Algorithmica manuscript No.
(will be inserted by the editor)

Polynomial-time approximation schemes for subset-connectivity problems in bounded-genus graphs

Glencora Borradaile · Erik D. Demaine · Siamak Tazari

Received: date / Accepted: date

Abstract We present the first polynomial-time approximation schemes (PTASes) for the following subset-connectivity problems in edge-weighted graphs of bounded-genus: Steiner tree, low-connectivity survivable-network design, and subset TSP. The schemes run in $\mathcal{O}(n \log n)$ time for graphs embedded on both orientable and nonorientable surfaces. This work generalizes the PTAS framework from planar graphs to bounded-genus graphs: any problem that is shown to be approximable by the planar PTAS framework of Borradaile, Klein, and Mathieu (2007) will also be approximable in bounded-genus graphs by our extension.

Keywords polynomial-time approximation scheme · bounded-genus graphs · Steiner tree · subset TSP

1 Introduction

In many practical scenarios of network design, input graphs have a natural drawing on the sphere or, equivalently, the plane. In most cases, these em-

G. Borradaile
Oregon State University
Tel.: +1-541-737-7280
Fax: +1-541-737-1300
E-mail: glencora@eecs.orst.edu

E. Demaine
Massachusetts Institute of Technology
Tel.: +1-617-253-6871
Fax: +1-617-258-8682
E-mail: edemaine@mit.edu

S. Tazari
Humboldt-Universität zu Berlin
Tel.: +49-30-2093-3094
Fax: +49-30-2093-3081
E-mail: tazari@informatik.hu-berlin.de

beddings have few crossings, either to avoid digging multiple levels of tunnels for fiber or cable or to avoid building overpasses in road networks. But a few crossings are common, and can easily come in bunches where one tunnel or overpass might carry several links or roads. Thus we naturally arrive at graphs of small (bounded) genus, which is the topic of this work.

We develop a PTAS framework for subset-connectivity problems on edge-weighted graphs of bounded-genus. In general, we are given a subset of the nodes, called *terminals*, and the goal is to connect the terminals together with some substructure of the graph by using cost within $1+\varepsilon$ of the minimum possible cost, for small ε. Our framework applies to three well-studied problems in this framework. In Steiner Tree, the substructure must be connected, and thus forms a tree. In Subset Tsp, the substructure must be a cycle; the cycle may traverse vertices and edges multiple times, but pays for each traversal. In $\{0,1,2\}$-edge-connectivity Survivable Network, the substructure must have $\min\{c_x, c_y\}$ edge-disjoint paths connecting vertices x and y, where each $c_x \in \{0,1,2\}$; we allow the substructure to include multiple copies of an edge in the graph, but pay for each copy separately. In particular, if $c_x = 1$ for all terminals x, then we obtain Steiner Tree; if $c_x = 2$ for all terminals x, then we obtain the minimum-cost 2-edge-connected multi-subgraph problem.

Our framework yields the first PTAS for all of these problems in bounded-genus graphs. These PTASes are efficient, running in $\mathcal{O}(f(\varepsilon,g)\,n + n\log n) = \mathcal{O}_{\varepsilon,g}(n\log n)$ time for graphs embedded on orientable surfaces and nonorientable surfaces. (We usually omit the mention of $f(\varepsilon,g)$ by assuming ε and g are constant, but we later bound $f(\varepsilon,g)$ as singly exponential in a polynomial in $1/\varepsilon$ and g.) In contrast, the problems we consider are APX-complete (and constant-factor-approximable) for general graphs.

We build upon the recent PTAS framework of Borradaile, Klein, and Mathieu [9] for subset-connectivity problems on planar graphs. In fact, our result is strictly more general: any problem to which the previous planar-graph framework applies can automatically be approximated by our framework as well, resulting in PTASes for bounded-genus graphs. For example, Borradaile and Klein [6] have recently given a PTAS for the $\{0,1,2\}$-edge-connectivity Survivable Network problem using the planar framework. This will imply a similar result in bounded-genus graphs. At the end of the paper, we discuss progress (since the conference version of this paper appeared in 2009) on PTAS development in planar graphs and how our paper fits in with this progress. In contrast to the planar-graph framework, our PTASes have the attractive feature that they run correctly on all graphs with the performance degrading with the genus.

Our techniques for attacking bounded-genus graphs includes the recent result of decompositions into bounded-treewidth graphs via contractions [14].[1] We also use a simplified version of an algorithm for finding a short sequence of

[1] Before the publication of [14], we used [15] together with fast algorithms for finding shortest noncontractible cycles [10] but this can be avoided now.

loops on a topological surface [18], and sophisticated dynamic programming. Our aim is to prove the following theorem:

Theorem 1 *There exists a* PTAS *for the* Steiner Tree*,* Subset Tsp*, and* $\{0,1,2\}$*-edge-connected* Survivable Network *problems in edge-weighted graphs of genus* g *with running time* $\mathcal{O}(2^{\mathrm{poly}(\varepsilon^{-1},g)}n + n\log n)$.

2 Preliminaries

All graphs $G = (V, E)$ have n vertices, m edges and are undirected with edge lengths (weights). The length of an edge e is denoted by $\ell(e)$; the sum of the lengths of the edges in a subgraph H or set of subgraphs $\mathcal{H}$ are denoted $\ell(H)$ and $\ell(\mathcal{H})$, respectively. The shortest distance between vertices x and y in a graph G is denoted $\mathrm{dist}_G(x, y)$. The boundary of a graph G embedded in the plane is denoted by ∂G. For an edge $e = uv$, we define the operation of *contracting* e as identifying u and v and removing all loops and duplicate edges.

2.1 Embedded graphs

We use the basic terminology for embeddings as outlined by Mohar and Thomassen [27]. In this paper, an embedding refers to a *2-cell embedding*, i.e., a drawing of the vertices and faces of the graph as points and arcs on a surface such that every face is homeomorphic to an open disc. Such an embedding can be described combinatorially by specifying the cyclic ordering of edges around each vertex of the graph. A combinatorial embedding of a graph G naturally induces such a 2-cell embedding on each subgraph of G. We only consider compact surfaces without boundary.

If a surface contains a subset homeomorphic to a Möbius strip, it is *nonorientable*; otherwise it is *orientable*. For a 2-cell embedded graph G with f facial walks, the number $g = 2 + m - n - f$ is called the Euler genus of the surface. The Euler genus is equal to twice the usual genus for orientable surfaces and equals the usual genus for nonorientable surfaces. The *dual* of an embedded graph G is defined as having the set of faces of G as its vertex set and having an edge between two vertices if the corresponding faces of G are adjacent. We denote the dual graph by $G^\star$ and identify each edge of G with its corresponding edge in $G^\star$. A cycle of an embedded graph is *contractible* if it can be continuously deformed to a point; otherwise it is *noncontractible*.

We say that two paths P and Q cross at vertex x if there is a component X of $P \cap Q$ that contains x such that, upon contracting X, edges of P and Q alternate in the cyclic embedding around the contracted vertex. A cycle is non-crossing if no two subpaths of the cycle cross.

We *cut along a 2-sided cycle* C by partitioning the edges adjacent to C into left and right edges and replacing C with two copies C_ℓ and C_r, adjacent to the left or right edges, accordingly. The inside of these new cycles is "patched" with two new faces. If the resulting graph is disconnected, the cycle is called *separating*, otherwise *nonseparating*. Cutting along a 1-sided cycle C on nonorientable surfaces is defined similarly, only that C is replaced by one bigger cycle C' that contains every edge of C exactly twice. We can likewise cut along a tree T by taking C to be a non-crossing Euler tour of T. Cutting along a tree creates a single new face corresponding to the inside of T. An example of these concepts is illustrated in Figure 3. Mohar and Thomassen give further technical details [27, pages 105–106].

2.2 Treewidth

Next we define the notions related to treewidth as introduced by Robertson and Seymour [29]. A *tree decomposition* of a graph G is a pair (T, χ), where $T = (I, F)$ is a tree and $\chi = \{\chi_i | i \in I\}$ is a family of subsets of $V(G)$, called *bags*, such that

1. every vertex of G appears in some bag of χ;
2. for every edge $e = uv$ of G, there exists a bag that contains both u and v;
3. for every vertex v of G, the set of bags that contain v form a connected subtree T_v of T.

The *width* of a tree decomposition is one less than the maximum size of a bag in χ. The *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, is the minimum width over all possible tree decompositions of G.

2.3 Input definition

The input graph is $G_0 = (V_0, E_0)$ and has genus g_0; the terminal set is Q. We assume G_0 is equipped with a combinatorial embedding; such an embedding can be found in linear time if the genus is known to be fixed [26]. Let $\mathcal{P}$ be the considered subset-connectivity problem. In Section 4.1, we show how to find a subgraph $G = (V, E)$ of G_0, such that for $0 \le \varepsilon \le 1$, there exists a $(1 + \varepsilon)$-approximate solution of $\mathcal{P}$ in G_0 that is also in G. Hence, we may use G instead of G_0 in the rest of the paper. Note that as a subgraph of G_0, G is automatically equipped with a combinatorial embedding.

Let OPT denote the length of an optimal Steiner tree spanning terminals Q. We define $\mathrm{OPT}_{\mathcal{P}}$ to be the length of an optimal solution to problem $\mathcal{P}$. For the problems that we solve, we require that $\mathrm{OPT}_{\mathcal{P}} = \Theta(\mathrm{OPT})$ and in particular that $\mathrm{OPT} \le \mathrm{OPT}_{\mathcal{P}} \le \mu\,\mathrm{OPT}$ for a constant $\mu > 1$. The constant μ will be used in Section 4 and is equal to 2 for both the subset TSP and $\{0, 1, 2\}$-edge-connectivity problems. This requirement is also needed for the planar case [6]. Because $\mathrm{OPT}_{\mathcal{P}} \ge \mathrm{OPT}$, upper bounds in terms of OPT hold for all the problems herein. As a result, we can safely drop the $\mathcal{P}$ subscript throughout the paper.

We show how to obtain a $(1 + c\varepsilon)\,\mathrm{OPT}_{\mathcal{P}}$ solution for a fixed constant c. To obtain a $(1+\varepsilon)\,\mathrm{OPT}_{\mathcal{P}}$ solution, we can simply use $\varepsilon' = \varepsilon/c$ as input to the algorithm.

2.4 Organization of the paper

In the next section, we overview the PTAS framework as developed by Borradaile, Klein and Mathieu [9]. This framework involves constructing a grid-like subgraph called the *mortar graph*, whose faces are referred to as *bricks*. They prove a *Structure Theorem* that shows that there is a near-optimal solution to $\mathcal{P}$ that crosses the boundary of each brick only a constant number of times. Given this construction and structural property, they develop two methods for obtaining a PTAS: by way of constructing a spanner, using the mortar graph as a skeleton, followed by dynamic programming [7] and more directly by dynamic programming over the bricks [8]. These methods result in PTASes with doubly and singly exponential dependence on $\mathrm{poly}(\varepsilon^{-1})$, respectively.

In the later sections, we generalize this framework to bounded-genus graphs. In Section 4, we show how to construct a mortar graph for bounded-genus graphs. We argue that a corresponding Structure Theorem follows directly from the planar framework. In Sections 5.1 and 5.3, we show how to generalize the two methods for obtaining a PTAS in planar graphs to bounded-genus graphs. Finally, in Section 6, we prove a Structure Theorem for the Subset Tsp problem. In planar graphs, a PTAS for this problem predated (and inspired) the planar PTAS framework. However, in order to obtain a PTAS for Subset Tsp in bounded-genus graphs, we must either generalize the direct PTAS of Klein [22] or prove a Structure Theorem so that the PTAS framework applies. We opt for the latter.

3 The planar framework (in abstract form)

In this section, we review the framework introduced by Klein [23] to obtain approximation schemes in planar graphs and how it was used by Borradaile et al. [9] for a PTAS for Steiner Tree in planar graphs. However, we already generalize the concepts to a more abstract form that can be applied to the bounded-genus case as well. In Section 3.6, we review a modified version of this framework that was introduced in [9] to obtain a PTAS with better dependence on $1/\varepsilon$.

Let G be a graph with edge weights and OPT the solution value of an optimization problem. The framework proposed by Klein [23] consists of the following four steps:

1. Spanner step: Find a subgraph G_{span} of G that has weight $\mathcal{O}(\mathrm{OPT})$ and contains a $(1+\frac{\varepsilon}{2})$-approximation of the optimal solution. G_{span} is called a *spanner* for the given problem.

2. Thinning step: Find and contract a set of edges of G_{span} of weight at most $\mathcal{O}(\varepsilon\,\mathrm{OPT})$ to obtain a graph G_{thin} that has *bounded treewidth*.
3. Dynamic-programming step: Use dynamic programming on graphs of bounded treewidth to find the optimal solution to the problem in G_{thin}.
4. Lifting step: Convert this solution to a solution in G by incorporating some of the edges that were contracted in the thinning step.

For planar graphs, it turns out that the most difficult step is the spanner construction. Borradaile et al. [9] introduce the concept of a *mortar graph* as an intermediate step and together with a *Structure Theorem* show how to obtain a spanner for the STEINER TREE problem. We are going to review these concepts, as needed for our work, below.

The technique used for thinning is called *contraction decomposition* and has been recently generalized to all classes of H-minor-free graphs [14], which include, in particular, bounded-genus graph. We review this technique in Section 3.4.

3.1 Mortar graph

In [9], Borradaile, Klein and Mathieu developed a PTAS for the STEINER TREE problem in planar graphs. The method involves finding a grid-like subgraph called the *mortar graph* that spans the input terminals and has length $\mathcal{O}(\mathrm{OPT})$. The set of feasible Steiner trees is restricted to those that cross between adjacent faces of the mortar graph only at a small number (per face of the mortar graph) of pre-designated vertices called *portals*. A Structure Theorem guarantees the existence of a nearly optimal solution (one that has length at most $(1+\varepsilon)\,\mathrm{OPT}$) in this set.

Here we define the mortar graph in a more general way that also applies to higher genus graphs. A path P in a graph G is *ε-short in G* if for every pair of vertices x and y on P, the distance from x to y along P is at most $(1+\varepsilon)$ times the distance from x to y in G: $\mathrm{dist}_P(x,y) \leq (1+\varepsilon)\,\mathrm{dist}_G(x,y)$. Given a graph G embedded on a surface and a set of terminals Q, a mortar graph is a subgraph of G with the following properties:

Definition 1 (Mortar Graph and Bricks) Given a graph G embedded on a surface of genus g, a set of terminals Q, and a number $0 < \varepsilon \leq 1$, consider a subgraph $\mathrm{MG} := \mathrm{MG}(G, Q, \varepsilon)$ of G spanning Q *such that each facial walk of* MG *encloses an area homeomorphic to an open disk*. For each face F of MG, we construct a *brick* B of G by cutting G along the facial walk ∂F; B is the planar subgraph of G embedded inside the face, including ∂F. We denote this facial walk as the *mortar boundary* ∂B of B. We define the *interior* of B as B without the edges of ∂B. We call MG a *mortar graph* if for some constants $\alpha(\varepsilon, g)$ and $\kappa(\varepsilon, g)$ (to be defined later), we have $\ell(\mathrm{MG}) \leq \alpha\,\mathrm{OPT}$ and every brick B satisfies the following properties:

1. B is planar.

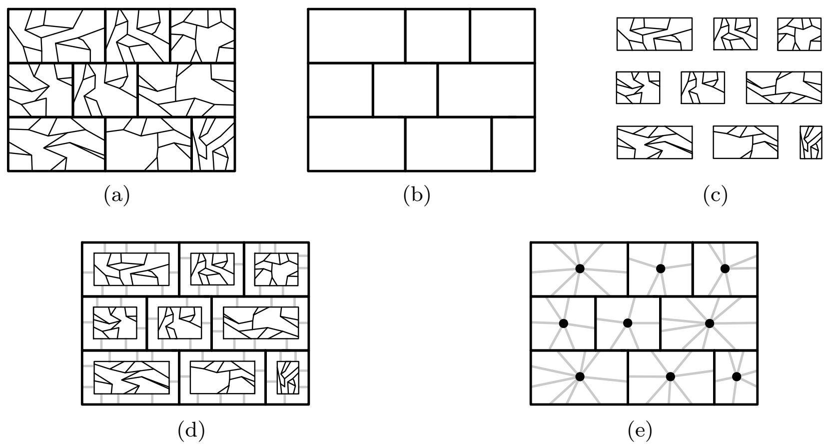


Fig. 1 (a) An input graph G with mortar graph MG given by bold edges in (b). (c) The set of bricks corresponding to MG. (d) A portal-connected graph, $\mathcal{B}^+(\mathrm{MG}, \theta)$. The portal edges are grey. (e) $\mathcal{B}^+(\mathrm{MG}, \theta)$ with the bricks contracted, resulting in $\mathcal{B}^{\div}(\mathrm{MG}, \theta)$. The dark vertices are brick vertices.

2. The boundary of B is the union of four paths in the clockwise order W, N, E, S.
3. Every terminal of Q that is in B is on N or on S.
4. N is 0-short in B, and every subpath of S is ε-short in B (unless both E and W are trivial paths, in which case every *proper* subpath of S is ε-short in B).
5. There exists a number $k \leq \kappa$ and vertices $s_0, s_1, s_2, \ldots, s_k$ ordered from left to right along S such that, for every i:

   for any vertex x of $S[s_i, s_{i+1})$, the distance from x to s_i along S is less than ε times the distance from x to N in B: $\mathrm{dist}_S(x, s_i) < \varepsilon \cdot \mathrm{dist}_B(x, N)$.

The mortar graph and the set of bricks are illustrated in Figures 1 (a), (b) and (c). We use the algorithm for finding a mortar graph in a planar graph as a black-box, and hence it is not necessary to repeat the details of the algorithm. We only present a rough sketch in order to introduce the main concepts that are relevant to this work (see Fig. 2):

1. Find a 2-approximate Steiner tree T [25] and cut the graph open along T to obtain a distinguished face H that we may think of as the outer face of the graph.
2. Find shortest paths between certain parts of H. These paths result in the N and S boundaries of the bricks.
3. Find shortest paths between certain vertices of the paths found in Step 2. These paths are called *columns*, do not cross each other, and have a natural order.

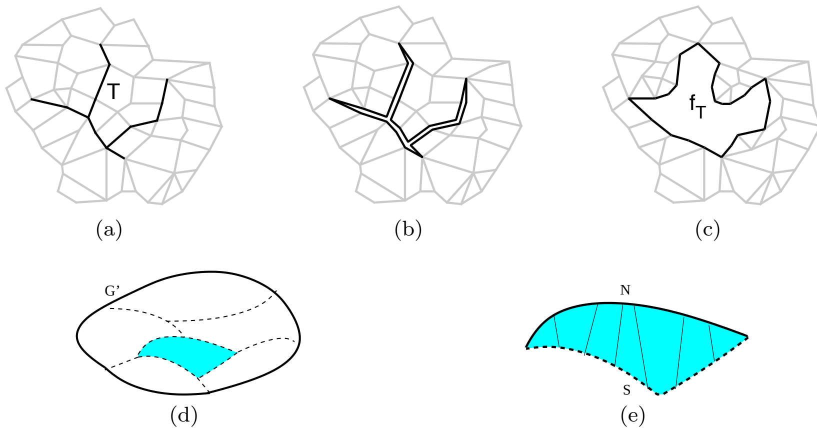


Fig. 2 Construction of a mortar graph: (a) a 2-approximate Steiner tree T (with dark edges); (b) splitting the 2-approximation along its Euler tour; (c) thinking of the newly created face as the outer face; (d) adding shortest paths that comprise the north and south boundaries of bricks with one shaded strip; (e) finding the columns inside each of the strips from S to N.

4. Take every κth path found in Step 3. These paths are called *supercolumns* and form the E and W boundaries of the bricks. We refer to κ as the *spacing* of the supercolumns.

The mortar graph is composed of the edges of T (equivalently, H) and the edges found in Steps 2 and 4. In [9], it is shown that the total length of the mortar graph edges is at most $9\varepsilon^{-1}\,\mathrm{OPT}$.

One crucial observation is that the mortar graph can be built based on any face H of a planar graph, as long as H contains all terminals and its weight is bounded in terms of OPT. Therefore, we state the mortar graph construction theorem of Borradaile et al. [9] in this more abstract form, based on the face H instead of OPT:

Theorem 2 ([9]) *Let $0 < \varepsilon \leq 1$ and G be a planar graph with outer face H containing the terminals Q and such that $\ell(H) \leq \alpha_0\,\mathrm{OPT}$, for some constant α_0. For $\alpha = (2\alpha_0 + 1)\varepsilon^{-1}$, there is a mortar graph $\mathrm{MG}(G, Q, \varepsilon)$ containing H whose length is at most $\alpha\,\mathrm{OPT}$ and whose supercolumns have length at most $\varepsilon\,\mathrm{OPT}$ with spacing $\kappa = \alpha_0\varepsilon^{-2}(1 + \varepsilon^{-1})$. The mortar graph can be found in $\mathcal{O}(n \log n)$ time.*

3.2 Structure Theorem

Along with the mortar graph, Borradaile et al. [9] define an operation $\mathcal{B}^+$ called *brick-copy* that allows a succinct statement of the Structure Theorem. For each brick B, a subset of θ vertices are selected as *portals* such that the distance along ∂B between any vertex and the closest portal is at most $\ell(\partial B)/\theta$. For every brick B, embed B in the corresponding face of MG and connect every portal of B to the corresponding vertex of MG with a zero-length *portal edge*: this defines $\mathcal{B}^+(\mathrm{MG}, \theta)$. $\mathcal{B}^+(\mathrm{MG}, \theta)$ is illustrated in Figure 1 (d). We denote the set of all portal edges by E_{portal}. The following simple observation also holds for bounded-genus graphs:

Observation 1 ([9]) *If A is a connected subgraph of $\mathcal{B}^+(\mathrm{MG}, \theta)$, then $A - E_{portal}$ is a connected subgraph of G spanning the same vertices of G.*

The Structure Theorem is the heart of the correctness of the PTASes. It essentially says that there is a constant θ depending polynomially on ε^{-1} such that in finding a near-optimal solution in G, we can restrict our attention to $\mathcal{B}^+(\mathrm{MG}, \theta)$. We will state the theorem in its exact form for bounded-genus graphs in Section 4.4.

3.3 Obtaining a spanner

Once a mortar graph MG is constructed, a spanner can be obtained as follows: for each brick B defined by MG and for each subset X of the portals of B, find the optimal Steiner tree for X in B (using the method of Erickson et al. [19]); the spanner is the union of all these trees over all bricks plus the edges of the mortar graph. The weight of the spanner is bounded by $2^{\mathrm{poly}(\varepsilon^{-1})}$ OPT and it can be constructed in $\mathcal{O}(n \log n)$ time. The fact that the spanner contains a near optimal solution follows from the Structure Theorem. See Section 5.1 for the exact details and proof for all considered subset-connectivity problems in bounded-genus graphs.

3.4 Contraction decomposition

Contraction decomposition is an algorithmic technique first introduced for planar graphs [23], then generalized to bounded-genus graphs [15], and recently to all H-minor-free graphs [14]. It essentially says that for a given H-minor-free graph G and parameter η, we can partition the edges of G into η parts such that contracting any part results in a graph of bounded treewidth.

The way to prove this theorem for planar graphs is similar to Baker's shifting technique for approximation schemes [3] but applied in the dual: the algorithm is reduces to a breadth-first search in the dual and numbering the layers periodically modulo k (however, the details are technically more involved). For bounded-genus graphs, an $\mathcal{O}(n \log n)$ algorithm was given in [15] but the result on H-minor-free graphs [14] contains an improved algorithm for bounded genus graphs that is essentially a breadth-first search in the radial graph and runs in linear time:

Theorem 3 (Contraction Decomposition [14]) *For a fixed genus g, and any integer $\eta \geq 2$ and for every graph G of Euler genus at most g, the edges* *of G can be partitioned into η sets such that contracting any one of the sets results in a graph of treewidth at most $\mathcal{O}(g \cdot \eta)$. Furthermore, such a partition can be found in linear time.*

3.5 PTAS via spanner

Together with the spanner obtained above, Klein's framework [23] – that was introduced in the beginning of this section – implies an $\mathcal{O}(n \log n)$ PTAS for Steiner Tree with doubly exponential dependence on ε^{-1} in planar graphs: by choosing $\eta = \mathcal{O}(\frac{\ell(G\text{span})}{\varepsilon\,\mathrm{OPT}})$, we obtain a set of edges of weight $\mathcal{O}(\varepsilon\,\mathrm{OPT})$ to perform the thinning step, and then the dynamic programming and lifting steps can easily follow.

3.6 PTAS via dynamic programming over bricks

In [9], Borradaile et al. present a PTAS that is singly exponential in a polynomial in ε^{-1} for Steiner Tree in planar graphs. The idea is to incorporate the spanner step into the dynamic-programming (DP) step and to use a somewhat modified thinning step. To this end, the operator *brick-contraction* $\mathcal{B}^{\div}$ is defined to be the application of the operation $\mathcal{B}^{+}$ followed by contracting each brick to become a single vertex of degree at most θ (see Figure 1(e)).

The method we present for bounded-genus graphs in this work is different and more generic than that of Borradaile et al. [9] and is given in Section 5.3. However, for the sake of completeness and to make the differences clear, we briefly review the earlier method below.

The thinning algorithm is applied to the mortar graph (as opposed to a spanner) and since the weight of the mortar graph is polynomial in ε^{-1}, we are able to save an exponent at this stage. However, in [9] the thinning is not done via a black-box contraction decomposition algorithm but rather by decomposing the mortar graph into parts of *bounded dual radius*, called *parcels*, such that the weight of the total boundary of all parcels is bounded by $\mathcal{O}(\varepsilon\,\mathrm{OPT})$. The operation $\mathcal{B}^{\div}$ is applied to each parcel individually and since the number of portals is constant per brick, the bounded dual radius property is maintained.

Afterwards, a *carving decomposition* of each parcel is found in such a way that the contracted bricks appear only at the leaves of the decomposition tree. A carving decomposition is a concept similar to a tree decomposition that allows for efficient DP algorithms for hard problems when the width is small – which is the case since the parcels have small dual radius. At those leaves of the decomposition tree which are contracted bricks, we compute a DP table containing optimal Steiner trees inside the brick for all possible noncrossing partitions of the portals of the brick using the algorithm of [19]. Then we combine these solutions with the rest of the parcel as we work our way up the carving decomposition tree – similar to standard DP algorithms on bounded treewidth graphs.

Finally, we are able to combine the solutions of all parcels into a global solution for the whole graph by incorporating the parcel boundaries since they have negligible weight (to be precise, a number of auxiliary terminals have to be added to the parcels to make this work but the details are not important in this work).

4 Mortar graph and Structure Theorem for bounded-genus graphs

We use Theorem 2 to prove the existence of a mortar graph for genus-g embedded graphs. This section is devoted to proving the following theorem:

Theorem 4 *Let an embedded edge-weighted graph G of Euler genus g, a subset of its vertices Q, an $0 < \varepsilon \leq 1$, and $\mu \geq 1$ be given. For $\alpha = (32\mu g + 9)\varepsilon^{-1}$, there is a mortar graph $\mathrm{MG}(G, Q, \varepsilon)$ of G such that the length of MG is $\leq \alpha\,\mathrm{OPT}$ and the supercolumns of MG have length $\leq \varepsilon\,\mathrm{OPT}$ with spacing $\kappa = (16\mu g + 4)\varepsilon^{-2}(1 + \varepsilon^{-1})$. The mortar graph can be found in $\mathcal{O}(n \log n)$ time.*

Let $G_0 = (V_0, E_0)$ be the input graph of genus g_0 and Q be the terminal set. In a first preprocessing step, we delete a number of unnecessary vertices and edges of G_0 to obtain a graph $G = (V, E)$ of genus $g \leq g_0$ that still contains every $(1 + \varepsilon)$-approximate solution for terminal set Q for all $0 \leq \varepsilon \leq 1$ while fulfilling certain bounds on the lengths of shortest paths. In the next step, we find a *cut graph* CG of G that contains all terminals and whose length is bounded by a constant times OPT. We cut G open along CG, so that it becomes a planar graph with a simple cycle σ as boundary, where the length of σ is twice that of CG. See Figure 3 for an illustration. Afterwards, the remaining steps of building the mortar graph can be the same as in the planar case, by way of Theorem 2.

For an edge $e = vw$ in G_0, we let

$$\mathrm{dist}_{G_0}(r, e) = \min\{\mathrm{dist}_{G_0}(r, v), \mathrm{dist}_{G_0}(r, w)\} + \ell(e)$$

and say that e is *at distance* $\mathrm{dist}_{G_0}(r, e)$ from r.

4.1 Preprocessing the input graph

Our first step is to apply the following preprocessing procedure:

Algorithm Preprocess(G_0, Q, μ).
Input. an arbitrary graph G_0, terminals $Q \subseteq V(G_0)$, a constant $\mu \geq 1$
Output. a preprocessed subgraph of G_0

1. Find a 2-approximate Steiner tree T_0 for Q and contract it to a vertex r.
2. Find a shortest-path tree rooted at r.
3. Delete all vertices v and edges e of G_0 with $\mathrm{dist}_{G_0}(r, v), \mathrm{dist}_{G_0}(r, e) > 2\mu\ell(T_0)$.

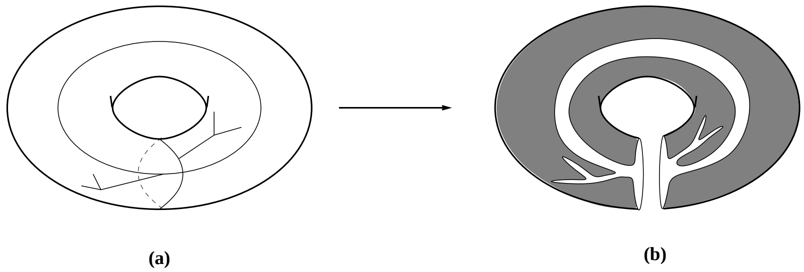


Fig. 3 (a) a cut graph of a tree drawn on a torus; (b) the result of cutting the surface open along the cut graph: the shaded area is homeomorphic to a disc and the white area is the additional face of the planarized surface.

Any deleted vertex or edge is at distance $> 2\mu\ell(T_0) \geq 2\mu\,\mathrm{OPT}$ from any terminal and hence can not be part of a $(1+\varepsilon)$-approximation for any $0 \leq \varepsilon \leq 1$. We call the resulting graph $G = (V, E)$ and henceforth use G instead of G_0 in our algorithm. The preprocessing step can be accomplished in linear time: step 1 using Müller-Hannemann and Tazari's algorithm [30] and step 2 using Henzinger et al.'s algorithm [21]. Trivially, we have

Proposition 1 *All vertices and edges of G are at distance at most $4\mu\,\mathrm{OPT}$ from T_0.*

4.2 Constructing a cut graph

A central fact that we use in this section and also in other parts of our work is the following observation [17]:

Observation 2 *Let G be a planar graph and T a spanning tree of G. Then the set of edges $E(G) - E(T)$ induces a spanning tree $T^\star$ in the dual $G^\star$. If T is a minimum spanning tree of G, then $T^\star$ is a maximum spanning tree of $G^\star$.*

A similar lemma also holds for bounded-genus graphs: if T is a (minimum) spanning tree of G and $T^\star$ a (maximum) spanning tree of $G^\star - E(T)$, then $T^\star$ is a (maximum) spanning tree of $G^\star$ and the size of the set of remaining edges $X := E(G) - E(T) - E(T^\star)$ is g, the Euler genus of G, by Euler's formula. Eppstein [16] defines such a triple $(T, T^\star, X)$ as a *tree-cotree decomposition* of G and shows that such a decomposition can be found in linear time for graphs on both orientable and nonorientable surfaces. Further he shows that a tree-cotree decomposition generates a cut graph. For an r-rooted spanning tree T and a non-tree edge e, we say that $\mathrm{loop}(T, e)$ is the *elementary cycle* formed by e and the paths from r to e's endpoints; Eppstein showed:

Lemma 1 (Lemma 2, [16]) *Given a tree-cotree decomposition $(T, T^\star, X)$, $\{loop(T, e) : e \in X\}$ is a cut graph.*

In order to construct a cut graph, we start again with a 2-approximation T_0 and contract it to a vertex r. Next, we look for a *system of loops* rooted at r: iteratively find short nonseparating cycles through r and cut the graph open along each cycle. Erickson and Whittlesey [18] showed that, for orientable surfaces, taking the *shortest* applicable cycle at each step results in the shortest system of loops through r. They suggest a linear-time algorithm using the tree-cotree decomposition $(T, T^\star, X)$ of Eppstein [16]. As we only need to bound (as opposed to minimize) the length of our cut graph, we use the following, simpler algorithm:

Algorithm $\textsc{Planarize}(G_0, Q, \mu)$**.**

Input. a graph G_0 of genus g, terminals $Q \subseteq V(G_0)$, a constant $\mu \geq 1$

Output. a preprocessed subgraph $G \subseteq G_0$ and a cut graph CG of G

1. Apply $\textsc{Preprocess}(G_0, Q, \mu)$ and let G be the obtained subgraph.
2. Find a 2-approximate Steiner tree T_0 for Q and contract it to a vertex r.
3. Find a shortest-paths tree SPT rooted at r.
4. Uncontract r and set $T_1 = T_0 \cup \mathrm{SPT}$. *($T_1$ is a spanning tree of G)*
5. Find a spanning tree $T_1^\star$ in $G^\star - E(T_1)$. *($T_1^\star$ is a spanning tree of $G^\star$)*
6. Let $X := E(G) - E(T) - E(T^\star)$.
7. Return $\mathrm{CG} := T_0 \cup \{\mathrm{loop}(T_1, e) : e \in X\}$ together with G.

Lemma 2 *The algorithm* $\textsc{Planarize}$ *returns a cut graph* CG *such that cutting G open along* CG *results in a planar graph G_p with a face f_σ whose facial walk σ*

(i) is a simple cycle;
(ii) contains all terminals (some terminals might appear more than once as multiple copies might be created during the cutting process); and
(iii) has length $\ell(\sigma) \leq 2(8\mu g + 2)\,\mathrm{OPT}$.

The algorithm can be implemented in linear time.

Proof Clearly, $(T_1, T_1^\star, X)$ is tree-cotree decomposition of G and so, by Lemma 1, CG is a cut graph. By Euler's formula, we get that $|X| = g$, the Euler genus of G.

Each edge $e = vw \in X$ completes a (nonseparating, not necessarily simple) closed walk as follows: a shortest path P_1 from T_0 to v, the edge e, a shortest path P_2 from w to T_0 and possibly a path P_3 in T_0. By Proposition 1, we know that e is at distance at most $4\mu\,\mathrm{OPT}$ from T_0 and so, both P_1 and P_2, and at least one of $\{P_1 \cup \{e\}, P_2 \cup \{e\}\}$ have length at most $4\mu\,\mathrm{OPT}$. Hence, we have that $\ell(P_1 \cup \{e\} \cup P_2) \leq 8\mu\,\mathrm{OPT}$. Because there are (exactly) g such cycles in CG, we get that

$$\ell(\mathrm{CG}) \leq g \cdot 8\mu\,\mathrm{OPT} + \ell(T_0) \leq (8\mu g + 2)\,\mathrm{OPT}\,.$$

Since CG is a connected cut graph and $T^\star \cap \mathrm{CG} = \varnothing$, cutting G open along CG results in a connected planar graph with boundary σ. Each edge of CG appears twice in σ and each edge of σ is derived from CG, so $\ell(\sigma) = 2\ell(\mathrm{CG})$ (see Fig. 3).

As mentioned in the previous section, T_0 and SPT can be computed in linear time on bounded-genus graphs [21,30]. $T_1^\star$ can be obtained, for example, by a simple breadth-first-search in the dual. The remaining steps can also easily be implemented in linear time. □

4.3 Proof of Theorem 4

We complete the construction of a mortar graph for genus-g embedded graphs.

Let G_p be the result of planarizing G as guaranteed by Lemma 2. G_p is a planar graph with boundary σ such that σ spans Q and has length $\leq 2(8\mu g + 2)\,\mathrm{OPT}$. Let MG be the mortar graph guaranteed by Theorem 2 as applied to G with σ as its outer face. Every edge of MG corresponds to an edge of G. Let MG′ be the subgraph of G composed of edges corresponding to MG. Every face f of MG (other than σ) corresponds to a face f' of MG′ and the interior of f' is homeomorphic to a disk on the surface in which G is embedded. It is easy to verify that MG′ is indeed a mortar graph of G; and the length bounds specified in the statement of the theorem follow directly from Theorem 2 and the bound on the length of σ.

4.4 Structure Theorem

Recall the operation $\mathcal{B}^+$ called brick-copy introduced in Section 3.2 and illustrated in Figure 1 (d). The precise statement of the Structure Theorem for bounded-genus graphs is as follows:

Theorem 5 (Structure Theorem) *Let $\mathcal{P}$ be one of the subset-connectivity problems* Steiner Tree*, $\{0,1,2\}$-edge-connectivity* Survivable Network*, or* Subset Tsp*. Let G be an edge-weighted graph embedded on a surface, $Q \subseteq V(G)$ a given set of terminals, and $0 < \varepsilon \leq 1$. Let $\mathrm{MG}(G,Q,\varepsilon)$ be a corresponding mortar graph of weight at most $\alpha\,\mathrm{OPT}$ and supercolumns of weight at most $\varepsilon\,\mathrm{OPT}$ with spacing κ. There exist constants $\beta(\varepsilon,\kappa)$ and $\theta(\alpha,\beta)$ depending polynomially on α and β such that*

$$\mathrm{OPT}_{\mathcal{P}}(\mathcal{B}^+(\mathrm{MG},\theta),Q) \leq (1+c\varepsilon)\,\mathrm{OPT}_{\mathcal{P}}(G,Q)\,,$$

where c is an absolute constant. Here $\beta = o(\varepsilon^{-2.5}\kappa)$ for Steiner Tree *and $\{0,1,2\}$-edge connectivity* Survivable Network *and $\beta = \mathcal{O}(\kappa)$ for* Subset Tsp*. (Recall that α and κ depend polynomially on ε^{-1} and g by Theorem 4.)*

It is due to our special way of defining and constructing a mortar graph for bounded-genus graphs that this theorem follows immediately as for the planar cases: the crucial point here is that our bricks are always planar – even when the given graph is embedded in a surface of higher genus. The Structure Theorem for Steiner Tree is proved in [9], the case of $\{0,1,2\}$-edge-connectivity Survivable Network is studied in [6], and we show that the theorem holds for Subset Tsp in Section 6. Note that for Subset Tsp, it is possible to obtain a singly exponential algorithm by following the spanner construction of Klein [22] after performing the planarizing step (Lemma 2). Our presentation here is chosen to unify the methods for all problems studied. Whenever we wish to apply our framework to a new problem, it is essential to prove a similar Structure Theorem for the considered problem.

5 Obtaining PTASes for bounded-genus graphs

We present two methods of obtaining polynomial-time approximation schemes. The first is a generalization of the framework of Klein [22] for planar graphs as introduced in Section 3. To this end, we show how to use the mortar graph for bounded-genus graphs to obtain a spanner in Section 5.1, and then generalize Klein's framework to higher genus graphs in Section 5.2.

In the second method, dynamic programming is done over the bricks of the mortar graph similar to the technique of Borradaile et al. [9] as introduced in Section 3.6. However, our technique that we describe in Section 5.3 is more general. The original technique required delving into the combinatorial structure of the mortar graph and used the somewhat limited concept of a carving decomposition in order to guide the dynamic program. We instead use the tree decomposition resulting from a black-box contraction decomposition algorithm to guide the dynamic program.

While both methods result in $\mathcal{O}(n \log n)$-time algorithms, the running time of the first method is doubly exponential in a polynomial in g and ε^{-1} while the second is singly exponential. The advantage of the first method is that it is simpler and more generic.

5.1 Spanner for subset-connectivity problems

A spanner is a subgraph of length $\mathcal{O}_{\varepsilon,g}(\mathrm{OPT})$ that contains a $(1+\varepsilon)$-approximate solution. Here we show how to find a spanner for bounded-genus graphs and the subset-connectivity problems considered in this paper. After a mortar graph is computed, the construction is, in fact, exactly the same as in the planar cases, namely:

> For each brick B defined by MG and for each subset X of the portals of B, find the optimal Steiner tree of X in B (using the method of Erickson et al. [19]). The spanner G_{span} is the union of all these trees over all bricks plus the edges of the mortar graph.

To prove the correctness of our spanner theorem for the case of $\{0,1,2\}$-edge-connectivity Survivable Network, we need to appeal to the following result of Borradaile and Klein, whose statement we have simplified here:

Theorem 6 ([6, Theorem 5]) *Consider an instance of the* $\{0,1,2\}$*-edge connectivity* Survivable Network *problem. There is a feasible solution* S *to this instance that is a subgraph of* $\mathcal{B}^+(MG)$ *such that*

– $\ell(S) \leq (1 + c\varepsilon)\,\mathrm{OPT}$ *where c is an absolute constant, and*
– *the intersection of S with any brick B is a set of O(1) trees, the set of leaves of which are portals.*

Theorem 7 (Spanner Theorem) *Let G be an edge-weighted graph embedded on a surface of Euler genus g and $Q \subseteq V(G)$ a given set of terminals. There exists a spanner $G_{span} \subseteq G$ such that*

G_{span} *is spanning:* G_{span} *contains a* $(1+c\varepsilon)$*-approximate solution to* Steiner Tree*,* $\{0, 1, 2\}$*-edge-connected* Survivable Network*, and* Subset Tsp*; and*
G_{span} *is short:* $\ell(G_{span}) \leq f(\varepsilon, g)\,\mathrm{OPT}$*;*

where the function $f(\varepsilon, g)$ is singly exponential in a polynomial in ε^{-1} and g, and c is an absolute constant. The spanner can be found in $\mathcal{O}(n \log n)$ time.

Proof Given a mortar graph $\mathrm{MG}(G, Q, \varepsilon)$ as guaranteed by Theorem 4, a spanner is constructed as specified above. As in [9], the time to find G_{span} is $\mathcal{O}(n \log n)$. It was proved in [9] that $\ell(G_{\mathrm{span}}) \leq (1 + 2^{\theta+1})\ell(\mathrm{MG})$. Therefore, $\ell(G_{\mathrm{span}}) \leq (1 + 2^{\theta+1})\alpha\,\mathrm{OPT}$ and $f(\varepsilon, g) = (1 + 2^{\theta+1})\alpha$ (recall that α and θ depend polynomially on ε^{-1} and g).

Now we show that G_{span} contains a near-optimal solution to each problem. For Steiner Tree, the proof follows directly from the Structure Theorem: the intersection of a minimal solution in $\mathcal{B}^+(\mathrm{MG}, \theta)$ with a brick B is a forest whose leaves are portals.

For $\{0, 1, 2\}$-edge-connected Survivable Network, we appeal to Theorem 6: By the Structure Theorem, there is a solution H in $\mathcal{B}^+(\mathrm{MG})$ that has length at most $(1 + c\varepsilon)\,\mathrm{OPT}$. For each brick B, let H_B be the intersection of H with B. H_B is the union of trees. Replace each tree with the Steiner tree spanning the same subset as found in the spanner construction. Let H' be the graph resulting from all such replacements: $\ell(H') \leq \ell(H) \leq (1 + c\varepsilon)\,\mathrm{OPT}$. By Observation 1, the edges of $H' - E_{\mathrm{portal}}$ induce a solution to the problem of length at most $(1 + c\varepsilon)\,\mathrm{OPT}$.

For Subset Tsp, the proof is similar. By the Structure Theorem, there is a tour T of the terminals Q in $\mathcal{B}^+(\mathrm{MG})$ that has length at most $(1+c\varepsilon)\,\mathrm{OPT}$. For each brick B, let K be a connected component of the intersection of T with B. Because the terminals are in MG and not in B, K is a path between portals of B: replace K with the Steiner tree (i.e., a shortest path) connecting these two portals found in the construction of the spanner (in fact, for Subset Tsp, it is sufficient to add only shortest paths between pairs of portals). Let T' be the tour resulting from all these replacements: $\ell(T') \leq \ell(T) \leq (1 + c\varepsilon)\,\mathrm{OPT}$. Appealing to Observation 1, the edges of $T' - E_{\mathrm{portal}}$ induce a solution of length at most $(1 + c\varepsilon)\,\mathrm{OPT}$. □

Let us remark that for Subset Tsp we can obtain a polynomial-sized spanner using the method of Klein [22] after applying Algorithm Planarize of Section 4.2. For (spanning) Tsp, the spanner can even be constructed in *linear* time using the algorithm of [23] after our planarizing step.[2]

[2] We would like to thank Christian Sommer for a discussion on this matter.

5.2 PTAS via spanner

We review the four steps of Klein's framework [23] (see Section 3) in our setting:

1. Spanner step: Find a spanner G_{span} of G according to Theorem 7.
2. Thinning step: Apply the Contraction Decomposition Theorem 3, with $\eta = f(\varepsilon, g)/\varepsilon$ (where $f(\varepsilon, g)$ is the function given in Theorem 7). Let $S_1, \ldots, S_\eta$ be the partition of the edges of G_{span} as guaranteed by the theorem. Let S^* be the set in the partition with minimum weight: $\ell(S^*) \leq \varepsilon\,\mathrm{OPT}$. Let G_{thin} be the graph obtained from G_{span} by contracting the edges of S^*. By Theorem 3, G_{thin} has treewidth at most $\mathcal{O}(g\varepsilon^{-1}f(\varepsilon, g))$.
3. Dynamic-programming step: Use dynamic programming (see, e.g. [24]) to find the optimal solution to the problem in G_{thin}.
4. Lifting step: Convert this solution to a solution in G by incorporating some of the edges of S^*. For Steiner Tree, at most one copy of each edge of S^* is introduced to maintain connectivity [9]. In the case of $\{0, 1, 2\}$-edge connected Survivable Network, at most two copies of each edge of S^* are required [6]. For Tsp and Subset Tsp, the method was explained in [23, 22].

Analysis of the running time. By Theorem 7, the time needed for the spanner construction is $\mathcal{O}(2^{\text{poly}(\varepsilon^{-1}, g)}n + n\log n)$. By Theorem 3, thinning takes linear time using [14]. Dynamic programming takes time $\mathcal{O}(2^{\text{poly}(\text{tw})}n)$ [11, 24]; since the treewidth of G_{thin} is $\mathcal{O}(g\varepsilon^{-1}f(\varepsilon, g))$ and $f(\varepsilon, g)$ is singly exponential in polynomials in g and ε^{-1}, this step is doubly exponential in polynomials in g and ε^{-1}. Lifting takes linear time. Hence, the overall running time is $\mathcal{O}(2^{2^{\text{poly}(\varepsilon^{-1}, g)}}n + n\log n)$. For the full Tsp the dependence on n is linear as the spanner construction takes linear time after the planarizing step [23].

5.3 PTAS via dynamic programming over the bricks

In this section, we develop a PTAS with singly exponential dependence on a polynomial in ε^{-1} and g for all subset-connectivity problems considered in this work. Even though the basic idea of dynamic programming over the bricks is similar to the planar case (cf. Section 3.6), our method is different and more general as we apply the contraction decomposition as a black-box and work with the resulting tree decomposition directly instead of using a carving decomposition. Note that this approach also facilitates generalizations to higher graph classes, like H-minor-free graphs, where the contraction decomposition is an extremely complicated algorithm [14].

The operation brick-contraction, $\mathcal{B}^{\div}$, is defined as first applying brick-copy, $\mathcal{B}^{+}$, and then contracting all the bricks (see Fig. 1(e)). We apply the contraction decomposition algorithm (Theorem 3 [14]) to $\mathcal{B}^{\div}(\mathrm{MG})$ and contract a set of edges $S^{\star}$ in $\mathcal{B}^{\div}(\mathrm{MG})$. However, we apply a special weight to portal edges so as to prevent them from being included in $S^\star$. Also, in $\mathcal{B}^\div(\mathrm{MG})$, we slightly modify the definition of contraction: after contracting an edge, we do not delete parallel portal edges. Because portal edges connect the mortar graph to the bricks, they are not parallel in the graph in which we find a solution via dynamic programming. The details are given below.

Algorithm THINNING(G, MG)**.**
Input. a graph G of fixed genus g, a mortar graph MG of G
Output. a set $S^\star \subseteq E(\mathcal{B}^\div(\mathrm{MG}))$,
a tree decomposition (T, χ) of $\mathcal{B}^\div(MG)/S^\star$

1. For each face F of MG, assign weight $\ell(\partial F)$ to each portal edge of $\mathcal{B}^\div(\mathrm{MG})$ enclosed in F;
2. Apply the Contraction Decomposition Theorem 3 to $\mathcal{B}^\div(\mathrm{MG})$ with $\eta := 3\theta\alpha\varepsilon^{-1}$ to obtain edge sets $S_1, \ldots, S_\eta$; let $S^\star$ be the set of minimum weight.
3. If $S^\star$ includes a portal edge e of a brick B enclosed in a face F of MG, add ∂F to $S^\star$ and mark B as ignored.
4. Let $\mathrm{MG}_{\mathrm{thin}} := \mathcal{B}^\div(\mathrm{MG})/S^\star$ (but do not delete parallel portal edges).
5. Let (T, χ) be a tree decomposition of width $\mathcal{O}(g \cdot \eta)$ of $\mathrm{MG}_{\mathrm{thin}}$.
6. For each vertex b of $\mathrm{MG}_{\mathrm{thin}}$ that represents an unignored contracted brick with portals $\{p_1, \ldots, p_\theta\}$:
   6.1. Replace every occurrence of b in χ with $\{p_1, \ldots, p_\theta\}$;
   6.2. Add a bag $\{b, p_1, \ldots, p_\theta\}$ to χ
   and connect it to a bag containing $\{p_1, \ldots, p_\theta\}$.
7. Reset the weight of the portal edges back to zero.
8. Return (T, χ) and $S^\star$.

Lemma 3 *The algorithm* THINNING(G, MG) *returns a set of edges $S^\star$ and a tree decomposition (T, χ) of $\mathcal{B}^\div(\mathrm{MG})/S^\star$, so that*

(i) the treewidth of (T, χ) is at most ξ where $\xi(\varepsilon, g) = \mathcal{O}(g\eta\theta) = \mathcal{O}(g^2\varepsilon^{-2}\theta^2)$; in particular, ξ is polynomial in ε^{-1} and g;
(ii) every brick is either
- *marked as ignored, or*
- *none of its portal edges are in $S^\star$; and*

(iii) $\ell(S^\star) \leq \varepsilon\,\mathrm{OPT}$.

Proof We first verify that (T, χ) is indeed a tree decomposition. For a vertex v and a tree decomposition (T', χ'), let T'_v denote the subtree of T' that contains v in all of its bags. Let us denote the tree decomposition of step (5) by (T^0, χ^0). For each brick vertex b and each of its portals p_i, we know that T^0_b is connected and $T^0_{p_i}$ is connected and that these two subtrees intersect; it follows that after the replacement in step (6.2), we have that $T_{p_i} = T^0_b \cup T^0_{p_i}$ is a connected subtree of T and hence, (T, χ) is a valid tree decomposition. Note that Theorem 3 guarantees a tree decomposition of width $\mathcal{O}(g\eta)$ if any of $S_1, \ldots, S_\eta$ are contracted; and in step (3), we only augment the set of edges to be contracted. Hence, the treewidth of (T^0, χ^0) is indeed $\mathcal{O}(g\eta)$ and with the construction in step (6.1), the size of each bag will be multiplied by a factor of at most θ. This shows the correctness of claim (i). The correctness of claim (ii) is immediate from the construction in step (3). It remains to verify claim (iii).

Let L denote the weight of $\mathcal{B}^{\div}(\mathrm{MG})$ after setting the weights of the portal edges according to step (1) of the algorithm. We have that

$$L \leq \ell(\mathrm{MG}) + \sum_F \ell(\partial F)\theta \leq \alpha\,\mathrm{OPT} + \theta \sum_F \ell(\partial F)$$
$$\leq \alpha\,\mathrm{OPT} + \theta \cdot 2\alpha\,\mathrm{OPT} \leq 3\theta\alpha\,\mathrm{OPT}\ .$$

Hence, the weight of $S^\star$, as selected in step (2), is at most $L/\eta \leq \frac{3\theta\alpha\,\mathrm{OPT}}{3\theta\alpha\varepsilon^{-1}} \leq \varepsilon\,\mathrm{OPT}$. The operation in step (3) does not add to the weight of $S^\star$: if ∂F is added to $S^\star$, the additional weight is subtracted when the corresponding portal-edge weights are set to zero in step (7). ☐

If a brick is "ignored" by Thinning, the boundary of its enclosing mortar graph face is completely added to $S^\star$. Because $S^\star$ can be added to the final solution, every potential connection through that brick can be rerouted through $S^\star$ around the boundary of the brick as explained in the lifting step in Section 5.2. The interior of the brick is not needed.

What we have now is a tree decomposition of $\mathcal{B}^{\div}(MG)/S^\star$ where some of the leaves of the decomposition contain a contracted brick with all its portal edges. We can perform standard dynamic-programming on this tree decomposition as described in [1,24,11] only with some extra processing in these special leaves. We briefly review the main idea of the DP and describe what has to be done in the leaves. Recall that we can always ensure that the tree is binary and root it at an arbitrary non-brick node that contains a terminal. For each node t of the tree decomposition let G_t denote the subgraph of $\mathcal{B}^{+}(MG)/S^\star$ that is induced by all the nodes and bricks that appear in the subtree rooted at t (note that here, we consider $\mathcal{B}^{+}(MG)$ and not $\mathcal{B}^{\div}(MG)$, i.e. the bricks are not contracted). We maintain a DP table that contains the following information at t:

– Subset Tsp: for each noncrossing *pairing* of the nodes in the bag t, we calculate a set of paths in G_t, one path for each pair, such that the collection of these paths covers all terminals in G_t.
– Steiner Tree: for each noncrossing *partition* of the nodes in the bag t, we calculate a set of Steiner trees in G_t, one tree for each part of the partition, such that the collection of these trees covers all terminals in G_t.
– $\{0,1,2\}$-edge connectivity Survivable Network: similar to Steiner Tree, the details are given in [6].

At those leaves of the decomposition which are not bricks, we can simply compute this information by brute force. In the leaves that contain bricks, we can compute optimal Steiner trees inside the brick for each subset of portals in polynomial time using the method of Erickson et al. [19].[3] In the case of Subset Tsp, we can use shortest paths as the interior of bricks does not contain terminals; and for Survivable Network, the method is described in [6]. The solution for each partition or pairing of the DP table is then simply the union of the solutions of the subsets that comprise them.

At an the interior node t of the tree decomposition, we find the solution for a given entry of the DP table by considering all combinations of entries of the children's tables with all choices of including the vertices introduced in t, and take the minimum consistent solution that can be obtained. See [1,24,11] for more details and [31] for an implementation-level exposition.

Analysis of the running time. As was shown for the planar Steiner Tree PTAS [9], the total time spent in the leaves of the dynamic program is $\mathcal{O}(4^{\theta} n)$. The rest of the dynamic program takes time $\mathcal{O}(2^{\text{poly}(\xi)} n)$ [24,11]. As contraction decomposition takes only linear time [14], the running time is dominated by the mortar graph construction which is $\mathcal{O}(n \log n)$ by Theorem 4. Hence, the total time is $\mathcal{O}(2^{\text{poly}(\xi)} n + n \log n)$ for the general case; in particular, this is singly exponential in ε^{-1} and g, as desired. This proves Theorem 1.

6 A Structure Theorem for Subset Tsp

Here we prove the Structure Theorem for Subset Tsp. While this theorem can be used to obtain a PTAS for the subset tour problem in planar graphs, a PTAS for this problem [22] predates the mortar graph framework.

Like Steiner Tree and Survivable Network, the Structure Theorem (Section 4.4) must be proved (Section 6.1) for the Subset Tsp problem. To this end, in this section, we state and prove a local structure theorem (Theorem 8, Figure 4). This local structure theorem describes how to replace the intersection of a tour with a brick to reduce the number of times the tour connects to the boundary of each brick. We only count a vertex as a connection if it is the endpoint of an edge that is not on the boundary of the brick. We call such a vertex a *joining vertex*. Only these connections contribute to the size of the dynamic-programming table. While the intersection of a tour with a brick is quite simple (a set of brick boundary-to-boundary paths), in modifying the tour we must be careful to maintain that our solution is still a tour.

We will use the following lemma due to Arora:

Lemma 4 (Patching Lemma [2]) *Let S be any line segment of length s and let π be a closed path that crosses S at least thrice. Then we can break the path in all but two of these places and add to it line segments lying on S of total length at most $3s$ such that π changes into a closed path π' that crosses S at most twice.*

[3] Note that polynomial time is not crucial here as the number of portals is constant.

This lemma applies to embedded graphs as well, as the patching process connects two subpaths of the tour that end on a common side of S by a subpath of S. When we apply the Patching Lemma, we do not allow automatic shortcuts; that is, the tour π' that we work with is a multi-subset of $\pi \cup S$. Only after application of the Patching Lemma do we use specific shortcuts in order to achieve the crossing properties that we need.

Theorem 8 (TSP Property of Bricks) *Let B be a brick of graph G with boundary $N \cup E \cup S \cup W$ (where E and W are supercolumns). Let T be a tour in G such that T crosses E and W at most 2 times each. Let H be the intersection of T with B. Then there is another subgraph of B, H', such that:*

(T1) H' has at most $\beta(\varepsilon)$ joining vertices with ∂B.
(T2) $\ell(H') \leq (1+5\varepsilon)\ell(H)$.
(T3) There is a tour in the edge set $(T \setminus H) \cup H'$ that spans the vertices in $\partial B \cap T$.

In the above, $\beta(\varepsilon) = \mathcal{O}(\kappa)$.

Proof We first remove ∂B from H. We partition the remaining edges 4 sets, each composed of minimal ∂B-to-∂B paths: $\mathcal{P}_{S\vee N} \cup \mathcal{P}_{E\vee W} \cup \mathcal{P}_{S\wedge N}$. Paths in $\mathcal{P}_{S\vee N}$ either start and end on S or start and end on N; paths in $\mathcal{P}_{E\vee W}$ start on E or W and end on ∂B; paths in $\mathcal{P}_{S\wedge N}$ start on S and end on N. For the constructions below, refer to Figure 4.

Since T crosses E and W at most 4 times, $|\mathcal{P}_{E\vee W}| \leq 4$. Therefore, this set of paths result in at most 8 joining vertices with ∂B.

For each path $P \in \mathcal{P}_{S\vee N}$, let $\widehat{P}$ be the minimal subpath of ∂B that spans P's endpoints. Let $\widehat{\mathcal{P}}_{S\vee N}$ be the resulting set of paths. As N is 0-short and S is ε-short, we have

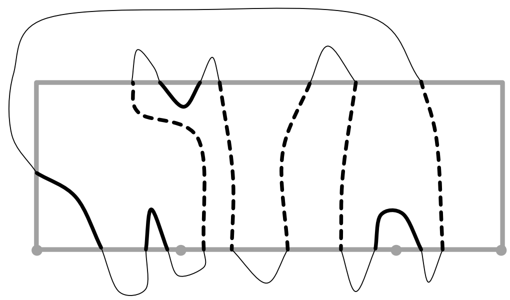
$$\ell(\widehat{\mathcal{P}}_{S\vee N}) \leq (1+\varepsilon)\ell(\mathcal{P}_{S\vee N}). \tag{1}$$

(In the case when E and W are trivial and P is a subpath connecting the endpoints of S, N is shorter than P.) Since $\widehat{\mathcal{P}}_{S\vee N}$ are subpaths of ∂B, they have no joining vertices with ∂B. Since paths in $\widehat{\mathcal{P}}_{S\vee N}$ correspond one-to-one with paths in $\mathcal{P}_{S\vee N}$, $(T \setminus \mathcal{P}_{S\vee N}) \cup \widehat{\mathcal{P}}_{S\vee N}$ is a tour. See Figure 4(a).

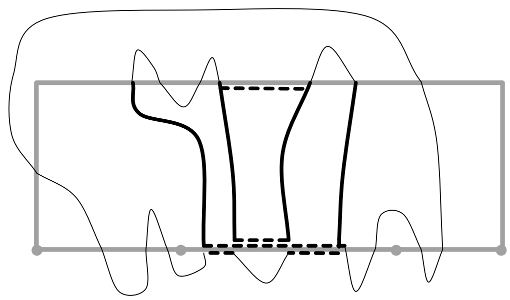
It remains to deal with the paths in $\mathcal{P}_{S\wedge N}$. Let $s_0, s_1, s_2, \ldots, s_k$ (where $k \leq \kappa$) be the vertices of S guaranteed by the properties of the bricks (see Definition 1). Let $\mathcal{X}_i$ be the subset of paths of $\mathcal{P}_{S\wedge N}$ that start on $S[s_i, s_{i+1})$, i.e., the vertices between s_i and s_{i+1} including s_i but not s_{i+1}.

If $|\mathcal{X}_i| > 2$, we do the following: Let P_i be the path in $\mathcal{X}_i$ whose endpoint x on S is closest to s_{i+1}. Let Q_i be the subpath of S from s_i to x. By the properties of the bricks, $\ell(Q_i) \leq \varepsilon\ell(P_i)$. Apply the Patching Lemma to the tour T and path Q_i; the new tour, T', crosses Q_i at most twice. However T' may still have many joining vertices with Q_i. Let $\mathcal{Q}_i$ be the subpaths of Q_i that are added to the tour.

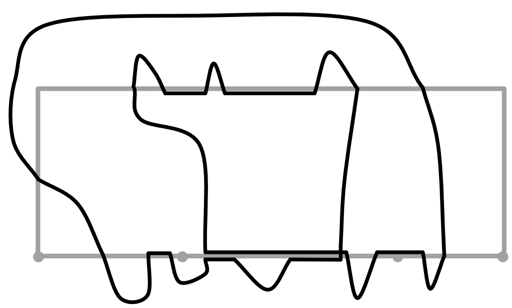
Let $\mathcal{L}_i$ be a maximal set of N-to-N paths in $\mathcal{X}_i \cup \mathcal{Q}_i$. $\mathcal{L}_i$ accounts for all but two (corresponding to the two crossings of T' with Q_i) of the joining vertices of T' with Q_i. For each path $L \in \mathcal{L}_i$, let $\widehat{L}$ be the minimal subpath of N that



(a)



(b)



(c)

Fig. 4 (a) A brick with a tour crossing through it. The bold paths are in H. The bold vertices are $s_0, s_1, s_2, \ldots, s_k$. The dotted paths are in $\mathcal{P}_{S \wedge N}$, the first four of which are in $\mathcal{X}_1$. (b) The patching process introduces the dotted paths on the lower boundary S of the brick and reroutes the tour to cross S twice between s_1 and s_2. The dotted subpath L of the top boundary N of the brick is used to replace the portion of the tour between its endpoints. (c) The tour after the entire construction given by Theorem 8.

spans L's endpoints and let $\widehat{\mathcal{L}}_i$ be the resulting set of paths. Replacing $\mathcal{L}_i$ with $\widehat{\mathcal{L}}_i$, we still have a tour, since the paths have a one-to-one correspondence. However, the resulting tour may no longer span all terminals on Q_i. Adding in two copies of Q_i fixes this. Since N is 0-short, $\ell(\widehat{\mathcal{L}}_i) \leq \ell(\mathcal{L}_i)$.

Based on the above description, let $\widehat{\mathcal{X}}_i = ((\mathcal{X}_i \cup \mathcal{Q}_i) \setminus \mathcal{L}_i) \cup \widehat{\mathcal{L}}_i \cup Q_i \cup Q_i$. Replacing $\mathcal{X}_i$ with $\widehat{\mathcal{X}}_i$, we still have a tour as argued above. Since the additional length added is at most 5 copies of Q_i, we have:

$$\ell(\widehat{\mathcal{X}}_i) \le \ell(\mathcal{X}_i) + 5\ell(Q_i) \le \ell(\mathcal{X}_i) + 5\varepsilon\ell(P_i) \le (1+5\varepsilon)\ell(\mathcal{X}_i). \tag{2}$$

Since $\mathcal{L}_i$ accounted for all but 2 of the joining vertices of T' with Q_i - so all but 4 of the joining vertices of $\mathcal{X}_i$ with ∂B - and $\widehat{\mathcal{L}}_i$ has no joining vertices with ∂B, $\widehat{\mathcal{X}}_i$ has at most 4 joining vertices with ∂B.

Let $\widehat{\mathcal{P}}_{S\wedge N} = \bigcup_i \widehat{\mathcal{X}}_i$. $\widehat{\mathcal{P}}_{S\wedge N}$ has at most 6κ joining vertices with ∂B and, by Equation (2),

$$\ell(\widehat{\mathcal{P}}_{S\wedge N}) \le (1+5\varepsilon)\ell(\mathcal{P}_{S\wedge N}). \tag{3}$$

Let $\widehat{H}$ be the union of the paths in $\mathcal{P}_{E\vee W}$, $\widehat{\mathcal{P}}_{S\vee N}$ and $\widehat{\mathcal{P}}_{S\wedge N}$. Combining Equations (1) and (3), we find that $\ell(\widehat{H}) \le (1+5\varepsilon)\ell(H)$. By construction, the edges in $(T\setminus H)\cup\widehat{H}$ contain a tour. $\widehat{H}$ has $6\kappa+8$ joining vertices with ∂B. □

6.1 Proof of the Structure Theorem for Subset Tsp

Using the TSP Property of Bricks, we prove the Structure Theorem (Section 4.4) for Subset Tsp.

Let T be the optimal tour spanning terminals Q in G. From T we build a tour $\widehat{T}$ spanning Q in $\mathcal{B}^+(MG)$ such that $\ell(\widehat{T}) \le (1+c\varepsilon)\ell(T)$.

Let C be a supercolumn. By the Patching Lemma, if T crosses C at least thrice, we can add to T at most three copies of C and create a new tour that crosses C at most twice. We do not apply any short cuts to this new tour, and let T_1 be the tour that is a multi-subgraph of $T\cup C$ that results from applying the Patching Lemma to each supercolumn. Because the sum of the weights of the supercolumns is at most $\varepsilon\,\mathrm{OPT}$,

$$\ell(T_1) \le (1+3\varepsilon)\ell(T). \tag{4}$$

Let B be a brick of G. Let H be the intersection of T_1 with B. By the construction above, T_1 satisfies the requirements of Theorem 8: let H' be the guaranteed subgraph of B. We replace H with H' in T_1. Let T_2 be the tour resulting from such replacements over all the bricks. Theorem 8 guarantees that

$$\ell(T_2) \le (1+5\varepsilon)\ell(T_1). \tag{5}$$

Now we map the edges of T_2 to a subgraph of $\mathcal{B}^+(MG)$. Every edge of G has at least one corresponding edge in $\mathcal{B}^+(MG)$. (Edges that are on the boundaries of bricks appear three times in $\mathcal{B}^+(MG)$; all other edges appear once.) For every edge e of T_2, we select one corresponding edge in $\mathcal{B}^+(MG)$ as follows: if e is an edge of MG select the corresponding mortar edge of $\mathcal{B}^+(MB)$, otherwise select the unique edge corresponding to e in $\mathcal{B}^+(MG)$. (An edge of $\mathcal{B}^+(MG)$ is a *mortar edge* if it is in MG.) This process constructs a subgraph T_3 of $\mathcal{B}^+(MG)$ such that

$$\ell(T_3) = \ell(T_2). \tag{6}$$

Because T_3 is not connected, we connect it via portal and mortar edges. Let V_B be the set of joining vertices of T_3 with ∂B for a brick B of $\mathcal{B}^+(MG)$. For any vertex v on the interior boundary ∂B of a brick, let p_v be the portal on ∂B that is closest to v, let P_v be the shortest v-to-p_v path along ∂B and let P'_v be the corresponding path of mortar edges. Let e be the portal edge corresponding to p_v. Add P_v, P'_v, and e to T_3. Repeat this process for every $v \in V_B$ and for every brick B, building a graph $\widehat{T}$. This completes the definition of $\widehat{T}$. From the construction, $\widehat{T}$ is a tour spanning the terminals Q in $\mathcal{B}^+(MG)$.

Now we analyze the increase in length:

$$\ell(\widehat{T}) \leq \ell(T_3) + \sum_{B \in \mathcal{B}} \sum_{v \in V_B} (\ell(P_v) + \ell(e) + \ell(P'_v)), \tag{7}$$

and we have:

$$\begin{aligned}
\sum_{B \in \mathcal{B}} \sum_{v \in V_B} \ell(P_v) + \ell(e) + \ell(P'_v) &= 2 \sum_{B \in \mathcal{B}} \sum_{v \in V_B} \ell(P_v), \text{ because } \ell(\text{portal edges}) = 0 \\
&\leq 2 \sum_{B \in \mathcal{B}} \sum_{v \in V_B} \ell(\partial B)/\theta, \text{ by the choice of portals} \\
&\leq 2 \sum_{B \in \mathcal{B}} \frac{\beta}{\theta} \ell(\partial B), \text{ by Theorem 8} \\
&\leq 2 \frac{\beta}{\theta} 2\alpha(\varepsilon^{-1}, g) \,\mathrm{OPT}\,, \text{ by Theorem 4} \\
&\leq \varepsilon \,\mathrm{OPT}\,, \text{ for } \theta = 4\varepsilon^{-1}\beta\alpha, \text{ as required.}
\end{aligned}$$

Combining Equations (4), (5), (6) and (7), we obtain $\ell(\widehat{T}) \leq (1 + 3\varepsilon)(1 + 5\varepsilon)\ell(T) + \varepsilon \,\mathrm{OPT} \leq (1 + c\varepsilon) \,\mathrm{OPT}$. The Structure Theorem is proved for the Subset Tsp problem.

7 Conclusion and outlook

We presented a framework to obtain PTASes on bounded-genus graphs for subset-connectivity problems, where we are given a graph, a set of terminal vertices and certain connectivity requirements among the terminals. Specifically, we obtained the first PTAS for Steiner Tree, Subset Tsp and $\{0, 1, 2\}$-edge-connected Survivable Network on bounded-genus graphs and our running time is $\mathcal{O}(n \log n)$ with a constant that is singly exponential in ε^{-1} and the genus of the graph. Our method is based on the framework of Borradaile et al. [9] for planar graphs; in fact, we generalize their work in the sense that basically any problem that is shown to admit a PTAS on planar graphs using their framework easily generalizes to bounded-genus graphs using the methods presented in this work. Since the publication of the conference version of this work, our method has also been applied to further problems such as Steiner Forest [5] and prize-collecting variants [4].

A natural question is to ask what other classes of graphs admit a PTAS for the problems discussed in this work. An important generalization of bounded-genus graphs are graphs that exclude a fixed graph H as a minor. Such H-*minor-free graphs* have earned much attention in recent years. Many hard optimization problems have been shown to admit PTASes and fixed-parameter algorithms on these classes of graphs; see, e.g., [20, 13, 12]. The recent contraction decomposition result for H-minor-free graphs of Demaine et al. [14] gives rise to the first PTAS for the full TSP in H-minor-graphs; however, this PTAS is not efficient as it uses a spanner of weight $\mathcal{O}(\log n \cdot \mathrm{OPT})$, and thus the running time of the PTAS is $n^{f(\varepsilon^{-1})}$. Subset-connectivity problems, specifically SUBSET TSP and STEINER TREE, remain important open problems [20, 14].

Very often, results on H-minor-free graphs are first shown for planar graphs, then extended to bounded-genus graphs, and finally obtained for H-minor-free graphs. This is due to the powerful decomposition theorem of Robertson and Seymour [28] that essentially says that every H-minor-free graph can be decomposed into a number of parts that are "almost embeddable" in a bounded-genus surface. We conjecture that our framework extends to H-minor-free graphs via this decomposition theorem. The advantage of our methodology is that handling weighted graphs and subset-type problems are naturally incorporated, and thus it might be possible to combine all the steps for a potential PTAS into a single framework for H-minor-free graphs based on what we presented in this work. Hence, whereas our work is an important step towards this generalization, still a number of hard challenges remain.

References

1. Arnborg, S., Proskurowski, A.: Linear time algorithms for NP-hard problems restricted to partial k-trees. Discrete Applied Mathematics **23**(1), 11–24 (1989)
2. Arora, S.: Approximation schemes for NP-hard geometric optimization problems: A survey. Mathematical Programming **97**(1–2), 43–69 (2003)
3. Baker, B.S.: Approximation algorithms for NP-complete problems on planar graphs. J. ACM **41**(1), 153–180 (1994)
4. Bateni, M., Chekuri, C., Ene, A., Hajiaghayi, M.T., Korula, N., Marx, D.: Prize-collecting Steiner problems on planar graphs. In: SODA'11: Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms, pp. 1028–1049. SIAM (2011)
5. Bateni, M., Hajiaghayi, M., Marx, D.: Approximation schemes for steiner forest on planar graphs and graphs of bounded treewidth. In: STOC '10: Proceedings of the 42nd annual ACM Symposium on Theory of Computing, pp. 211–220. ACM (2010)
6. Borradaile, G., Klein, P.: The two-edge connectivity survivable network problem in planar graphs. In: ICALP '08: Proceedings of the 35th International Colloquium on Automata, Languages and Programming, *LNCS*, vol. 5125, pp. 485–501. Springer (2008)
7. Borradaile, G., Klein, P.N., Mathieu, C.: A polynomial-time approximation scheme for Steiner tree in planar graphs. In: SODA '07: Proceedings of the 18th annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1285–1294 (2007)
8. Borradaile, G., Klein, P.N., Mathieu, C.: Steiner tree in planar graphs: An $O(n \log n)$ approximation scheme with singly exponential dependence on epsilon. In: WADS '07: Proceedings of the 10th Workshop on Algorithms and Data Structures, *LNCS*, vol. 4619, pp. 275–286. Springer (2007)
9. Borradaile, G., Klein, P.N., Mathieu, C.: An $O(n \log n)$ approximation scheme for Steiner tree in planar graphs. ACM Transactions on Algorithms **5**(3) (2009)

10. Cabello, S., Chambers, E.W.: Multiple source shortest paths in a genus g graph. In: SODA '07: Proceedings of the 18th annual ACM-SIAM Symposium on Discrete Algorithms, pp. 89–97. SIAM (2007)
11. Chimani, M., Mutzel, P., Zey, B.: Improved Steiner tree algorithms for bounded treewidth. In: IWOCA'11: Revised Selected Papers of the 22nd International Workshop on Combinatorial Algorithms, *Lecture Notes in Computer Science*, vol. 7056, pp. 374–386. Springer (2011)
12. Demaine, E.D., Fomin, F.V., Hajiaghayi, M., Thilikos, D.M.: Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. J. ACM **52**(6), 866–893 (2005)
13. Demaine, E.D., Hajiaghayi, M.: Bidimensionality: New connections between FPT algorithms and PTASs. In: SODA '05: Proceedings of the 16th annual ACM-SIAM Symposium on Discrete Algorithms, pp. 590–601 (2005)
14. Demaine, E.D., Hajiaghayi, M., Kawarabayashi, K.: Contraction decomposition in H-minor-free graphs and algorithmic applications. In: STOC'11: Proceedings of the 43rd Symposium on Theory of Computing, pp. 441–450. ACM (2011)
15. Demaine, E.D., Hajiaghayi, M., Mohar, B.: Approximation algorithms via contraction decomposition. In: SODA '07: Proceedings of the 18th annual ACM-SIAM Symposium on Discrete Algorithms, pp. 278–287. SIAM (2007)
16. Eppstein, D.: Dynamic generators of topologically embedded graphs. In: SODA '03: Proceedings of the 14th annual ACM-SIAM Symposium on Discrete Algorithms, pp. 599–608. SIAM (2003)
17. Eppstein, D., Italiano, G., Tamassia, R., Tarjan, R., Westbrook, J., Yung, M.: Maintenance of a minimum spanning forest in a dynamic planar graph. J. Algorithms **13**(1), 33–54 (1992). Special issue for 1st SODA
18. Erickson, J., Whittlesey, K.: Greedy optimal homotopy and homology generators. In: SODA '05: Proceedings of the 16th annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1038–1046. SIAM (2005)
19. Erickson, R.E., Monma, C.L., Arthur F. Veinott, J.: Send-and-split method for minimum-concave-cost network flows. Math. Oper. Res. **12**(4), 634–664 (1987)
20. Grohe, M.: Local tree-width, excluded minors, and approximation algorithms. Combinatorica **23**(4), 613–632 (2003)
21. Henzinger, M.R., Klein, P.N., Rao, S., Subramanian, S.: Faster shortest-path algorithms for planar graphs. Journal of Computer and System Sciences **55**(1), 3–23 (1997)
22. Klein, P.N.: A subset spanner for planar graphs, with application to subset TSP. In: STOC '06: Proceedings of the 38th annual ACM Symposium on Theory of Computing, pp. 749–756 (2006)
23. Klein, P.N.: A linear-time approximation scheme for TSP in undirected planar graphs with edge-weights. SIAM J. Comput. **37**(6), 1926–1952 (2008)
24. Korach, E., Solel, N.: Linear time algorithm for minimum weight Steiner tree in graphs with bounded treewidth. Manuscript (1990)
25. Mehlhorn, K.: A faster approximation algorithm for the Steiner problem in graphs. Information Processing Letters **27**, 125–128 (1988)
26. Mohar, B.: A linear time algorithm for embedding graphs in an arbitrary surface. SIAM Journal on Discrete Mathematics **12**(1), 6–26 (1999)
27. Mohar, B., Thomassen, C.: Graphs on Surfaces. The John Hopkins University Press (2001)
28. Robertson, N., Seymour, P.: Graph minors. XVI. Excluding a non-planar graph. J. Comb. Theory Ser. B **89**(1), 43–76 (2003)
29. Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of tree-width. Journal of Algorithms **7**(3), 309–322 (1986)
30. Tazari, S., Müller-Hannemann, M.: Shortest paths in linear time on minor-closed graph classes, with an application to Steiner tree approximation. Discrete Applied Mathematics **157**, 673–684 (2009)
31. Tazari, S., Müller-Hannemann, M.: Dealing with large hidden constants: Engineering a planar Steiner tree PTAS. ACM Journal of Experimental Algorithmics **16**(3) (2011). Article 3.16. Special Issue on ALENEX'09