Technical Report

TR-2004-025

Approximate Minimum 2-Connected Subgraphs in Weighted Planar Graphs

by

Andre Berger, Artur Czumaj, Michelangelo Grigni, Hairong Zhao

Mathematics and Computer Science

EMORY UNIVERSITY

Approximate Minimum 2-Connected Subgraphs in Weighted Planar Graphs

André Berger* Artur Czumaj† Michelangelo Grigni* Hairong Zhao†

Abstract

We consider the problems of finding the minimum-weight 2-connected spanning subgraph in edge-weighted planar graphs and its variations. We first give a PTAS for the problem of finding minimum-weight 2-edge-connected spanning subgraphs where duplicate edges are allowed. Then we present a new greedy spanner construction for edge-weighted planar graphs. From this we derive quasi-polynomial time approximation schemes for the problems of finding the lightest 2-edge-connected or biconnected spanning subgraph in such graphs. We also design efficient approximation schemes for the variant where vertices have non-uniform (1 or 2) connectivity constraints.

1 Introduction

The design of graphs that resist edge and/or vertex removal is essential in algorithmic graph theory, and has numerous applications in computer science and operations research. The classical k-connectivity problems are perhaps the most extensively studied problems in network design. We are given a graph G with n vertices and a nonnegative weight $w(e)$ on each edge e, and we want to find a k-edge or k-vertex connected *spanning* subgraph S (a subgraph containing all vertices), such that its total edge weight $w(S)$ equals OPT, the minimum possible.

In this paper we consider approximation algorithms for the case when $k = 2$: we either want S to be a 2-edge-connected (2-EC) spanning subgraph (a 2-ECSS), or a 2-vertex-connected (2-VC or biconnected) spanning subgraph (a 2-VCSS). We also consider some variations of these two problems. The first variation is a relaxation of the 2-ECSS problem: we want to find a minimum weight 2-EC spanning sub-*multigraph* (2-ECSSM), meaning that we may use each edge of G multiple times. Another variant is the *1-2-connectivity problem*: each vertex v is assigned a connectivity type $r_v \in \{1, 2\}$. The problem is to find a minimum weight spanning subgraph such that for any pair of vertices $v, u \in V$, there are at least $r_{uv} = \min\{r_u, r_v\}$ edge-disjoint or vertex-disjoint paths. We denote the 1-2-edge-connectivity by {1,2}-EC, and 1-2-vertex-connectivity by {1,2}-VC. We also consider the relaxed 1-2-edge-connectivity problem where each edge may be used more than once.

A *c-approximation algorithm* always produces a *c-approximate* solution: a solution S such that $w(S) \leq c \cdot \mathrm{OPT}$. The constant $c \geq 1$ is the *approximation guarantee* of the algorithm. Many

*Department of Mathematics and Computer Science, Emory University, Atlanta GA 30322, USA. Email: aberge2@emory.edu, mic@mathcs.emory.edu. Research supported in part by NSF grant CCR-0208929.

†Department of Computer Science, New Jersey Institute of Technology, Newark NJ 07102, USA. Email: {czumaj,hairong}@cis.njit.edu. Research supported in part by NSF grants ITR-CCR-0313219 and CCR-0105701.

polynomial time approximation algorithms are already known for these problems, see the survey [18] and more recent advances [4, 10, 11, 17, 20]. We would prefer a *polynomial-time approximation scheme (PTAS)*, which is an approximation algorithm taking both G and c as inputs, and running in polynomial time for each fixed $c > 1$. However, both the 2-ECSS and 2-VCSS problems are max-SNP-hard [6], even for unweighted graphs or when duplicate edges are allowed; therefore we do not expect a PTAS. But this does not preclude a PTAS for restricted classes of graphs: in particular, a PTAS exists for both problems in *geometric graphs* of constant dimension [6], and also in *unweighted planar graphs* [5].

In fact, the approximation schemes of [5] allow weighted planar graphs, but then those algorithms run in time $n^{O((1/\varepsilon)(w(G)/\mathrm{OPT}))}$, where $\varepsilon = c - 1$. The ratio $w(G)/\mathrm{OPT}$ appearing in the exponent could be arbitrarily large. For both problems in weighted planar graphs, the best known subexponential-time approximation guarantee is still 2 [19, 24], which is achieved by polynomial-time algorithms working for general weighted graphs. On the other hand, besides the PTAS in [5], there are much better constant approximation guarantees known for unweighted graphs. For example, there exists a 5/4-approximation algorithm for the unweighted 2-ECSS problem [17], and a 4/3-approximation algorithm for the unweighted 2-VCSS problem [27].

The 1-2-connectivity problem, as described above, is a basic non-trivial variant of the *survivable network design problem* [13–15, 23, 26]. For both, the unweighted {1,2}-ECSS and the unweighted {1,2}-VCSS problem, Krysta [21] gives 3/2-approximation algorithms. If the graph is weighted, the best known result for {1,2}-ECSS is a 2-approximation algorithm, due to Jain [16], which in fact solves the more general problem where $r_v \leq k$ for any k. For the {1,2}-VCSS problem with arbitrary weights, Fleischer [9] has given a 2-approximation algorithm, which actually solves the {0,1,2}-VCSS problem. A PTAS for geometric version of these problems has been presented in [8].

The discrepancy between approximation guarantees of the unweighted and weighted versions of these problems is a phenomenon that is known also for general graphs. A striking discrepancy is known for the k-VCSS problem ($k \gg 2$), which admits a $(1+1/k)$-approximation for the unweighted case [3], while for the weighted version of the problem the best known approximation guarantee is $O(\log n)$ [4] (see also [20]). In general, a PTAS for unweighted graphs in some family does not seem to imply a PTAS for weighted graphs in the same family. In particular, the existence of a sub-exponential time approximation scheme for these problems in weighted planar graphs has remained as a major open question in the area.

1.1 Light Spanners

Let G be a weighted graph. We use $d_G(u, v)$ to denote the weighted shortest path distance between the vertices u and v in G. An *s-spanner* of G is a spanning subgraph H of G such that $d_H(u, v) \leq s \cdot d_G(u, v)$ for all u, v. A spanner provides an approximate representation of the shortest path metric (1-connectivity) in G, but it may be much lighter than G.

For general graphs, a simple greedy algorithm [1] computes an s-spanner H for any $s > 1$, and in planar graphs this spanner has weight $w(H) \leq (1 + 2/(s - 1))\mathrm{MST}(G)$, where $\mathrm{MST}(G)$ is the weight of a minimum spanning tree in G. Since $\mathrm{MST}(G) \leq \mathrm{OPT}$ for all the problems we consider, this bounds the ratio $w(H)/\mathrm{OPT}$ in terms of just s. If all weighted graphs in a graph family have spanners with such a bound on $w(H)/\mathrm{OPT}$ (depending only on s), then we say the family has *light spanners* for this problem. Light spanners are known to be very useful for solving some optimization problems on graphs. For example, planar graphs have light spanners for metric-TSP: the first step in the metric-TSP PTAS for weighted planar graphs is to replace the input

graph with an accurate enough s-spanner (using [1]), thus effectively bounding $w(G)/\text{OPT}$ for the remainder of the algorithm. Spanners are also used in complete geometric graphs to design efficient PTAS's for geometric TSP and related problems [25], and to design PTAS's for the 2-edge and 2-vertex-connectivity problems [7, 8].

But this approach of replacing the input graph with an s-spanner fails for the 2-ECSS and 2-VCSS problems. The reason is that a spanner does not have to be 2-connected. This problem does not exist in [7] and [8], because in the metric case one can always reduce these two problems to the case where duplicate edges are allowed. Then one can show there is a near optimal solution that uses only spanner edges. However, in our case, edges are arbitrarily weighted. Therefore, the spanner may not contain the optimal or a near optimal solution in most cases. Naturally, one may think to use light *fault-tolerant* spanners (see, e.g., in [22]), which are subgraphs that persist as s-spanners even after deleting some constant number of vertices or edges. Unfortunately, this concept is not useful in our setting, since simple examples show that light fault-tolerant spanners do not exist in weighted planar graphs, not even for a single edge deletion.

1.2 New Contributions

We present efficient approximation schemes for all the above mentioned problems in weighted planar graphs. We first present a simple PTAS for the 2-ECSSM problem in Section 2. Our central result, developed in Sections 3 through 5, is an approximation scheme for the 2-ECSS problem finding a solution with weight at most $(1+\varepsilon)\cdot\text{OPT}$ in $n^{O((\log n)(\log(1/\varepsilon)/\varepsilon))}$ time; this is a quasi-polynomial time approximation scheme (QPTAS). Alongside that we also sketch a similar QPTAS for the 2-VCSS problem, with the same performance. Finally in Section 6, we consider {1,2}-ECSS and {1,2}-VCSS problems: we show there is a PTAS for the {1,2}-ECSSM problem, and a QPTAS for each of the {1,2}-ECSS and {1,2}-VCSS problems.

Despite the difficulties listed in Section 1.1, we do use light spanners as follows. First, we modify the greedy construction to produce a light planar spanner H^*, so that it is 2-EC (or 2-VC), and the weight of each omitted edge is bounded by s times the length of a path within the face of H^* containing that edge. While H^* need not contain an approximate solution S, we do put a bound on the number of edges S "crossing" each face of H^* (Theorem 4).

Our central algorithmic approach for 2-ECSS may be roughly summarized as follows. We apply a planar separator theorem (from [5]) to H^*, rather than G. We then exhaustively guess the edges of S outside H^* which "cross" the separator; this is the super-polynomial part of our algorithm. We repeat the technique for all subproblems that occur in the recursive decomposition. We reuse other techniques of [5] to improve the running time. We also introduce some new analysis bounding the number of subproblems that occur during our dynamic programming (Theorems 6 and 7).

2 A PTAS for the Minimum-Weight 2-ECSSM of Planar Graphs

In this problem, we may use each edge of G multiple times. However, it never helps to use an edge more than twice, so we may cap all edge multiplicities at two. The given weighted planar graph G should be connected, but it need not be 2-EC. We now sketch a simple PTAS for this problem, running in $n^{O(1/\varepsilon^2)}$ time.

We first show that there is a $(1+\varepsilon)$-approximate 2-ECSSM that uses only edges from a s-spanner, where s depends on ε. Given G and $\varepsilon > 0$, we choose s so that $s^2 \le 1+\varepsilon$. We compute

an s-spanner H in G by the greedy spanner algorithm [1], with weight $w(H) = O((1/\varepsilon) \cdot \mathrm{OPT})$. Suppose S^* is a 2-ECSSM in G with $w(S^*) = \mathrm{OPT}$. For each edge e of S^* not in H, we remove e and add a shortest path from H of total weight at most $s \cdot w(e)$. When we add the path, we add the edges with multiplicity, but capped at two. The result of all these modifications is another 2-ECSSM S, using only edges from H, each edge used at most twice, with $w(S) \leq s \cdot \mathrm{OPT}$.

Now we apply the known 2-ECSS s-approximation algorithm [5] to the graph H', which is H with each edge duplicated. (One may check that [5] allows parallel edges, or alternatively just use a standard reduction from H' back to a simple graph.) The 2-ECSS found in H' is what we want: it is a 2-ECSSM in G with weight at most $s \cdot w(S) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$. The time used is $n^{O((1/(s-1))(w(H')/\mathrm{OPT}))} = n^{O(1/\varepsilon^2)}$, as claimed.

In summary, we have the following theorem.

Theorem 1 *Let $\varepsilon > 0$ and let G be a connected weighted planar graph with n vertices. There is an algorithm running in time $n^{O(1/\varepsilon^2)}$ that outputs a 2-ECSSM of G whose weight is at most $(1 + \varepsilon)$ times the minimum.*

Of course the above technique does not work for the 2-ECSS problem, because we are not allowed to duplicate edges from G in a 2-ECSS. Instead our 2-ECSS approximation scheme must consider the possibility that the near-optimal S needs some "extra" edges from outside the spanner. In Sections 3 and 4 we develop a new type of light planar spanner, and we limit the number and arrangement of those extra edges outside the spanner.

3 Augmented Planar Spanners

Here we present a new greedy algorithm constructing s-spanners in weighted planar graphs, resembling the standard greedy algorithm [1] for general graphs. Just as in the standard algorithm, we take a connected weighted graph G and a parameter $s \geq 1$, and produce an s-spanner H. Unlike the general algorithm, our G must be planar, and for each edge e of G not in H we guarantee that $s \cdot w(e)$ is at least the length of some path in the face of H containing e. We also provide our algorithm with a third argument: a "seed" spanning subgraph A, containing edges that must appear in H. In Section 4 we will use A to enforce some 2-connectivity properties in the spanner.

Suppose G is a weighted *plane graph* (that is, an embedded planar graph) and H is a subgraph. A *chord* of H is an edge of G not in H. Note that H and e inherit embeddings from G. For each chord e we define $w_H(e)$ as the length of the shortest walk connecting the endpoints of e, along the boundary of the face of H containing e.

More precisely, if the endpoints of e are disconnected in H, then we define $w_H(e) = +\infty$. Otherwise e connects two vertices in a component of H, and e is embedded in some face f of this component. The boundary of f is a cyclic walk of (oriented) edges, with total weight $w(f)$; note that a cut-edge may appear twice in the boundary (once per orientation), and its weight would then count twice in $w(f)$. Similarly a cut-vertex may appear multiple times. The edge e splits the boundary sequence into two walks P_1 and P_2, both connecting the endpoints of e, with $w(P_1) + w(P_2) = w(f)$. Now we define $w_H(e) = \min(w(P_1), w(P_2))$ (see Figure 1).

Given G, s, and A as above, we compute $H = \mathrm{Augment}(G, s, A)$ as follows:

Augment(G, s, A):
    $H \leftarrow A$

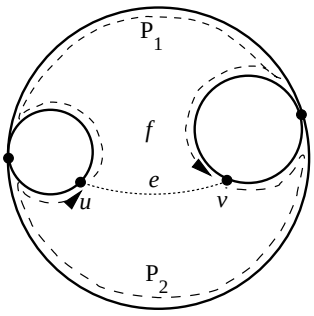

Figure 1: A non-simple face f, a chord e of f, and walks P_1 and P_2.

for all edges e of G in non-decreasing $w(e)$ order **do**
  if e is not in H **and** $s \cdot w(e) < w_H(e)$ **then**
    add e to H
return H

Note $A \subseteq H \subseteq G$. If A is empty (has all vertices of G but no edges), then this is like the general greedy spanner algorithm [1], except that we have w_H in place of d_H.

Theorem 2 *If G, s, and A are as above, then $H = \text{Augment}(G, s, A)$ is an s-spanner of G.*

Proof: It suffices to show that each edge of G is s-approximated in H. For e not in H, at the moment it was rejected we had $w_H(e) \leq s \cdot w(e)$. Note that $w_H(e)$ may only decrease after that, so $d_H(e) \leq w_H(e) \leq s \cdot w(e)$ at the end of the algorithm. □

Theorem 3 *Suppose G, s, and A are as above, $s > 1$, A is connected, and $H = \text{Augment}(G, s, A)$. Then $w(H) \leq (1 + 2/(s-1)) \cdot w(A)$.*

Proof: We need to show that the weight of all edges in H but not A is at most $(2/(s-1)) \cdot w(A)$. Suppose e is such an edge; then e is not a cut edge in H, so it is bounded by two distinct faces. Let f be either face bounding e. We first claim that $w(f) > (1 + s) \cdot w(e)$. To see this, consider the *last* edge e' added to f. The boundary of f is a path P plus e', such that $w_H(e') \leq w(P)$ and $s \cdot w(e') < w_H(e')$. Adding $w(e')$ to both sides of $s \cdot w(e') < w(P)$, and noting $w(e) \leq w(e')$, we get the claim.

For each face f of A, let E_f be the set of edges in H crossing the interior of f. Since the sum of $w(f)$ over all faces of A is $2 \cdot w(A)$, it suffices to show that $w(E_f) \leq (1/(s-1)) \cdot w(f)$. Note the edges dual to E_f define a tree on the faces of H inside f. Orient this dual tree away from some arbitrarily chosen root: now for each $e \in E_f$, we have chosen an adjacent face f_e of H (only the root was not picked). For each $e \in E_f$ we know $w(f_e) - 2 \cdot w(e) > (s-1) \cdot w(e)$, from the previous paragraph. Summing these inequalities over all $e \in E_f$, we get at most $w(f)$ on the left hand side, and exactly $(s-1) \cdot w(E_f)$ on the right. □

4 Spanners and 2-EC Subgraphs

Suppose we are given a weighted plane 2-EC graph G, where we want to find a $(1+\varepsilon)$-approximate 2-ECSS. We first compute a useful subgraph H^*, as follows:

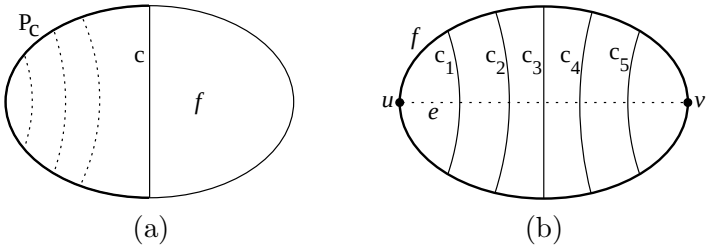

Figure 2: (a) Face f (oval) with chord c, path P_c (bold), and chords removed by the chord move at c (dotted). (b) Face f with a face-edge e (dashed) crossed by five chords.

1. Compute a 2-approximate 2-ECSS A, in polynomial time.
2. Compute $H^* = \text{Augment}(G, \sqrt{2}, A)$.

The constant $\sqrt{2}$ here is not critical, just convenient. By Theorem 3, H^* is a 12-approximate 2-ECSS. In the next theorem, we see that for every $\varepsilon > 0$, this H^* has nice intersection properties with some $(1+\varepsilon)$-approximate 2-ECSS in G.

Given a face f in H^*, the chords of f are the edges of G embedded inside this face, according to G's embedding. A *face-edge* e of f is an abstract edge connecting two vertices of f; unlike a chord, a face-edge is not necessarily an edge of G. (If vertices appear more than once on f, we must specify which appearances we want as the endpoints of e.) We say e *crosses* a chord c if: c is a chord of the same face f, their endpoints are distinct vertex appearances on f, and they appear in cyclic "*ecec*" order around the boundary of f. Note that we may embed e inside f so e intersects only the crossed chords.

Suppose S is a 2-ECSS in G, and an edge c of S is not in H^*. Then c is a chord of some face f of H^*. Let P_c be the path in f connecting the endpoints of c, such that $w(P_c) \leq \sqrt{2} \cdot w(c)$. Then the *chord move at* c is the following modification of S: add to S all the edges of P_c that were not already in S, and remove from S any chords inside the cycle $c \cup P_c$ (see Figure 2(a)). Since H^* is 2-EC, the cycle has no repeated edges, and therefore S is still a 2-ECSS after the chord move. The chord move is *improving* if $w(S)$ decreases; this happens whenever $w(P_c)$ (or $\sqrt{2} \cdot w(c)$) is less than the weight of the discarded chords. Any non-trivial chord move brings S closer to H^* (in Hamming distance), therefore at most $O(n)$ improving chord moves apply to any given 2-ECSS S.

Lemma 1 *Suppose S is a 2-ECSS, e is a face-edge in face f, C is a set of chords crossing e, $\sqrt{2} \cdot \min_{c \in C} w(c) > \max_{c \in C} w(c)$, and no chord in C gives an improving chord move. Then $|C| \leq 4$.*

Proof: If not, S has five chords crossing e as in Figure 2(b). But then we have an improving chord move at c_3, since the discarded chords ($\{c_1, c_2\}$ or $\{c_4, c_5\}$) weigh more than $\sqrt{2} \cdot w(c_3)$. □

Now we argue that by accepting a small additive error in our 2-ECSS, we may assume it has only a small number of chords crossing a given face-edge:

Theorem 4 *Suppose G and H^* are as above, $\varepsilon > 0$, S is a 2-ECSS, f is a face in H^*, and e is a face-edge in f. Then there exists a 2-ECSS S' such that $w(S') \leq w(S) + \varepsilon \cdot w(C'_f)$, where C'_f is the set of chords in S' crossing e, $C'_f \subseteq S$, and $|C'_f| = O(\log(1/\varepsilon))$.*

Proof: First we may suppose that S has no improving chord move at a chord crossing e, since such a move could only remove some chords crossing e. Let C_f be the set of chords in S crossing e. Arrange C_f in "left to right" order, according to how they intersect e. Let $c_0 \in C_f$ be the chord with maximum weight. Say that a chord $c \in C_f$ is *short* if $w(c) \leq \varepsilon \cdot w(c_0)/(2\sqrt{2})$. Now if there are short chords to the left of c_0, perform a chord move at the rightmost one, c_l. Similarly if there are short chords to the right of c_0, perform a chord move at the leftmost one, c_r. S' is the result of these (at most) two chord moves; note that C'_f contains no short chords except possibly c_l and c_r.

Map each non-short chord $c \in C'_f$ to the real number $\log(w(c_0)/w(c))$, a point in the real interval $I = [0, \log(1/\varepsilon) + 3/2]$. By Lemma 1, each semi-open subinterval of I of length $1/2$ receives at most four of these points. This implies $|C'_f| = O(\log(1/\varepsilon))$.

The chord moves in f increased $w(S')$ by at most $\sqrt{2}(w(c_l) + w(c_r)) \leq \varepsilon \cdot w(c_0)$, which is at most $\varepsilon \cdot w(C'_f)$. □

Remarks: In the 2-VCSS case, the initial A should be a 2-approximate 2-VCSS, so that H^* is a 12-approximate 2-VCSS. Then in the chord move the cycle has no repeated vertices, therefore S remains a 2-VCSS after the move. In Lemma 1 and Theorem 4, the only properties of H^* that we needed were that it was 2-EC (or 2-VC), and that $w_{H^*}(e) \leq \sqrt{2} \cdot w(e)$ for each chord e.

5 The 2-ECSS and 2-VCSS Approximation Schemes

In this section, we use our new spanner construction to design a QPTAS for the 2-ECSS problem in weighted planar graphs. We also sketch a similar QPTAS for the 2-VCSS problem. Roughly speaking, our algorithms are extensions of those in [5]: we first recursively decompose the graph G into small pieces, then use dynamic programming to solve the problems in each piece. As we mentioned earlier, the algorithms in [5] run in time exponential in $O(w(G)/\mathrm{OPT})$, which may be very large in weighted graphs. We overcome this difficulty by recursively decomposing the spanner H^* as constructed in Section 4, instead of G. On the other hand, since H^* may not contain a $(1+\varepsilon)$-approximate solution, we cannot completely ignore G after we obtain H^*. So, after each step of the decomposition, we need to add some crossing edges back into each piece. Of course, we do not know exactly which ones to add. Fortunately, Theorem 4 implies that each time we cut a face in H^*, it is enough to guess only $O(\log(1/\varepsilon))$ crossing edges not in H^*, and we pay an error at most ε times the weight of these guessed edges that we commit to our solution. Thus we bound both the number of guessed edges as well as the size of the interface between each piece and the remaining subgraphs of G. This in turn bounds the running time of our algorithm.

5.1 Modified Miller's Planar Separator

Our recursive decomposition of H^* is driven by this planar separator theorem from [5]:

Theorem 5 *Given a connected plane graph G with non-negative weights on its edges and vertices, and given a positive integer k, we may apply a near linear-time algorithm to find cycles and a Jordan curve as described below, such that:*

1. *We have a collection $\mathcal{C}$ of at most three edge-disjoint simple cycles in G, such that if we choose the point at infinity appropriately, they have disjoint interiors. Each interior has at most half the total vertex weight, and the cycles have edge weight $O(1/k)$ times the total edge weight.*

2. *Suppose we delete the cycle interiors and contract each cycle to a point with zero weight. In this embedded graph, we have a Jordan curve J passing through $O(k)$ vertices and no edges, so the interior and exterior of J both have at most $2/3$ the total vertex weight of G.*

By considering the choices made by the above algorithm, we may show the following:

Theorem 6 *Given G and k as in Theorem 5 but with unknown vertex weights, there exists a list of $O(n^2)$ separations (each separation being a pair $(\mathcal{C}, J)$ as above), such that for any vertex weighting of G, some separation in this list satisfies the properties of Theorem 5.*

5.2 Approximation Schemes

We first describe the approximation scheme for the 2ECSS problem. To find a near-optimal 2-ECSS in a weighted planar graph G, we first preprocess G to get a desirable spanner H^* as described in Section 4. Next, let $k = O((\log n)/\varepsilon)$. We decompose H^* by applying Theorem 5 into at most five pieces: the three (pinched) cycle interiors, and the interior and the exterior of the Jordan curve J. As in [5], we commit all the cycle edges to the solution (our first source of error). Then, different from the unweighted case [5], we have to *guess* $O(k \log(1/\varepsilon))$ edges of a near-optimal solution crossing J, for $n^{O(k \log(1/\varepsilon))}$ possibilities. Each piece now has $O(k \log(1/\varepsilon))$ "portal" vertices: the vertices along J together with the endpoints of the guessed edges. The portals of each piece form the interface between the pieces.

After each decomposition we apply a weighting scheme on the new portals (see [2, 5]), insuring that every piece of H^* in the recursive decomposition accumulates $O(k \log(1/\varepsilon))$ portals overall (new or old). Even with the new weights, each piece still has weight at most constant fraction (larger than $2/3$, say $5/6$) of its parent weight.

Then we define subproblems in each piece. The decomposition of H^* implicitly results in a decomposition of G. Thus, for each small piece H' with portal set P_H, we have a corresponding subgraph G' of G with some cycles contracted. A portal in H' is still a portal in G'. As in [5], we define external edge-connectivity types $t_{G'}$ of G', which describe how these portals may be connected outside this piece in the $(1+\varepsilon)$-approximate 2-ECSS S. Given a graph H with portal set P, its edge-connectivity type is essentially represented by a forest with portals as leaves. It can be obtained by contracting the internal (does not contain portals as internal vertices) paths and 2-edge-connected components. For a graph H, there are at most $2^{O(|P|)}$ types formed by the spanning subgraphs of H. For each external type of G', the problem is to find approximately a minimum weight subgraph of G' that is compatible with $t_{G'}$.

Note that by the way we construct H^*, Theorem 4 still holds for H'. So we can solve subproblems of G' by recursively decomposing H'. Given the solutions to the subproblems on each piece, we can then take the best compatible combination of solutions to find our approximate solution for G.

The approximation scheme for the 2-VCSS problem follows the same framework, and we only mention the differences. First, we redefine H^* as remarked at the end of Section 4. Second, the operations of obtaining the vertex-connectivity type of H are much more complex because cycle contraction is not always safe. We remedy this by using the same techniques as in [5].

5.3 Analysis

The error of our final solution comes from two sources. First, each time a face of H^* (or its pieces) is cut by a Jordan curve, we guess $O(\log(1/\varepsilon))$ crossing edges. If we guess these edges optimally

(they were edges in some original optimal S^*) then by Theorem 4 we may pay an additive error of at most $\varepsilon/2$ times the weight of these guessed edges. Summing over the entire assembly of a possible solution, the total of these errors is at most $(\varepsilon/2)$OPT.

The second source of error is the weight of all the cycles contracted and committed to our solution during the recursive calls. Remember we apply the separator theorem to the light spanner H^* and its subgraphs, so in each level the cost of the contracted cycles has cost at most $O(w(H^*)/k)$. Since each piece has weight at most constant fraction of its parent weight, the depth of the recursive calls is $O(\log n)$. So the total error is $O((w(H^*)/k)\log n)$, where $k = O((\log n)/\varepsilon)$ and $w(H^*) = O(OPT)$. By an appropriate choice of the leading constant defining k, this is at most $(\varepsilon/2)\cdot$OPT. Now accounting for both types of error, our final solution still has cost at most $(1+\varepsilon)$OPT.

If we follow a pure recursive approach (without dynamic programming), then the time is $T(n) \leq n^{O(k\log(1/\varepsilon))}T(5n/6)$, with solution $n^{O((1/\varepsilon)\log(1/\varepsilon)(\log^2 n))}$. We may improve this bound by dynamic programming and a more careful count of subproblems. Because of the portal weighting scheme, and the fact that we are trying many different portal arrangements within each piece, a piece may be separated in many different ways. However, Theorem 6 shows that a piece is partitioned in only $O(n^2)$ different ways, no matter how many portal arrangements we try. This implies:

Theorem 7 *The total number of distinct pieces (contracted subgraphs) of the original H^* that occur during our recursive decomposition is $n^{O(\log n)}$. Therefore the number of distinct subproblems (a piece, $p = O(k\log(1/\varepsilon))$ portals selected in the piece, and an external connectivity type on those portals) is $n^{O(\log n)}n^{O(p)}2^{O(p)} = n^{O(k\log(1/\varepsilon))}$.*

Now by dynamic programming with $n^{O(k\log(1/\varepsilon))}$ time per subproblem, we have:

Theorem 8 *Let $\varepsilon > 0$ and let G be a 2-EC weighted planar graph with n vertices. There is an algorithm running in time $n^{O((\log n)(\log(1/\varepsilon)/\varepsilon))}$ that outputs a 2-ECSS H of G such that $w(H) \leq (1+\varepsilon)\cdot$OPT.*

Using similar arguments as above and 2-VCSS techniques from [5], we have:

Theorem 9 *Let $\varepsilon > 0$, and let G be a 2-VC weighted planar graph with n vertices. There is an algorithm running in time $n^{O((\log n)(\log(1/\varepsilon)/\varepsilon))}$ that outputs a 2-VCSS H of G such that $w(H) \leq (1+\varepsilon)\cdot$OPT.*

6 Extensions to the $\{1,2\}$-Connectivity Problem in Planar Graphs

In this section, we extend our results to the $\{1,2\}$-connectivity problems in planar graphs. Again we focus on the algorithm for the $\{1,2\}$-ECSS problem only. The algorithm for the $\{1,2\}$-VCSS problem can be obtained similarly.

First consider the relaxed version of the $\{1,2\}$-ECSS problem, where we are allowed to duplicate edges. We call it the $\{1,2\}$-ECSSM problem. As in Section 2, we can show that there is a $(1+\varepsilon)$-approximate $\{1,2\}$-ECSSM that uses only edges from a light $(1+\varepsilon)$-spanner H. So instead of G, we can work in H with duplicated edges.

The main difference is the dynamic programming part. We need to redefine the connectivity types to reflect the non-uniform connectivity requirement. For this, we can combine our definitions and those used in [8]. Informally, the main difference is that each time we contract a 2-connected component or path, we assign the highest connectivity requirement among all contracted vertices

to the new vertex. This increases the number of types from $2^{O(|P|)}$ to $2^{O(2|P|)}$, where P is the set of portals in the given graph. We again obtain a PTAS with running time $n^{O(1/\varepsilon^2)}$.

Now consider the $\{1,2\}$-ECSS problem. We first find a 2-approximate solution A using algorithms from [16] (or [9] for $\{1,2\}$-VCSS). Then we augment A into a light spanner H^* as in Section 4. Using similar arguments as in the proof of Theorem 4, we can show that there is a $(1+\varepsilon)$-approximate $\{1,2\}$-ECSS S so that for each picked face-edge e, only $O(\log(1/\varepsilon))$ edges of S cross e. Now redefine the connectivity types as above. Finally, we use dynamic programming to solve the problem. The running time is still dominated by the number of subproblems $n^{O((\log n)(\log(1/\varepsilon)/\varepsilon)}$. Hence, we get a QPTAS in this case.

Our results in this section are summarized as follows.

Theorem 10 *Let $\varepsilon > 0$ and let G be a weighted planar graph with n vertices. There is an algorithm running in time $n^{O(1/\varepsilon^2)}$ that outputs a $\{1,2\}$-ECSSM of G whose weight is at most $(1+\varepsilon)\cdot OPT$.*

Theorem 11 *Let $\varepsilon > 0$ and let G be a weighted planar graph with n vertices. There is an algorithm running in time $n^{O((\log n)(\log(1/\varepsilon)/\varepsilon))}$ that outputs a $\{1,2\}$-ECSS H of G such that $w(H) \leq (1+\varepsilon)\cdot$ OPT.*

Theorem 12 *Let $\varepsilon > 0$, and let G be a weighted planar graph with n vertices. There is an algorithm running in time $n^{O((\log n)(\log(1/\varepsilon)/\varepsilon))}$ that outputs a $\{1,2\}$-VCSS H of G such that $w(H) \leq (1+\varepsilon)\cdot$ OPT.*

References

[1] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9:81–100, 1993.

[2] S. Arora, M. Grigni, D. Karger, P. Klein, and A. Woloszyn. A polynomial time approximation scheme for weighted planar graph TSP. *9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 33–41, 1998.

[3] J. Cheriyan and R. Thurimella. Approximating minimum-size k-connected spanning subgraphs via matching. *SIAM Journal on Computing*, 30(2):528-560, 2000.

[4] J. Cheriyan, S. Vempala, and A. Vetta. An approximation algorithm for the minimum-size k-vertex connected subgraph. *SIAM Journal on Computing*, 32(4):1050–1055, 2003.

[5] A. Czumaj, M. Grigni, P. Sissokho, and H. Zhao. Approximation schemes for minimum 2-edge-connected and biconnected subgraphs in planar graphs. *15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 489–498, 2004.

[6] A. Czumaj and A. Lingas. On approximability of the minimum-cost k-connected spanning subgraph problem. *10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 281–290, 1999.

[7] A. Czumaj and A. Lingas. Fast approximation schemes for Euclidean multi-connectivity problems. *27th Annual International Colloquium on Automata, Languages and Programming*, pp. 856–868, 2000.

[8] A. Czumaj, A. Lingas, and H. Zhao Polynomial-time approximation schemes for the Euclidean survivable network design problem. *29th Annual International Colloquium on Automata, Languages and Programming*, pp. 973–984, 2002.

[9] L. Fleischer. A 2-approximation for minimum cost $\{0, 1, 2\}$ vertex connectivity. *8th International Integer Programming and Combinatorial Optimization Conference*, pp. 115–129, 2001.

[10] H. N. Gabow. An ear decomposition approach to approximating the smallest 3-edge connected spanning subgraph of a multigraph. *13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 84–93, 2002.

[11] H. N. Gabow. Better performance bounds for finding the smallest k-edge connected spanning subgraph of a multigraph. *14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 460–469, 2003.

[12] H. N. Gabow, M. X. Goemans, and D. P. Williamson. An efficient approximation algorithm for the survivable network design problem. *Mathematical Programming B*, 82:13–40, 1998.

[13] M. Grötschel and C. L. Monma. Integer polyhedra arising from certain network design problems with connectivity constraints. *SIAM Journal on Discrete Mathematics*, 3(4):502–523, 1990.

[14] M. Grötschel, C. L. Monma, and M. Stoer. Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Operations Research*, 40(2):309–330, 1992.

[15] M. Grötschel, C. L. Monma, and M. Stoer. Design of survivable networks. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Handbooks in Operations Research and Management Science*, volume 7: Network Models, chapter 10, pp. 617–672. North-Holland, Amsterdam, 1995.

[16] K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[17] R. Jothi, B. Raghavachari, and S. Varadarajan. A 5/4-approximation algorithm for minimum 2-edge-connectivity. *14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 725–734, 2003.

[18] S. Khuller. Approximation algorithms for finding highly connected subgraphs. In D. S. Hochbaum, ed., *Approximation Algorithms for $\mathcal{NP}$-Hard Problems*, PWS Publishing Company, 1996.

[19] S. Khuller and U. Vishkin. Biconnectivity approximations and graph carvings. *Journal of the ACM*, 41(2):214–235, March 1994.

[20] G. Kortsarz and Z. Nutov. Approximation algorithm for k-node connected subgraphs via critical graphs. *36th Annual ACM Symposium on Theory of Computing*, pp. 138–145, 2004.

[21] P. Krysta. Approximating minimum size 1,2-connected networks, *Discrete Applied Mathematics*, 125:267–288, 2003.

[22] C. Levcopoulos, G. Narasimhan, and M. H. M. Smid. Efficient algorithms for constructing fault-tolerant geometric spanners. *30th Annual ACM Symposium on Theory of Computing*, pp. 186–195, 1998.

[23] C. L. Monma and D. F. Shallcross. Methods for designing communications networks with certain two-connected survivability constraints. *Operations Research*, 37(4):531–541, July 1989.

[24] M. Penn and H. Shasha-Krupnik. Improved approximation algorithms for weighted 2- and 3-vertex connectivity augmentation problems. *Journal of Algorithms*, 22(1):187–196, January 1997.

[25] S. B. Rao and W. D. Smith. Approximating geometrical graphs via "spanners" and "banyans". *30th Annual ACM Symposium on Theory of Computing*, pp. 540–550, 1998.

[26] M. Stoer. *Design of Survivable Networks*, volume 1531 of *Lect. Notes in Math.*, Springer-Verlag, 1992.

[27] S. Vempala and A. Vetta Factor 4/3 approximations for minimum 2-connected subgraphs. *3rd APPROX*, pp. 262–273, 2000.