}_p\, v_i + j - i.$$

By the simplicity of the path and the maximality of i, the new definitions are well founded. We have $\mathsf{path}_p\, v_j \in \mathcal{P}$ because that path is obtained by replacing a prefix

of a prefix of P with $\mathsf{path}_p\, v_i$, so all three invariants are maintained. □

For some well behaved choices of $\mathcal{P}$, there exist $O\big(|E|\big)$- or $O\big(|E| \log |E|\big)$-time algorithms that instantiate the framework described in the proof. We mostly take for granted these standard graph algorithms (e.g., depth-first search and the shortest paths algorithm of Dijkstra [1959]).

2.7 Topological order

Let X be a finite set of cardinality n. An **enumeration** of X is a sequence $\langle x_1, x_2, \ldots, x_n \rangle$ that satisfies $\{x_1, x_2, \ldots, x_n\} = X$. Let $G = (E, \pi)$ be an embedded graph and p be an arborescence with root r. A **topological order** is an enumeration $\langle v_1, v_2, \ldots, v_n \rangle$ of $\mathrm{Vertices}\, G$ that satisfies the following requirements.

- $v_1 = r$.
- For every $j \in \{2, 3, \ldots, n\}$, we have $\mathsf{tail}_G\, p(v_j) \in \{v_1, v_2, \ldots, v_{j-1}\}$.

Observe that $\mathsf{head}_G\, p(v_j) = v_j$ and thus

$$v_j \notin \big\{\mathsf{tail}_G\, p(v_2),\ \ldots,\ \mathsf{tail}_G\, p(v_j),\ \mathsf{head}_G\, p(v_2),\ \ldots,\ \mathsf{head}_G\, p(v_{j-1})\big\}.$$

In other words, none of the edges $\mathsf{edge}\, p(v_1)$, $\mathsf{edge}\, p(v_2)$, $\ldots$, $\mathsf{edge}\, p(v_{j-1})$ is incident to v_j, and $\mathsf{tail}_G\, p(v_j) \neq v_j$.

Lemma 6. *Let $G = (E, \pi)$ be an embedded graph and p be an arborescence with root r. There exists a topological order.*

Proof. Let $\langle v_1, v_2, \ldots, v_n \rangle$ be an enumeration of Vertices G such that, for every pair $i, j \in \{1, 2, \ldots, n\}$, if $\mathsf{depth}_p\, v_i < \mathsf{depth}_p\, v_j$, then $i < j$. Since $w = r$ is the unique solution to the equation $\mathsf{depth}_p\, w = 0$, we have $v_1 = r$. For every $j \in \{2, 3, \ldots, n\}$,

$$\mathsf{depth}_p\, \mathsf{tail}_G\, p(v_j) = \mathsf{depth}_p\, v_j - 1 < \mathsf{depth}_p\, v_j,$$

so $\mathsf{tail}_G\, p(v_j) \in \{v_1, v_2, \ldots, v_{j-1}\}$, as required. □

2.8 Paths that avoid an arborescence

Let $G = (E, \pi)$ be an embedded graph and p be an arborescence. In conjunction with Lemma 5, the following lemma implies that, in the dual of G, there exists an arborescence q having no edges in common with p.

Lemma 7. *Let $G = (E, \pi)$ be an embedded graph and let p be an arborescence with root r. For every pair of faces s, t, there exists an st-path Q in the dual of G such that* Edges $Q \cap$ Edges $p = \varnothing$.

Proof. Let $\langle v_1, v_2, \ldots, v_n \rangle$ be a topological order, which exists by Lemma 6. For every $k \in \{1, 2, \ldots, n\}$, define

$$F_k = \big\{\mathsf{edge}\, p(v_i) : i \in \{2, 3, \ldots, k\}\big\}.$$

Observe that $F_1 = \varnothing$ and $F_n =$ Edges p. We prove by induction on k that there exists an st-path Q such that Edges $Q \cap F_k = \varnothing$.

The claim for $k = 1$ is just that there exists an st-path. The existence of p implies that G is connected and hence, by Lemma 2, that the dual of G is connected,

which suffices to prove the claim. For $k > 1$, we are given an st-path Q' with $\mathrm{Edges}\, Q' \cap F_{k-1} = \varnothing$ and must find an st-path Q such that $\mathrm{Edges}\, Q \cap F_k = \varnothing$. Let $a = p(v_k)$, so that $T_k = T_{k-1} \cup \{a\}$. It suffices to find a $(\mathsf{right}_G\, a)(\mathsf{left}_G\, a)$-path Q'' such that $\mathrm{Edges}\, Q'' \cap F_k = \varnothing$, because then we derive Q by replacing in Q' each occurrence of $\langle \mathsf{right}_G\, a, \mathsf{rev}\, a, \mathsf{left}_G\, a \rangle$ by Q'', and each occurrence of $\langle \mathsf{left}_G\, a, a, \mathsf{right}_G\, a \rangle$ by $\mathsf{rev}\, Q''$.

Intuitively, Q'' is the path around v_k that avoids a. As there are only finitely many darts with head v_k, let j be the minimum nonnegative integer for which there exists a nonnegative integer $i < j$ satisfying $\pi^i(a) = \pi^j(a)$. Since π is a permutation, $\pi^{j-i}(a) = a$, and $i = 0$. Observe that $\mathsf{right}_G\, \pi^{j-1}(a) = \mathsf{right}_G\, \mathsf{rev}\, \pi^j(a) = \mathsf{left}_G\, a$ and define the $(\mathsf{right}_G\, a)(\mathsf{left}_G\, a)$-path

$$Q'' = \big\langle \mathsf{right}_G\, a,\ \pi(a),\ \mathsf{right}_G\, \pi(a),\ \pi^2(a),\ \ldots,\ \mathsf{right}_G\, \pi^{j-1}(a) \big\rangle.$$

Every dart in Q'' has head v_k. By the minimality of j, the path Q'' does not contain the dart a, and it follows from the properties of the topological order that $\mathrm{Edges}\, Q'' \cap F_k = \varnothing$. □

2.9 Chains, cycles, and boundaries

Let X be a set. For every $y \in X$ and $n \in \mathbb{Z}$ and $\phi, \phi' \in \mathbb{Z}^X$, we define, abusing notation,

$$[y](x) \triangleq \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$$

$$n(x) \triangleq n$$

$$(\phi + \phi')(x) \triangleq \phi(x) + \phi'(x)$$

$$(-\phi)(x) \triangleq -\phi(x)$$

$$\phi - \phi' \triangleq \phi + (-\phi')$$

$$(n\phi)(x) \triangleq n\phi(x).$$

The map ϕ is **carried by** a set $Y \subseteq X$ if and only if, for every $x \in X \setminus Y$, we have $\phi(x) = 0$. We write $\phi \leq \phi'$ in case $\phi' - \phi \in \mathbb{N}^X$, and $\phi \geq \phi'$ in case $\phi' \leq \phi$. We write $\phi < \phi'$ in case $\phi \leq \phi'$ and $\phi \neq \phi'$, and $\phi > \phi'$ in case $\phi' < \phi$. The relation $\leq$ is a partial order.

Reflexive $\phi \leq \phi$.

Antisymmetric If $\phi \leq \phi'$ and $\phi' \leq \phi$, then $\phi = \phi'$.

Transitive If $\phi \leq \phi'$ and $\phi' \leq \phi''$, then $\phi \leq \phi''$.

Let $G = (E, \pi)$ be an embedded graph. We define **face chains** (2-chains), **edge chains** (1-chains), and **vertex chains** (0-chains).

$$\operatorname{FaceChains} G \triangleq \mathbb{Z}^{\operatorname{Faces} G} \qquad \operatorname{EdgeChains} E \triangleq \mathbb{Z}^{E}$$

$$\operatorname{VertexChains} G \triangleq \mathbb{Z}^{\operatorname{Vertices} G}$$

We extend the domain of every edge chain ψ to include $\operatorname{Darts} E$.

$$\psi\big((e, \sigma)\big) \triangleq \sigma\psi(e).$$

We extend the domain of $[\cdot]$ to include darts and paths.

$$\big[(e, \sigma)\big] \triangleq \sigma[e] \qquad \big[\langle v_0, a_1, v_1, \ldots, v_\ell\rangle\big] \triangleq \sum_{i=1}^{\ell} [a_i]$$

We define two **boundary operators**. The first, ∂_2^G, maps face chains to edge chains.

$$\partial_2^G \phi \triangleq \sum_{f \in \operatorname{Faces} G} \phi(f) \sum_{a \in f} [a]$$

The second, ∂_1^G, maps edge chains to vertex chains.

$$\partial_1^G \psi \triangleq \sum_{a \in \operatorname{Darts} E} \psi(a)[\mathsf{head}_G\, a].$$

We sometimes omit the superscripts when the choice of G is clear. Observe that, when P is an st-path, we have $\partial_1^G[P] = [t] - [s]$. The boundary operators are linear.

$$\partial_2^G(\phi + \phi') = \partial_2^G\phi + \partial_2^G\phi' \qquad \partial_1^G(\psi + \psi') = \partial_1^G\psi + \partial_1^G\psi'$$

$$\partial_2^G(n\phi) = n(\partial_2^G\phi) \qquad \partial_1^G(n\psi) = n(\partial_1^G\psi)$$

We define two sets of edge chains. The first, the kernel of ∂_1^G, is the set of all **cycles**. The second, the image of ∂_2^G, is the set of all **boundaries**.

$$\operatorname{Cycles} G \triangleq \{\psi : \psi \in \operatorname{EdgeChains} G,\ \partial_1^G \psi = 0\}$$
$$\operatorname{Boundaries} G \triangleq \{\partial_2^G \phi : \phi \in \operatorname{FaceChains} G\}$$

The following lemma implies that every boundary is a cycle.

Lemma 8. *Let $G = (E, \pi)$ be an embedded graph. For every $\phi \in \operatorname{FaceChains} G$, we have $\partial_1^G \partial_2^G \phi = 0$.*

Proof. By the linearity of ∂_1^G,

$$\partial_1^G \partial_2^G \phi = \sum_{f \in \operatorname{Faces} G} \phi(f) \sum_{a \in f} \partial_1^G [a],$$

so it suffices to show that, for every face f,

$$\begin{aligned}
\sum_{a \in f} \partial_1^G [a] &= \sum_{a \in f} \big([\mathsf{head}_G\, a] - [\mathsf{tail}_G\, a]\big) \\
&= \sum_{a \in f} [\mathsf{head}_G\, a] - \sum_{a \in f} \big[\mathsf{tail}_G\, \mathsf{rev}\, \pi(a)\big] && \text{since } \big\{(\mathsf{rev} \circ \pi)^{-1}(a) : a \in f\big\} = f \\
&= 0 && \text{since } \mathsf{tail}_G\, \mathsf{rev}\, \pi(a) = \mathsf{head}_G\, a. \qquad \square
\end{aligned}$$

The following lemma relates the notion of frontier to the boundary operator.

Lemma 9. *Let $G = (E, \pi)$ be an embedded graph and F be a set of faces. Defining $\phi = \sum_{f \in F} [f]$,*

$$\operatorname{Frontier}_G F = \big\{\mathsf{head}_G\, a : a \in \operatorname{Darts} E,\ (\partial_2^G \phi)(a) \neq 0\big\}.$$

Proof. A vertex v belongs to $\operatorname{Frontier}_G F$ if and only if there exists a dart $a \in v$ such that either $\mathsf{right}_G\, a$ or $\mathsf{left}_G\, a$ belongs to F but not both, which in turn holds if and only if $(\partial_2^G \phi)(a) = \phi(\mathsf{right}_G\, a) - \phi(\mathsf{left}_G\, a) \neq 0$. □

2.10 Fundamental cycles

Let $G = (E, \pi)$ be an embedded graph and p be an arborescence with root r. For every dart a, we define the **fundamental cycle**

$$\mathsf{fundamentalCycle}_p\, a \triangleq \big[\mathsf{path}_p\, \mathsf{tail}_G\, a * \langle \mathsf{tail}_G\, a,\ a,\ \mathsf{head}_G\, a\rangle * \mathsf{rev}\, \mathsf{path}_p\, \mathsf{head}_G\, a\big],$$

which is carried by the set $\operatorname{Edges} p \cup \{a\}$. We check quickly that $\mathsf{path}_p\, \mathsf{tail}_G\, a * \langle \mathsf{tail}_G\, a,\ a,\ \mathsf{head}_G\, a\rangle * \mathsf{path}_p\, \mathsf{head}_G\, a$ is an rr-path and thus that

$$\partial_1^G\, \mathsf{fundamentalCycle}_p\, a = [r] - [r] = 0,$$

from which we conclude that $\mathsf{fundamentalCycle}_p\, a$ is indeed a cycle.

Observe that $(\mathsf{fundamentalCycle}_p\, a)(a) = 1$. The following lemma has the useful corollary that, for every cycle ψ,

$$\psi = \sum_{e \in E \setminus \operatorname{Edges} p} \psi(e)\, \mathsf{fundamentalCycle}_p\, (e, 1).$$

Lemma 10. *Let $G = (E, \pi)$ be an embedded graph and p be an arborescence. If $\psi \in \operatorname{Cycles} G$ is carried by $\operatorname{Edges} p$, then $\psi = 0$.*

Proof. By Lemma 6, let $v_1, v_2, \ldots, v_n$ be a topological order. We show inductively

for every $k \in \{0,\ 1,\ \ldots,\ n-2\}$ that $\psi\big(p(v_{n-k})\big) = 0$. Since $(\partial_1^G \psi)(v_{n-k}) = 0$,

$$\sum_{a \in v_{n-k}} \psi(a) = 0.$$

This sum contains the term $\psi\big(p(v_{n-k})\big)$, so it suffices to show that every other term is zero. For every dart $a \notin \operatorname{Edges} p$, we have $\psi(a) = 0$. Every dart $a \in v_{n-k} \setminus \big\{p(v_{n-k})\big\}$ satisfies $\mathsf{edge}\, a \in \operatorname{Edges} p$ only if $p(\mathsf{tail}_G\, a) = \mathsf{rev}\, a$. In this case, $\mathsf{depth}_p\, \mathsf{tail}_G\, a = \mathsf{depth}_p\, v_{n-k} + 1 > \mathsf{depth}_p\, v_{n-k}$, so by the definition of topological order, the index of $\mathsf{tail}_G\, a$ is greater than $n-k$. We conclude by the inductive hypothesis that $\psi(a) = 0$. □

2.11 Planarity

A connected nonempty embedded graph $G = (E, \pi)$ is **planar** if and only if the following equation is satisfied.

$$|E| = \big(|\operatorname{Vertices} G| - 1\big) + \big(|\operatorname{Faces} G| - 1\big)$$

Observe that this definition is invariant under duality. Planarity has many important consequences. We prove first the existence of **interdigitating** arborescences.

Lemma 11. *Let $G = (E, \pi)$ be a planar embedded graph. For every arborescence p and face o, there exists an arborescence q with root o in the dual of G such that* $\operatorname{Edges} p \cap \operatorname{Edges} q = \varnothing$ *and* $\operatorname{Edges} p \cup \operatorname{Edges} q = E$.

Proof. The existence of q such that $\operatorname{Edges} p \cap \operatorname{Edges} q = \varnothing$ follows from Lemmas 7 and 5. Since $|\operatorname{Edges} p| = |\operatorname{Vertices} G| - 1$ and $|\operatorname{Edges} q| = |\operatorname{Faces} G| - 1$, we conclude

by planarity that $\operatorname{Edges} p \cup \operatorname{Edges} q = E$. □

Next, we show that, in planar embedded graphs, the notions of cycle and boundary coincide. By Lemmas 8 and 10 and the linearity of ∂_2^G, it suffices to show that every nonzero fundamental cycle is a boundary. The face chain that bears witness to this fact is given in the following lemma.

Lemma 12. *Let $G = (E, \pi)$ be a planar embedded graph and p, q be interdigitating arborescences. Let $e \in E \setminus \operatorname{Edges} p = \operatorname{Edges} q$ be arbitrary and let f be the face such that $\mathsf{edge}\, q(f) = e$.*

$$\mathsf{fundamentalCycle}_p\, q(f) = \partial_2^G \sum_{g \in \operatorname{Descendants}_q f} [g]$$

Proof. Let

$$\phi = \sum_{g \in \operatorname{Descendants}_q f} [g].$$

For every dart a, we have $(\partial_2^G \phi)(a) = \phi(\mathsf{right}_G\, a) - \phi(\mathsf{left}_G\, a)$. For every face g other than the root of q,

$$\phi(g) - \phi\big(\mathsf{left}_G\, q(g)\big) = \begin{cases} 1 & \text{if } g = f \\ 0 & \text{if } g \neq f, \end{cases}$$

since there exists no face g with $g \notin \operatorname{Descendants}_q f$ but $\mathsf{left}_G\, q(g) \in \operatorname{Descendants}_q f$, and the unique face g such that $g \in \operatorname{Descendants}_q f$ but $\mathsf{left}_G\, q(g) \notin \operatorname{Descendants}_q f$ is f. The conclusion follows from Lemmas 8 and 10. □

2.12 Edge deletion

Let $G = (E, \pi)$ be an embedded graph and let $e \in E$ be an edge. We define the embedded graph $G - e \triangleq (E', \pi')$ as follows.

$$E' \triangleq E \setminus \{e\}$$

$$\pi'(a) \triangleq \begin{cases} \pi(a) & \text{if } \mathsf{edge}\,\pi(a) \neq e \\ \pi\big(\pi(a)\big) & \text{if } \mathsf{edge}\,\pi(a) = e \text{ and } \mathsf{edge}\,\pi\big(\pi(a)\big) \neq e \\ \pi\Big(\pi\big(\pi(a)\big)\Big) & \text{if } \mathsf{edge}\,\pi(a) = e \text{ and } \mathsf{edge}\,\pi\big(\pi(a)\big) = e. \end{cases}$$

Let $g_1 = \mathsf{right}_G\,(e, 1)$ and $g_{-1} = \mathsf{right}_G\,(e, -1)$. We say that e is a **bridge** if and only if $g_1 = g_{-1}$. We prove some properties about edge deletion that we use later.

Lemma 13. *Suppose that e is not a bridge. There exists a map h from* $\operatorname{Faces} G$ *to* $\operatorname{Faces}(G - e)$ *that, for every $a \in \operatorname{Darts}\big(E \setminus \{e\}\big)$, satisfies $h(\mathsf{right}_G\, a) = \mathsf{right}_{G-e}\, a$. For every path from f_1 to f_2 in the dual of G, there exists a path from $h(f_1)$ to $h(f_2)$ in the dual of $G - e$ obtained by deleting every dart in* $\operatorname{Darts}\{e\}$*. If G is connected, then $G - e$ is connected. If G additionally is planar and $E \neq \{e\}$, then $G - e$ is planar.*

For every $\psi \in \operatorname{Cycles} G$ carried by $E \setminus \{e\}$, we have $\psi' = \psi|_{E\setminus\{e\}} \in \operatorname{Cycles}(G - e)$. If, moreover, there exists $\phi' \in \operatorname{FaceChains}(G - e)$ such that $\partial_2^{G-e}\phi' = \psi'$, then $\partial_2^G(\phi' \circ h) = \psi$. Conversely, if there exists $\phi \in \operatorname{FaceChains} G$ such that $\partial_2^G \phi = \psi$, then there exists $\phi' \in \operatorname{FaceChains}(G - e)$ such that $\phi' \circ h = \phi$ and $\partial_2^{G-e}\phi' = \psi'$.

Proof. The property required of h immediately suggests a definition, which we verify to be well founded. If $\pi\big((e, 1)\big) = (e, -1)$ and $\pi\big((e, -1)\big) = (e, 1)$, then e is the sole edge of a connected component that simply is deleted without affecting the rest of

G. Otherwise, the remaining darts of g_1 and g_{-1} are spliced into a single face of $G - e$, which we call g'. The other faces are preserved exactly.

The transfer of paths from dual to dual is valid because there is a common face corresponding to g_1 and g_{-1}. Connectivity follows by Lemma 2. Observe that, if there exists a vertex $\{a\} \in \operatorname{Vertices} G$, then $\mathsf{rev}\,\pi(a) = \mathsf{rev}\,a$, so $\mathsf{right}_G\,a = \mathsf{left}_G\,a$, and $\mathsf{edge}\,a \neq e$. There exists, then, a one-to-one correspondence between $\operatorname{Vertices}(G - e)$ and $\operatorname{Vertices} G$ where each vertex in $G - e$ is a subset of its image. Planarity follows from the fact that, in the circumstances specified, $G-e$ has, compared to G, one fewer edge, as many vertices, and one fewer face. The transfer of cycles and boundaries between G and $G - e$ is a tedious exercise in algebra. □

2.13 Shortest paths

Let $G = (E, \pi)$ be a connected embedded graph. Let $\lambda \in \mathbb{N}^{\operatorname{Darts} E}$ map each dart to its **length**. The **length** of a path $P = \langle v_0, a_1, v_1, \ldots, v_\ell \rangle$ is

$$\mathsf{length}_\lambda\,P \triangleq \sum_{i=1}^{\ell} \lambda(a_i).$$

We sometimes omit the subscript when the choice of λ is clear. The **distance** from a vertex s to a vertex t is

$$d_{G,\lambda}(s,t) \triangleq \min\{\mathsf{length}_\lambda\,P : P \text{ is an } st\text{-path}\}.$$

The minimum is well founded because the length of a path is a nonnegative integer and, by the assumption of connectivity, there exists at least one st-path. A **shortest st-path** is an st-path whose length is $d_{G,\lambda}(s,t)$. The following lemma is the **triangle**

inequality.

Lemma 14. *For every triple of vertices s, t, u, we have $d(s, u) \leq d(s, t) + d(t, u)$.*

Proof. The concatenation of a shortest st-path and a shortest tu-path is an su-path of length $d(s, t) + d(t, u)$. □

We often consider **symmetric** λ, that is, λ satisfying $\lambda = \lambda \circ \mathsf{rev}$. Under this hypothesis, distances are symmetric as well.

Lemma 15. *Suppose that λ is symmetric. For every pair of vertices s, t, we have $d(s, t) = d(t, s)$.*

Proof. The reverse of a shortest st-path is a ts-path of length $d(s, t)$. The reverse of a shortest ts-path is an st-path of length $d(t, s)$. □

A **shortest path arborescence** is an arborescence p with root r such that, for every vertex v, the path $\mathsf{path}_p\, v$ is a shortest rv-path.

Lemma 16. *For every vertex r, there exists a shortest path arborescence with root r.*

Proof. It suffices to verify the three hypotheses of Lemma 5 when $\mathcal{P}$ is the collection of all shortest paths. The first, that, for every vertex v, there exists an rv-shortest path, is clear. The second is that every subpath of a shortest path is a shortest path. Let P be a shortest st-path and Q be a uv-subpath of P. There exists a decomposition $P_1 * P_2 * P_3 = P$ such that Q is a uv-subpath of P_2. Let R be a

shortest uv-path. Since $P_1 * R * P_3$ is an st-path,

$$\mathsf{length}\, P = \mathsf{length}\, (P_1 * P_2 * P_3) \leq \mathsf{length}\, (P_1 * R * P_3)$$
$$\mathsf{length}\, Q \leq \mathsf{length}\, P_2 \leq \mathsf{length}\, R,$$

which implies that Q is a shortest uv-path. The third hypothesis is that, for every pair of rv-paths P_1, P_1' and every vw-path P_2, if $P_1 * P_2$ and P_1' are shortest paths, then $P_1' * P_2$ is a shortest path. Since $\mathsf{length}\, P_1' \leq \mathsf{length}\, P_1$, we have $\mathsf{length}\, (P_1' * P_2) \leq \mathsf{length}\, (P_1 * P_2)$. □

Lemma 17 (Henzinger, Klein, Rao, and Subramanian [1997])**.** *There exists a linear-time algorithm that, given a planar embedded graph, computes a shortest path arborescence.*

2.14 Epsilon nets

Let $G = (E, \pi)$ be a connected embedded graph with lengths $\lambda \in \mathbb{N}^{\mathrm{Darts}\, E}$. With respect to a real parameter $\epsilon > 0$ and a set of vertices K, an ϵ-**net** is a set of vertices C such that, for every $k \in K$, there exists $c \in C$ with $d(c, k) < \epsilon$.

Lemma 18. *There exists a linear-time algorithm that, given $\epsilon > 0$ and a path $P = \langle v_0, a_1, v_1, \ldots, v_\ell \rangle$, computes an ϵ-net C for $K = \{v_0, v_1, \ldots, v_\ell\}$ such that $|K| \leq 1 + \epsilon^{-1}\, \mathsf{length}\, P$.*

Proof. The algorithm is the obvious greedy one.

$L \leftarrow 0$
$C \leftarrow \{v_0\}$
$i \leftarrow 0$

for $j \leftarrow 1$ to ℓ by 1
    loop invariant: C is an ϵ-net for $\{v_0, v_1, \ldots, v_{j-1}\}$
    loop invariant: $|C| \leq 1 + \epsilon^{-1} \sum_{h=1}^{i} \mathsf{length}\, a_h$
    loop invariant: $L = \sum_{h=i+1}^{j-1} \mathsf{length}\, a_h$
    loop invariant: $v_i \in C$
    $L \leftarrow L + \mathsf{length}\, a_j$
    if $L \geq \epsilon$ then
        $L \leftarrow 0$
        $C \leftarrow C \cup \{v_j\}$
        $i \leftarrow j$
    end if
end for

This algorithm is clearly linear-time. Given the invariants, whose proofs are straightforward, the correctness of this algorithm is clear as well. □

2.15 Multitriangulations

A **multitriangulation** (by analogy with "multigraph") is an embedded graph in which each face is comprised of at most three darts. The following lemma implies that, for problems concerning distances between vertices of a connected embedded graph with lengths, we may assume without loss of generality that we are dealing with a multitriangulation.

Lemma 19. *Let $G = (E, \pi)$ be a connected embedded graph with lengths $\lambda \in \mathbb{N}^{\operatorname{Darts} E}$. There exists a linear-time algorithm to compute a connected multitriangulation $G' = (E', \pi')$ and lengths $\lambda' \in \mathbb{N}^{\operatorname{Darts} E'}$ having the following properties.*

- $E \subseteq E'$.
- *For every $v' \in \operatorname{Vertices} G'$, there exists $v \in \operatorname{Vertices} G$ such that $v \subseteq v'$. (This*

implies the existence of a natural one-to-one correspondence between Vertices G *and* Vertices G'*.)*

- *For every pair* $a, b \in \operatorname{Darts} E$*,*

$$d_{G,\lambda}(\mathsf{head}_G\, a,\ \mathsf{head}_G\, b) = d_{G',\lambda'}(\mathsf{head}_{G'}\, a,\ \mathsf{head}_{G'}\, b).$$

- *If* G *is planar, then* G' *is planar as well.*

Proof. We induct the quantity

$$\Phi G = \sum_{f \in \operatorname{Faces} G} \max\bigl(|f| - 3,\ 0\bigr).$$

We use Φ also as a potential function for analyzing the running time of the obvious algorithm that applies greedily and repeatedly the subdivision operation described below. Observe that $0 \le \Phi G \le 2|E| - 3$. We have $\Phi G = 0$ if and only if G is a multitriangulation, in which case we take $G' = G$ and $\lambda' = \lambda$.

If $\Phi G > 0$, then let $a_1 \in \operatorname{Darts} E$ satisfy $|\mathsf{right}_G\, a_1| > 3$. Let $e' \notin E$ be a new edge and define

$$E' = E \cup \{e'\} \qquad a' = (e, 1)$$
$$a_2 = \mathsf{rev}\, \pi(a_1) \qquad a_3 = \mathsf{rev}\, \pi(a_2).$$

Observe that $a_1 \neq a_2 \neq a_3 \neq a_1$. We define π' so as to **subdivide** $\mathsf{right}\, a_1$.

$$\pi'(a) = \begin{cases} a' & \text{if } a = a_1 \\ \pi(a_1) & \text{if } a = a' \\ \mathsf{rev}\, a' & \text{if } a = a_3 \\ \pi(a_3) & \text{if } a = \mathsf{rev}\, a' \\ \pi(a) & \text{if } a \notin \{a_1,\ a',\ a_3,\ \mathsf{rev}\, a'\} \end{cases}$$

The lengths of a' and $\mathsf{rev}\, a'$ are defined to be the lengths of paths that can replace them. Distances do not increase because every dart in G exists in G' and has the same length. Distances do not decrease because both a and a' can be avoided by using two other darts having the same total length.

$$\lambda'(a) = \begin{cases} \lambda(\mathsf{rev}\, a_3) + \lambda(\mathsf{rev}\, a_2) & \text{if } a = a' \\ \lambda(a_2) + \lambda(a_3) & \text{if } a = \mathsf{rev}\, a' \\ \lambda(a) & \text{if } a \notin \{a',\ \mathsf{rev}\, a'\} \end{cases}$$

Compared to G, the embedded graph G' has one fewer face of cardinality $|\mathsf{right}_G\, a_1|$, one more face of cardinality $|\mathsf{right}_G\, a_1| - 1$, and one more face of cardinality 3. We conclude that $\Phi G' = \Phi G - 1$. Compared to G, the embedded graph G' has the same number of vertices, one more edge, and one more face, so if G is planar, then G' is planar as well. $\square$

CHAPTER **Three**

Steiner forest

Given an undirected graph G, lengths $\lambda \in \mathbb{Z}_+^{E(G)}$, and a list of pairs of **terminals** $s_1, t_1, s_2, t_2, \ldots, s_m, t_m \in V(G)$, the **Steiner forest problem** is to find a set of edges $F \subseteq E(G)$ minimizing $\mathsf{cost}_\lambda F \triangleq \sum_{e \in F} \lambda_e$ subject to the condition that, in the subgraph $\big(V(G), F\big)$, for every $i \in \{1, 2, \ldots, m\}$, the vertices s_i and t_i are connected.

In this chapter, we present an approximation algorithm for graphs G of bounded branchwidth, which is the bottleneck of the first polynomial-time approximation scheme for planar G, due to Bateni et al. [2011].

Theorem 20 (Eisenstat, Klein, and Mathieu [2012])**.** *There exists a polynomial-time approximation scheme for the Steiner forest problem in planar graphs, which uses the reduction of Bateni et al. [2011] to the problem of approximating Steiner forest in graphs of bounded branchwidth. This scheme is efficient in the sense that its running time is bounded by a fixed polynomial times a function of the approximation parameter ϵ.*

The original write-up makes a number of other improvements to the running time, resulting in a near linear-time scheme.

The Steiner forest problem generalizes the Steiner tree problem, also NP-hard, which involves connecting each of a given set of terminals to each other terminal. In turn, Steiner tree generalizes the polynomial-time solvable minimum spanning tree problem, and in fact, there is a simple 2-approximation for Steiner tree that involves computing the minimum spanning tree in a complete graph on the terminals, where the edge lengths are distances in the original graph. Better approximations are known: at the end of a long sequence of combinatorial approximation algorithms by several authors, Robins and Zelikovsky [2005] gave a polynomial-time 1.55-approximation, which was followed by a 1.39-approximation based on linear programming and a new technique for randomized rounding [Byrka, Grandoni, Rothvoss, and Sanità, 2013]. Conversely, Bern and Plassmann [1989] showed that, unless P = NP, there exists a constant $c > 1$ such that no polynomial-time c-approximation exists. This lower bound applies to the special case of Steiner tree where the length of every edge is either 1 or 2. Bern and Plassmann also gave a 4/3-approximation for this case, followed much later by a 1.25-approximation due to Berman, Karpinski, and Zelikovsky [2009].

On Steiner forest in general graphs, there has been much less progress. Agrawal, Klein, and Ravi [1991] gave a 2-approximation, which is still the best ratio known. Goemans and Williamson [1995] later gave another 2-approximation, via a technique that generalizes to other connectivity problems. For the special case with edge lengths 1 and 2, Berman, Karpinski, and Zelikovsky [2010] gave a 3/2-approximation.

For Steiner forest in planar graphs, the aforementioned polynomial-time approximation scheme due to Bateni et al. was the first. The new ingredients were a

partitioning method and an approximation scheme for Steiner forest in graphs of bounded branchwidth, which they combined with a spanner construction introduced by Borradaile, Klein, and Mathieu [2009] in their approximation scheme for Steiner tree in planar graphs. Both schemes as well as ours are examples of a paradigm pioneered by Klein [2008] in approximation schemes for the planar traveling salesman problem and later the subset traveling salesman problem [Klein, 2006]. The steps are to find a short subgraph that approximately preserves the objective, called a *spanner*; partition the spanner, often via Baker's technique; solve each part (approximately) optimally, often using a dynamic program; and then assemble the partial solutions.

Steiner tree was studied originally in a Euclidean setting, where the terminals are points in a Euclidean space, and the tree consists of line segments between arbitrary points. Arora [1998] gave a polynomial-time approximation scheme for low-dimensional Euclidean spaces (e.g., the plane), upon which Borradaile, Klein, and Mathieu [2008] built their scheme for Euclidean Steiner forest. Independently, Mitchell [1999] gave another polynomial-time approximation scheme for Steiner tree in the Euclidean plane.

3.1 Branch decompositions

A **cluster** $C = (V, E, \partial C)$ consists of an undirected graph (V, E) together with a set of **boundary vertices** $\partial C \subseteq V$. A **branch subdecomposition** is a set of related clusters specified recursively as follows. The base case of the recursion is that every one-edge cluster C gives rise to a branch subdecomposition $\mathcal{B} = \{C\}$, with root cluster C. The inductive case is, given two branch subdecompositions

$\mathcal{B}_1, \mathcal{B}_2$ with root clusters $C_1 = (V_1, E_1, \partial C_1)$ and $C_2 = (V_2, E_2, \partial C_2)$ respectively satisfying $V_1 \cap V_2 \subseteq \partial C_1 \cap \partial C_2$ and $E_1 \cap E_2 = \varnothing$, for every $\partial C \subseteq \partial C_1 \cup \partial C_2$ satisfying $(\partial C_1 \cup \partial C_2) \setminus (\partial C_1 \cap \partial C_2) \subseteq \partial C$, the set $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2 \cup \{C\}$ is a branch subdecomposition, where the root cluster C is defined by $C = (V_1 \cup V_2, E_1 \cup E_2, \partial C)$. The **width** of a branch subdecomposition $\mathcal{B}$ is $\max\{|\partial C| : C \in \mathcal{B}\}$. A **branch decomposition** is a branch subdecomposition whose root cluster C satisfies $\partial C = \varnothing$. The notion of branch decomposition is due to Robertson and Seymour [1991].

3.2 An exact dynamic program

In this section, we present a dynamic program for Steiner forest on a branch decomposition. While this program does not run in polynomial time, it is a useful step in presenting our efficient approximation. Like many previous algorithms that operate on a branch decomposition, this algorithm is a straightforward bottom-up dynamic program. The interesting parts are how the subproblems are specified (conformance) and how they can be combined (compatibility).

An **equivalence relation** is a binary relation that is reflexive, symmetric, and transitive. We make use of three definitions involving equivalence relations. The first is, given an equivalence relation $\equiv$ on a set X and an element $x \in X$, we have $x/{\equiv} \triangleq \{y : y \equiv x\}$, the equivalence class to which x belongs. The second is, given an equivalence relation $\equiv$ on a set X and a subset $Y \subseteq X$, we have $Y/{\equiv} \triangleq \{x/{\equiv} : x \in Y\}$, the set of equivalence classes generated by Y. The third is, given two equivalence relations $\equiv_1, \equiv_2$ on sets X_1, X_2 respectively, the notation $\equiv_1 \vee \equiv_2$ means the finest equivalence relation $\equiv$ on $X_1 \cup X_2$ such that, for $j \in \{1, 2\}$ and every $x, y \in X_j$, if $x \equiv_j y$, then $x \equiv y$.

We first generalize the Steiner forest problem to the following **subproblem**. The inputs are a cluster $C = (V, E, \partial C)$, lengths $\lambda \in \mathbb{Z}_+^E$, pairs of terminals $s_1, t_1, s_2, t_2, \ldots, s_m, t_m$ not necessarily belonging to V, equivalence relations $\equiv^{\text{ext}}, \equiv^{\text{int}}$ on ∂C, and a map $\alpha : A \to \partial C/(\equiv^{\text{ext}} \vee \equiv^{\text{int}})$, where

$$A \triangleq \big(\{s_i : t_i \notin V \setminus \partial C\} \cup \{t_i : s_i \notin V \setminus \partial C\}\big) \cap (V \setminus \partial C)$$

is the set of **active** terminals. The goal is to find a **subsolution** $F \subseteq E$ minimizing $\mathsf{cost}_\lambda F$ subject to several connectivity conditions.

Inactive terminals For every $j \in \{1, 2, \ldots, m\}$ with $s_i, t_i \in V \setminus \partial C$, at least one of two possibilities holds. The first possibility is that there exists a path in the graph (V, F) from s_i to t_i. The second possibility is that there exist vertices $s', t' \in \partial C$ with s' $(\equiv^{\text{ext}} \vee \equiv^{\text{int}})$ t' and paths in the graph (V, F) from s to s' and from t to t'.

Boundary vertices For every pair of vertices $u, v \in \partial C$ with $u \equiv^{\text{int}} v$, there exists a path in the graph (V, F) from u to v.

Active terminals For every active terminal $u \in A$, there exists a path in the graph (V, F) from u to some boundary vertex in the equivalence class $\alpha(u)$.

For clusters C with $\partial C = \varnothing$, it can be seen that this subproblem simplifies to the Steiner forest problem. For later use, we make two definitions.

$$\hat{A} \triangleq A \cup \partial C \qquad\qquad \hat{\alpha}(u) \triangleq \begin{cases} \alpha(u) & \text{if } u \in A \\ u/(\equiv^{\text{ext}} \vee \equiv^{\text{int}}) & \text{if } u \in \partial C. \end{cases}$$

Observe that the "Active terminals" condition extends readily to the analogous state-

ment involving $\hat{A}$ and $\hat{\alpha}$.

3.2.1 Compatibility

The dynamic program works by joining repeatedly two subsolutions into one. With respect to an instance of the Steiner forest problem on a branch decomposition $\mathcal{B}$, and a particular cluster $C \in \mathcal{B}$, a **configuration** is a triple $(\equiv^{\text{ext}}, \equiv^{\text{int}}, \alpha)$ of inputs needed to specify a subproblem for C. Compatibility is a sufficient condition for a successful join. With respect to three clusters $C_0, C_1, C_2 \in \mathcal{B}$ where C_0 is the parent of C_1 and C_2, three configurations $(\equiv_j^{\text{ext}}, \equiv_j^{\text{int}}, \alpha_j)$ for $j \in \{0,1,2\}$ are **compatible** if and only if the following conditions are met.

Parent's internal For every $u, v \in \partial C_0$ with $u \equiv_0^{\text{int}} v$, the relation u $(\equiv_1^{\text{int}} \vee \equiv_2^{\text{int}})$ v holds.

Parent's connectivity For every $u, v \in \partial C_0$, the relation u $(\equiv_0^{\text{ext}} \vee \equiv_0^{\text{int}})$ v holds if and only if u $(\equiv_0^{\text{ext}} \vee \equiv_1^{\text{int}} \vee \equiv_2^{\text{int}})$ v.

Children's external For $j \in \{1,2\}$ and every $u, v \in \partial C_j$ with $u \equiv_j^{\text{ext}} v$, the relation u $(\equiv_0^{\text{ext}} \vee \equiv_{3-j}^{\text{int}})$ v holds.

Parent's assignment For every $u \in A_0$, there exist $j \in \{1,2\}$ and $v \in \hat{\alpha}_j(u)$ and $w \in \alpha_0(u)$ such that v $(\equiv_1^{\text{int}} \vee \equiv_2^{\text{int}})$ w.

Deactivated terminals For every terminal pair s_i, t_i satisfying $\{s_i, t_i\} \subseteq V_j \setminus \partial C_j$ for $j = 0$ but neither $j = 1$ nor $j = 2$, there exist $k \in \{1,2\}$ and $s' \in \hat{\alpha}_k(s_i)$ and $t' \in \hat{\alpha}_{3-k}(t_i)$ such that s' $(\equiv_0^{\text{ext}} \vee \equiv_1^{\text{int}} \vee \equiv_2^{\text{int}})$ t'.

Following a technical proposition, we prove the sufficiency of compatibility in joining subsolutions.

Proposition 21. *Let $\equiv_1, \equiv_2$ be equivalence relations on sets X_1, X_2 respectively. For every $x \in X_1$ and $y \in X_2$ with $x\ (\equiv_1 \vee \equiv_2)\ y$, there exists $z \in X_1 \cap X_2$ such that $x \equiv_1 z$ and $z\ (\equiv_1 \vee \equiv_2)\ y$.*

Lemma 22. *Fix an instance of Steiner forest on a branch decomposition $\mathcal{B}$. Let $C_0, C_1, C_2 \in \mathcal{B}$ be arbitrary clusters such that C_0 is the parent of C_1 and C_2 and let $(\equiv_j^{\text{ext}}, \equiv_j^{\text{int}}, \alpha_j)$ for $j \in \{0, 1, 2\}$ be compatible configurations. For every subsolution $F_1 \subseteq E_1$ with respect to C_1 and $F_2 \subseteq E_2$ with respect to C_2, the set $F_0 = F_1 \cup F_2$ is a subsolution with respect to C_0.*

Proof. We first prove the "Boundary vertices" property for F_0. Fix arbitrary $u, v \in \partial C_0$ with $u \equiv_0^{\text{int}} v$. By compatibility ("Parent's internal"), we have $u\ (\equiv_1^{\text{int}} \vee \equiv_2^{\text{int}})\ v$. It follows that there exists a path from u to v with subpaths belonging alternately to F_1 and F_2, which path is in F_0.

Now we prove the "Active terminals" property for F_0. Fix an arbitrary terminal $u \in A_0$. By compatibility ("Parent's assignment"), there exist $j \in \{1, 2\}$ and $v \in \hat{\alpha}_j(u)$ and $w \in \alpha_0(u)$ such that $v\ (\equiv_1^{\text{int}} \vee \equiv_2^{\text{int}})\ w$. By the "Active terminals" property for C_j, there exists a path in F_j from u to some $v' \in \hat{\alpha}_j(u)$, which by definition satisfies $v'\ (\equiv_j^{\text{ext}} \vee \equiv_j^{\text{int}})\ v$. Again by compatibility ("Children's external"), we have $v'\ (\equiv_0^{\text{ext}} \vee \equiv_1^{\text{int}} \vee \equiv_2^{\text{int}})\ w$, which implies by Proposition 21 and "Parent's connectivity" that there exists $w' \in \alpha_0(u)$ such that $v'\ (\equiv_1^{\text{int}} \vee \equiv_2^{\text{int}})\ w'$. We conclude that there exists a path in F_0 from u to w', via v'.

Finally, we prove the "Inactive terminals" property for F_0. For every terminal pair s_i, t_i not connected by F_1 or F_2, there exist vertices $s', t' \in \partial C_1 \cup \partial C_2$ with

s' ($\equiv_0^{\text{ext}} \vee \equiv_1^{\text{int}} \vee \equiv_2^{\text{int}}$) t' and paths in F_0 from s to s' and from t to t'. This statement follows in one of two ways. The first is that there exists $j \in \{1, 2\}$ such that s_i, t_i are subject to the "Inactive terminals" condition for F_j. The second is by compatibility ("Deactivated terminals") and the "Active terminals" conditions for F_1 and F_2. In both cases, we invoke compatibility again ("Children's external") to replace $\equiv_j^{\text{ext}}$ by $\equiv_0^{\text{ext}} \vee \equiv_{3-j}^{\text{int}}$. If s' ($\equiv_1^{\text{int}} \vee \equiv_2^{\text{int}}$) t', then there exists a path in F_0 from s to t. Otherwise, there exists a vertex $u \in \partial C_0$ belonging to the same equivalence class of $\equiv_0^{\text{ext}} \vee \equiv_1^{\text{int}} \vee \equiv_2^{\text{int}}$ as s' and t'. We apply Proposition 21 twice as before to obtain $s'', t'' \in \partial C_0$ with s'' ($\equiv_0^{\text{ext}} \vee \equiv_0^{\text{int}}$) t'' and paths from s to s'' and from t to t'', as required. □

3.2.2 A structure lemma

Of course, the definition of compatibility would be useless if it excluded valid solutions. We address this concern with the following lemma.

Lemma 23. *Let $G = (V, E)$ and λ and $s_1, t_1, s_2, t_2, \ldots, s_m, t_m$ be an instance of Steiner forest. Let $\mathcal{B}$ be a branch decomposition having root cluster $(V, E, \varnothing)$. For every solution $F \subseteq E$, there exists a collection of compatible configurations $(\equiv_C^{\text{ext}}, \equiv_C^{\text{int}}, \alpha_C)$ indexed by clusters $C \in \mathcal{B}$ such that, for every $C \in \mathcal{B}$, the set $F \cap E_C$ is a subsolution with respect to C, where $E_C \subseteq E$ is the set of edges in C.*

In particular, we choose $\equiv_C^{\text{ext}}$ to be the connectivity relation on ∂C induced by the edges $F \setminus E_C$ and $\equiv_C^{\text{int}}$ to be the connectivity relation on ∂C induced by the edges $F \cap E_C$. For every active terminal $u \in A_C$, we define $\alpha_C(u) = v/(\equiv_C^{\text{ext}} \vee \equiv_C^{\text{int}})$, where $v \in \partial C$ is some vertex to which u is connected by edges in $F \cap E_C$.

Proof. There are eight different conditions that need to be proved: three for being a subsolution and five for compatibility. We fix an arbitrary cluster $C \in \mathcal{B}$ and start with the "Inactive terminals" condition. Let i satisfy $s_i, t_i \in V_C \setminus \partial C$ and consider a path in F from s_i to t_i. Unless this path uses only edges in E_C, there exist initial and final subpaths in $F \cap E_C$ from s_i to some $s' \in \partial C$ and from some $t' \in \partial C$ to t_i respectively. Since s' and t' are connected by ∂C-to-∂C subpaths alternately contained by $F \setminus E_C$ and $F \cap E_C$, implying that $s' \, (\equiv_C^{\text{ext}} \vee \equiv_C^{\text{int}}) \, t'$, as required.

After the "Boundary vertices" condition, which follows immediately from the definition of $\equiv_C^{\text{int}}$, the only condition remaining for being a subsolution is "Active terminals". In fact, "Active terminals" is satisfied almost by definition as well; we just need to show that some suitable path exists. Every (active) terminal $u \in A_C \subseteq V_C \setminus \partial C$ has a mate $v \notin V_C \setminus \partial C$, which implies the existence of a path from u to v in F. Clearly, some prefix of this path using only edges in $F \cap E_C$ touches some vertex in ∂C.

To finish, we prove compatibility. Let $C_j = (V_j, E_j, \partial C_j) \in \mathcal{B}$ indexed by $j \in \{0, 1, 2\}$ be clusters such that C_0 is a join of C_1 and C_2. Let $(\equiv_j^{\text{ext}}, \equiv_j^{\text{int}}, \alpha_j)$ indexed by $j \in \{0, 1, 2\}$ be the configurations associated with those clusters (abbreviating the subscript C_j to j). Of the five conditions to be checked, the first is "Parent's internal". Let $u, v \in \partial C_0$ satisfy $u \equiv_0^{\text{int}} v$. By the definition of $\equiv_0^{\text{int}}$, there exists a path in $F \cap E_0$ from u to v. This path alternates ∂C_0-to-∂C_0 subpaths in $F \cap E_1$ and $F \cap E_2$, which implies that $u \, (\equiv_1^{\text{int}} \vee \equiv_2^{\text{int}}) \, v$. The second and third conditions are "Parent's connectivity" and "Children's external", which are proved by similar arguments about subpaths.

The fourth condition is "Parent's assignment". Let $u \in A_0$ be an arbitrary terminal. Since $A_0 \subseteq V_0 = V_1 \cup V_2$, there exists $j \in \{1, 2\}$ such that $u \in V_j$, which

implies that $u \in \hat{A}_j = A_j \cup \partial C_j$. We now apply the "Active vertices" condition twice. In $F \cap E_j$, there exists a path from u to some vertex $v \in \hat{\alpha}_j(u)$. In $F \cap E_0$, there exists a path from u to some vertex $w \in \alpha_0(u)$. It follows that there exists a path from v to w in $F \cap E_0$ and hence that $v \; (\equiv_1^{\text{int}} \vee \equiv_2^{\text{int}}) \; w$.

The fifth and final condition is "Deactivated terminals". Fix an arbitrary terminal pair s_i, t_i that satisfies $\{s_i, t_i\} \subseteq V_j \setminus \partial C_j$ for $j = 0$ but neither $j = 1$ nor $j = 2$. There exists $k \in \{1, 2\}$ such that $s_i \in \hat{A}_k$ and $t_i \in \hat{A}_{3-k}$. Since s_i, t_i are mates, there exists a path in F from one to the other. Combining this path with the paths existing by "Active terminals" from s_i to some vertex $s' \in \hat{\alpha}_k(s_i)$ and from some vertex $t' \in \hat{\alpha}_k(t_i)$ to t_i, we conclude that there exists a path from s' to t' in F and hence that $s' \; (\equiv_0^{\text{ext}} \vee \equiv_1^{\text{int}} \vee \equiv_2^{\text{int}}) \; t'$. □

3.3 An approximating dynamic program

The number of configurations dictates up to fixed polynomial factors the running time of the previous algorithm. The main contributor to this number is α, since the number of possibilities for $\equiv^{\text{ext}}$ and $\equiv^{\text{int}}$ is independent of $|V|$. In this section, we define a polynomial-size set of **representative** possibilities for α, giving rise to a polynomial number of **representative** configurations that choose α to be one of these possibilities. We prove an approximate structure lemma to the effect that, for every solution, there exists a relatively small patch such that a dynamic program considering only representative configurations can find a solution at least as good as the patched one. This dynamic program runs in polynomial time.

Let $\epsilon > 0$ be the approximation parameter and k be the width of the branch

decomposition. Without loss of generality, we assume that there exists a positive integer n such that $\epsilon^{-1} = n$. With respect to a cluster C and arbitrary choices for $\equiv^{\text{ext}}, \equiv^{\text{int}}$, the **representative** choices of α are those that can be constructed in the following manner. We choose an enumeration $\langle u_1, u_2, \ldots, u_{k'} \rangle$ of a subset of ∂C. For each boundary vertex u_i, we choose a radius $r_i > 0$. For convenience, we let r_i be arbitrary, but for the purpose of the dynamic program, it suffices to try ∞ and the at most $|V|\big(|V|+1\big)/2$ distinct positive distances in C. Finally, for each boundary vertex u_i, we choose a set of centers $Z_i \subseteq V$ with $|Z_i| < 2k\epsilon^{-1}+1$. We define pairwise disjoint sets A_i as follows.

$$A'_i = A \cap \bigcup_{z \in Z_i} \big\{v : v \in V,\ d(z,v) < r_i\big\} \qquad\qquad A_i = A'_i \setminus \bigcup_{j=1}^{i-1} A'_j$$

Whenever these sets have union A, we define α by stipulating, for every $v \in A_i$, that $\alpha(v) = u_i/(\equiv^{\text{ext}} \vee \equiv^{\text{int}})$. It is clear that, for every choice of ϵ and k, there exists an exponent p such that the number of representative α is at most $|V|^p$.

Lemma 24. *Let $\epsilon > 0$ be the approximation parameter. Let $G = (V, E)$ and λ and $s_1, t_1, s_2, t_2, \ldots, s_m, t_m$ be an instance of Steiner forest. Let $\mathcal{B}$ be a branch decomposition having root cluster $(V, E, \varnothing)$. For every solution $F \subseteq E$, there exists a solution $F' \supseteq F$ with $\mathsf{cost}_\lambda F' \leq (1+\epsilon)\,\mathsf{cost}_\lambda F$ and a collection of compatible* representative *configurations $(\equiv^{\text{ext}}_C, \equiv^{\text{int}}_C, \alpha_C)$ indexed by clusters $C \in \mathcal{B}$ such that, for every $C \in \mathcal{B}$, the set $F' \cap E_C$ is a subsolution with respect to C, where $E_C \subseteq E$ is the set of edges in C.*

In particular, we choose $\equiv^{\text{ext}}_C$ to be the connectivity relation on ∂C induced by the edges $F' \setminus E_C$ and $\equiv^{\text{int}}_C$ to be the connectivity relation on ∂C induced by the edges $F' \cap E_C$. For every active terminal $u \in A_C$, we define $\alpha_C(u) = v/(\equiv^{\text{ext}}_C \vee \equiv^{\text{int}}_C)$, where $v \in \partial C$ is some vertex to which u is connected by edges in $F' \cap E_C$.

Proof. We construct F' and show that the resulting configurations are representative. The rest of the proof follows Lemma 23.

A **component** is a maximal nonempty set of edges $T \subseteq F$ such that the graph (V, T) is connected. For every component T, the **index** of T is the maximum depth of a cluster $C \in \mathcal{B}$ such that T is a subset of the edges of C but is not incident to ∂C. To construct F', we first determine the witnesses for α being representative. Consider a particular cluster $C \in \mathcal{B}$ and let $\langle T_1, T_2, \ldots, T_{k'} \rangle$ be an enumeration with nonincreasing index of the components in (V, F) that contain an edge incident to ∂C. Observe that, by maximality of each T_i, we have $k' \leq |\partial C| \leq k$. Let $\langle u_1, u_2, \ldots, u_{k'} \rangle$ be a corresponding enumeration of a subset of ∂C such that every connected component T_i contains an edge incident to u_i. Let $r_i = (2k)^{-1} \epsilon \, \mathsf{cost}_\lambda \, T_i$. There exists a path of length less than $2 \, \mathsf{cost}_\lambda \, T_i$ that visits every edge in T_i at least once and no other edges, so by Lemma 18, there exists a set of centers Z_i with $|Z_i| < 2k\epsilon^{-1} + 1$ such that every center in Z_i is incident to T_i and such that A'_i contains every active terminal incident to T_i.

In general, we have $A_i \neq A'_i$, and the assignment α determined by our witnesses does not comport with the assignment dictated by F. After initializing $F' \leftarrow F$, we take the following action for every cluster C. For every pair i, j with $i < j$ and $A'_i \cap A'_j \neq \varnothing$, we update $F' \leftarrow F' \cup P$, where P is a shortest path from some vertex incident to T_i to some vertex incident to T_j, unless T_i and T_j already have been joined in this way. The cumulative effect of these updates is to rectify every disagreement between the assignments, by connecting the offending boundary vertices.

The only remaining issue is bounding $\mathsf{cost}_\lambda \, (F' \setminus F)$. We "charge" the cost of each P to T_i and show that each component T is charged at most $2k$ times in the amount of at most $(2k)^{-1} \epsilon \, \mathsf{cost}_\lambda \, T$ each time, which, by the fact that distinct components

are disjoint, proves the stated bound, $\epsilon \, \mathsf{cost}_\lambda \, F$.

Each time T_i is charged for P, there exist $j > i$, an active terminal $v \in A'_j$, and a center $z \in Z_i$ such that $\mathsf{cost}_\lambda \, P \leq d(z, v) < r_i = (2k)^{-1} \epsilon \, \mathsf{cost}_\lambda \, T_i$. To bound the number of charges, let T be an arbitrary component and $C_0 \in \mathcal{B}$ be the deepest cluster such that T is a subset of the edges of C_0 but is not incident to ∂C_0. In other words, C_0 is the cluster whose depth determines the index of T. Let T' be a component that causes T to be charged. There exists a cluster $C' \in \mathcal{B}$ such that both T and T' are incident to $\partial C'$. Since C' is hence a proper descendant of C_0, the latter has two children, which we call C_1 and C_2.

We consider three cases. The first case is that T' is not a subset of the edges of C_0. In this case, T', being connected, is incident to ∂C_0. The second case is that, for every $i \in \{1, 2\}$, we have that T' is not a subset of the edges of T_i. Being connected, T' is incident to $\partial C_1 \cap \partial C_2$. The third case is that there exists $i \in \{1, 2\}$ such that T' is a subset of the edges of T_i. Since the index of T' is not greater than the index of T, we conclude that T' is incident to ∂C_i. In all cases, T' is incident to $\partial C_1 \cup \partial C_2 \supseteq \partial C_0$, and $|\partial C_1 \cup \partial C_2| \leq 2k$. With more work, this bound can be improved to $\lfloor \frac{3}{2} k \rfloor - 1$. □

3.4 An efficient approximating dynamic program

While the approximation scheme of the previous section is polynomial-time, the exponent of the polynomial depends on ϵ and k. In this section, we present a scheme that runs in time bounded by a fixed polynomial times a constant depending on ϵ and k, that is, an **efficient** scheme. This scheme is not strictly better than previous one

because it makes a weaker guarantee about the quality of the solution. Specifically, given an instance of Steiner forest $G = (V, E)$ and λ and $s_1, t_1, s_2, t_2, \ldots, s_m, t_m$, the quality of the output solution F' satisfies for every solution F the inequality $\mathsf{cost}_\lambda F' \leq \mathsf{cost}_\lambda F + \epsilon\, \mathsf{cost}_\lambda E$. The approximation term $\epsilon\, \mathsf{cost}_\lambda E$ in general may be quite a lot larger than $\epsilon\, \mathsf{cost}_\lambda F$, but in the context of the approximation scheme for planar graphs, this is not a problem.

Our main goal is to reduce the number of representative configurations to polylogarithmic and prove a new structure lemma. (With more work, this number can be reduced to polyloglog [Eisenstat et al., 2012].) The low-hanging fruit is to reduce the number of possibilities for r_i and Z_i by tolerating some imprecision in the former and adjusting each element of the latter to the nearest element of a fixed r_i-net. There still may be too many possibilities, but in that case, part of the graph is very dense in the sense of there being edges with large total cost within a short distance of one another. We contract that part and charge to it the cost of patching over it.

We massage the input graph to make it easier to handle. First, at the expense of doubling the width, balance the branch decomposition to have depth at most $O\big(\log_2 |E|\big)$ [Eisenstat et al., 2012]. Second, subdivide each edge e into

$$\left\lceil \frac{\lambda(e)}{\frac{1}{4}\epsilon |E|^{-1}\, \mathsf{cost}_\lambda E} \right\rceil$$

edges of length 1, which, relatively speaking with respect to the scaling factor in the denominator, increases the cost of each solution by less than $\frac{1}{4}\epsilon\, \mathsf{cost}_\lambda E$. No solution cost is strictly decreased. The number of edges in the new graph is $O\big(\epsilon^{-1}|E|\big)$. The new branchwidth is at most $\max\{2k, 2\}$. The depth of the branch decomposition is defined to be $h - 1 = O\big(\log_2 \epsilon^{-1}|E|\big)$, so that each edge belongs to at most h clusters.

We redefine the notion of a representative configuration with respect to a cluster $C = (V, E, \partial C)$. We start out by contracting the edges belonging to a dense region near the boundary. Let $d_{C,\lambda}$ denote distances in the metric graph specified by C and λ and define

$$\begin{aligned}\operatorname{VertexMargin}_\lambda z &= \big\{v : u \in \partial C,\ d_{C,\lambda}(u, v) \leq z\big\} \\ \operatorname{EdgeMargin}_\lambda z &= \big\{\{v, w\} : \{v, w\} \in E,\ \{v, w\} \subseteq \operatorname{VertexMargin}_\lambda z\big\}.\end{aligned}$$

Let $\theta \in \mathbb{N}$ be maximal such that

$$\mathsf{cost}_\lambda \operatorname{EdgeMargin}_\lambda \theta \geq 8hk^2\epsilon^{-1}\theta.$$

This definition is well founded because $\theta = 0$ is a solution, the left-hand side is bounded in θ, and the right-hand side is not. Recalling that λ has been massaged to be the constant function 1, define

$$\lambda'(e) = \begin{cases} 0 & \text{if } e \in \operatorname{EdgeMargin}_\lambda \theta \\ 1 & \text{if } e \notin \operatorname{EdgeMargin}_\lambda \theta. \end{cases}$$

Observe that $\operatorname{EdgeMargin}_{\lambda'} z = \operatorname{EdgeMargin}_\lambda (\theta + z)$, from which we conclude by the maximality of θ that, for every $z > 0$,

$$\mathsf{cost}_\lambda \operatorname{EdgeMargin}_{\lambda'} z < 8hk^2\epsilon^{-1}z.$$

We use this bound to control the number of possibilities for Z_i.

As before, α is computed from r_i and Z_i. Each radius r_i now belongs to the set $\{2^{q-3} : q \in \mathbb{N},\ 2^q < 2\,\mathsf{cost}_\lambda E\}$, which, by the fact that $\mathsf{cost}_\lambda E = |E|$, has logarithmic size. Each set of centers Z_i is a subset of a fixed r_i-net chosen at the outset that covers

$\mathrm{VertexMargin}_\lambda\, 16k\epsilon^{-1}r_i$. The set Z_i has cardinality at most $16k\epsilon^{-1}+1$, and, by k invocations of Lemma 18, the r_i-net has cardinality less than $256hk^3\epsilon^{-2}+k$, which is logarithmic. We conclude that the total number of representative configurations now is polylogarithmic. The seemingly arbitrary quantities are justified by the proof of the following lemma.

Lemma 25. *Let $\epsilon > 0$ be the approximation parameter. Let $G = (V, E)$ and λ and $s_1, t_1, s_2, t_2, \ldots, s_m, t_m$ be an instance of Steiner forest. Let $\mathcal{B}$ be a branch decomposition having root cluster $(V, E, \varnothing)$. For every solution $F \subseteq E$, there exists a solution $F' \supseteq F$ with $\mathsf{cost}_\lambda F' \leq \mathsf{cost}_\lambda F + \epsilon\, \mathsf{cost}_\lambda E$ and a collection of compatible* representative *configurations $(\equiv_C^{\mathrm{ext}}, \equiv_C^{\mathrm{int}}, \alpha_C)$ indexed by clusters $C \in \mathcal{B}$ such that, for every $C \in \mathcal{B}$, the set $F' \cap E_C$ is a subsolution with respect to C, where $E_C \subseteq E$ is the set of edges in C.*

In particular, we choose $\equiv_C^{\mathrm{ext}}$ to be the connectivity relation on ∂C induced by the edges $F' \setminus E_C$ and $\equiv_C^{\mathrm{int}}$ to be the connectivity relation on ∂C induced by the edges $F' \cap E_C$. For every active terminal $u \in A_C$, we define $\alpha_C(u) = v/(\equiv_C^{\mathrm{ext}} \vee \equiv_C^{\mathrm{int}})$, where $v \in \partial C$ is some vertex to which u is connected by edges in $F' \cap E_C$.

Proof. We present only the arguments not present in the proof of Lemma 24. Given a component T_i, we make the unique choice of r_i satisfying $\frac{1}{8}r_i \leq (2k)^{-1}\epsilon\, \mathsf{cost}_\lambda T_i < \frac{1}{4}r_i$. Accordingly, $T_i \subseteq \mathrm{EdgeMargin}_\lambda\, 16k\epsilon^{-1}r_i$. We construct an r_i-net Z_i' covering T_i as before and then obtain Z_i by replacing each vertex in Z_i' with the closest vertex in the fixed r_i-net of which Z_i must be a subset. This Z_i is a $2r_i$-net covering T_i, and the total patching cost relative to λ' is at most $\frac{1}{2}\epsilon\, \mathsf{cost}_\lambda E$. This patch, however, might use edges of zero cost. For every cluster $C \in \mathcal{B}$, for each of $k-1$ patches, we need at most $2k$ paths of length θ. By our choice of θ, the total cost of these paths is at most $\frac{1}{4}h^{-1}\epsilon\, \mathsf{cost}_\lambda E_C$. Each edge belongs to at most h clusters, so the total over

all clusters is $\frac{1}{4}\epsilon\,\mathsf{cost}_\lambda\,E$. Together with the term $\frac{1}{4}\epsilon\,\mathsf{cost}_\lambda\,E$ incurred at the outset, the total cost of the patch is $\epsilon\,\mathsf{cost}_\lambda\,E$. □

CHAPTER **Four**

Ball cover

Given an embedded graph $G = (E, \pi)$, symmetric lengths $\lambda \in \mathbb{N}^{\operatorname{Darts} E}$, **clients** $K \subseteq \operatorname{Vertices} G$, and a **radius** $r > 0$, the **ball cover** problem is to find a set of **centers** $C \subseteq \operatorname{Vertices} G$ minimizing $|C|$ subject to the condition that, for every client k, there exists a center c such that $d_{G,\lambda}(c, k) < r$. In other words, the ball cover problem is to find a minimum-cardinality r-net C with respect to K.

A straightforward objective-preserving reduction from set cover via dominating set establishes that, for general graphs, we should not hope to achieve a sublogarithmic approximation ratio with a polynomial-time algorithm [Feige, 1998, Dinur and Steurer, 2013]. In this chapter, we prove the following theorem.

Theorem 26 (Eisenstat, Klein, and Mathieu [2014])**.** *There exists a polynomial-time bicriteria approximation scheme for the ball cover problem in planar embedded graphs that, for every parameter $\epsilon > 0$, gives a radius-$(1 + \epsilon)r$ solution with at most $1 + \epsilon$ times as many centers as the smallest radius-r solution.*

The original write-up of this work includes extensions to the k-center problem, bounded-genus graphs, center costs, and penalties for leaving clients uncovered.

Prior to this result, there was an approximation scheme known for unit-length edges and a class of graphs that includes planar graphs, due to Demaine, Fomin, Hajiaghayi, and Thilikos [2005]. At the time of writing, it is not known for the ball cover problem whether it is necessary to let the approximation scheme use a radius larger than its benchmark or whether there exists an efficient approximation scheme (exponent of the polynomial running time independent of ϵ). Our scheme uses techniques from the inefficient approximation scheme for the planar traveling salesman due to Arora, Grigni, Karger, Klein, and Woloszyn [1998a], and we attempted without success to emulate the approach of Klein [2008], who gave an efficient scheme. (This approach is described in the previous chapter, on Steiner forest.)

Our approximation scheme uses the algorithm of Lemma 19 to obtain a multitriangulation with distances equivalent to the given embedded graph, decomposes that multitriangulation recursively, and applies a dynamic program to the decomposition. The decomposition technique is based on a divide-and-conquer algorithm due to Arora et al. [1998a]. We present two versions of the dynamic program. The first solves the problem exactly, but in exponential time. The second runs in polynomial time and uses the tolerances allowed by the approximation to reduce greatly the number of subproblems. We use an encoding technique due to Arora, Raghavan, and Rao [1998b] to control the number of configurations.

The problem of covering by balls of a uniform radius is closely related to the k-center problem, which asks for the minimum radius necessary to find a cover consisting of k balls. In turn, the k-center problem is related to the k-medians problem, where the objective is the sum rather than the maximum of the radii, and

the facility location problem, where the hard limit k is replaced with a linear cost for each chosen center (facility). There are approximation schemes for instances of the latter two problems in low-dimensional Euclidean spaces [Arora, Raghavan, and Rao, 1998b, Kolliopoulos and Rao, 2007], and it is surprising that counterparts for planar graphs are not known to exist. It is more typical that, when a scheme for the Euclidean plane is devised, a counterpart for planar graphs appears relatively soon afterward (see, e.g., the previous chapter).

For general graphs, polynomial-time 2-approximations to the k-center problem were discovered independently by Gonzalez [1985] and by Hochbaum and Shmoys [1985]. This ratio is the best possible under plausible complexity-theoretic assumptions, due to the aforementioned reduction from dominating set: when all of the edges are unit-length, the difference between radius 1 and radii in the interval $(1, 2)$ is moot. Since dominating set is hard even in planar graphs [Garey and Johnson, 1979], we must allow the approximation scheme more centers if we are to hope for a ratio less than 2. Note, though, that not many more centers are required, as evidenced by the polynomial-time approximation scheme for dominating set in planar graphs due to Baker [1994].

In the Euclidean setting, Feder and Greene [1988] gave a 1.822 hardness of approximation result for k-center and an efficient implementation of the previously proposed 2-approximations. In contrast, Hochbaum and Maass [1985] gave a polynomial-time approximation scheme for covering by fixed-radius balls. The discrepancy, again, is because of the flexibility in the number of balls. They used an independently developed shifting technique similar to that of Baker. Geometric covering by other shapes has been much studied since; see, for example, the polynomial-time approximation scheme due to Gonzalez [1991] for fixed-size Cartesian products of intervals. We note some other work on geometric covering that has turned to bicriteria approximation

as a way to avoid impossibility results: Agarwal and Procopiuc [2003], Feldman, Fiat, Sharir, and Segev [2007], Har-Peled and Lee [2012].

4.1 Facial carving decompositions

A **facial carving decomposition** of an embedded graph $G = (E, \pi)$ is a complete rooted unordered binary tree whose leaves are in one-to-one correspondence with $\operatorname{Faces} G$. We identify each node of this tree with the set of faces whose corresponding leaves are descendants of that node. Each leaf, then, is identified with the singleton set containing its corresponding face, and the root is identified with $\operatorname{Faces} G$.

Assume now that G is a planar multitriangulation with symmetric lengths $\lambda \in \mathbb{N}^{\operatorname{Darts} E}$. Let p be a shortest path arborescence with root s (Lemma 16). We give an algorithm due to Arora et al. [1998a] to construct, for some absolute constant b, a facial carving decomposition of depth at most $\log_b |\operatorname{Faces} G|$ such that, for every node F, there exist 8 shortest paths with tail s such that, for every vertex $v \in \operatorname{Frontier}_G F$, at least one of those paths contains v.

With respect to p, let q be an interdigitating arborescence with root o (Lemma 7). Define a relation $\preceq$ on $\operatorname{Faces} G$ where $f \preceq g$ if and only if $f \in \operatorname{Descendants}_q g$. This relation is a partial order.

Reflexive We have $f \in \operatorname{Descendants}_q f$.

Antisymmetric If $f \in \operatorname{Descendants}_q g$ and $g \in \operatorname{Descendants}_q f$, then $f = g$.

Transitive If $f \in \operatorname{Descendants}_q g$ and $g \in \operatorname{Descendants}_q h$, then $f \in \operatorname{Descendants}_q h$.

Suppose that $f \preceq g_1$ and $f \preceq g_2$. Then there exists a pair of nonnegative integers k_1, k_2 such that $(\mathsf{left}_G \circ q)^{k_1}(f) = g_1$ and $(\mathsf{left}_G \circ q)^{k_2}(f) = g_2$. If $k_1 \leq k_2$, then $g_1 \preceq g_2$. If $k_2 \leq k_1$, then $g_2 \preceq g_1$. We conclude that the set $\{g : g \in \operatorname{Faces} G,\ f \preceq g\}$ is totally ordered. By $f_1 \vee f_2$, we denote the unique $\preceq$-minimum element of the set $\{g : g \in \text{Faces},\ f_1 \preceq g,\ f_2 \preceq g\}$. This element is the **leafmost common ancestor**.

We now construct the facial carving decomposition recursively. Let F be the node whose subtree we currently are constructing. We ensure that F is expressed as the intersection of four or fewer sets of the form $\operatorname{Descendants}_q f$ or $\operatorname{Faces} G \setminus \operatorname{Descendants}_q f$. (The empty intersection is taken to be $\operatorname{Faces} G$.) By Proposition 4 and Lemmas 9 and 12, we ensure in this way that, for every node F, there exist 8 shortest paths with tail s such that, for every vertex $v \in \operatorname{Frontier}_G F$, at least one of those paths contains v.

In tandem, the following properties, which apply to every triple of nodes F_0, F_1, F_2 such that F_1, F_2 are the children of F_0, imply that $\log_b |\operatorname{Faces} G|$ is an upper bound on the depth of the decomposition.

- If F_0 cannot be expressed as the intersection of three or fewer sets, then both F_1, F_2 can.
- If F_0 can be expressed as the intersection of three or fewer sets, then for every $i \in \{1, 2\}$, we have $|F_i| < \frac{2}{3}\big(|F_0| - 1\big) + 1$.

For every node F with grandchild F', we have $|F'| + \frac{1}{5} \leq \frac{2}{3}\big(|F| + \frac{1}{5}\big)$, so the depth of the decomposition is at most

$$2 \log_{\frac{3}{2}} \frac{|\operatorname{Faces} G| + \frac{1}{5}}{1 + \frac{1}{5}},$$

from which bound we prove the existence of the absolute constant b.

Consider those sets of the form $\text{Descendants}_q f$ or $\text{Faces}\, G \setminus \text{Descendants}_q f$ whose intersection equals F. Without loss of generality, at most one of these sets is of the form $\text{Descendants}_q f$, since for every pair of faces f_1, f_2, if there exists a face $f_3 \in \text{Descendants}_q f_1 \cap \text{Descendants}_q f_2$, then $f_1 \preceq f_2$ or $f_2 \preceq f_1$, and either $\text{Descendants}_q f_1$ or $\text{Descendants}_q f_2$ can be omitted from the intersection.

Suppose that the intersection contains a set $\text{Descendants}_q f_1$. There exists a triple of faces f_2, f_3, f_4 such that

$$F = \text{Descendants}_q f_1 \setminus (\text{Descendants}_q f_2 \cup \text{Descendants}_q f_3 \cup \text{Descendants}_q f_4).$$

Assume without loss of generality that, for every $i \in \{2, 3, 4\}$, we have $f_i \prec f_1$; otherwise, omit the corresponding set and follow the case to be presented shortly. Assume by similar logic that $f_1 \prec o$ and that $f_2 \not\preceq f_3 \not\preceq f_4 \not\preceq f_2$. Consider the set $\{f_2 \vee f_3,\ f_2 \vee f_4,\ f_3 \vee f_4\}$. Assume to the contrary that it is a singleton set $\{g\}$. Since $g \preceq f_1 \prec o$, we have $g \neq o$. There exist three distinct faces h such that $\mathsf{left}_G\, q(h) = g$, which is impossible because g is comprised of $q(g)$ and at most two other darts. Assume without loss of generality that $g = f_2 \vee f_3$ is the minimum element. Then $f_4 \not\preceq f_2 \vee f_3$, so we choose child nodes

$$F \cap \text{Descendants}_q g = \text{Descendants}_q g \setminus (\text{Descendants}_q f_2 \cup \text{Descendants}_q f_3)$$
$$F \setminus \text{Descendants}_q g = \text{Descendants}_q f_1 \setminus (\text{Descendants}_q g \cup \text{Descendants}_q f_4).$$

We argue similarly when there exist $f_1, f_2, f_3, f_4 \in \text{Faces}\, G$ such that

$$F = \bigcup_{i \in \{1,2,3,4\}} (\text{Faces}\, G \setminus \text{Descendants}_q f_i).$$

Now suppose that F is the intersection of at most three approved sets. We show that there exists a face $g \in \operatorname{Faces} G$ such that

$$|F \cap \operatorname{Descendants}_q g| < \tfrac{2}{3}\big(|F| - 1\big) + 1$$
$$|F \setminus \operatorname{Descendants}_q g| < \tfrac{2}{3}\big(|F| - 1\big) + 1.$$

We find g by walking leafward in the arborescence q, maintaining the invariant that $|F \cap \operatorname{Descendants}_q g| \geq \frac{1}{3}\big(|F| - 1\big)$. By an averaging argument and the fact that $|o| \leq 3$, there exists an initial choice for g satisfying $\mathsf{left}_G\, q(g) = o$ and the invariant. While possible, we update $g \leftarrow g'$, where g' satisfies $\mathsf{left}_G\, q(g') = g$ and the invariant. At the end, by another averaging argument, $|F \cap \operatorname{Descendants}_q g| < \frac{2}{3}\big(|F| - 1\big) + 1$.

4.2 An exact dynamic program

Let $G = (E, \pi)$ be a planar multitriangulation with symmetric lengths $\lambda \in \mathbb{N}^{\operatorname{Darts} E}$. Fix a facial carving decomposition with the properties described in the previous section. In this section, we give an algorithm to solve the ball cover problem. The core of this algorithm is a straightforward bottom-up dynamic program on a facial carving decomposition. The interesting aspect is the interplay between which solutions for a node are allowed (conformance) and which ones can be combined (compatibility).

4.2.1 Interfaces and conformance

For a decomposition node F, an **interface** ι is a map $\iota \in \left(\mathbb{N} \cup \{\infty\}\right)^{\text{Frontier } F}$. We stipulate that, for every nonnegative integer n,

$$n < \infty \qquad\qquad \infty \leq \infty$$

$$\infty + n = \infty \qquad\qquad \infty + \infty = \infty$$

and limit our algebraic manipulations to the facts that addition is still commutative and associative and that $\leq$ is still a total order that respects addition. A set of centers C **conforms** to ι if and only if, for every $w \in \text{Frontier } F$ satisfying $\iota(w) < \infty$, there exists some $v \in C$ with $d(v, w) \leq \iota(w)$. The subproblems in the dynamic program are defined with respect to an **internal** interface ι^{int} and an **external** interface ι^{ext} that specify the coverage produced and consumed by the subsolution.

4.2.2 Compatibility

Compatibility is defined with respect to three pairs of interfaces $\iota_0^{\text{int}}, \iota_0^{\text{ext}}$ and $\iota_1^{\text{int}}, \iota_1^{\text{ext}}$ and $\iota_2^{\text{int}}, \iota_2^{\text{ext}}$ with respect to a parent-children triple of nodes F_0, F_1, F_2, i.e., $F_0 = F_1 \cup F_2$ and $F_1 \cap F_2 = \varnothing$. These interfaces are **compatible** if and only if both of two conditions hold.

Parent's internal For every $w \in \text{Frontier } F_0$ satisfying $\iota_0^{\text{int}}(w) < \infty$, there exist $i \in \{1, 2\}$ and $v \in \text{Frontier } F_i$ such that $\iota_i^{\text{int}}(v) + d(v, w) \leq \iota_0^{\text{int}}(w)$.

Children's external For every $i \in \{1, 2\}$ and $w \in \text{Frontier } F_i$ satisfying $\iota_i^{\text{ext}}(w) < \infty$, at least one of two possibilities holds. The first possibility is that there exists

$v \in \operatorname{Frontier} F_{3-i}$ such that $\iota^{\text{int}}_{3-i}(v) + d(v, w) \leq \iota^{\text{ext}}_i(w)$. The second possibility is that there exists $v \in \operatorname{Frontier} F_0$ such that $\iota^{\text{ext}}_0(v) + d(v, w) \leq \iota^{\text{ext}}_i(w)$.

4.2.3 Total conformance

Before we define total conformance, we need an additional definition. Fix an arbitrary map $\varphi \in (\operatorname{Darts} E)^{\operatorname{Vertices} G}$ such that, for every vertex v, we have $\varphi(v) \in v$. For every node F, define

$$\operatorname{Content} F \triangleq \big\{v : v \in \operatorname{Vertices} G,\ \mathsf{right}_G\, \varphi(v) \in F\big\}.$$

Observe that $\operatorname{Content} F \subseteq \operatorname{Interior} F \cup \operatorname{Frontier} F$.

Total conformance is a property of a subsolution $C \subseteq \operatorname{Vertices} G$ with respect to a node F. The subsolution C is **totally conforming** if and only if there exists a family of interface pairs $\iota^{\text{int}}_{F'}, \iota^{\text{ext}}_{F'}$ indexed by nodes $F' \subseteq F$ such that the following properties hold.

Conformance For every node $F' \subseteq F$, the subsolution $C \cap \operatorname{Content} F'$ conforms to the interface $\iota^{\text{int}}_{F'}$.

Compatibility Every parent-children triple of nodes $F_0, F_1, F_2 \subseteq F$ has compatible interface pairs.

Coverage For every node $F' \subseteq F$ and every client $w \in K \cap \operatorname{Content} F'$, at least one of two possibilities holds. The first possibility is that there exists $v \in C \cap \operatorname{Content} F'$ with $d(v, w) < r$. The second possibility is that there exists $w \in \operatorname{Frontier} F'$ with $\iota^{\text{ext}}_{F'}(v) + d(v, w) < r$.

The following proposition is an easy consequence of the coverage property and the fact that $\operatorname{Content}\operatorname{Faces} G = \operatorname{Vertices} G$.

Proposition 27. *A totally conforming subsolution C with respect to the root node* $\operatorname{Faces} G$ *is feasible.*

The following lemma shows that compatible totally conforming subsolutions compose under union.

Lemma 28. *Let F_0, F_1, F_2 be a parent-children triple of nodes. Let $\iota_0^{\mathrm{int}}, \iota_0^{\mathrm{ext}}$ and $\iota_1^{\mathrm{int}}, \iota_1^{\mathrm{ext}}$ and $\iota_2^{\mathrm{int}}, \iota_2^{\mathrm{ext}}$ be three pairs of interfaces that are compatible. If for every $i \in \{1, 2\}$ the subsolution C_i is totally conforming via a family of interfaces with $\iota_i^{\mathrm{int}}, \iota_i^{\mathrm{ext}}$ for F_i, then $C_0 = C_1 \cup C_2$ is totally conforming via a family of interfaces with $\iota_0^{\mathrm{int}}, \iota_0^{\mathrm{ext}}$ for F_0.*

Proof. We need to check conformance and coverage at F_0. The other requirements are hypotheses. For conformance, let $w \in \operatorname{Frontier} F_0$ satisfy $\iota_0^{\mathrm{int}}(w) < \infty$. By compatibility (parent's internal), there exist $i \in \{1, 2\}$ and $v \in \operatorname{Frontier} F_i$ such that $\iota_i^{\mathrm{int}}(v) + d(v, w) \leq \iota_0^{\mathrm{int}}(w)$. By conformance for F_i, there exists a center $u \in C_i \subseteq C_0$ with $d(u, v) \leq \iota_i^{\mathrm{int}}(u)$. In consequence,

$$d(u, w) \leq d(u, v) + d(v, w) \leq \iota_i^{\mathrm{int}}(v) + d(v, w) \leq \iota_0^{\mathrm{int}}(w).$$

As for coverage, let $x \in K \cap \operatorname{Content} F_0$. Either $x \in K \cap \operatorname{Content} F_1$ or $x \in K \cap \operatorname{Content} F_2$; let $i \in \{1, 2\}$ satisfy $x \in K \cap \operatorname{Content} F_i$. If there exists $w \in C_i \cap \operatorname{Content} F_i$ with $d(w, x) < r$, then $w \in C_0 \cap \operatorname{Content} F_0$, and we're done. If not, then there exists $w \in \operatorname{Frontier} F_i$ with $\iota_i^{\mathrm{ext}}(w) + d(w, x) < r$. By compatibility (children's external), at least one of two possibilities holds. The first possibility is that there exists $v \in \operatorname{Frontier} F_{3-i}$ such that $\iota_{3-i}^{\mathrm{int}}(v) + d(v, w) \leq \iota_i^{\mathrm{ext}}(w)$. In this case,

by conformance, there exists $u \in C_{3-i} \cap \text{Content } F_{3-i} \subseteq C_0 \cap \text{Content } F_0$ such that $d(u,v) \leq \iota^{\text{int}}_{3-i}(v)$, and

$$\begin{aligned} d(u,x) &\leq d(u,v) + d(v,w) + d(w,x) \\ &\leq \iota^{\text{int}}_{3-i}(v) + d(v,w) + d(w,x) \\ &\leq \iota^{\text{ext}}_{i}(w) + d(w,x) \\ &< r. \end{aligned}$$

The second possibility is that there exists $v \in \text{Frontier } F_0$ such that $\iota^{\text{ext}}_0(v) + d(v,w) \leq \iota^{\text{ext}}_i(w)$. Then

$$\begin{aligned} \iota^{\text{ext}}_0(v) + d(v,x) &\leq \iota^{\text{ext}}_0(v) + d(v,w) + d(w,x) \\ &\leq \iota^{\text{ext}}_i(w) + d(w,x) \\ &< r. \end{aligned}$$

□

The following proposition is a partial converse for Lemma 28.

Proposition 29. *Let C be an totally conforming subsolution with respect to some node F. Then C is totally conforming with respect to every node $F' \subseteq F$.*

4.2.4 A structure lemma

Having proved that the output of the dynamic program will be feasible, we now prove that it will be optimal.

Lemma 30. *Let C be a set of centers covering every client. Then C is totally conforming.*

Proof. We stipulate that $\min \varnothing = \infty$. For every node F, we set

$$\iota_F^{\mathrm{int}}(w) \triangleq \min \big\{ d(v,w) : v \in C \cap \operatorname{Content} F \big\}$$
$$\iota_F^{\mathrm{ext}}(w) \triangleq \min \big\{ d(v,w) : v \in C \setminus \operatorname{Content} F \big\},$$

and conformance is immediate. To check compatibility, let F_0, F_1, F_2 be a parent-children triple with associated interfaces $\iota_i^{\mathrm{int}} = \iota_{F_i}^{\mathrm{int}}$ and $\iota_i^{\mathrm{ext}} = \iota_{F_i}^{\mathrm{ext}}$. The property "parent's internal" requires that, for every $w \in \operatorname{Frontier} F_0$ satisfying $\iota_0^{\mathrm{int}}(w) < \infty$, there exist $i \in \{1,2\}$ and $v \in \operatorname{Frontier} F_i$ such that $\iota_i^{\mathrm{int}}(v) + d(v,w) \le \iota_0^{\mathrm{int}}(w)$. Fix w; by the definition of $\iota_0^{\mathrm{int}}(w)$, there exists $u \in C \cap \operatorname{Content} F_0$ such that $d(u,w) = \iota_0^{\mathrm{int}}(w)$. Since $\operatorname{Content} F_0 = \operatorname{Content} F_1 \cup \operatorname{Content} F_2$, let $i \in \{1,2\}$ satisfy $u \in \operatorname{Content} F_i$ and consider a particular shortest path from u to w. The fact that $w \in \operatorname{Frontier} F_0$ implies that $w \notin \operatorname{Interior} F_i$, so there exists by Lemma 3 a vertex $v \in \operatorname{Frontier} F_i$ such that

$$\iota_0^{\mathrm{int}}(w) = d(u,w) = d(u,v) + d(v,w) \ge \iota_i^{\mathrm{int}}(v) + d(v,w).$$

The property "children's external" requires that, for every $i \in \{1,2\}$ and $w \in \operatorname{Frontier} F_i$ satisfying $\iota_i^{\mathrm{ext}}(w) < \infty$, at least one of two possibilities holds. Fix i and w and let $u \in C \setminus \operatorname{Content} F_i$ satisfy $d(u,w) = \iota_i^{\mathrm{ext}}(w)$. In case $u \in \operatorname{Content} F_{3-i}$, there exists a vertex $v \in \operatorname{Frontier} F_{3-i}$ with $d(u,w) = d(u,v) + d(v,w)$, which implies that

$$\iota_{3-i}^{\mathrm{int}}(v) + d(v,w) \le d(u,v) + d(v,w) = d(u,w) = \iota_i^{\mathrm{ext}}(w),$$

which is the first acceptable possibility. In case $u \notin \operatorname{Content} F_0$, there exists a vertex $v \in \operatorname{Frontier} F_0$ with $d(u,w) = d(u,v) + d(v,w)$, and

$$\iota_0^{\mathrm{ext}}(v) + d(v,w) \le d(u,v) + d(v,w) = d(u,w) = \iota_i^{\mathrm{ext}}(w),$$

which is the second acceptable possibility.

The final property in need of verification is "coverage": for every node F and every client $w \in K \cap \operatorname{Content} F$, at least one of two possibilities holds. Let $u \in C$ satisfy $d(u, w) < r$. If $u \in \operatorname{Content} F$, then the first acceptable possibility is immediate. Otherwise, $u \in C \setminus \operatorname{Content} F$, and there exists a vertex $v \in \operatorname{Frontier} F$ with $d(u, v) + d(v, w) = d(u, w)$. In this case,

$$\iota_F^{\text{ext}}(v) + d(v, w) \leq d(u, v) + d(v, w) = d(u, w) < r,$$

which implies the second acceptable possibility. □

4.3 An approximating dynamic program

In this section, we show how to modify and extend the dynamic program presented in the previous section to trade precision for a polynomial running time. The first step is a partitioning technique due to Baker [1994] called **shifting** that has been used in many approximation schemes for planar graphs. The second step is to reduce the number of configurations that the dynamic program needs to consider, via techniques of Arora et al. [1998b].

4.3.1 Shifting

Assume without loss of generality that there exists a positive integer n such that $\epsilon = n^{-1}$. Choose uniformly at random a **shift** $\sigma \in \{0, 2r, \ldots, 2(\epsilon^{-1} - 1)r\}$ and

partition the client set C into sets

$$C_i = \left\{x : x \in C,\ 2i\epsilon^{-1}r \leq d(s,x) + \sigma < 2(i+1)\epsilon^{-1}r\right\}$$

for $i \in \mathbb{N}$, where s is the root of the shortest path arborescence used to construct the decomposition. Observe that almost all of these sets are empty.

The algorithm covers each C_i independently and takes the output to be the union of these covers. The purpose of this partition is to limit the length of the frontier under consideration to $O(\epsilon^{-1}r)$, which we exploit in the next subsection. The drawback is that, even if each individual cover is optimal, the union of covers in general is not. We show that expected fraction of extra centers is at most ϵ of the optimal solution and derandomize by minimizing over ϵ^{-1} outcomes for σ.

Let K be a feasible cover for C. For each i, let

$$K_i = \left\{u : u \in K,\ 2i\epsilon^{-1}r - r \leq d(s,u) + \sigma < 2(i+1)\epsilon^{-1}r + r\right\}.$$

First we verify that K_i is a cover for C_i. Let $x \in C_i$ be arbitrary and let $u \in K$ satisfy $d(x,u) = d(u,x) < r$. Then, by two applications of the triangle inequality,

$$d(s,x) - r < d(s,x) - d(u,x) \leq d(s,u) \leq d(s,x) + d(x,u) < d(s,x) + r,$$

which implies that $u \in K_i$. Second, we compute the expected number of sets K_i to which a particular $u \in K$ belongs. Let $Z = \big(d(s,u) + \sigma\big) \bmod 2\epsilon^{-1}r$. The vertex u belongs to exactly two sets if $Z < r$ or $Z \geq 2\epsilon^{-1}r - r$; otherwise, it belongs to exactly one set. For σ, exactly one outcome out of ϵ^{-1} causes the above condition to hold. We conclude that the probability that u belongs to two sets is ϵ.

4.3.2 Quantization

If the dynamic program is to run in polynomial time, then we need to effect a drastic reduction in the number of interfaces that it considers. We call the interfaces that satisfy the conditions set out in the following subsections **representative interfaces**.

Several techniques are required. We exploit the relaxation of the covering radius from r to $(1+\epsilon)r$ by reducing the precision of the set of representative interfaces. Every distance not less than $(1+\epsilon)r$ can be replaced by ∞ without affecting compatibility or total conformance. Recall that, by our construction of the decomposition, for each node, the frontier vertices lie on the union of at most 8 shortest paths from s. Moreover, the depth of the decomposition is at most $\log_b |\text{Faces}\, G|$. Define

$$\delta = \max\left\{ \left\lfloor \frac{\epsilon r}{6 \log_b |\text{Faces}\, G|} \right\rfloor,\ 1 \right\} > \frac{\epsilon r}{12 \log_b |\text{Faces}\, G|}.$$

We consider to be representative only those interfaces in $D^{\text{Frontier}\, F}$, where $D = \big\{ n\delta : n \in \mathbb{N},\ n\delta < (1+\epsilon)r \big\} \cup \{\infty\}$.

4.3.3 Portals

Observe that, after applying Baker's technique, the distance from the root vertex s to each client to be covered belongs to the interval $\{x, x+1, \ldots, x+2\epsilon^{-1}r-1\}$. Accordingly, all of the centers and frontier vertices that could participate meaningfully in the dynamic program have distances belonging to the interval $\big\{x-r+1,\ x-r+2,\ \ldots,\ x+(2\epsilon^{-1}+1)r-1\big\}$. It is not necessary to track the others.

By exploiting again, moreover, the relaxed covering radius, it is also not necessary to track every frontier vertex. For each node F, apply the algorithm of Lemma 18 on each of at most 8 shortest paths to compute, with respect to the subset of frontier vertices

$$\big\{v : v \in \text{Frontier}\, F,\ x - r < d(s,v) < x + (2\epsilon^{-1} + 1)r\big\},$$

a δ-net $\text{Portals}\, F$ of cardinality at most

$$8\big(1 + \delta^{-1}(2\epsilon^{-1} + 2)r\big) < 192(\epsilon^{-2} + \epsilon^{-1}) \log_b |\text{Faces}\, G| + 8.$$

A **portal** is an element of $\text{Portals}\, F$. The portal analog of Lemma 3 is that, for every st-path P where s belongs to the interior and t does not, there exists another st-path P' that satisfies $\mathsf{length}\, P' < \mathsf{length}\, P + 2\delta$ and contains a portal. We change the definitions used by the exact dynamic program by replacing globally $\text{Frontier}\, F$ with $\text{Portals}\, F$.

4.3.4 A relaxed Lipschitz condition

Let $m = \big\lceil (1+\epsilon) r \delta^{-1} \big\rceil \delta$ be the least multiple of δ satisfying $m \geq (1+\epsilon)r$. Given an interface ι, we define, for every portal v,

$$\iota'(v) = \min\big\{\iota(v), m\big\}.$$

In other words, we derive ι' by treating ∞ as m, a radius slightly too large to be covering. The final step in controlling the number of representative interfaces is that they are required to satisfy the following **relaxed Lipschitz condition**, which is,

for every pair of portals v, w,

$$\iota'(v) + d(v, w) > \iota'(w) - \delta.$$

We bound the number of representative interfaces examining each of the 8 shortest paths separately and hence bounding a Cartesian product. Fix a particular shortest path and let

$$\langle v_1, v_2, \ldots, v_\ell \rangle$$

be the subsequence of vertices selected from the path in constructing the corresponding δ-net. Letting $D = 2(\epsilon^{-1} + 2)r - 2$,

$$\sum_{i=2}^{\ell} d(v_{i-1}, v_i) \leq D.$$

We now show that there exist a polynomial number of possibilities for the sequence

$$\big\langle \iota'(v_1),\ \iota'(v_2) - \iota'(v_1),\ \iota'(v_3) - \iota'(v_2),\ \ldots,\ \iota'(v_\ell) - \iota'(v_{\ell-1}) \big\rangle,$$

given which it is possible to reconstruct, for every i, the value of $\iota(v_i)$. Using the relaxed Lipschitz condition and the fact that $\iota'(v_i)$ is a multiple of δ, the number of sequences is bounded by the number of possibilities for $\iota(v_1)$, a logarithmic quantity, times

$$\prod_{i=2}^{\ell} \Big(2\lceil \delta^{-1} d(v_{i-1}, v_i) \rceil + 1 \Big) \leq 5^{\ell-1} \prod_{i=2}^{\ell} \max \big\{ \delta^{-1} d(v_{i-1}, v_i),\ 1 \big\}.$$

The inequality is derived by considering separately the cases $d(v_{i-1}, v_i) \leq \delta$ and $d(v_{i-1}, v_i) > \delta$. Since ℓ is logarithmic, the factor $5^{\ell-1}$ is polynomial. Via the inequality of arithmetic and geometric means, the other factor can be bounded above by $\big(D\delta^{-1}/(\ell - 1)\big)^{\ell-1}$, which, because $D\delta^{-1}$ and ℓ are both logarithmic, is polynomial.

4.3.5 An approximate structure lemma

Lemma 31. *Let C be a set of centers covering every client. Then C is totally conforming with radius $(1+\epsilon)r$ via representative interfaces.*

Proof. For every node F, let $\iota_F^{\text{int}}, \iota_F^{\text{ext}}$ be the interfaces from the construction of the original structure lemma, Lemma 30. Let $h = \lfloor \log_b |\text{Faces}\, G| \rfloor$ and define

$$\hat{\iota}_F^{\text{int}}(v) = \mu\big(\iota_F^{\text{int}}(v) + (h - \mathsf{depth}\, F)3\delta\big)$$
$$\hat{\iota}_F^{\text{ext}}(v) = \mu\big(\iota_F^{\text{ext}}(v) + (h - 1 + \mathsf{depth}\, F)3\delta\big)$$
$$\mu(z) = \begin{cases} \lceil z\delta^{-1} \rceil \delta & \text{if } z \le m - \delta \\ \infty & \text{if } z > m - \delta. \end{cases}$$

The effect of μ is to round its argument up to the nearest multiple of δ less than $(1+\epsilon)r$, or ∞.

First, we verify that the newly defined interfaces are representative interfaces. This is tedious but straightforward. By the triangle inequality, an original interface ι satisfies, for every pair of frontier vertices v, w with $\iota(w) < \infty$,

$$\iota(v) + d(v, w) \ge \iota(w).$$

Its counterpart $\hat{\iota}$ is defined as

$$\hat{\iota}(v) = \mu\big(\iota(v) + \gamma\big),$$

where γ is constant. Now we derive the inequality

$$\hat{\iota}'(v) + d(v, w) > \hat{\iota}'(w) - \delta,$$

where $\hat{\iota}'(x) = \min\left\{\hat{\iota}(x), m\right\}$. Observe that $\iota(x)+\gamma+\delta > \hat{\iota}'(x)$ and that, if $\iota(x)+\gamma \leq m$, then $\iota(x) + \gamma \leq \hat{\iota}'(x)$. The first case of the inequality to be proved is when $\iota(v) + \gamma \leq m$; combine the previous two inequalities with the triangle inequality. The second is when $\iota(v) + \gamma > m$, in which case we observe that the left-hand side is at least m, and the right-hand side is at most $m - \delta$.

Second, we verify total conformance. Since the revised interfaces have values no less than their originals, conformance carries over. Coverage with radius $(1 + \epsilon)r$ holds because the error term $(h - 1 + \mathsf{depth}\, F)3\delta + \delta$, which is an upper bound on $\hat{\iota}_F^{\text{ext}}(v) - \iota_F^{\text{ext}}(v)$, is less than ϵ. Finally, for compatibility, the error term adds 2δ slack to each of the relevant inequalities after rounding up, which pays for a detour of one-way distance at most δ to a portal (observe that the root cluster has no frontier vertices). □

CHAPTER **Five**

Multiple-source shortest paths

Let $G = (E, \pi)$ be a connected embedded graph with lengths $\lambda \in \mathbb{N}^{\operatorname{Darts} E}$ and let $f_\infty \in \operatorname{Faces} G$. **Multiple-source shortest paths** (MSSP) is the problem of computing a shortest path arborescence for each root vertex incident to f_∞, that is, each vertex $\mathsf{head}_G\, a$ where $a \in f_\infty$. In this chapter, we prove the following theorem.

Theorem 32 (Eisenstat and Klein [2013])**.** *There exists an algorithm that solves the multiple-source shortest paths problem for planar G that runs in time $O\big(|E| + \sum_{a \in \operatorname{Darts} E} \lambda(a)\big)$.*

Our algorithm is based on the work of Cabello et al. [2013], who simplified a previous $O\big(|E| \log |E|\big)$-time algorithm for real lengths due to Klein [2005] and generalized it to bounded-genus graphs. With slightly different techniques, Eisenstat and Klein [2013] also obtained a linear-time algorithm for single-source sink-single maximum flow in unit-capacity planar graphs. Substantially simpler than a previous algorithm

with the same, optimal running time due to Brandes and Wagner [2000], this algorithm is based on the $O\big(|E| \log |E|\big)$-time algorithm for real capacities of Borradaile and Klein [2009]. The proof of correctness borrows liberally from Erickson [2010], who first pointed out the similarity of the latter algorithm to that of Cabello et al. and who summarized some of the long, intertwined history of shortest paths and maximum flow in planar graphs. One hint of this similarity is that both algorithms can be viewed as implementations of the simplex method, supported by the same collection of data structures.

Since its introduction by Klein, the multiple-source shortest paths problem has appeared as a primitive in many planar graph algorithms; see Eisenstat and Klein [2013] for details. Somewhat surprisingly, however, the only public experimental work to date involving multiple-source shortest paths, at least of which we are aware, is presented in the next chapter of this dissertation, on centrality measures. Tazari and Müller-Hannemann [2008] declined to implement an efficient multiple-source shortest paths subroutine for their algorithms engineering work on the Steiner tree approximation algorithm of Borradaile et al. [2009] and found that, while their implementation was faster than some existing heuristics at producing solutions with comparable quality, almost all of the running time was spent computing shortest paths. The maximum flow algorithm of Borradaile and Klein was implemented by Schmidt, Toppe, and Cremers [2009], who found it to be faster on instances arising in computer vision than algorithms designed specifically for computer vision. The latter remain heavily used, though, because they handle multiple sources, multiple sinks, and three-dimensional grids, which aren't planar.

The special case of multiple-source shortest paths in weighted rectilinear grids with diagonals has been investigated in connection with string alignment problems. Schmidt [1998] and Tiskin [2007, 2008] have given efficient algorithms for such prob-

lems, for which multiple-source shortest paths is, as of this work, an equally asymptotically efficient alternative. Their proofs of correctness use the same noncrossing property of shortest paths.

5.1 Implicit representation of multiple arborescences

Let $\{b_0, b_1, \ldots, b_{\ell-1}\} = f_\infty$ be an enumeration such that, for every i, we have $b_{i+1} = \mathsf{rev}\,\pi(b_i)$, where we define $b_\ell = b_0$. We specify an embedded graph $\hat{G} = (\hat{E}, \hat{\pi})$, obtained by subdividing the cycle f_∞ into a wheel with a double spoke to $\mathsf{head}_G\, b_0$. If G is planar, then so is $\hat{G}$. Formally, let $\{e_0, e_1, \ldots, e_\ell\}$ be a set of edges disjoint from E.

$$\hat{E} = E \cup \{e_0, e_1, \ldots, e_\ell\}$$

$$\hat{\pi}(a) = \begin{cases} (e_\ell, 1) & \text{if } a = b_0 \\ (e_0, 1) & \text{if } a = (e_\ell, 1) \\ \mathsf{rev}\, b_1 & \text{if } a = (e_0, 1) \\ (e_i, 1) & \text{if } \exists i \in \{1,\ 2,\ \ldots,\ \ell - 1\},\ a = b_i \\ \mathsf{rev}\, b_{i+1} & \text{if } \exists i \in \{1,\ 2,\ \ldots,\ \ell - 1\},\ a = (e_i, 1) \\ (e_\ell, -1) & \text{if } a = (e_0, -1) \\ (e_{i-1}, -1) & \text{if } \exists i \in \{1, 2, \ldots, \ell\},\ a = (e_i, -1) \\ \pi(a) & \text{otherwise.} \end{cases}$$

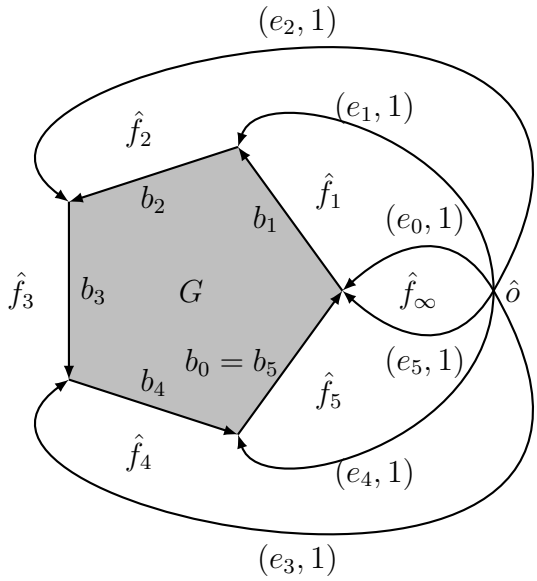


Figure 5.1: The embedded graph $\hat{G}$.

For each vertex $v = \mathsf{head}_G\, a$ of G, the corresponding vertex of $\hat{G}$ is denoted $\hat{v} = \mathsf{head}_{\hat{G}}\, a$. In $\hat{G}$, there is one additional vertex, which we call $\hat{o} = \mathsf{head}_{\hat{G}}\,(e_0, -1)$. For each face $f = \mathsf{right}_G\, a \neq f_\infty$ of G, the corresponding face of $\hat{G}$ is denoted $\hat{f} = \mathsf{right}_{\hat{G}}\, a$. Let $\hat{f}_\infty = \mathsf{right}_{\hat{G}}\,(e_0, -1)$ and, for each $i \in \{1, 2, \ldots, \ell\}$, let $\hat{f}_i = \mathsf{right}_{\hat{G}}\, b_i$. See Figure 5.1.

The output of the MSSP algorithms that we describe is an arborescence $\hat{p}_0$ rooted at $\hat{o}$, a sequence of **pivot** darts $a_1, a_2, \ldots, a_n \in \operatorname{Darts} \hat{E}$, and indices $0 = j(0) < j(1) < \cdots < j(\ell) = n$. The pivots define a sequence of arborescences inductively as follows.

$$\hat{p}_{j+1}(\hat{v}) = \begin{cases} a_{j+1} & \text{if } \mathsf{head}_{\hat{G}}\, a_{j+1} = \hat{v} \\ \hat{p}_j(\hat{v}) & \text{if } \mathsf{head}_{\hat{G}}\, a_{j+1} \neq \hat{v} \end{cases}$$

Each of these arborescences necessarily is rooted at $\hat{o}$. Define a map p_j with respect to G by taking $p_j(v) = \hat{p}_j(\hat{v})$ whenever the latter belongs to $\operatorname{Darts} E$. For each i, the map $p_{j(i)}$ is a shortest path arborescence rooted at $\mathsf{head}_G\, b_i$.

5.2 Algorithmic framework

The framework that we are about to describe is due to Cabello et al. [2013]. It is equivalent to running the dual simplex method on the linear program for shortest paths. We start by computing an initial shortest path arborescence p_0 rooted at $\mathsf{head}_G\, b_0$ and defining the arborescence

$$\hat{p}_0(\hat{v}) = \begin{cases} (e_0, 1) & \text{if } v = \mathsf{head}_{\hat{G}}\,(e_0, 1) \\ p_0(v) & \text{if } v \neq \mathsf{head}_{\hat{G}}\,(e_0, 1). \end{cases}$$

For efficient updates, we compute also **reduced lengths** λ'_0, where

$$\lambda'_0(a) = d_{G,\lambda}(\mathsf{head}_G\, b_0,\ \mathsf{tail}_G\, a) + \lambda(a) - d_{G,\lambda}(\mathsf{head}_G\, b_0,\ \mathsf{head}_G\, a).$$

For every dart a in the image of p_0, we have $\lambda'_0(a) = 0$. Observe also that $\lambda'_0 - \lambda \in \text{Boundaries}\, G^*$, where G^* is the dual of G. It is well known that, for every alternative length map $\lambda' \in \mathbb{N}^{\text{Darts}\, E}$ satisfying $\lambda' - \lambda \in \text{Boundaries}\, G^*$, every pair of vertices s, t, and every pair of st-paths P_1, P_2,

$$\mathsf{length}_\lambda(P_1) - \mathsf{length}_\lambda(P_2) = \mathsf{length}_{\lambda'}(P_1) - \mathsf{length}_{\lambda'}(P_2).$$

In consequence, both length maps give rise to the same shortest paths. We maintain over time an arborescence p_j and reduced lengths λ'_j that satisfy the invariant that every dart a in the image of the map p_j satisfies $\lambda'_j(a) = 0$. Whenever p_j is an arborescence, each of its paths has length 0, so p_j is a shortest path arborescence.

Each round consists of one or more pivots. In round i, counting from 0, the first dart that enters the primal arborescence is $a_{j(i)+1} = (e_{i+1}, 1)$; we set $\lambda'_{j(i)+1} = \lambda'_{j(i)}$.

In subsequent pivots, the dart a_{j+1} that enters is chosen in conjunction with λ'_{j+1} so that $\lambda'_{j+1}(a_{j+1}) = 0$ and $\lambda'_{j+1} - \lambda'_j \in \text{Boundaries}\, G^*$. Since $\text{Boundaries}\, G^*$ is closed under addition, this guarantees inductively that $\lambda'_{j+1} - \lambda \in \text{Boundaries}\, G^*$. The last pivot of round i is the one that displaces the dart $(e_i, 1)$.

For each i and each $j(i) < j < j(i+1)$, we define a vertex chain $\chi_j \in \{0,1\}^{\text{Vertices}\, G}$ as follows. For each vertex v, if $\mathsf{path}_{\hat{p}_j}\, \hat{v}$ contains $(e_i, 1)$, then we set $\chi_j(v) = 0$. If $\mathsf{path}_{\hat{p}_j}\, \hat{v}$ contains $(e_{i+1}, 1)$, then we set $\chi_j(v) = 1$. We observe that χ_j is total and well defined. Since $\sum_{v \in \text{Vertices}\, G} \chi_j(v)$ is nonnegative; bounded above by $|\text{Vertices}\, G|$; and, as we will ensure within each round, increasing in j; each round terminates after at most $|\text{Vertices}\, G|$ pivots.

Let $\omega_j = \partial_2^{G^*} \chi_j \in \text{Boundaries}\, G^* \cap \{0, 1, -1\}^E$. Since $\chi_j(\mathsf{head}_G\, b_i) = 0$ and $\chi_j(\mathsf{head}_G\, b_{i+1}) = 1$, there exists, by the connectivity of G, a dart a such that $\omega(a) = 1$. Let $z = \min\left\{\lambda'_j(a) : \omega_j(a) = 1\right\}$ and define $\lambda'_{j+1} = \lambda'_j - z\omega_j$. By our choice of z, we have $\lambda'_{j+1} \geq 0$. There exists, moreover, a dart a_{j+1} such that $\omega_j(a_{j+1}) = 1$ and $\lambda'_{j+1}(a_{j+1}) = 0$. The head of this dart and its descendants have their paths switched from $(e_i, 1)$ to $(e_{i+1}, 1)$.

5.3 Use of the interdigitating arborescence

Starting with this section, we assume that G and thus $\hat{G}$ are planar. For each i and each $j(i) < j < j(i+1)$, define as follows an arborescence p'_j, consisting of $\mathsf{rev}\, b_{i+1}$

together with the darts in p_j.

$$p'_j(v) = \begin{cases} b_{i+1} & \text{if } v = \mathsf{head}_G\, b_{i+1} \\ p_j(v) & \text{if } v \neq \mathsf{head}_G\, b_{i+1} \end{cases}$$

For each i, define $p'_{j(i)} = p_{j(i)}$. For each j, let $\mathsf{rev} \circ q'_j$ be the interdigitating arborescence (in G) rooted at f_∞. For each i and each $j(i) < j < j(i+1)$, let

$$q_j(f) = \begin{cases} \mathsf{rev}\, b_{i+1} & \text{if } f = f_\infty \\ q'_j(f) & \text{if } f \neq f_\infty \end{cases}$$

Observe that $\mathsf{left}_G \circ q_j$ is the identity map on $\operatorname{Faces} G$. For each i, define $q_{j(i)} = q'_{j(i)}$. For each i and each $j(i) < j < j(i+1)$, the vertex chain χ_j is the indicator function for $\operatorname{Descendants}_{p'_j} \mathsf{head}_G\, b_{i+1}$. It follows by the planarity of G and Lemma 12 that the darts a satisfying $(\partial_2^{G^*} \chi_j)(a) = 1$ are $\mathsf{rev}\, b_{i+1}$ and the darts in $\mathsf{path}_{\mathsf{rev} \circ q'_j} \mathsf{right}_G\, \mathsf{rev}\, b_{i+1}$. Put another way, these darts comprise the image under q_j of the unique cycle of the map $\mathsf{right}_G \circ q_j$ from $\operatorname{Faces} G$ to $\operatorname{Faces} G$.

The algorithm of Cabello et al. [2013] uses a top tree [Alstrup, Holm, Lichtenberg, and Thorup, 2005] to manage λ'_j and q'_j and the task of finding a_{j+1}, with each pivot requiring logarithmic time. The algorithm that we describe in the next section works directly with q_j.

5.4 Algorithm

In this section, we present our MSSP algorithm (Figure 5.2) and sketch the difficult parts of a proof that it is a correct instance of the framework. This algorithm,

```
    initialize Ĝ, λ′, p̂, q

    procedure Evert(c′, f′)              # change the root of q to f′
        loop                             # and insert the cycle dart c′ from f′
            c″ ← q(f′)
            q(f′) ← c′
            exit when f′ = f
            c′ ← rev c″
            f′ ← left_G c′
        end loop
    end procedure

    j ← 0
    f ← f_∞
L1  for i ← 0 to ℓ − 1 by 1
        j(i) ← j
        a ← (e_{i+1}, 1)
L2      loop
            j ← j + 1                    # pivot the dart a in and the dart c out
            a_j ← a
            c ← p̂(head_Ĝ a)
            p̂(head_Ĝ a) ← a
            exit when c = (e_i, 1)
S3          Evert(rev c, left_G rev c)
L4          loop                         # find the cycle dart with minimum λ′
                a ← q(f)                 # by counting down
                exit when λ′(a) = 0
                λ′(a), λ′(rev a) ← λ′(a) − 1, λ′(rev a) + 1
                f ← right_G a
            end loop
        end loop
S5      Evert(⊥, f_∞)
L6      loop                             # fix up λ′
            exit when f = f_∞
            a ← q(f)
            λ′(a), λ′(rev a) ← λ′(a) − 1, λ′(rev a) + 1
            f ← right_G a
        end loop
    end for
    j(ℓ) ← j
```

Figure 5.2: Pseudocode for the linear-time multiple-source shortest paths algorithm.

moreover, uses the **leafmost pivot rule** to break ties. The analysis of the running time is subtle and deferred to the next section. We omit the subscript j on λ' and $\hat{p}$ and q to emphasize that these maps are updated in place. We use $\perp$ to make undefined values first-class.

Loop L1 iterates through ℓ new roots. Loop L2 effects one root change but leaves λ' in an inconsistent state; statement S5 and loop L6 clean up. Statements S3 and S5 maintain q. The effect of S3 is to remove a from q and insert $\mathsf{rev}\, c$; some darts in q must be reversed if $\mathsf{left}_G\, \mathsf{rev}\, c$ is not already the root face. The effect of S5 is to restore the root to be f_∞.

Loops L4 and L6 are the most interesting because they update the reduced lengths λ'. The key new idea of this algorithm is not to update λ' all at once. In this way, we avoid reviewing the leafmost darts every time a dart with reduced length zero is found. To show, however, that our algorithm pivots in a correct manner that is consistent with the leafmost rule, we need a correspondence between λ' and the framework's λ'_j. We thus stipulate the invariant that there exists $z' \in \mathbb{N}$ such that

$$\lambda' = \lambda'_j - z'\omega_j + [\mathsf{rev}\, \mathsf{path}_{\mathsf{rev} \circ q'_j}\, f].$$

Intuitively, $\mathsf{rev}\, \mathsf{path}_{\mathsf{rev} \circ q'_j}\, f$ excludes those darts that have had their reduced lengths successfully but tentatively decreased by 1.

This invariant is straightforward to verify for loops L4 and L6. In L4, the only complication is that, whenever f changes from f_∞ to some other value, we need to increment z'. Let $z = z' - 1$ if $f \neq f_\infty$, or $z = z'$ if $f = f_\infty$. Just before a pivot, we

have $\lambda'_{j+1} = \lambda'_j - z\omega_j$. Since the edge chain

$$(z - z')\omega_j + [\mathsf{rev}\,\mathsf{path}_{\mathsf{rev}\circ q'_j}\, f] - [\mathsf{rev}\,\mathsf{path}_{\mathsf{rev}\circ q'_{j+1}}\, f]$$

belongs to Cycles G^* and is carried by Edges $(\mathsf{rev} \circ q'_{j+1})$, by Lemma 10, there exists $z'' \in \{0, 1\}$ such that

$$(z - z')\omega_j + [\mathsf{rev}\,\mathsf{path}_{\mathsf{rev}\circ q'_j}\, f] = -z''\omega_{j+1} + [\mathsf{rev}\,\mathsf{path}_{\mathsf{rev}\circ q'_{j+1}}\, f],$$

and the invariant is preserved across pivots.

Finally, we must verify that the pivoted dart a_{j+1} is eligible and leafward of other eligible darts, where "leafward" means appearing beforehand in the sequence

$$q(f_\infty), (q \circ \mathsf{right}_G \circ q)(f_\infty), (q \circ \mathsf{right}_G \circ q \circ \mathsf{right}_G \circ q)(f_\infty), \ldots.$$

If $\omega_j(a) \neq 1$, then $f \neq f_\infty$, and it follows that $\lambda'(a) \geq 1$, so a is not pivoted. Otherwise, the fact that $\lambda'(a) = 0$ implies, in concert with the inequality $\lambda'(a) \geq 0$, that a is eligible. Every dart a' leafward of a satisfies $\lambda'_j(a') > \lambda'_j(a)$ and so is not eligible.

5.5 Analysis of the running time

Every step of the algorithm in Figure 5.2 can be charged to one of two types of operations: inserting a dart into q, the interdigitating arborescence for the dual of G, and decreasing by 1 the reduced length of a dart in that arborescence. To prove that the running time is $O\big(|E| + \sum_{a \in \mathrm{Darts}\, E} \lambda(a)\big)$, it suffices to show that each dart

is inserted into q at most once. Our analysis generalizes that of Cabello et al. by handling degeneracy and counting insertions made by Evert.

For every vertex $v \in \operatorname{Vertices} \hat{G}$ and every time j, we define

$$\psi_j^v \triangleq [\mathsf{path}_{\hat{p}_j}\, v].$$

Observe that $\partial_1^{\hat{G}}(\psi_0^v - \psi_j^v) = 0$. For every v and j, there exists by Lemmas 10 and 12 a unique solution $\phi_j^v \in \operatorname{FaceChains} \hat{G}$ to the equations

$$\phi_j^v(\hat{f}_\infty) = 0 \qquad\qquad \partial_2^{\hat{G}} \phi_j^v = \psi_0^v - \psi_j^v.$$

See Figure 5.3. Clearly, $\phi_0^v = 0$. By Lemma 12 again and some tedious work on the correspondence between G and $\hat{G}$, we can prove also that $\phi_{j'}^v \geq \phi_j^v$ for every time $j' \geq j$. (Intuitively, every path is no less counterclockwise than the last.) As a special case, $\phi_j^v \geq \phi_0^v = 0$. This much does not require leafmost pivots, but the following lemmas do.

Lemma 33. *Assume leafmost pivots. For every v and j, we have $\phi_j^v \leq 1$.*

Proof. Assume to the contrary that there exist v and j and $\hat{f}$ such that $\phi_j^v(\hat{f}) > 1$. Without loss of generality, let $k = \phi_j^v(\hat{f})$ and assume that $\phi_j^v \leq k$. From $\hat{G}$, construct an embedded graph H by iteratively deleting every edge e that is not a bridge in the current iterate of H but that satisfies $\phi_j^v\big(\mathsf{right}_{\hat{G}}(e, 1)\big) = \phi_j^v\big(\mathsf{right}_{\hat{G}}(e, -1)\big)$. Let $\langle c_1, c_2, \ldots, c_m \rangle$ enumerate in clockwise order (the boundary of) the face in H corresponding to $\hat{f}$. By Lemma 13 and the maximality of k (to rule out the reverse orientations), it holds for every dart c_i that c_i appears in $\mathsf{path}_{\hat{p}_0}\, v$ or c_i appears in $\mathsf{rev}\,\mathsf{path}_{\hat{p}_j}\, v$ or $\mathsf{edge}\, c_i$ is a bridge in H. Moreover, $\mathsf{edge}\, c_i \in E$ (i.e., $\mathsf{edge}\, c_i$ appears in the unmodified input, G), since ϕ_f^v maps to 0 or 1 every face incident to at least

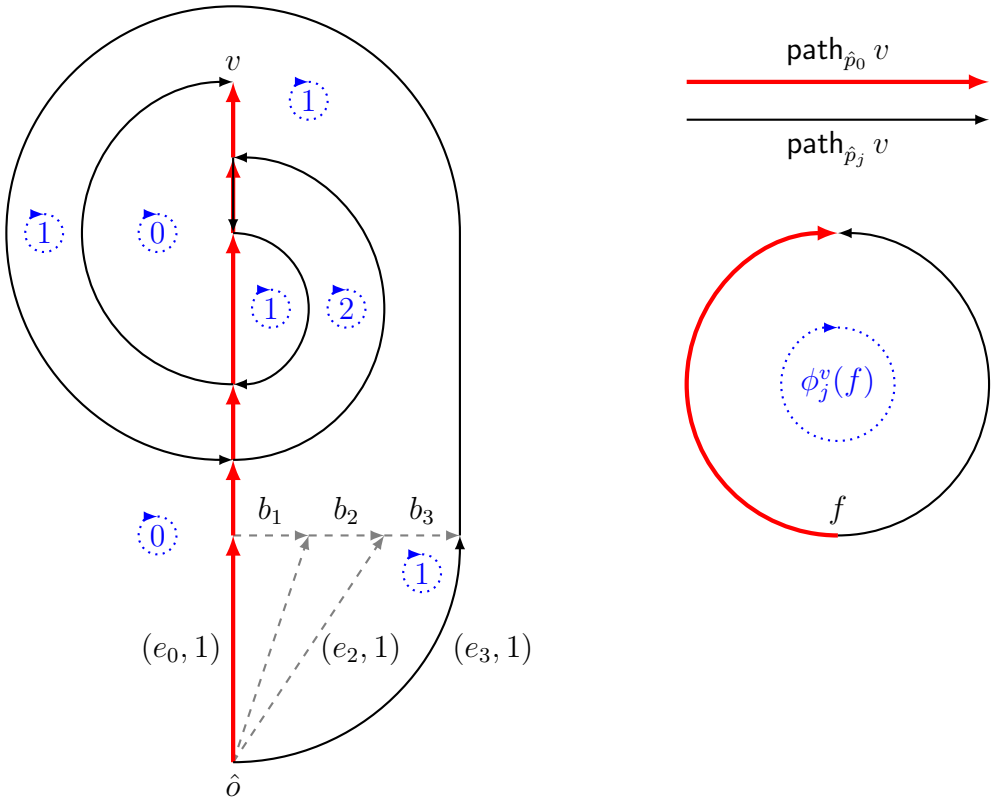


Figure 5.3: The construction of ϕ_j^v.

one edge $e \in \hat{E} \setminus E$. Every path in the dual of $\hat{G}$ from $\hat{f}$ to $\hat{f}_\infty$ includes one of these darts whose edge is not a bridge. Since $\hat{f}$ and $\hat{f}_\infty$ have corresponding faces f and f_∞ respectively in G, the same statement holds for paths in the dual of G.

Let $j' < j$ be the last time at which at least one dart $\mathsf{rev}\, c_i$ appearing in $\mathsf{path}_{\hat{p}_j}\, v$ is not present in $\hat{p}_{j'}$. By the maximality of j', the next pivot, $a_{j'+1}$, is $\mathsf{rev}\, c_i$. The search path $\mathsf{rev}\, \mathsf{path}_{\mathsf{rev} \circ q_j}\, \mathsf{left}_G\, \mathsf{rev}\, c_i$ necessarily includes a dart in $\mathsf{path}_{\hat{p}_0}\, v$, since all of the other boundary darts are in $\hat{p}_j$. As we show, this dart necessarily has reduced length zero, violating the leafmost pivot rule.

Assume without loss of generality that no $\mathsf{edge}\, c_i$ is a bridge. (If $\mathsf{edge}\, c_i$ is a bridge and c_i appears before $\mathsf{rev}\, c_i$, then we apply this argument inductively to the subsequence from c_i exclusive to $\mathsf{rev}\, c_i$ exclusive, and to the subsequence from $\mathsf{rev}\, c_i$ exclusive to the end followed by the subsequence from the beginning to c_i exclusive.) The relevant property of the sequence now is that it is a path from some vertex to itself (a cycle), obtained by concatenating subpaths alternately of $\mathsf{path}_{\hat{p}_0}\, v$ and $\mathsf{path}_{\hat{p}_j}\, v$, with all darts appearing in the unmodified input, G. Let $w_1, w_2, \ldots, w_r$ be consecutive vertices from rootmost to leafmost in $\mathsf{path}_{\hat{p}_0}\, v$, where w_1 is the rootmost vertex that $\mathsf{path}_{\hat{p}_0}\, v$ has in common with the cycle, and w_r is the leafmost. There exists a sequence of pairs $\big(\alpha(1), \beta(1)\big), \big(\alpha(2), \beta(2)\big), \ldots, \big(\alpha(z), \beta(z)\big)$ such that $\alpha(1) = r$ and $\beta(z) = 1$ and, for every i, we have $\beta(i) < \alpha(i+1)$, and the cycle contains the subpath of $\mathsf{rev}\, \mathsf{path}_{\hat{p}_j}\, v$ from $w_{\alpha(i)}$ to $w_{\beta(i)}$ and the subpath of $\mathsf{path}_{\hat{p}_0}\, v$ from $w_{\beta(i)}$ to $w_{\alpha(i+1)}$. We conclude that, since all of these subpaths are shortest, every dart on the subpath of $\mathsf{path}_{\hat{p}_0}\, v$ from $w_{\beta(i)}$ to $w_{\alpha(i)}$ has reduced length zero, and the union of the intervals $\big[\beta(i), \alpha(i)\big]$ is $[1, r]$. □

Lemma 34. *Assuming leafmost pivots, each dart is inserted into the dual arborescence q at most once.*

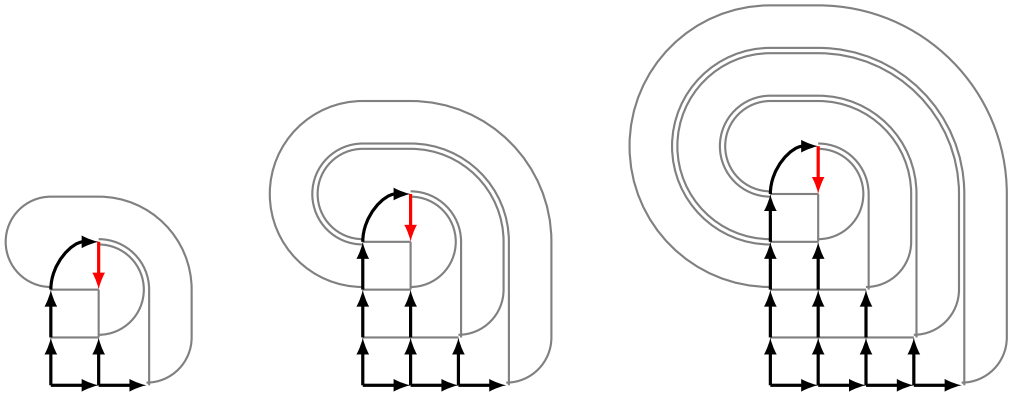


Figure 5.4: Members for $k \in \{5, 6, 7\}$ of a family of planar embedded graphs with k edges incident to the infinite face, on which MSSP without the leafmost pivot rule can pivot the red edge k times. The maximum with the leafmost pivot rule is twice. (The length of every dart is 0.)

Proof. Fix a dart c. For every j, let $\rho_j \in \operatorname{FaceChains} \hat{G}$ satisfy the equations

$$\rho_j(\hat{f}_\infty) = 0 \qquad\qquad \partial_2^G \rho_j = \mathsf{fundamentalCycle}_{\hat{p}_j}\, c.$$

Observe that, for every j,

$$\rho_j = \rho_0 - \phi_j^{\mathsf{tail}_{\hat{G}}\, c} + \phi_j^{\mathsf{head}_{\hat{G}}\, c}.$$

The dart c is present in q_j if and only if $\rho_j(\mathsf{left}_{\hat{G}}\, c) = -1$. The times j at which c is inserted satisfy

$$\phi_{j-1}^{\mathsf{tail}_{\hat{G}}\, c}(\mathsf{left}_{\hat{G}}\, c) < \phi_j^{\mathsf{tail}_{\hat{G}}\, c}(\mathsf{left}_{\hat{G}}\, c),$$

and by Lemma 33, there can be only one. □

5.6 Necessity of a rule for breaking ties

When multiple darts are eligible to be pivoted, the correctness of MSSP does not require a specific choice. Nevertheless, if this choice is made in an adversarial manner

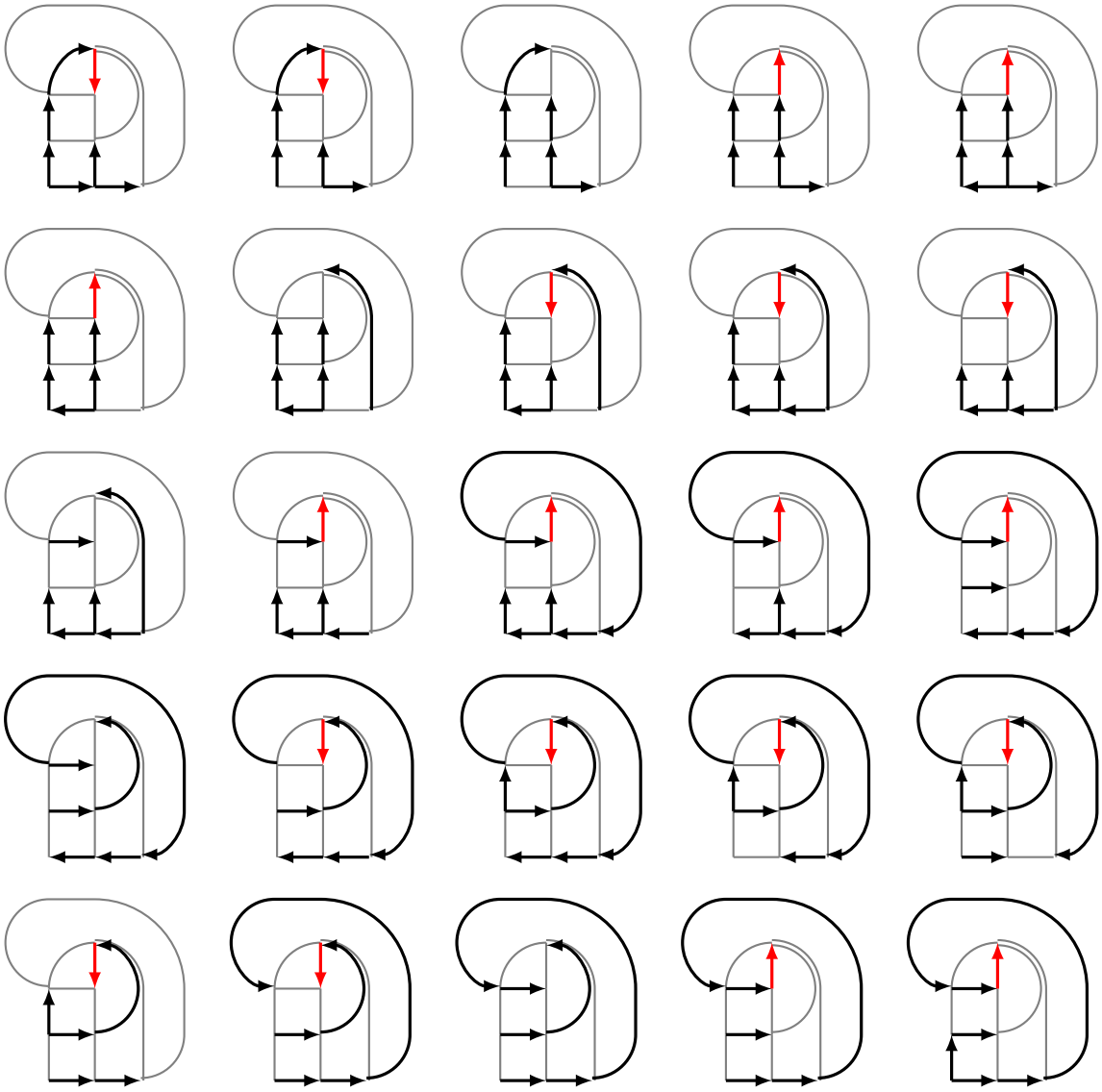


Figure 5.5: A complete execution of MSSP without the leafmost pivot rule on the $k = 5$ member of the family depicted in Figure 5.4. The red edge is pivoted 5 times.

without regard to the leafmost pivot rule, then the running time may be significantly more than linear. Figure 5.4 depicts a family of problematic inputs, where one distinguished edge can be pivoted an unbounded number of times (see Figure 5.5). By subdividing that distinguished edge and adjusting the executions accordingly, we find that the worst-case number of pivots is $\Omega(m^{3/2})$, where m is the number of edges in the embedded graph.

5.7 Implementation notes

MSSP can be implemented with a dynamic tree that is either edge-weighted or node-weighted. An edge-weighted tree can provide an efficient general-purpose Evert operation but practically is half as fast as a node-weighted tree and requires the reduced length of one of the edges of the cycle separating the old root from the new to be maintained out of band. The cost of Evert for node-weighted trees results from the fact that we need to simulate edge weights. When a path is reversed, the weights of the edges on the path must be moved from one node corresponding to an endpoint to the other. A happy consequence of Lemma 34, however, is that the Evert operations issued require only linear time over the course of the entire algorithm.

It is possible to generalize MSSP to change between roots that are not adjacent. Unfortunately, this generalization seems to be mutually exclusive with the leafmost pivot rule, as there is no longer a common face with respect to which "leafmost" can be defined. The most natural faces from which to start the search are those incident to the edge most recently cut.

MSSP can be implemented comfortably with fixed-width integers and without

the possibility of integer overflow. Assuming that the combined length of the two darts of each edge can be represented, every derived vector of reduced lengths can be represented as well, even if some distances cannot.

CHAPTER **Six**

Centrality measures

Let G be a graph with edge lengths and let d be the associated metric on the set of vertices V. The **farness** of a vertex v is $\sum_{w \in V} d(v, w)$, and the **closeness** of v is the reciprocal of that quantity. Under the simplifying assumption that, for every pair of vertices s, t, there exists a unique shortest st-path, the **betweenness** of a vertex v is

$$\frac{\Big|\big\{(s,t) : \{s,t\} \subseteq V \setminus \{v\},\ d(s,v) + d(v,t) = d(s,t)\big\}\Big|}{\big(|V|-1\big)\big(|V|-2\big)}.$$

Both definitions are due to Freeman [1978–1979]. The condition $d(s, v) + d(v, t) = d(s, t)$ coincides with the proposition that v lies on the shortest st-path. **Nondegeneracy** assumptions like these are commonly made in the computational geometry literature (see, for example, the work of Cabello et al. [2013]).

In this chapter, we give algorithms to compute these centrality measures in planar graphs from the output of multiple-source shortest paths. Our algorithms are based

on the node-weighted dynamic tree data structure of Sleator and Tarjan [1985]. We do not prove a worst-case time bound better than the trivial bound $O\big(|E|^2 \log |E|\big)$ but instead present evidence that these algorithms are significantly more efficient in practice than previous exact algorithms.

For betweenness centrality in general graphs, the $O\Big(|V|\big(|E| + |V| \log |V|\big)\Big)$-time algorithm due to Brandes [2001], which computes shortest path trees from each vertex, is still asymptotically fastest among known exact algorithms. This running time is not good enough for large graphs, so a variety of approximate approaches have been developed; see, e.g., the paper of Riondato and Kornaropoulos [2014] for the newest and a summary of the others. The situation with closeness centrality is similar.

On the problem of betweenness centrality in road networks specifically, the system PHAST of Delling et al. [2013] is the only work known to us, and we use it as a point of reference in our experimental evaluation. PHAST computes all-pairs shortest paths but creates opportunities for data parallelism via the contraction hierarchies technique of Geisberger, Sanders, Schultes, and Delling [2008], one of several techniques for shortest paths found by [Abraham, Fiat, Goldberg, and Werneck, 2010] to exploit the low highway dimension of typical road networks, a measure that they introduced.

6.1 Sleator–Tarjan trees

Sleator and Tarjan [1985] present a data structure that represents a rooted forest where each node has a real value. This data structure supports the following oper-

ations, each of which runs in amortized time $O(\log n)$. Note that we have altered some of the operation names and omitted, among others, the operations for ancestor minimum, eversion, and leafmost common ancestor.

New() Returns a new node, different from all nodes returned previously by New(). This node has value 0 and is the only node in the tree to which it belongs.

Root(v) Returns the root node of the tree to which node v belongs.

Parent(v) Returns the parent node of node v. Precondition: $\text{Root}(v) \neq v$.

Link(v, w) Inserts an arc to node v from node w, which becomes v's parent. Preconditions: $\text{Root}(v) = v$ and $\text{Root}(w) \neq v$.

Cut(v, w) Deletes the arc to node v from node w. Preconditions: $\text{Root}(v) \neq v$ and $\text{Parent}(v) = w$.

Value(v) Returns the value of node v.

Update(v, x) Adds x to the value of each of the nodes

$$v,\ \text{Parent}(v),\ \text{Parent}\big(\text{Parent}(v)\big),\ \ldots,\ \text{Root}(v),$$

the ancestors of v.

The term "dynamic tree" refers to the data structure of Sleator and Tarjan as well as data structures with similar interfaces developed subsequently, such as top trees [Alstrup et al., 2005].

6.2 Sleator–Tarjan trees with generalized updates

In this section, we show how to generalize the values in a Sleator–Tarjan tree from real numbers under addition to arbitrary groups with computable products and inverses. The resulting data structure supports noncommutative updates with both ancestor and descendant scope. We prefer Sleator–Tarjan trees as a base on account of their superior practical performance [Tarjan and Werneck, 2010].

6.2.1 Groups

A **group** consists of a set G and a binary operation $G \times G \to G$ denoted by concatenation that together satisfy several axioms.

Associativity For every $x, y, z \in G$, we have $(xy)z = x(yz)$.

Identity There exists an element $e \in G$ such that, for every $x \in G$, we have $ex = x = xe$. It follows that e is unique.

Inverse For every $x \in G$, there exists an element $x^{-1} \in G$ such that $x^{-1}x = e = xx^{-1}$. It follows that x^{-1} is unique and that $(x^{-1})^{-1} = x$.

An **endomorphism** is a map $h : G \to G$ such that, for every $x, y \in G$, we have $h(xy) = h(x)h(y)$. It follows that $h(e) = e$ and that, for every $x \in G$, we have $h(x^{-1}) = h(x)^{-1}$. The map h is **idempotent** if and only if, for every $x \in G$, we have $h\big(h(x)\big) = h(x)$.

6.2.2 Generalized updates

Given a group G and an idempotent endomorphism h, we specify a generalized update operation.

New() The initial value of a new node is still the identity element, now denoted e.

Update(v, x) For each node w in the tree to which v belongs: if w is an ancestor of v, then update w's value from y to yx; otherwise, update w's value from y to $yh(x)$.

When G is the real numbers under addition and $h(x) = e$, we recover the old specification.

6.2.3 Operational underpinning

Sleator–Tarjan trees perform operations on the **virtual tree** by manipulating an internal linked collection of binary trees called the **actual tree**. Both trees have the same set of nodes, but their parent relationships are in general different. Every node in the actual tree has at most one **left child**, at most one **right child**, and any number of **middle children**. We refer to left and right children as **solid children** and to middle children as **dashed children**. A **solid descendant** of a node is the node itself, or a solid child, or a solid child of a solid child, etc. A **solid ancestor** of a node v is a node of which v is a solid descendant. A **solid root** is a node that is not a solid child. The key internal operation Expose(v) rearranges the actual tree so that v is the actual root and the solid descendants of v are the ancestors of v. In turn, Expose(v) depends on two other internal operations, Rotate(u, v, w, x) and

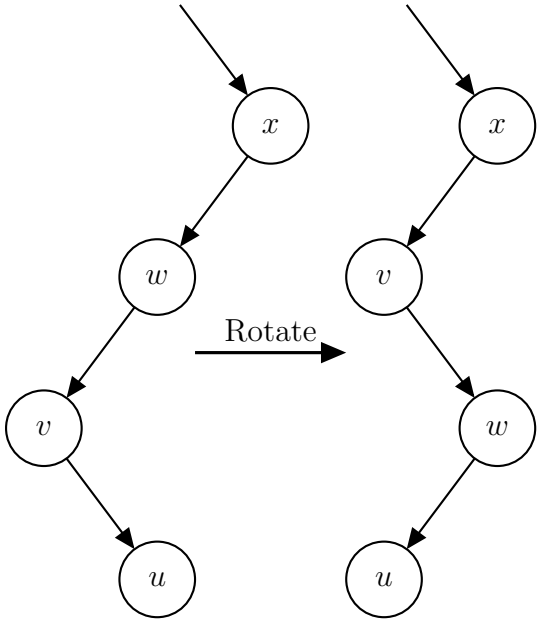
Splice(v, w, x).

Sleator–Tarjan trees support Update(v, x) by not storing each value directly. Instead, each node has an associated quantity Δval, and the value of a node v is the sum $\delta_0 + \delta_1 + \cdots + \delta_\ell$ where δ_i is the Δval of the solid ancestor i parents above v. Note that this sum ends just after the first solid root encountered while moving rootward from v. Updates to the values of the virtual ancestors of v are effected by calling Expose(v) and then adding to v's Δval. Value(v) can call Expose(v) and then return v's Δval.

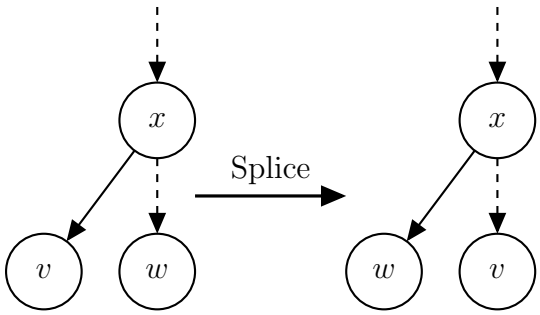
To support generalized updates, we redefine the value of a node v to be the group product $\delta_0\delta_1 \cdots \delta_\ell h(\delta_{\ell+1}\delta_{\ell+2} \cdots \delta_{\ell'})$ where $\delta_0, \ldots, \delta_\ell$ are the Δvalues of the solid ancestors of v and $\delta_{\ell+1}, \ldots, \delta_{\ell'}$ are the Δvalues of the other actual ancestors, all in rootward order. Update(v, x) is carried out as before, by calling Expose(v) and multiplying v's Δval on the right by x. All we have to do is to show that each of four topological operations on the actual tree can be made to preserve values. See Figure 6.1.

6.3 Algorithms

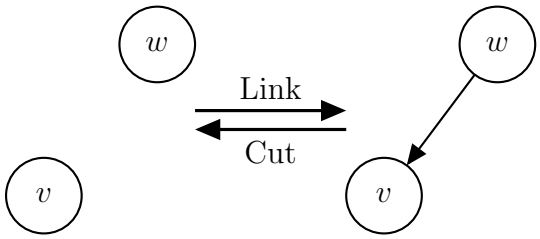
In this section, we describe our algorithms to compute closeness and betweenness centrality. Both algorithms use multiple-source shortest paths to compute all shortest path trees. Both involve a dynamic tree with generalized updates over the group of 3×3 unit upper triangular matrices with integer entries, under matrix multiplication.



On Rotate(u, v, w, x):
$v.\Delta\text{val} \leftarrow v.\Delta\text{val}\ w.\Delta\text{val}$
$u.\Delta\text{val} \leftarrow u.\Delta\text{val}\ v.\Delta\text{val}$
$u.\Delta\text{val} \leftarrow u.\Delta\text{val}\ (w.\Delta\text{val})^{-1}$
$w.\Delta\text{val} \leftarrow w.\Delta\text{val}\ (v.\Delta\text{val})^{-1}$



On Splice(v, w, x):
$v.\Delta\text{val} \leftarrow v.\Delta\text{val}\ x.\Delta\text{val}$
$v.\Delta\text{val} \leftarrow v.\Delta\text{val}\ \big(h(x.\Delta\text{val})\big)^{-1}$
$w.\Delta\text{val} \leftarrow w.\Delta\text{val}\ h(x.\Delta\text{val})$
$w.\Delta\text{val} \leftarrow w.\Delta\text{val}\ (x.\Delta\text{val})^{-1}$



On Link(v, w):
$v.\Delta\text{val} \leftarrow v.\Delta\text{val}\ (w.\Delta\text{val})^{-1}$

On Cut(v, w):
$v.\Delta\text{val} \leftarrow v.\Delta\text{val}\ w.\Delta\text{val}$

Figure 6.1: Maintenance of the Δval field after topological changes to the actual tree. Nodes with a tailless solid incoming arc have a solid parent, a dashed parent, or no parent. Nodes with a tailless dashed incoming arc have a dashed parent or no parent. Nodes with no incoming arc have no parent.

Rational entries are not required because

$$\begin{bmatrix} 1 & a & c \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -a & ab-c \\ 0 & 1 & -b \\ 0 & 0 & 1 \end{bmatrix}.$$

For implementations of these algorithms, it suffices to store only the three entries above the diagonal.

6.3.1 Computing closeness centrality

Since closeness and farness are reciprocally related, we focus on computing farness. We pipe the output of multiple-source shortest paths to a dynamic tree that maintains, for each node, the sum of distances to its descendants, i.e., the farness of that node with respect to the nodes in its subtree of the shortest path tree. After each root change, we trivially query the farness of the new root.

In order to maintain farness (f), we maintain two other quantities for each node. The first is the distance to that node from the root of its tree (d). The second is its descendant count (n). We pack these into a 3×3 matrix as follows.

$$\begin{bmatrix} 1 & -d & f \\ 0 & 1 & n \\ 0 & 0 & 1 \end{bmatrix}$$

The reason for storing $-d$ and not d will become clear shortly. We let

$$h\left(\begin{bmatrix} 1 & -d & f \\ 0 & 1 & n \\ 0 & 0 & 1 \end{bmatrix}\right) = \begin{bmatrix} 1 & -d & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and verify that h is an idempotent endomorphism. Intuitively, updates to d affect descendants, and updates to n and f affect ancestors.

We initialize the value of each node to

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Before linking an arc to v from w of length ℓ, we make the following updates. After each cut, we do the inverse. First, we retrieve the matrix entries for v and w.

$$\begin{bmatrix} 1 & -d_v & f_v \\ 0 & 1 & n_v \\ 0 & 0 & 1 \end{bmatrix} \leftarrow \text{Value}(v) \qquad \begin{bmatrix} 1 & -d_w & f_w \\ 0 & 1 & n_w \\ 0 & 0 & 1 \end{bmatrix} \leftarrow \text{Value}(w).$$

Second, we increase by $\ell + d_w$ the distance of each node in v's subtree from the root, which is changing from v to $\text{Root}(w)$.

$$\text{Update}\left(v, \begin{bmatrix} 1 & -(\ell + d_w) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)$$

Third, we update the descendant count and farness of each of w's ancestors.

$$\text{Update}\left(w,\ \begin{bmatrix} 1 & 0 & f_v + n_v(\ell + d_w) \\ 0 & 1 & n_v \\ 0 & 0 & 1 \end{bmatrix}\right)$$

This final update bears some explanation. Let's work through an example for some ancestor x.

$$\begin{bmatrix} 1 & -d_x & f_x \\ 0 & 1 & n_x \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & f_v + n_v(\ell + d_w) \\ 0 & 1 & n_v \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -d_x & f_v + n_v(\ell + d_w - d_x) + f_x \\ 0 & 1 & n_v + n_x \\ 0 & 0 & 1 \end{bmatrix}$$

The descendant count is incremented by n_v. The farness is incremented by $f_v + n_v(\ell + d_w - d_x)$. The quantity $\ell + d_w - d_x$ is the distance from x to v. Letting u range over the descendants of v, which are n_v in number, the difference between x's new and old farness is

$$\begin{aligned} \sum_u d(x,u) &= \sum_u \big(d(v,u) + d(x,v)\big) \\ &= \sum_u d(v,u) + n_v d(x,v) \\ &= f_v + n_v(\ell + d_w - d_x), \end{aligned}$$

as desired.

6.3.2 Computing betweenness centrality

Recall our nondegeneracy assumption that, for every pair of vertices s, t, there exists a *unique* shortest st-path. As before, we pipe the output of multiple-source shortest paths to a dynamic tree. The value of each node v is comprised of three quantities. The first, n_v, is the number of *proper* descendants. The second, b_v, is

$$\Big|\big\{(s,t) : s \in R \setminus \{v\},\ t \in V \setminus \{v\},\ d(s,v) + d(v,t) = d(s,t)\big\}\Big|,$$

where R is the set of roots that have been considered so far. At the end, when $R = V$, the betweenness centrality of v is

$$\frac{b_v}{\big(|V|-1\big)\big(|V|-2\big)}.$$

The third quantity is k_v, the number of times that b_v has been updated. It's needed to make the algebra work out. We define

$$h\left(\begin{bmatrix} 1 & n & b \\ 0 & 1 & k \\ 0 & 0 & 1 \end{bmatrix}\right) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & k \\ 0 & 0 & 1 \end{bmatrix},$$

which ensures ancestor scope on updates to n and b.

Before linking an arc to node v from node w, we update the descendant count of each of w's ancestors as follows. After each cut, we do the inverse.

$$\begin{bmatrix} 1 & n_v & b_v \\ 0 & 1 & k_v \\ 0 & 0 & 1 \end{bmatrix} \leftarrow \text{Value}(v) \qquad \text{Update}\left(w,\ \begin{bmatrix} 1 & n_v+1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)$$

For each vertex v, when we have built the shortest path tree with root v, we apply the following update.

$$\begin{bmatrix} 1 & n_v & b_v \\ 0 & 1 & k_v \\ 0 & 0 & 1 \end{bmatrix} \leftarrow \text{Value}(v) \qquad \text{Update}\left(w, \begin{bmatrix} 1 & 0 & -n_v \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}\right)$$

The effect of this update on v is to increment k_v.

$$\begin{bmatrix} 1 & n_v & b_v \\ 0 & 1 & k_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -n_v \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & n_v & b_v \\ 0 & 1 & 1 + k_v \\ 0 & 0 & 1 \end{bmatrix}$$

The effect on v's proper descendants u is to increment b_u by n_u, the number of shortest paths from v that pass through u on the way to another vertex.

$$\begin{bmatrix} 1 & n_u & b_u \\ 0 & 1 & k_u \\ 0 & 0 & 1 \end{bmatrix} h\left(\begin{bmatrix} 1 & 0 & -n_v \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}\right) = \begin{bmatrix} 1 & n_u & b_u \\ 0 & 1 & k_u \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & n_u & n_u + b_u \\ 0 & 1 & 1 + k_u \\ 0 & 0 & 1 \end{bmatrix}$$

6.4 Experimental results

We implemented our algorithm for betweenness centrality in C. The source code and scripts to replicate the results in this section are available at `http://www.davideisenstat.com/mississippi/`.

The machine that we used for our benchmarks was a commodity workstation running Linux 3.2, with 16 GiB of DDR3-1066 RAM and a quad-core AMD Phenom II X4 955 processor clocked at 3.2 GHz. We used Clang 3.2 with the optimization

flag `-O3` to compile our C source code with POSIX Threads for parallelization. Our test graphs were derived from the U.S. Census Bureau's 2012 TIGER/Line data for the road networks of the United States and its territories. The whole graph has 28.1 million vertices and 35.7 million edges. The edge lengths were approximately proportional to physical distances.

We compare our implementation against PHAST and GPHAST [Delling et al., 2013], which also compute shortest path trees for each possible root. The benchmarks reported for PHAST in that paper were performed on a workstation comparable to ours, with 12 GiB of DDR3-1066 RAM and a quad-core Intel Core i7-920 processor clocked at 2.66 GHz (with greater throughput per clock cycle than the 3.2-GHz AMD). This workstation was running Windows 7 Enterprise, and the authors used Microsoft Visual C++ 2010 to compile their C++ source code with OpenMP for parallelization. Their test graph is from the Ninth DIMACS Implementation Challenge and was derived from 2000 TIGER/Line data on the U.S. road network, with edge lengths proportional to distance. It has only 23.9 million vertices. GPHAST used additionally an NVIDIA GeForce GTX 580 graphics processing unit.

6.4.1 Comparison of dynamic trees

Tarjan and Werneck [2010] compared several implementations of dynamic trees and found that, with one exception, Sleator–Tarjan trees (ST-V) dominate the competition on change-heavy workloads like multiple-source shortest paths. The exception is a naive tree where each node has just a singly linked parent and an explicit value (LIN-V). LIN-V is faster on small trees but slower on large ones and not amenable to updates with descendant scope.

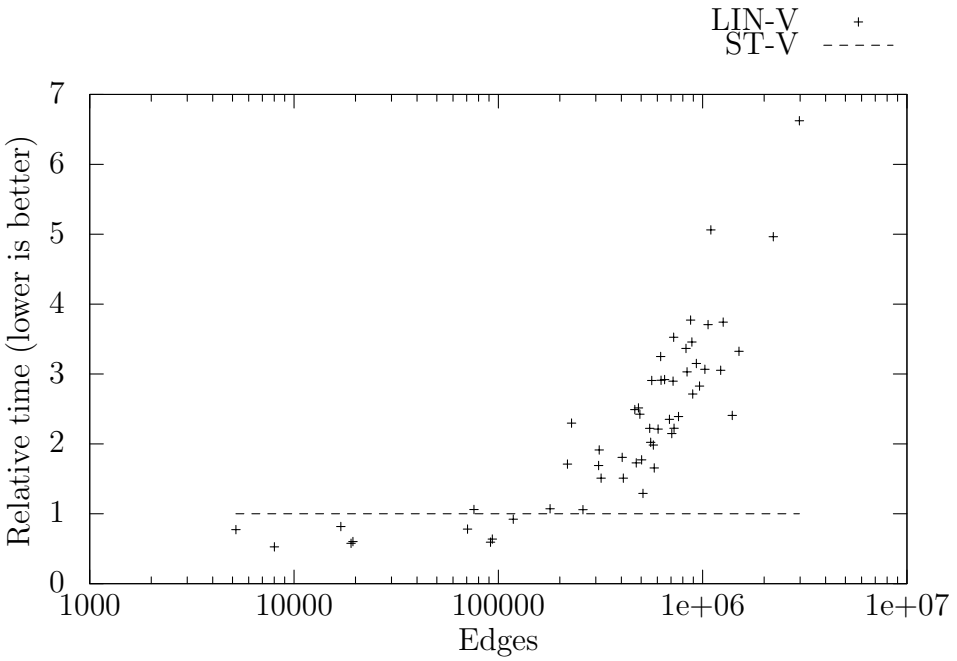


Figure 6.2: Comparison of the performance of dynamic trees based on splay trees and singly linked lists for all-roots multiple-source shortest paths on graphs derived from the road networks of U.S. states and territories.

In Figure 6.2, we plot the relative performance of ST-V and LIN-V for all-roots multiple-shortest paths on the states and territories of the U.S. We measure single-threaded performance without the betweenness calculation. ST-V is competitive with LIN-V on the smaller graphs and significantly outperforms it on the larger ones.

6.4.2 Scalability

Like PHAST and GPHAST, our algorithm for betweenness centrality is embarrassingly parallel. We store the traveling salesman tour in a work queue and let each thread dequeue a fixed fraction of its share of the remaining work as needed. From one core to four, we observe a speedup factor of 3.7 in computing betweenness centrality for the entire U.S. road network.

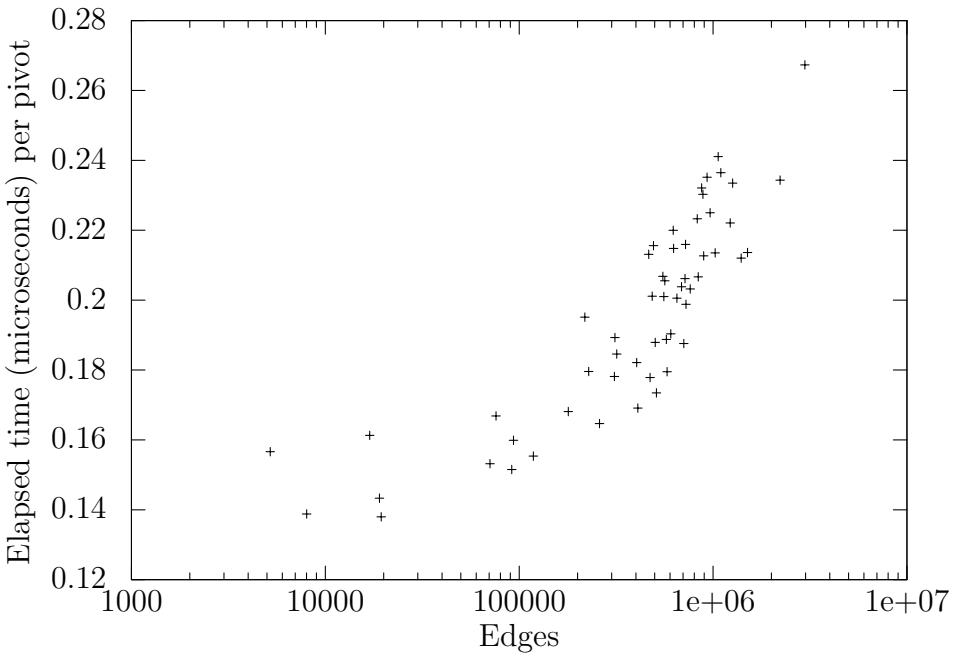


Figure 6.3: Elapsed time in microseconds per pivot when computing betweenness centrality with multiple-source shortest paths on graphs derived from the road networks of U.S. states and territories.

An all-roots MSSP computation makes a number of pivots, each of which in theory requires logarithmic time. In Figure 6.3, we plot the elapsed time per pivot with semi-log axes. The datum for the whole U.S. is not plotted; the time per pivot is 0.40 microseconds. The inflection point seems plausibly explained by caching effects, and we find nothing else to disprove the theory of logarithmic costs.

In Figure 6.4, we plot the number of pivots per edge. The datum for the whole U.S. again is not plotted, on which input there are 574 pivots per edge. This quantity depends mainly on the order in which roots are visited. On the theory that shorter distances between roots means fewer pivots, we use a 2-approximate traveling salesman path derived from a minimum spanning tree. This is considerably better than a random order, whose estimated expected pivot count is several orders of magnitude higher, but comparable to depth-first search on the initial shortest path tree. We have not devoted any significant effort to investigating alternative approaches.

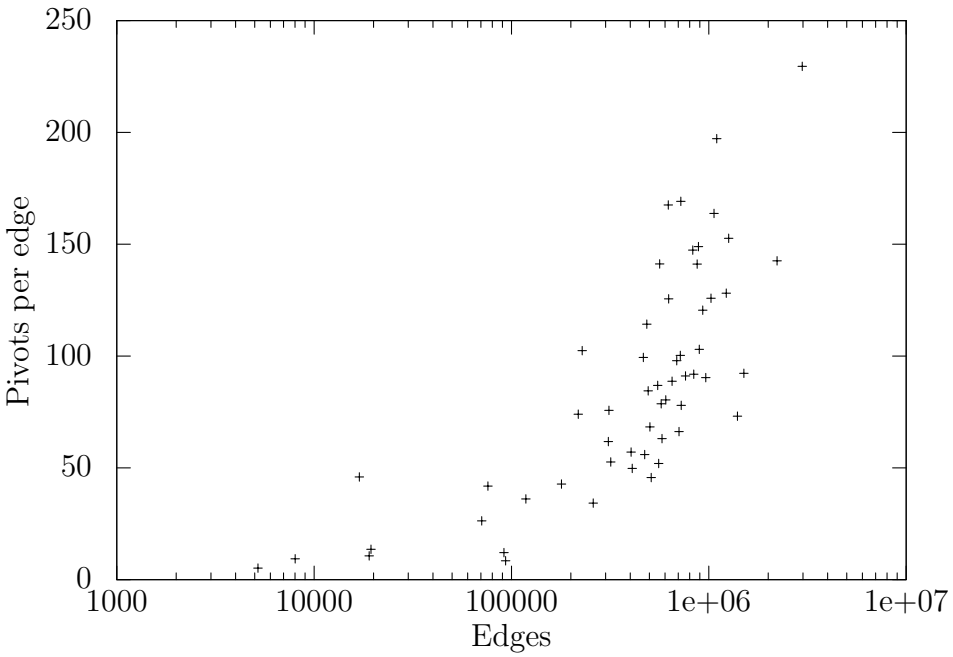


Figure 6.4: Pivots per edge when computing betweenness centrality with multiple-source shortest paths on graphs derived from the road networks of U.S. states and territories.

As the number of edges increases, the number of pivots per edge tends to increase as well, but much less than linearly. This is crucial to the excellent performance of our algorithm demonstrated in the next subsection.

6.4.3 Comparison with Dijkstra, PHAST, and GPHAST

Table 6.1 presents our results alongside those reported by Delling et al. [2013]. The total running time of our implementation is 137 minutes. Our speedup is attributable largely to algorithms that avoid the need to do linear work for each vertex, however good the constant may be. Note, however, that PHAST and GPHAST handle graphs that are not planar, while our algorithm does not.

Algorithm	Time (ms) per vertex
Dijkstra	947.75
PHAST	28.81
GPHAST	4.65
Our algorithm	0.29

Table 6.1: Elapsed time in milliseconds per vertex when computing all shortest path trees on graphs derived from the U.S. road network. Our algorithm additionally computes betweenness centrality.

BIBLIOGRAPHY

Ittai Abraham, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. Highway Dimension, Shortest Paths, and Provably Efficient Algorithms. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 782–793, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics. ISBN 978-0-898716-98-6. URL `http://dl.acm.org/citation.cfm?id=1873601.1873665`.

Pankaj K Agarwal and Cecilia M Procopiuc. Approximation algorithms for projective clustering. *Journal of Algorithms*, 46(2):115–139, 2003. ISSN 0196-6774. doi: http://dx.doi.org/10.1016/S0196-6774(02)00295-X. URL `http://www.sciencedirect.com/science/article/pii/S019667740200295X`.

Ajit Agrawal, Philip Klein, and R. Ravi. When Trees Collide: An Approximation Algorithm for the Generalized Steiner Problem on Networks. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 134–144, New York, NY, USA, 1991. ACM. ISBN 0-89791-397-3. doi: 10.1145/103418.103437. URL `http://doi.acm.org/10.1145/103418.103437`.

Stephen Alstrup, Jacob Holm, Kristian De Lichtenberg, and Mikkel Thorup. Maintaining Information in Fully Dynamic Trees with Top Trees. *ACM Trans. Algorithms*, 1(2):243–264, October 2005. ISSN 1549-6325. doi: 10.1145/1103963.1103966. URL `http://doi.acm.org/10.1145/1103963.1103966`.

Sanjeev Arora. Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and Other Geometric Problems. *J. ACM*, 45(5):753–782, September 1998. ISSN 0004-5411. doi: 10.1145/290179.290180. URL `http://doi.acm.org/10.1145/290179.290180`.

Sanjeev Arora, Michelangelo Grigni, David Karger, Philip Klein, and Andrzej Woloszyn. A Polynomial-time Approximation Scheme for Weighted Planar Graph TSP. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '98, pages 33–41, Philadelphia, PA, USA, 1998a. Society for Industrial and Applied Mathematics. ISBN 0-89871-410-9. URL `http://dl.acm.org/citation.cfm?id=314613.314632`.

Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation Schemes for Euclidean K-medians and Related Problems. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 106–113, New

York, NY, USA, 1998b. ACM. ISBN 0-89791-962-9. doi: 10.1145/276698.276718. URL http://doi.acm.org/10.1145/276698.276718.

Brenda S. Baker. Approximation Algorithms for NP-complete Problems on Planar Graphs. *J. ACM*, 41(1):153–180, January 1994. ISSN 0004-5411. doi: 10.1145/174644.174650. URL http://doi.acm.org/10.1145/174644.174650.

Mohammadhossein Bateni, Mohammadtaghi Hajiaghayi, and Dániel Marx. Approximation Schemes for Steiner Forest on Planar Graphs and Graphs of Bounded Treewidth. *J. ACM*, 58(5):1–37, October 2011. ISSN 0004-5411. doi: 10.1145/2027216.2027219. URL http://doi.acm.org/10.1145/2027216.2027219.

Piotr Berman, Marek Karpinski, and Alexander Zelikovsky. 1.25-Approximation Algorithm for Steiner Tree Problem with Distances 1 and 2. In Frank Dehne, Marina Gavrilova, Jörg-Rüdiger Sack, and Csaba D. Tóth, editors, *Algorithms and Data Structures*, volume 5664 of *Lecture Notes in Computer Science*, pages 86–97. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-03366-7. doi: 10.1007/978-3-642-03367-4_8. URL http://dx.doi.org/10.1007/978-3-642-03367-4_8.

Piotr Berman, Marek Karpinski, and Alexander Zelikovsky. A 3/2-Approximation Algorithm for Generalized Steiner Trees in Complete Graphs with Edge Lengths 1 and 2. In Otfried Cheong, Kyung-Yong Chwa, and Kunsoo Park, editors, *Algorithms and Computation*, volume 6506 of *Lecture Notes in Computer Science*, pages 15–24. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-17516-9. doi: 10.1007/978-3-642-17517-6_4. URL http://dx.doi.org/10.1007/978-3-642-17517-6_4.

Marshall Bern and Paul Plassmann. The Steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32(4):171–176, 1989. ISSN 0020-0190. doi: http://dx.doi.org/10.1016/0020-0190(89)90039-2. URL http://www.sciencedirect.com/science/article/pii/0020019089900392.

G. Borradaile, P.N. Klein, and C. Mathieu. A Polynomial-Time Approximation Scheme for Euclidean Steiner Forest. In *Foundations of Computer Science, 2008. FOCS '08. IEEE 49th Annual IEEE Symposium on*, pages 115–124, Oct 2008. doi: 10.1109/FOCS.2008.59.

Glencora Borradaile and Philip Klein. An O(N Log N) Algorithm for Maximum St-flow in a Directed Planar Graph. *J. ACM*, 56(2):1–30, April 2009. ISSN 0004-5411. doi: 10.1145/1502793.1502798. URL http://doi.acm.org/10.1145/1502793.1502798.

Glencora Borradaile, Philip Klein, and Claire Mathieu. An O(N Log N) Approximation Scheme for Steiner Tree in Planar Graphs. *ACM Trans. Algorithms*, 5(3):1–31, July 2009. ISSN 1549-6325. doi: 10.1145/1541885.1541892. URL http://doi.acm.org/10.1145/1541885.1541892.

Ulrik Brandes. A faster algorithm for betweenness centrality*. *The Journal of Mathematical Sociology*, 25(2):163–177, 2001. doi: 10.1080/0022250X.2001.9990249. URL http://dx.doi.org/10.1080/0022250X.2001.9990249.

Ulrik Brandes and Dorothea Wagner. A Linear Time Algorithm for the Arc Disjoint Menger Problem in Planar Directed Graphs. *Algorithmica*, 28(1):16–36, 2000.

Jarosław Byrka, Fabrizio Grandoni, Thomas Rothvoss, and Laura Sanità. Steiner Tree Approximation via Iterative Randomized Rounding. *J. ACM*, 60(1):1–33, February 2013. ISSN 0004-5411. doi: 10.1145/2432622.2432628. URL `http://doi.acm.org/10.1145/2432622.2432628`.

S. Cabello, E. Chambers, and J. Erickson. Multiple-Source Shortest Paths in Embedded Graphs. *SIAM Journal on Computing*, 42(4):1542–1571, 2013. doi: 10.1137/120864271. URL `http://epubs.siam.org/doi/abs/10.1137/120864271`.

Daniel Delling, Andrew V. Goldberg, Andreas Nowatzyk, and Renato F. Werneck. PHAST: Hardware-accelerated shortest path trees. *Journal of Parallel and Distributed Computing*, 73(7):940–952, 2013. ISSN 0743-7315. doi: http://dx.doi.org/10.1016/j.jpdc.2012.02.007. URL `http://www.sciencedirect.com/science/article/pii/S074373151200041X`. Best Papers: International Parallel and Distributed Processing Symposium (IPDPS) 2010, 2011 and 2012.

Erik D. Demaine, Fedor V. Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter Algorithms for (K, R)-center in Planar Graphs and Map Graphs. *ACM Trans. Algorithms*, 1(1):33–47, July 2005. ISSN 1549-6325. doi: 10.1145/1077464.1077468. URL `http://doi.acm.org/10.1145/1077464.1077468`.

E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. ISSN 0029-599X. doi: 10.1007/BF01386390. URL `http://dx.doi.org/10.1007/BF01386390`.

Irit Dinur and David Steurer. Analytical Approach to Parallel Repetition. *CoRR*, abs/1305.1979, 2013.

David Eisenstat and Philip N. Klein. Linear-time Algorithms for Max Flow and Multiple-source Shortest Paths in Unit-weight Planar Graphs. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 735–744, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2029-0. doi: 10.1145/2488608.2488702. URL `http://doi.acm.org/10.1145/2488608.2488702`.

David Eisenstat, Philip Klein, and Claire Mathieu. An Efficient Polynomial-time Approximation Scheme for Steiner Forest in Planar Graphs. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 626–638. SIAM, 2012. URL `http://dl.acm.org/citation.cfm?id=2095116.2095169`.

David Eisenstat, Philip N. Klein, and Claire Mathieu. Approximating k-center in planar graphs. In Chandra Chekuri, editor, *SODA*, pages 617–627. SIAM, 2014. ISBN 978-1-61197-338-9, 978-1-61197-340-2.

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 '10, pages 794–804, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics. ISBN 978-0-898716-98-6. URL `http://dl.acm.org/citation.cfm?id=1873601.1873666`.

Tomás Feder and Daniel Greene. Optimal Algorithms for Approximate Clustering. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 434–444, New York, NY, USA, 1988. ACM. ISBN 0-89791-264-0. doi: 10.1145/62212.62255. URL http://doi.acm.org/10.1145/62212.62255.

Uriel Feige. A Threshold of Ln N for Approximating Set Cover. *J. ACM*, 45(4): 634–652, July 1998. ISSN 0004-5411. doi: 10.1145/285055.285059. URL http://doi.acm.org/10.1145/285055.285059.

Dan Feldman, Amos Fiat, Micha Sharir, and Danny Segev. Bi-criteria Linear-time Approximations for Generalized K-mean/Median/Center. In *Proceedings of the Twenty-third Annual Symposium on Computational Geometry*, SCG '07, pages 19–26, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-705-6. doi: 10.1145/1247069.1247073. URL http://doi.acm.org/10.1145/1247069.1247073.

Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, 1978–1979. ISSN 0378-8733. doi: http://dx.doi.org/10.1016/0378-8733(78)90021-7. URL http://www.sciencedirect.com/science/article/pii/0378873378900217.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. ISBN 0716710447.

Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In Catherine C. McGeoch, editor, *Experimental Algorithms*, volume 5038 of *Lecture Notes in Computer Science*, pages 319–333. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-68548-7. doi: 10.1007/978-3-540-68552-4_24. URL http://dx.doi.org/10.1007/978-3-540-68552-4_24.

M. Goemans and D. Williamson. A General Approximation Technique for Constrained Forest Problems. *SIAM Journal on Computing*, 24(2):296–317, 1995. doi: 10.1137/S0097539793242618. URL http://epubs.siam.org/doi/abs/10.1137/S0097539793242618.

Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38(0):293–306, 1985. ISSN 0304-3975. doi: http://dx.doi.org/10.1016/0304-3975(85)90224-5. URL http://www.sciencedirect.com/science/article/pii/0304397585902245.

Teofilo F. Gonzalez. Covering a set of points in multidimensional space. *Information Processing Letters*, 40(4):181–188, 1991. ISSN 0020-0190. doi: http://dx.doi.org/10.1016/0020-0190(91)90075-S. URL http://www.sciencedirect.com/science/article/pii/002001909190075S.

Sariel Har-Peled and Mira Lee. Weighted geometric set cover problems revisited. *JoCG*, 3(1):65–85, 2012.

Monika R Henzinger, Philip Klein, Satish Rao, and Sairam Subramanian. Faster Shortest-Path Algorithms for Planar Graphs. *Journal of Computer and System Sciences*, 55(1):3–23, 1997. ISSN 0022-0000. doi: http://dx.doi.org/10.1006/

jcss.1997.1493. URL http://www.sciencedirect.com/science/article/pii/S0022000097914938.

Dorit S. Hochbaum and Wolfgang Maass. Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI. *J. ACM*, 32(1):130–136, January 1985. ISSN 0004-5411. doi: 10.1145/2455.214106. URL http://doi.acm.org/10.1145/2455.214106.

Dorit S. Hochbaum and David B. Shmoys. A Best Possible Heuristic for the k-Center Problem. *Mathematics of Operations Research*, 10(2):180–184, 1985. doi: 10.1287/moor.10.2.180. URL http://pubsonline.informs.org/doi/abs/10.1287/moor.10.2.180.

P. Klein. A Linear-Time Approximation Scheme for TSP in Undirected Planar Graphs with Edge-Weights. *SIAM Journal on Computing*, 37(6):1926–1952, 2008. doi: 10.1137/060649562. URL http://epubs.siam.org/doi/abs/10.1137/060649562.

Philip N. Klein. Multiple-source Shortest Paths in Planar Graphs. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 146–155, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics. ISBN 0-89871-585-7. URL http://dl.acm.org/citation.cfm?id=1070432.1070454.

Philip N. Klein. A Subset Spanner for Planar Graphs,: With Application to Subset TSP. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 749–756, New York, NY, USA, 2006. ACM. ISBN 1-59593-134-1. doi: 10.1145/1132516.1132620. URL http://doi.acm.org/10.1145/1132516.1132620.

S. Kolliopoulos and S. Rao. A Nearly Linear-Time Approximation Scheme for the Euclidean k-Median Problem. *SIAM Journal on Computing*, 37(3):757–782, 2007. doi: 10.1137/S0097539702404055. URL http://epubs.siam.org/doi/abs/10.1137/S0097539702404055.

Richard J. Lipton, Donald J. Rose, and Robert Endre Tarjan. Generalized Nested Dissection. *SIAM Journal on Numerical Analysis*, 16(2):346–358, 1979. ISSN 00361429. URL http://www.jstor.org/stable/2156840.

J. Mitchell. Guillotine Subdivisions Approximate Polygonal Subdivisions: A Simple Polynomial-Time Approximation Scheme for Geometric TSP, k-MST, and Related Problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999. doi: 10.1137/S0097539796309764. URL http://epubs.siam.org/doi/abs/10.1137/S0097539796309764.

Matteo Riondato and Evgenios M. Kornaropoulos. Fast Approximation of Betweenness Centrality Through Sampling. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 413–422, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2351-2. doi: 10.1145/2556195.2556224. URL http://doi.acm.org/10.1145/2556195.2556224.

Neil Robertson and P.D Seymour. Graph minors. X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991. ISSN 0095-8956. doi: http://dx.doi.org/10.1016/0095-8956(91)90061-N. URL http://www.sciencedirect.com/science/article/pii/009589569190061N.

G. Robins and A. Zelikovsky. Tighter Bounds for Graph Steiner Tree Approximation. *SIAM Journal on Discrete Mathematics*, 19(1):122–134, 2005. doi: 10.1137/S0895480101393155. URL http://epubs.siam.org/doi/abs/10.1137/S0895480101393155.

F.R. Schmidt, E. Toppe, and D. Cremers. Efficient planar graph cuts with applications in Computer Vision. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 351–356, June 2009. doi: 10.1109/CVPR.2009.5206863.

J. Schmidt. All Highest Scoring Paths in Weighted Grid Graphs and Their Application to Finding All Approximate Repeats in Strings. *SIAM Journal on Computing*, 27(4):972–992, 1998. doi: 10.1137/S0097539795288489. URL http://epubs.siam.org/doi/abs/10.1137/S0097539795288489.

Daniel Dominic Sleator and Robert Endre Tarjan. Self-adjusting Binary Search Trees. *J. ACM*, 32(3):652–686, July 1985. ISSN 0004-5411. doi: 10.1145/3828.3835. URL http://doi.acm.org/10.1145/3828.3835.

Robert E. Tarjan and Renato F. Werneck. Dynamic Trees in Practice. *J. Exp. Algorithmics*, 14:5–23, January 2010. ISSN 1084-6654. doi: 10.1145/1498698.1594231. URL http://doi.acm.org/10.1145/1498698.1594231.

Siamak Tazari and Matthias Müller-Hannemann. Dealing with Large Hidden Constants: Engineering a Planar Steiner Tree PTAS. *J. Exp. Algorithmics*, 16:1–33, November 2008. ISSN 1084-6654. doi: 10.1145/1963190.2025382. URL http://doi.acm.org/10.1145/1963190.2025382.

Alexander Tiskin. Semi-local longest common subsequences in subquadratic time. *Journal of Discrete Algorithms*, 6(4):570–581, 2008. ISSN 1570-8667. doi: http://dx.doi.org/10.1016/j.jda.2008.07.001. URL http://www.sciencedirect.com/science/article/pii/S157086670800049X. Selected papers from the 1st Algorithms and Complexity in Durham Workshop (ACiD 2005) 1st Algorithms and Complexity in Durham Workshop (ACiD 2005).

Alexandre Tiskin. Semi-local string comparison: algorithmic techniques and applications. *CoRR*, abs/0707.3619, 2007.