Approximation Schemes for Minimum 2-Edge-Connected and Biconnected Subgraphs in Planar Graphs

Artur Czumaj* Michelangelo Grigni† Papa Sissokho‡ Hairong Zhao*

Abstract

Given an undirected graph, finding either a minimum 2-edge-connected spanning subgraph or a minimum 2-vertex-connected (biconnected) spanning subgraph is MaxSNP-hard. We show that for planar graphs, both problems have a polynomial time approximation scheme (PTAS) with running time $n^{O(1/\varepsilon)}$, where n is the graph size and ε is the relative error allowed.

When the planar graph has edge costs, we approximately solve the analogous min-cost subgraph problems in time $n^{O(\gamma/\varepsilon)}$, where γ is the ratio of the total edge cost to the optimum solution cost.

1 Introduction

Given an undirected graph G, the *minimum 2-edge-connectivity problem* is to find a spanning subgraph with a minimum number of edges which is 2-edge connected (remaining connected after removing any one edge). The *minimum biconnectivity problem* is similar, except that the subgraph must be 2-vertex-connected (biconnected, remaining connected after removing any one vertex). When G has edge costs, we define the *minimum-cost* 2-edge-connectivity and biconnectivity problems similarly, where now we minimize the total edge cost of the subgraph.

All these problems are NP-hard, even in the unweighted planar case. Furthermore they are MaxSNP-hard for general graphs [6, 10] and for bounded-degree graphs [7]. Much effort has been put into achieving polynomial-time constant approximation ratio algorithms, see e.g., [4, 5, 6, 11, 16]. In arbitrary unweighted graphs, the best currently known approximation ratios are $\frac{5}{4}$ for 2-edge-connectivity [11] and $\frac{4}{3}$ for biconnectivity [16].

We would prefer a *polynomial time approximation scheme* (PTAS): an algorithm taking an instance G and a positive ε, returning a solution with cost at most $(1+\varepsilon)$ times optimum, and running in time that is polynomial for each fixed ε. The MaxSNP-hardness results imply that there is no hope of a PTAS in general (unless P=NP), but a PTAS could still exist for special cases. Indeed, based on the framework of Arora [2], a PTAS was found [7, 8] for the problem of finding a minimum-cost k-vertex (or k-edge) connected spanning subgraph in complete Euclidean graphs in bounded dimension.

In this paper we consider the case that the input graph G is planar. We show a PTAS for the minimum 2-edge-connectivity problem and a PTAS for the minimum biconnectivity problem, both running in time $n^{O(1/\varepsilon)}$. More generally, when the planar graph has edge weights our algorithms approximately solve the minimum-cost problems in time $n^{O(\gamma/\varepsilon)}$, where γ is the ratio of the total edge cost to the optimum solution cost. This is a PTAS when γ is bounded; note that γ is bounded (by 3) when the edge weights are uniform.

Our general approach resembles the approximation schemes for metric-TSP in planar graphs [3, 12, 13]. We use a separator theorem, hierarchical decomposition, and dynamic programming. Our separator finds low-cost cycles in a planar graph so that after contracting those cycles (and committing their edges to our approximate solution), the remaining graph has a logarithmic size vertex separator. Using this, we recursively divide the input graph G into pieces, forming a decomposition tree $\mathcal{T}$ of logarithmic depth. Each piece has a logarithmic number of "portal" vertices connecting it to the rest of G. For each piece, we enumerate all the different ways that some subgraph of G (outside this piece) may influence the connectivity constraints within this piece. We call these the *external types* of the piece, and we show that the number of such types is a simple exponential in the number of portals. For each piece in $\mathcal{T}$ and for each external type, we must find a near minimum cost subgraph H of the piece, so that H together with the external type can meet the global connectivity constraints. We solve these problems by dynamic programming, working up $\mathcal{T}$ from the leaves to the root G.

*Department of Computer Science, New Jersey Institute of Technology, Newark NJ 07102, USA. Email: czumaj@cis.njit.edu, hairong@cis.njit.edu.

†This work was supported by NSF Grants CCR-9820931 and CCR-0208929. Department of Mathematics and Computer Science, Emory University, Atlanta GA 30322, USA. Email: mic@mathcs.emory.edu.

‡Department of Mathematics, Illinois State University, Normal IL 61790, USA. Email: psissok@ilstu.edu.

In the following, 2-EC denotes "2-edge-connected", 2-VC denotes "2-vertex-connected" (or biconnected), 2-ECSS denotes "2-edge-connected spanning subgraph", 2-VCSS denotes "2-vertex-connected spanning subgraph", and $c(H)$ denotes the total edge cost of a subgraph H. Our main results are summarized in the following two theorems.

THEOREM 1.1. *Let $\varepsilon > 0$, let G be a 2-EC planar graph with edge costs, and let* OPT *be the minimum cost of a 2-ECSS in G. There is an algorithm taking inputs G and ε, running in time $n^{O(c(G)/(\mathrm{OPT}\cdot\varepsilon))}$, and producing a 2-ECSS H in G such that $c(H) \le (1+\varepsilon)\cdot \mathrm{OPT}$.*

THEOREM 1.2. *Let $\varepsilon > 0$, let G be a 2-VC planar graph with edge costs, and let* OPT *be the minimum cost of a 2-VCSS in G. There is an algorithm taking inputs G and ε, running in time $n^{O(c(G)/(\mathrm{OPT}\cdot\varepsilon))}$, and producing a 2-VCSS H in G such that $c(H) \le (1+\varepsilon)\cdot \mathrm{OPT}$.*

As remarked above, each claimed algorithm is a PTAS when the ratio $\gamma = c(G)/\mathrm{OPT}$ is bounded. In particular, Theorems 1.1 and 1.2 imply a PTAS for the *unweighted* minimum 2-edge-connectivity and the minimum biconnectivity problems.

COROLLARY 1.1. *Let $\varepsilon > 0$, let G be a 2-EC planar graph, and let* OPT *be the minimum number of edges of a 2-ECSS in G. There is an algorithm taking inputs G and ε, running in time $n^{O(1/\varepsilon)}$, and producing a 2-ECSS H in G that has at most $(1+\varepsilon)\cdot$ OPT edges.*

COROLLARY 1.2. *Let $\varepsilon > 0$, let G be a 2-VC planar graph, and let* OPT *be the minimum number of edges of a 2-VCSS in G. There is an algorithm taking inputs G and ε, running in time $n^{O(1/\varepsilon)}$, and producing a 2-VCSS H in G that has at most $(1+\varepsilon)\cdot$ OPT edges.*

Our main technical contributions are the modified separator theorem, the notion of types (both 2-EC and 2-VC, each represented by small graphs), and the simple exponential bounds on the number of external types possible within a portal face. In this abstract we concentrate on Theorem 1.1, with only a sketch of the new ideas required for Theorem 1.2.

2 Cuts and k-EC Types

In this paper, graphs are undirected, without self-loops but possibly with parallel edges. Each edge e has a nonnegative cost c_e; a subgraph or minor H inherits edge costs from its parent graph, and $c(H)$ denotes the total edge cost of H. Suppose $G = (V, E)$ is a graph and S_1, S_2 are disjoint subsets of its vertex set $V = V(G)$. S_1 and S_2 are *separated* if there is no path in G from a vertex of S_1 to a vertex of S_2. An edge set $F \subseteq E$ *separates* S_1 and S_2 if they are separated in $G - F$; we also say that F is an *edge cut* for S_1 and S_2. Similarly, a *vertex cut* for S_1 and S_2 is some $U \subseteq V - (S_1 \cup S_2)$ such that S_1 and S_2 are separated in $G - U$. We let $\mathrm{Cut}^e_G(S_1, S_2)$ denote a minimum size edge cut, and $\mathrm{C}^e_G(S_1, S_2) = |\mathrm{Cut}^e_G(S_1, S_2)|$ is the *edge capacity* between S_1 and S_2. We define $\mathrm{Cut}^v_G(S_1, S_2)$ and $\mathrm{C}^v_G(S_1, S_2)$ similarly. We may efficiently compute such min cuts using max-flow. For $P \subseteq V$, we say that a (vertex or edge) cut *crosses* P if it separates some S_1 and S_2 such that $S_1 \cup S_2 = P$.

A *cycle* has no repeated vertex, but it may consist of two vertices joined by two parallel edges. For $e \in E$, G/e denotes the graph obtained by contracting e (that is, identifying its endpoints). If C is a cycle of G, G/C is the graph obtained by contracting the edges of C. After contraction we discard self-loops, but we retain parallel edges. A *minor* of graph G is a graph obtained from G through a series of such edge contractions and edge/vertex deletions.

DEFINITION 2.1. *A* bipartition *of a set P is a pair of nonempty subsets $\{S_1, S_2\}$ such that $S_1 \cup S_2 = P$ and $S_1 \cap S_2 = \emptyset$.*

Suppose G is a graph, $P \subseteq V(G)$, and k is a positive integer. The (k$-EC, P$)-type *of G is a table t indexed by bipartitions $\{S_1, S_2\}$ of P, and holding the values $t(S_1, S_2) = \min(k, \mathrm{C}^e_G(S_1, S_2))$.*

Suppose t_1 and t_2 are ($k-EC, P$)-types of two graphs sharing the vertex subset P; they are compatible *iff $t_1(S_1, S_2) + t_2(S_1, S_2) \ge k$ for all $\{S_1, S_2\}$.*

Intuitively, the (k$-EC, P$)-type describes how P is crossed by edge cuts using less than k edges. We are usually only interested in the type when G is ($k-EC, P$)*-safe*, meaning that all edge cuts of G not crossing P use at least k edges. The relevance of such types to the k-ECSS problem follows from this simple claim:

CLAIM 2.1. *Suppose H_1 and H_2 are graphs with disjoint edge sets, and $V(H_1) \cap V(H_2) = P$. Then $H_1 \cup H_2$ is k-EC iff:*

1. *H_1 is ($k-EC, P$)-safe,*
2. *H_2 is ($k-EC, P$)-safe, and*
3. *the ($k-EC, P$)-types of H_1 and H_2 are compatible.*

We may abbreviate the third condition above by saying that H_1 and H_2 (or H_1 and the type of H_2, or vice versa) are *compatible*.

Suppose G_1 and G_2 are edge disjoint graphs with $V(G_1) \cap V(G_2) = P$. To solve the k-ECSS problem in $G_1 \cup G_2$, it suffices to do the following:

1. For each possible type t_1 of a subgraph of G_1, find a min-cost (k$-EC, P$)-safe spanning subgraph H_2 of G_2 compatible with t_1.
2. For each possible type t_2 of a subgraph of G_2, find a min-cost (k$-EC, P$)-safe spanning subgraph H_1 of G_1 compatible with t_2.
3. Consider all pairs of H_1 from Step 1 and H_2 from Step 2. Return the min-cost compatible pair.

We will use a similar approach in our algorithm; in particular we need a polynomial bound on the number of distinct subgraph types considered. We may succinctly represent the (k$-EC, P$)-type of G by a smaller graph $t(G)$ which contains P and has the same type. In particular a minor of G contains P as long as we have neither deleted a vertex of P, nor contracted two vertices of P together.

In the special case of $k = 2$, we shall construct such a $t(G)$ from G by applying the following rules, until none apply:

1. If a cycle C has a chord (an edge $e \notin E(C)$ connecting two vertices of C), delete the chord.
2. If a cycle C has at most one vertex in P, contract C to a point.
3. If a vertex $v \notin P$ has degree 2, contract it with a neighbor.

The correctness of the above follows from the observation that all 0-edge cuts and 1-edge cuts of P are invariant under the above rules. In our application we need to consider the situation where G is embedded in a disk with the vertices of P on the boundary. In the next two lemmas we bound the size of $t(G)$, and we also bound the total number of possible (2$-EC, P$)-types induced by subgraphs of G.

LEMMA 2.1. *Suppose G is a (2$-EC, P$)-safe planar graph embedded in the disk, with the vertices of P on the disk boundary. Then $t(G)$ is a planar graph embedded in the same way, with $O(|P|)$ vertices.*

Proof. By considering the three rules used to form $t(G)$, we see that it is also a planar (2$-EC, P$)-safe graph embedded in the disk with P on the boundary. Every internal face f of $t(G)$ has at least two portals. If f has exactly two portals, we draw an arc e_f inside f between those two portals. If f has $d \geq 3$ portals, we draw a cycle of d arcs within f connecting the portals. These arcs form an outerplanar graph A on the portals.

We claim A has no parallel edges. Suppose instead that two portals $p, q \in P$ are connected by two parallel arcs a_1 and a_2, from faces f_1 and f_2. Since all faces must involve at least two portals, we can choose a_1 and a_2 consecutive at p, so that f_1 and f_2 share at least one edge. Now consider the part of $t(G)$ drawn between a_1 and a_2: it has no cycles (by rule 2) and is connected, so it is a tree. Because $t(G)$ is (2$-EC, P$)-safe, it is a path from p to q. By rule 3 it must be an edge directly between p and q. But then it is a chord between f_1 and f_2, so it should have been deleted by rule 1.

Therefore A is a simple outerplanar graph on vertex set P, so it has less than $2|P|$ arcs. Further if we add arcs from the outer faces of $t(G)$ (those faces bounded by a segment of the boundary), we still have at most three parallel arcs per pair of portals, therefore at most $6|P|$ arcs. For each $p \in P$, its degree in $t(G)$ is at most the number of adjacent arcs; therefore the sum of the degree of p in $t(G)$, over all $p \in P$, is at most $\ell = 12|P|$.

Now if we erase each portal and an infinitesimal neighborhood around it, the graph $t(G)$ is transformed into a forest (by rule 2) with ℓ leaves, and all internal vertices of degree at least 3 (by rule 3). Then $t(G)$ has less than ℓ vertices not in P, or in other words $t(G)$ has less than $13|P|$ vertices overall.

LEMMA 2.2. *With G and P embedded as in the previous lemma, the number of distinct (2$-EC, P$)-types defined by subgraphs of G is $2^{O(|P|)}$.*

Proof. Let H be a subgraph of G containing P. By trimming H, we can make it (2$-EC, P$)-safe without changing its (2$-EC, P$)-type. In the previous proof we saw that $t(H)$ can be described by a planar forest T with at most $12|P|$ leaves, internal vertices of degree at least three, and each leaf labeled by some $p \in P$, where the labels for a given p are on consecutive leaves. By standard tree counting techniques, there are $2^{O(|P|)}$ such graphs, and therefore at most that many distinct (2$-EC, P$)-types.

3 Planar Separators

We want to approximately solve the 2-ECSS problem in a planar graph G embedded on a sphere. If we can find a low-cost simple cycle C in G, then we may divide our problem into subproblems by contracting C. This follows from two observations:

FACT 3.1. *For any subgraph H of G containing C, H is a 2-ECSS in G iff H/C is a 2-ECSS in G/C. (This does not use planarity.)*

FACT 3.2. *When we contract C, the sphere pinches into two spheres kissing at the new contracted vertex (a cut-point). Therefore the 2-ECSS problem in G/C is equivalent to two disjoint 2-ECSS problems, one on each sphere.*

Therefore to approximately solve the 2-ECSS problem in G, we contract C and approximately solve the two

independent 2-ECSS subproblems. Then we lift the edges of those two solutions back to G and add the edges of C, to get a 2-ECSS in G. The additive error of this solution (the difference between its cost and the optimal cost) is at most the sum of the errors of the two subproblems plus $c(C)$.

We will not always be lucky enough to find a light cycle which does a good enough job of separating G, therefore we consider a more general kind of separator combining cycles with a *Jordan cut*: a Jordan cut of G is a closed Jordan curve in the embedding of G that does not cross (intersect the interior of) any edge. Given a Jordan cut J, every edge is either in the interior or the exterior of J, but those vertices and faces intersected by J are not counted in either the interior or the exterior of J.

The following theorem is a modification of Miller's planar separator theorem, as already used for the planar TSP [3, 12, 15]. Our main technical change is to use up to three "caps" of Miller's cycle tree (the root cap and at most two leaf caps) as contractible cycles. Due to space limitations, we omit the details here.

THEOREM 3.1. *Let G be a plane graph with n vertices, with non-negative weights on its vertices and faces, and non-negative costs on its edges. Let W be the total weight and let M be the total cost. Given parameter $k \geq 1$, in $O(n \log n)$ time we may find a subgraph F of G and a Jordan cut J (as described below) so that:*

1. *F is the union of at most three vertex-disjoint simple cycles (maybe none). Their total edge cost $c(F)$ is $O(M/k)$. The cycles are non-nesting, meaning we may choose an embedding of G so that their interiors are disjoint.*
2. *The interior of each C_i has weight at most $(2/3)W$.*
3. *Let G' be the embedded graph that results after we "pinch off" the interiors of the cycles (G' is G/F minus the interior parts from each cycle). Each new contracted vertex has weight 0.*
4. *J is a Jordan cut of G' passing through the new contracted vertices, and the interior and exterior of J each have weight at most $(2/3)W$.*
5. *Q, the set of vertices of G' on J, has size at most k.*

See Figure 1. When we apply the above theorem, we say Q is the set of new[1] *portal* vertices introduced by the separator. The original graph G has been divided into at most five parts of weight at most $(2/3)W$: the (up to three) pinched cycle interiors, the interior of J, and the exterior of J. We let G_1 denote Q together with the subgraph of G' interior to J, and we let G_2 denote Q together with the subgraph of G' exterior to J. In this way $G_1 \cup G_2 = G'$, $E(G_1) \cap E(G_2) = \emptyset$, and $V(G_1) \cap V(G_2) = Q$. In the inherited embedding of G_1 (or G_2) all vertices of Q appear on a single new face which we call a *portal face*; the old faces that intersected J are gone. When solving 2-ECSS in G, we will see that the "pinched off" cycle interiors become independent 2-ECSS problems, but the subproblems in G_1 and G_2 are dependent because they share Q.

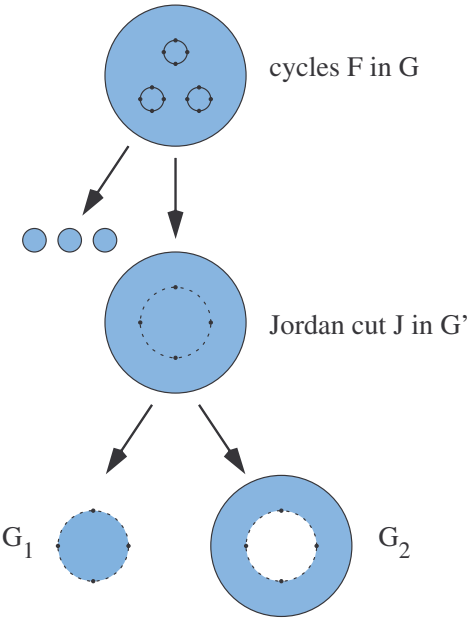


Figure 1: The separator theorem. The Jordan cut vertices become portals in G_1 and G_2.

4 The 2-ECSS Algorithm

We are given as input an embedded planar 2-EC graph G_0 with n vertices, non-negative edge costs, and a parameter $\varepsilon > 0$. We assign weight 1 to each vertex and weight 0 to each face. By existing approximation algorithms we estimate $\mathrm{OPT}(G_0)$, the minimum cost of a 2-ECSS, within a constant factor. We fix an integer $k = \Theta((\gamma/\varepsilon) \log n)$, where $\gamma = c(G_0)/\mathrm{OPT}(G_0)$.

We build a rooted decomposition tree $\mathcal{T}$ from G_0 as follows. Each node of $\mathcal{T}$ stores an embedded planar graph G, which has edge costs, vertex/face weights, and some distinguished subset P of "portal" vertices. The root of $\mathcal{T}$ stores G_0 itself, with no portals. Each node of $\mathcal{T}$ has at most five children, defined inductively as follows.

Let G be the graph stored at a node of $\mathcal{T}$, and let W

[1] We are reserving "P" to denote all portals in a graph, new or old.

be its total vertex/face weight. If W is $O(k^2)$, then this node is a leaf of $\mathcal{T}$. Otherwise, apply Theorem 3.1 to partition G into at most five pieces (up to three pinched cycle interiors and G_1, G_2), each of total weight at most $(2/3)W$. Uncontracted portal vertices from G remain as portals in each piece where they appear; the graphs G_1 and G_2 each get at most k new portals, the set Q. In G_1 and G_2, we assign a weight of $W/(12k)$ to each new portal, and weight $W/12$ to the new portal face. With these new weights, G_1 and G_2 still have total weight at most $(5/6)W$. By the separator properties, we see that each G in $\mathcal{T}$ is $(2-EC, P)$-safe, and that the tree $\mathcal{T}$ has depth $O(\log n)$ and size $O(n \log n)$.

By our construction P is the set of portals in G, which have been introduced by some Jordan cut but not yet cut off by a cycle contraction or another Jordan cut. It follows from our portal/hole weighting scheme [3, 12] that $|P|$ is $O(k)$, and G has $O(1)$ portal faces. Each portal face contains a hole made by a Jordan cut at some ancestor of G in $\mathcal{T}$. Note that a Jordan cut might cut (simply) across an existing portal face, in which case some old portals may appear on the new portal face, but this still counts as a single portal face. Or in terms of an embedding on a sphere with holes, all old holes crossed by the Jordan cut disappear, with segments of their boundaries incorporated into the one new hole boundary.

G is connected via P to the rest of G_0 (really a pinched and contracted version of G_0) which can be embedded as disjoint pieces, one in each portal face of G. Therefore the $(2-EC, P)$-type imposed on P by the rest of G_0 decomposes into independent types, one in each portal face of G. By applying Lemma 2.2 to each portal face, we may bound and enumerate the $2^{O(|P|)} = n^{O(\gamma/\varepsilon)}$ different $(2-EC, P)$-types that may be imposed on P by the rest of G_0. Call this list the list of *external types* for G. It is more efficient, although not essential for our purposes, if we represent each external type t of G as a planar graph of size $O(|P|)$ (see Lemma 2.1) embedded in the portal faces of G. In particular at the root of $\mathcal{T}$ the input graph G_0 has no portals, and therefore it has the "empty" external type t_0.

Having computed $\mathcal{T}$ and these external type lists, we may now define a set of subproblems that we want to approximately solve:

Definition 4.1. *For G in $\mathcal{T}$ (with portal set P) and an external type t for G, the* subproblem (G, t) *is this: find a min-cost $(2-EC, P)$-safe spanning subgraph H of G which is compatible with t, or else declare that (G, t) is infeasible (no such H).*

Checking feasibility is simple: just check whether t is compatible with G itself. The total number of subproblems (over all choices of G and t) is $n^{O(\gamma/\varepsilon)}$, and the subproblem (G_0, t_0) is our original 2-ECSS problem. We will approximately solve each subproblem starting at the leaves of $\mathcal{T}$ and finishing at the root. We use dynamic programming, storing our solutions to avoid recomputation.

In the base case, G is a leaf of $\mathcal{T}$ and has size $N = O(k^2)$. Then we apply an enumerative method based on Lipton-Tarjan separators [14] to exactly solve such subproblems in $2^{O(\sqrt{N})} = n^{O(\gamma/\varepsilon)}$ time (this may be regarded as a continuation of our method, using Jordan cuts without cycle contractions). We omit details.

Otherwise G is not a leaf, and has up to five children in $\mathcal{T}$ as found by Theorem 3.1. We have a specific external type t, which decomposes into an independent external type in each portal face of G.

For each child G_C of G which was pinched off in the interior of some cycle C, let t_C be the subtype of t induced by the portal faces inside C, and lookup our solution H_C to the subproblem (G_C, t_C). We lift the edges of H_C and the edges of C to be part of our approximate solution H for the (G, t) subproblem.

Now we must consider the two remaining children G_1 and G_2. As in Theorem 3.1, let G' be what is left of G after we contract the (up to three) cycles and pinch off their interiors; so $G_1 \cup G_2 = G'$. Let t' denote the external type induced by t in the portal faces of G'. Not knowing the optimal choice of external types t_1 and t_2 for G_1 and G_2, we try them all. That is, for every pair (t_1, t_2) where subproblems (G_1, t_1) and (G_2, t_2) were found feasible, we lookup their solutions H_1 and H_2 and check whether $H' = H_1 \cup H_2$ is compatible with t'. We take the cheapest compatible H' found, and lift its edges back to G. These edges of H', together with the C and H_C edges mentioned earlier, comprise our approximate solution H for the (G, t)-subproblem.

Although G' is not actually associated with a node of $\mathcal{T}$, note that we can still speak sensibly of the (G', t') subproblem as defined above. In fact it would be a simple matter to reformulate $\mathcal{T}$ as a binary tree including G': at each internal node of $\mathcal{T}$ we would either pinch one cycle, or apply a Jordan cut.

4.1 Analysis Our algorithm solves $n^{O(\gamma/\varepsilon)}$ subproblems, each in $n^{O(\gamma/\varepsilon)}$ time, so the total running time is $n^{O(\gamma/\varepsilon)}$.

Consider a feasible subproblem (G, t). By planarity, t decomposes into independent types in each portal face, and these faces cannot cross a cycle; therefore each cycle-pinched subproblem (G_C, t_C) and the remaining subproblem (G', t') are all feasible. Taking the external type on G_1 induced by $G_2 \cup t'$ as t_1, we see that (G_1, t_1)

is feasible. Supposing (by induction) that our algorithm found some solution H_1 for (G_1, t_1), then $H_1 \cup t'$ induces an external type t_2 on G_2 such that (G_2, t_2) is also feasible. Therefore by induction up $\mathcal{T}$, our algorithm finds some solution for each feasible (G, t)-subproblem.

Now suppose H is the solution our algorithm finds for a feasible subproblem (G, t). Define the *error* on (G, t) as the difference between the found cost $c(H)$ and the minimum possible cost. For each pinched cycle C (up to three), by Facts 3.1 and 3.2 subproblem (G, t) will inherit the error of (G_C, t_C) plus an additional additive error of at most $c(C)$.

After pinching cycles, the remaining error of (G, t) is that from (G', t'). Recall $G' = G_1 \cup G_2$; let H^* be the unknown optimal solution for (G', t'). Let t_1^* denote the external type of G_1 induced by $(H^* \cap G_2) \cup t'$, and similarly let t_2^* denote the external type of G_2 induced by $(H^* \cap G_1) \cup t'$. Then (G_i, t_i^*) has the optimal solution $H^* \cap G_i$ (for $i = 1, 2$), and (t_1^*, t_2^*) is a compatible type pair considered by our algorithm; if we solved these two subproblems optimally, our solution cost would be $c(H^*)$. Therefore our error on (G', t') is at most the sum of our errors on (G_1, t_1^*) and (G_2, t_2^*), even though we might not actually find our best solution H' using this pair. Therefore error terms simply add at a Jordan cut.

Therefore the total error of our root problem (G_0, t_0) is at most the sum of $c(C)$ over all cycles contracted in $\mathcal{T}$. We see that for any level of $\mathcal{T}$, the total edge cost of that level is at most $c(G_0)$, therefore the total edge cost of all cycles contracted on that level is $O(c(G_0)/k)$. Summing over all $O(\log n)$ levels of $\mathcal{T}$, the total error from all levels of $\mathcal{T}$ is $O((c(G_0)/k) \log n)$. By an appropriate choice of the leading constant defining k, this is at most $\varepsilon \cdot \mathrm{OPT}(G_0)$. Therefore our final solution has cost at most $(1+\varepsilon)\mathrm{OPT}(G_0)$, proving Theorem 1.1.

5 Sketch of the 2-VCSS Algorithm

Here we sketch the 2-VCSS algorithm claimed in Theorem 1.2. The main idea is similar to the 2-ECSS algorithm. Given the input plane graph G_0, we use the separator theorem to decompose G_0 hierarchically into small pieces. For each piece, we use types to enumerate the different ways that the "rest" of the graph may influence the connectivity constraints within this piece. Then, we use dynamic programming to approximately find the minimum subgraph compatible for each type of each piece. Similarly we can prove that the number of external types of each piece is a simple exponential in the number of portals (Lemma 5.2), yielding the same running time analysis. Again the only source of error is in the weight of the separating cycle edges, yielding the same error analysis.

However, there are some difficulties preventing us from using the same technique to solve the 2-VCSS problem. The principle difficulty is cycle contraction. In the 2-ECSS algorithm, the decomposition, type definition and dynamic programming are all performed on the minors of G_0, and we showed this is sufficient. But it is no longer reasonable to contract the cycles in the 2-VCSS problem, because this changes the problem (in particular, it may introduce a false cutvertex). Therefore for each node in $\mathcal{T}$, we keep a pair of graphs, the compressed subgraph G as defined in the 2-ECSS algorithm and its corresponding *uncompressed* subgraph $\mathbb{G}$, that is the subgraph of G_0 induced by all vertices which appear (after cycle contractions) as some vertex in G. As before, the separator theorem is still applied to G, and the decomposition of $\mathbb{G}$ is obtained naturally. But the type and dynamic programming will be defined on $\mathbb{G}$.

The uncompressed graph $\mathbb{G}$ also contains portal vertices and portal faces, which are the preimages of the portals and portal faces in G. Specifically, let J be the Jordan cut when we apply the separator theorem to G. Let p be a vertex on J. If p is mapped to a single vertex of $\mathbb{G}$, then p is a new portal that will be contained in the subgraphs of $\mathbb{G}$. Otherwise, p is mapped to many vertices which are on a series of cycles in $\mathbb{G}$, then at most 2 of these vertices will be specified as new portals of $\mathbb{G}$. These two vertices are where the Jordan cut J would intersect $\mathbb{G}$ if we uncontract the cycles represented by p. Thus we still have $O(k)$ portals in $\mathbb{G}$. Each portal appears on some portal face in $\mathbb{G}$ corresponding to the portal faces of G, and the portal faces identify where "the rest of G_0" would appear in our embedding. The edges of each separating cycle will appear in $\mathbb{G}$ as *hard edges* which are committed to our solution as in the 2-ECSS algorithm.

We must also redevelop the notion of types in the biconnected context, so that we may again use types to characterize the possible counterparts of an uncompressed graph; this is the point of Lemma 5.1 below. For convenience, we develop types simply as graphs (rather than abstract cut tables represented by graphs, as in Section 2). We begin with a definition analogous to "(k$-EC, P$)-safe" graphs.

DEFINITION 5.1. *For a graph $G = (V_G, E_G)$ with a portal set $P \subseteq V_G$, any graph $H = (V_H, E_H)$ for which $V_H \cap V_G \subseteq P$ is called a* counterpart *(of G). G is called* ($2-VC, P$)-safe *if it has a counterpart H such that $G \cup H$ is biconnected.*

5.1 Types of ($2-VC, P$)-safe planar graphs The type of a ($2-VC, P$)-safe graph is a *graph* describing a simplified characterization of the biconnectivity infor-

mation of the portals in the graph.

Definition of the type. Our definition of type is operational and it is oriented towards Lemmas 5.1 and 5.2 below. Let $H = (V_H, E_H)$ be any $(2{-}VC, P)$-safe graph (does not have to be planar) with a distinguished portal set P and a distinguished set of hard edges $E_{H,r}$. We define the type $t(H)$ of H by performing a series of operations on H. See Figure 2 for an illustration.

1. Let H_r be the graph consisting all portals of P and all the edges of $E_{H,r}$. We construct a simplified graph $\widehat{H_r}$ from H_r that is a forest formed by a collection of (possibly connected) stars[2]:
   (a) All vertices of H_r are included in $\widehat{H_r}$.
   (b) Every portal of P is marked as a *super-portal*.
   (c) For each block of H_r (hard block) with at least two vertices, create a new vertex in $\widehat{H_r}$ as a *super-portal* and connect by an edge the super-portal to every vertex in the block.
   (d) Mark all cutvertices and the end vertices of the isolated edges in H_r as super-portals in $\widehat{H_r}$.
   (e) Assign distinct IDs to all super-portals.
2. Replace the subgraph H_r of H by the graph $\widehat{H_r}$ and remove those vertices that have no ID and are adjacent only to a super-portal (notice that the removed vertices are all connected only to vertices of a hard block of H_r). Let the obtained graph be $\widehat{H}$.
3. For each block of $\widehat{H}$ that is not a bridge, if it has exactly two cutvertices and does not contain a super-portal, contract it into a single vertex.
4. Repeat the following two operations until none apply.
   (a) Remove all chords in any cycle of $\widehat{H}$.
   (b) For each path π of $\widehat{H}$ with no super-portals as internal vertices and all internal vertices with degree exactly 2, *contract* π to an edge with the same end vertices as π.
5. Let H' be the graph obtained after Steps 1–4. The type $t(H)$ is a graph with some vertices *labeled* (having IDs, these are vertices corresponding to super-portals and will be called portal nodes) and the other vertices not labeled that is constructed from H' as follows:
   (a) Add each cutvertex x in H' to $t(H)$, and refer to x as a cutvertex in $t(H)$. If it is a super-portal p in H', then it is a portal-node in $t(H)$ with the ID of p.
   (b) For each block in H' that does not contain any super-portals, contract it to a vertex in $t(H)$. We refer to this vertex in $t(H)$ as a *block-vertex*.
   (c) For each block in H' containing exactly one super-portal p, contract this block to a block-vertex. If the block is not a bridge and p is not a cutvertex in H', then this block-vertex is a portal node in $t(H)$ with the ID of p.
   (d) Connect a block-vertex by an edge in $t(H)$ to each cutvertex adjacent to it or to the endvertices of the bridge if the block-vertex is obtained from the bridge in H'.
   (e) If a block has two or more super-portals, then keep the block in $t(H)$ as it is in H'.
   (f) Finally, for each path π of $t(H)$ with no portal-nodes as internal vertices and all internal vertices with degree exactly 2, *contract* π to a single edge with the same end vertices as π.

Note the concept of *super-portal* can be applied to any graph H that contains portals and hard edges, and it is completely determined by the portal set and the hard edge set of H.

Properties of the types. We list now some basic properties of the types. First of all, the number of labels of $t(H)$ is identical to the number of super-portals of H. Secondly, let us notice that the type $t(H)$ is uniquely defined. This allows us to say that two $(2{-}VC, P)$-safe graphs H_1 and H_2 on the same vertex set and with the same set of portals and hard edges have *the same type* if $t(H_1)$ and $t(H_2)$ are identical with respect to their IDs. Thirdly, all vertices that are not portal-nodes have degree at least 3. Next, we notice that if H is biconnected and the portal set and hard edges are empty, then $t(H)$ contains a single vertex (block-vertex). Finally, if we consider the subgraph of H induced by non-portals and the corresponding subgraph in $t(H)$, we only removed the cutvertices of H that are internal "vertices" on "paths" consisting only degree 2 internal vertices[3]. Thus, the connections among portals in H are represented by the connections of portal nodes in $t(H)$. This property can be summarized in the following lemma.

LEMMA 5.1. *Let G and H be two $(2{-}VC, P)$-safe graphs that have the same set of portals P and the same*

[2]Notice a similarity of this construction to the standard construction of cutvertex-trees, see, e.g., [9].

[3]Some vertices are in fact blocks with two cutvertices.

set of hard edges E_r. Let $t(H)$ be the type of H. Then for any spanning subgraph G' of G that contains all hard edges in E_r, $t(G')$ and $t(H)$ have the same set of portal-nodes with respect to IDs, and $G' \cup H$ is biconnected if and only if $t(G') \cup t(H)$ is 2-edge connected and the cutvertices of $t(G') \cup t(H)$ are block-vertices of $t(H)$ or $t(G')$ or the super-portals resulting from contraction of hard blocks as defined in the type.

Next, we consider the number of types. Specifically, we consider the number of *external types* of $(2{-}VC, P)$-safe *plane* graphs. Let G be a $(2{-}VC, P)$-safe plane graph with a hard edge set E_r and a portal face set F. Let the super-portal set of G determined by P and E_r be P_s. If H be a counterpart of G that can be embedded in F, then we say the type $t(H)$ of H is an external type of G with respect to F. We will focus on the counterpart H of G that also is $(2{-}VC, P)$-safe and shares the same set of super-portals P_s (i.e., H and G have the same set of portals and hard edges). Then, the type $t(H)$ is called an external type of G with respect to P_s and F. Our goal is to show that the number of external types of G with respect to P_s and F is small, that is, it is only exponential in $|P_s|$. Now we state the main lemma bounding the number of types; we omit the proof in this abstract.

LEMMA 5.2. *Let $G = (V_G, E_G)$ be a $(2{-}VC, P)$-safe plane graph with a hard edges set E_r, a super-portal set P_s and a portal face set F. G is embedded in a way such that all super-portals of P_s will be on the border of the portal faces if we replace the graph G_r consisting of P and E_r with $\widehat{G_r}$ as in the definition of type. Then the number of external types of G with respect to P_s and F induced by $(2{-}VC, P)$-safe graphs is at most $2^{O(|P_s|)}$.*

Let $(\mathbb{H}, H)$ be a pair stored in a node of $\mathcal{T}$ where $\mathbb{H}$ is the uncompressed $(2{-}VC, P)$-safe graph. Suppose $\mathbb{H}$ has portal set P and hard edge set E_r. Let t be an external type of $\mathbb{H}$. We say $\mathbb{H}$ is compatible with t if $t(\mathbb{H}) \cup t$ is 2-edge connected and the cutvertices of $t(\mathbb{H}) \cup t$ are block-vertices of $t(\mathbb{H})$ or t or super-portals representing hard blocks. As in the 2-ECSS algorithm, we define the subproblem instance to be for $\mathbb{H}$ and a compatible external type t finding a min-cost $(2{-}VC, P)$-safe subgraph of $\mathbb{H}$ which is compatible with t.

By Lemma 5.2, the number of possible external types is determined by the number of super-portals of $\mathbb{H}$. We now show the number of super-portals in each subgraph obtained from separator theorem is $O(k)$. By definition, the number of super-portals of $\mathbb{H}$ is at most the number of portals in it plus the number of separating cycles (hard cycles) and the cutvertices between these cycles. Because the depth of the decomposition tree is at most $O(\log n)$ and at each node we introduce at most 3 "cycles" [4], the number of hard "cycles" is at most $O(\log n)$. Hence the number of super-portals of $\mathbb{H}$ is $O(k) + O(\log n) = O(k)$. By Lemma 5.2, $\mathbb{H}$ has at most $2^{O(k)} = n^{O(\gamma/\varepsilon)}$ external types.

6 Concluding Remarks

A deficiency of our algorithms is their dependence on the ratio $\gamma = c(G)/\mathrm{OPT}$. We know of no hardness result justifying this dependency, and indeed a very similar dependency in the context of the TSP was eliminated using a "spanner" construction [1, 3, 13] which safely prunes some heavy edges out of G.

Another direction of research is to extend our methods to similar problems, such as the 3-ECSS problem in planar graphs, or the 2-ECSS and 2-VCSS problems in more general graph families. We remark on some of these problems below; in all cases, progress depends on the development of appropriate separator theorems.

Planar 3-ECSS: Here we need to generalize Lemma 2.2; let us remark that counting planar Gomory-Hu trees is not sufficient. Also cycle contraction no longer works here, because the cycle vertices are not 3-edge connected. However we can modify our separator theorem to replace cycles with *bicycles*: a bicycle is two nested cycles, with all in-between faces visible from one of the two cycles. The connecting endpoints on the cycles are 3-edge connected, and we may safely contract them to a cut-point. The bicycle edges surviving this contraction are committed to the solution by resetting their costs to zero.

Forbidden minor 2-ECSS (or 2-VCSS): Here we have two difficulties: first, the usual separator theorems do not produce cycles (just trees). Also the appropriate generalization of Lemma 2.2 may require a careful application of the Robertson-Seymour theory, which appears difficult. For the special case of bounded-genus graphs, both of these issues look more tractable.

References

[1] I. Althöfer, G. Das, D. P. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete Comput. Geom.*, 9(1):81–100, 1993. An early version appeared in SWAT'90, LNCS V. 447.

[2] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5): 753–782, 1998.

[4] They may not be simple cycles in $\mathbb{H}$, but be glued with a set of other cycles from previous recursive calls.

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

[4] J. Cheriyan, A. Sebö, and Z. Szigeti. An improved approximation algorithm for minimum size 2-edge connected spanning subgraphs. *Proc. 6th IPCO, LNCS*, 1412:126–136, 1998.

[5] J. Cheriyan, S. Vempala, and A. Vetta. Approximation algorithms for minimum-cost k-vertex connected subgraphs. *Proc. 34th ACM Symposium on Theory of Computing*, pages 306–312, 2002.

[6] B. Csaba, M. Karpinski, and P. Krysta. Approximability of dense sparse instances of minimum 2-connectivity, TSP and path problems. *Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 74–83, 2002.

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

[8] A. Czumaj and A. Lingas. Fast approximation schemes for Euclidean minimum-cost multi-connectivity. *Proc. 27th Annual Intl. Colloq. on Automata, Languages, and Programming (ICALP), LNCS*, 1853:856–868, 2000.

[9] R. Diestel. Graph theory. Springer-Verlag, New York, 2000.

[10] C. G. Fernandes. A better approximation ratio for the minimum size k-edge-connected spanning subgraph problem. *Journal of Algorithms*, 28:105–124, 1988.

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

[12] M. Grigni, E. Koutsoupias, and C. Papadimitriou. An approximation scheme for planar graph TSP. *Proc. 36th IEEE Symposium on Foundations of Computer Science*, pages 640–645, 1995.

[13] M. Grigni and P. Sissokho. Light spanners and approximate TSP in weighted graphs with forbidden minors. *Proc 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 852–857, 2002.

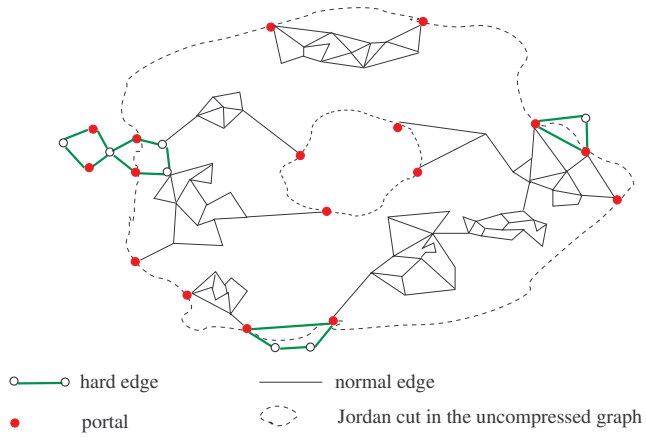
[14] R. Lipton and R. Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, 1980.

[15] G. L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *Journal of Computer and System Sciences*, 32:265–279, 1986.

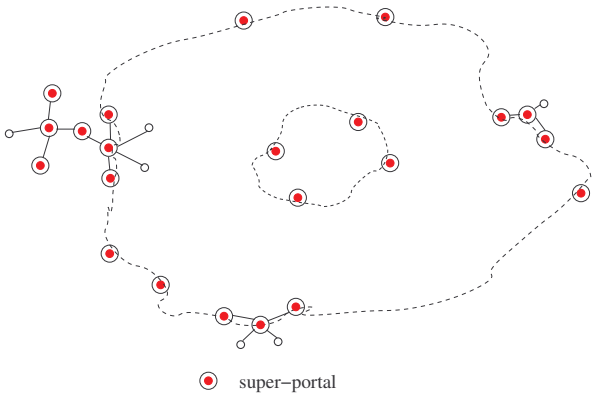
[16] S. Vempala and A. Vetta. Factor 4/3-approximations for minimum 2-connected subgraphs. *Proc. 3rd Workshop APPROX, LNCS*, 1913:262–273, 2000.

A Figures illustrating the type of a $(2-VC, P)$-safe graph

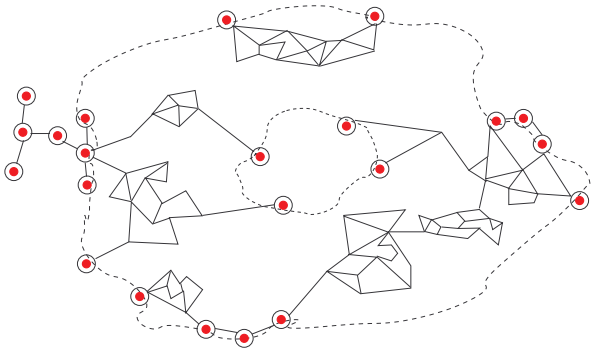
Figure 2: Type of a $(2-VC, P)$-safe graph used in the 2-VCSS Algorithm



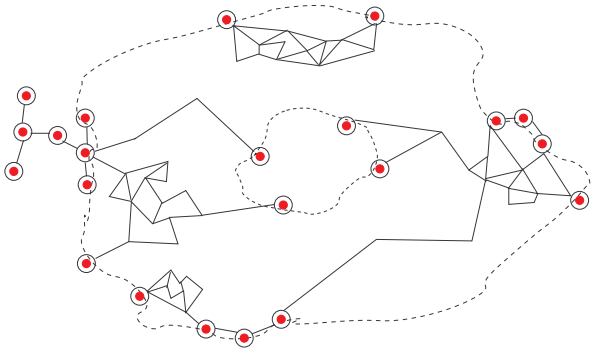
(a) A $(2-VC, P)$-safe graph H



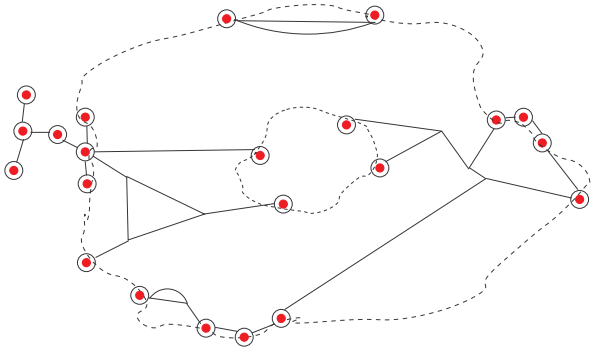
(b) $\widehat{H_r}$ obtained after step 1



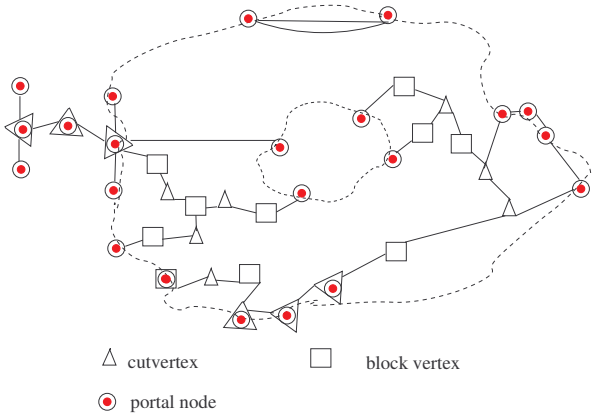
(c) $\widehat{H}$ obtained after step 2



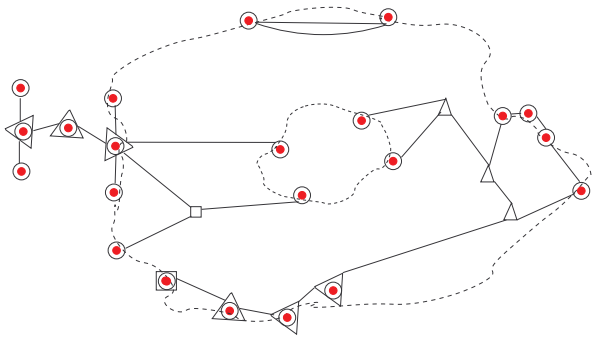
(d) $\widehat{H}$ after step 3



(e) H' obtained after step 4



(f) graph obtained after step 5a-5f



(g) the type $t(H)$ of H