A Polynomial-time Approximation Scheme for Planar Multiway Cut

MohammadHossein Bateni * MohammadTaghi Hajiaghayi † Philip N. Klein ‡ Claire Mathieu §

Abstract

Given an undirected graph with edge lengths and a subset of nodes (called the *terminals*), a *multiway cut* (also called a *multi-terminal cut*) problem asks for a subset of edges with minimum total length and whose removal disconnects each terminal from the others. It generalizes the min st-cut problem but is NP-hard for planar graphs and APX-hard for general graphs. We give a polynomial-time approximation scheme for this problem on planar graphs. We prove the result by building a novel "spanner" for multiway cut on planar graphs which is of independent interest.

*Department of Computer Science, Princeton University, Princeton, NJ 08540; Email: mbateni@cs.princeton.edu. Part of the work was done while the author was a summer intern at Microsoft Research, New England. The author is also with Center for Computational Intractability, Princeton, NJ 08540.

†Computer Science Department, University of Maryland, College Park, MD 20742; Email: hajiagha@cs.umd.edu.

‡Computer Science Department, Brown University, Providence, RI 02912; Email: klein@cs.brown.edu.

§Computer Science Department, Brown University, Providence, RI 02912; Email: claire@cs.brown.edu.

1 Introduction

The *multiway cut problem* (a.k.a. *multi-terminal cut problem*) is a classic problem of combinatorial optimization. Given an undirected graph with edge lengths and a subset of nodes called the *terminals*, the goal is to find a subset of edges with minimum total length whose removal disconnects each terminal from the others. It is a natural generalization of the problem of finding a minimum-length st-cut. A variant of multiway cut was first proposed in T. C. Hu's 1969 book [23]. Many applications proposed for multiway cut include image processing, chip design and parallel and distributed computing.

When the set of terminals has cardinality k, it is sometimes called the *k-terminal cut problem* or the *k-way cut problem*. The study of its computational complexity was inaugurated in 1983 by Dahlhaus, Johnson, Papadimitriou, Seymour, and Yannakakis [14][1]. Their results have guided the agenda for subsequent research:

1. For planar graphs, the problem can be solved in polynomial time for fixed k but is NP-hard when k is unbounded.
2. For general graphs, there is a simple 2-approximation algorithm disconnecting each terminal from the others by a min-cut, but for any fixed $k \geq 3$, the problem is APX-hard.

Result 1 spawned many papers giving improved running times for the case of planar graphs and fixed k; see, e.g., [22, 10, 24, 4]. Polyhedral work, e.g. [13, 11], addresses branch-and-cut methods for the problem.

Result 2 led to approximation algorithms with improved approximation ratios. Using a geometric relaxation, Calinescu, Karloff and Rabani [9] achieved an approximation factor $3/2$. Karger, Klein, Stein, Thorup and Young [25] obtained an improved—and currently the best—approximation factor 1.3438 for general k, and better factors for specific k, including $k = 3$. Cunningham and Tang [12] independently achieved the same result for $k = 3$.

By APX-hardness, no polynomial-time approximation scheme exists for general graphs if $P \neq NP$. However, for dense graphs with unit-length edges, there is a polynomial-time approximation scheme [1, 18]. Finally a generalization of multiway cut called *multicut* in which we want to disconnect only a set of k given pairs (instead of all pairs) has also been studied extensively. The current best approximation factor for multicut on general graphs is $O(\log k)$ [19] and for planar graphs is $O(1)$ [29], though the problem even on unweighted trees of height one (stars) is APX-hard [20].

The above results suggest a natural question : is there a polynomial-time approximation scheme (PTAS) for multiway cut on planar graphs? In this paper, we answer in the affirmative.

Theorem 1. *There is a polynomial-time approximation scheme for the multiway cut problem on planar graphs. Its runtime is $O(f(\epsilon)n^c)$, where $f(\epsilon)$ is a function of ϵ independent of n and c is an absolute constant independent of ϵ.*

We observe that the special case of multiway cut on planar graphs in which all terminals are on the outside face can be reduced (see Figure 1) via planar duality to the *Steiner tree problem on planar graphs*. In 2007 Borradaile, Klein and Mathieu [6] obtained a PTAS for that problem using a technique called *brick decomposition*. However, when terminals are not necessarily on a common face of a planar graph, an optimal multiway cut can have several connected components in the dual. Steiner tree techniques alone will therefore not suffice. Bateni, Hajiaghayi and Marx [3] recently generalized [6] to obtain a PTAS for *Steiner forest in planar graphs* using a new primal-dual technique called *prize-collecting clustering*.

Our new algorithm builds on the brick decomposition from [6], the prize-collecting clustering from [3], and techniques for finding short cycles enclosing prescribed amounts of weight from a paper by Park and Phillips [28].

In the interest of space, several proofs and figures appear in the appendix.

2 Overview

We first give a high-level overview of the techniques. Let (G_{in}, T_{in}) be the input instance of multiway cut. The details of the approach as well as a formal description of our algorithm $\texttt{Solve}(G_{in}, T_{in})$ can be found

[1] The work was first known in an unpublished but widely circulated extended abstract. Their complete paper was published in 1994.

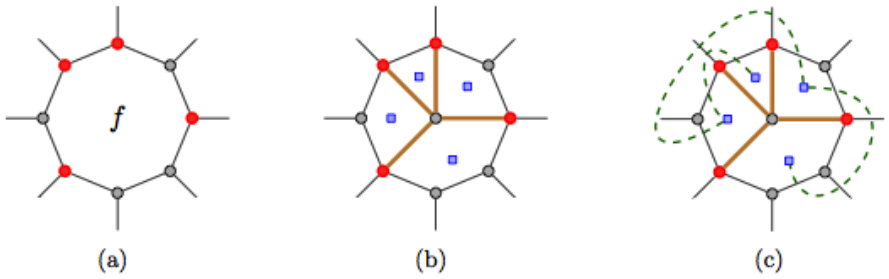

Figure 1: Reduction to Steiner tree if all terminals lie on one face. (a) shows a planar multiway cut instance with all terminals lie on one face f. Terminals are in red, Steiner vertices are in black. (b) We add a vertex at the center of f and connect it to the terminals by edges of infinite length. This creates four new faces denoted by blue squares. The Steiner tree instance, defined on the dual graph, asks to connect these faces to each other. (c) shows one such solution with dashed edges. The primal edges corresponding to the dashed edges cut all the terminals of the original instance from one another.

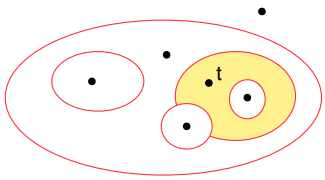

Figure 2: The multiway cut solution may not form a single connected component in the dual graph G^*. Here the terminals are depicted by black dots, and the optimal solution is shown as red cycles in the dual graph. Notice that even the portion serving one particular terminal may form more than one connected component in the dual. For instance, the region corresponding to the terminal t, painted yellow above, does not have a connected boundary.

in the next sections.

In the easy special case where all the terminals lie on one face of G_{in}, the simplicity of this case can be explained, perhaps, by the fact that there exists an optimal solution whose edges are *connected* in G^*. If terminals lie on different faces, the optimal solution may be *not connected* in the dual (see Figure 2). Moreover, the subset of edges "serving" a terminal need not be connected; it may consist of several disjoint cycles in G^*. This makes the problem much more difficult than the Steiner forest problem.

Let $\mathsf{OPT}(G_{in}, T_{in})$ denote the length of minimum multiway cut for the terminals T_{in} in G_{in}. The core of the algorithm tries to construct a *spanner* that has total length $\epsilon^{-c}\mathsf{OPT}(G_{in}, T_{in})$, for some constant c, but also includes a $(1+\epsilon)$-approximate multiway cut.

Theorem 2. *There is a universal constant c such that for any constant $\epsilon > 0$, we can construct in polynomial time a spanner of length $\epsilon^{-c}\mathsf{OPT}$ which contains a multiway cut solution of length at most $(1+\epsilon)\mathsf{OPT}$.*

Proving Theorem 1 given Theorem 2 In one sentence, starting from the spanner we incorporate a Baker-like technique [2] to reduce the problem to a bounded-treewidth graph that can then be solved using standard techniques. Since this part of the argument—that having a spanner implies Theorem 1—is by now well-understood (see e.g. [27, 15, 16]), the rest of the proof of Theorem 1 follows from Theorem 2. Refer to Appendix A.1 for more details.

Outline of the proof of Theorem 2 We focus on proving Theorem 2 in the rest of the paper.

First we construct a graph called the *skeleton* that guides the algorithm in construction of the spanner. The construction of the skeleton, and then of the spanner from the skeleton, consist of several steps, briefly described here; the details can be found in the appropriate sections.

- Each terminal is assigned a weight that is equal to the length of the minimum cut separating it from the other terminals. The sum of these weights, denoted by W, is at most $2\,\mathsf{OPT}(G_{in}, T_{in})$; see Lemmas 4 and 5. Throughout the paper, the analysis is done by charging to weights of terminals.
- We remove some edges to break (G_{in}, T_{in}) into disjoint instances such that, in each instance, the length of any cut $\delta(S)$ is at least ϵ times the sum of the weights of all the terminals in S; see Lemma 6 and Corollary 8. This is achieved by a greedy algorithm, as explained in Section 4.

- For each terminal t, we select several cycles in the planar dual (cuts in the primal) that enclose t, so that for each terminal t, at least one of the cycles selected for t intersects the component that immediately encloses t in a near-optimal solution. Refer to Theorem 10 for the analysis of the procedure described in Section 6.
- We next add several *ears* to each face of each component of the skeleton so that each resulting face contains at most one terminal that is relevant for that component. The connected components that this step produces are called *blobs*. This step is explained in Section 6.
- We run the prize-collecting clustering procedure to augment a graph formed from the union of the blobs. This procedure is explained in Section 5 and is invoked in Section 6. The connected components that this step produces are called *clusters*. The main property is that the near-optimal solution (almost) does not connect different components of the resulting skeleton The detailed properties, summarized in Theorem 9, are used in Section 7 and again in the proof of Theorem 21. Although we do not know of a direct reduction, we invoke one piece of the Steiner forest algorithm of [3], i.e., the prize-collecting clustering procedure, in this part of our algorithm. This defines the spanner.
- For each cluster, we build a brick decomposition starting from that cluster and add it in.
- Finally, for each brick, we guess the relevant part of the near-optimal solution by exhaustive search (there are only a constant number of possibilities) and add it in (see Section 8). This defines the spanner.

3 Background

3.1 Terminology

We use $G[V']$ to denote the subgraph of G induced by a subset V' of the vertex set of G. The *cut* $\delta(S)$ of a connected graph G defined by a set S of vertices is the set of edges having one endpoint in S and one endpoint not in S. The cut is called *simple* if $G[S]$ and $G[V - S]$ are both connected graphs.

For a connected planar embedded graph G, one face is designated as f_∞, the infinite face. A simple cycle C in G *encloses* a face f if (in the dual) C separates f from f_∞. It encloses an edge e if it encloses a face having e on its boundary. It *strictly* encloses e if in addition e is not part of C. More generally, a subgraph H encloses a face or edge if H has a cycle that encloses the face or edge. The *outer boundary* of H is the set of edges that are part of H and not strictly enclosed by H. We say a cycle *properly encloses* a subgraph if the cycle encloses the subgraph and shares no vertices with the subgraph. We say a subgraph is *properly in* a face if the subgraph is in the face and shares no vertices with the face. We say a subgraph A *encloses* a subgraph or face if some cycle of A encloses the subgraph or face.

3.2 Finding short cycles and paths

Using the transfer-function technique of [28], we can obtain efficient algorithms for the following computational problems. In each instance, the input includes a planar graph G with edge lengths length$(\cdot)$ and integer face weights weight$(\cdot)$.

1. Find a minimally enclosing simple cycle C such that length$(C) \leq \epsilon$ weight(faces enclosed by C).
2. Given in addition a face f, a weight interval $[\ell, u)$ and a length threshold τ, find a maximally enclosing cycle C that has length at most τ, that encloses f, and that encloses an amount of weight in the interval $[\ell, u)$.
3. Given in addition a cycle C and a set F of faces enclosed by C, find a shortest path enclosed by C, starting and ending on C, that separates some given pair of faces in F.

Theorem 3. *For each of the problems above, there is an algorithm whose running time is polynomial in the size of G and the maximum weight.*

For a proof, see Appendix B. Note that Theorem 3 assumes the weights are integers, and that the running time of the subroutines depends on the largest weight. In order to use these subroutines in our algorithm, we make sure to define weights (approximately equal to minimum cuts) so that they are all integer multiples of a number η, and that the largest weight is at most $\epsilon^{-1}\eta$ times the number of terminals. It follows that the running time of each subroutine is polynomial in the size of the graph.

4 The preprocessing phase

Let (G_{in}, T_{in}) denote the input instance. For each terminal t, let mincut(t) denote a minimum cut separating that terminal from $T_{in} - \{t\}$. The main algorithm first assigns each terminal t a *weight*, denoted by weight(t), which is essentially equal to the value of mincut(t); the value is rounded so that all scaled weights are small integers, thus enabling the algorithms of Appendix B to run in polynomial time. The weight of a nonterminal is zero. For a set S of vertices, weight(S) is defined to be $\sum_{v \in S}$ weight(v).

The algorithm then repeatedly finds short simple cuts, and uses them to cut up the graph into pieces for which the multiway cut problem is solved independently using Algorithm `SolveSpecial`, to be described later.

The main step is finding a minimal set S of vertices such that $\delta(S)$ is small with respect to weight(S). We can restrict our attention to sets S such that S is connected and $V(G) - S$ is connected. In this case, $\delta(S)$ is a simple cycle in the planar dual. Therefore this step can use the algorithm outlined in Appendix B.

Lemma 4 (Dahlhaus et al. [14])**.** $\sum_t$ length(mincut(t)) $\leq 2\,OPT(G_{in}, T_{in})$.

The proof of the following lemma follows from the definition of weight(t).

Lemma 5. $\sum_t$ length(mincut(t)) $\leq \sum_t$ weight$(t) \leq (1+\epsilon) \sum_t$ length(mincut(t)).

Lemma 6. *The cuts $\delta(S)$ added to M in Step * of* `Solve` *have total length at most* $2\epsilon(1+\epsilon)\,\mathsf{OPT}(G_{in}, T_{in})$.

The step marked * invokes `SolveSpecial` on a graph obtained by contracting all vertices in $V(G) - S$ to a single supervertex, which is not designated a terminal.

Lemma 6 implies that if `SolveSpecial`$(G', \hat{v}', T')$ returns a $(1 + c_2\epsilon)$ approximate solution for the multiway-cut instance (G', T'), then `Solve` returns a $(1 + (c_2 + 2(1+\epsilon))\epsilon)$-approximate solution. What have we gained? `SolveSpecial` takes as input instances that have additional structure captured by the following lemma.

Lemma 7. *Consider an invocation* `SolveSpecial`$(G', \hat{v}', T')$ *in Step * or Step ‡ of* `Solve`*. For any proper subset X of $V(G') - \{\hat{v}'\}$, we have* length$(\delta(X)) >$ ϵ weight(X).

```
Solve(G_in, T_in):
Input: planar instance (G_in, T_in) of multiway cut
Output: multiway cut of cost (1 + O(ε))OPT(G_in, T_in)

    for each terminal t, compute mincut(t).
    let η := ε Σ_{t∈T_in} length(mincut(t))/|T_in|
    for each terminal t, let weight(t) := smallest multiple of η that is ≥ length(mincut(t))
    for each nonterminal v, let weight(v) := 0

    select a nonterminal vertex v̂
    initialize (G, T) := (G_in, T_in) and M := ∅
    while possible
        find a minimal set S of vertices such that v̂ ∉ S and length(δ(S)) ≤ ε weight(S)
        let G' := G/(V(G) − S) be the graph obtained by merging all vertices of V(G) − S
        into a single vertex v̂'.
*       M := M ∪ δ(S) ∪ SolveSpecial(G', v̂', T ∩ S)
        delete S from G and T
‡   M := M ∪ SolveSpecial(G, v̂, T)
    return M

SolveSpecial(G, v̂, T):
Input: instance (G, T) such that length(δ(X)) > ε weight(X) for every X ⊂ V(G) − {v̂}
Output: multiway cut of cost at most (1 + c_2ε)OPT(G, T)
    H := BuildSkeleton(G, v̂, T)
    H' := BuildSpanner(G, v̂, H, T)
    Reduce the problem to a bounded-treewidth instance
    Use dynamic programming on the bounded-treewidth instance
```

The rest of the paper describes `SolveSpecial`$(G, \hat{v}, T)$. The algorithm is best described as operating on the planar dual G^* of G. The vertex $\hat{v}$ of G is the infinite face of G^*. Each simple cycle C in G^* partitions the faces of G^* into two sets according to which side of the cycle the faces lie. As defined in subsection 3.1, the set that does not contain $\hat{v}$ is the set of *enclosed* faces. We use e to denote both an edge in G and the corresponding dual edge in G^*. The terminals of the primal are faces of the dual, so weight$(\cdot)$ is assigned to the faces of the dual graph. Lemma 7, in the dual graph G^*, implies the following.

Corollary 8. *Every simple cycle C of G^* has length at least ϵ times the weight enclosed by C unless C is the boundary of the infinite face.*

5 Prize-collecting clustering

In our algorithm, we use the `PC-Clustering` algorithm of Bateni et al. [3] as a subroutine. Theorem 9 summarizes its guarantees, which we will use in the following way. There is a cost weight(v) associated with *ignoring* each vertex v. We let $\phi(v) = \epsilon^{-2}\text{weight}(v)$ for all vertices v, invoke the procedure, and apply the theorem on a near-optimal solution L to obtain Q. Then, the total cost of Q is $\sum_{v\in Q} \text{weight}(v) \leq \epsilon^2\text{length}(L) \approx \epsilon^2\mathsf{OPT}$. Thus, all these vertices are going to be ignored from consideration, paying only a negligible cost.

Theorem 9 (prize-collecting)**.** *Let G be a graph with edge lengths such that each vertex v has a potential $\phi(v)$, and let H be the subgraph of G output by the `PC-Clustering` algorithm executed on (G, ϕ). Then*

1. length$(H) \leq 2\sum_v \phi(v)$.
2. *For any subgraph L of G, there is a set Q of vertices such that*
   1. *$\sum_{v\in Q} \phi(v) \leq$ length(L); and*
   2. *If two vertices $v_1, v_2 \notin Q$ are connected by L, they are in the same connected component of H.*

6 Building a skeleton (Algorithm `BuildSkeleton`)

First we give the algorithm for building the skeleton, then we give the structural properties achieved by the construction in Theorem 10, Theorem 15, Lemma 28, and Corollary 17.

6.1 The *skeleton* algorithm

Recomputing weights. The min-cuts of the terminals might be smaller in this instance than in the original instance (since from the original graph, we only contract edges in the primal and thus delete them in the dual to reach to each instance), so the algorithm recomputes the min-cuts and reassigns the weights. Since the weights cannot increase, the resulting weights still satisfy Corollary 8. As before, the sum of the weights is at most twice the optimum value for the instance. Let W be the sum of the weights. If it is needed all values are rounded again based on the number of vertices in each subinstance so that all scaled weights are integers.

Embedding. The algorithm now operates on an instance (G, T) such that Corollary 8 holds. A particular nonterminal $\hat{v}$ denotes the infinite face. All the terminals form finite faces.

Step 1: Cycles. For each terminal t, for each length range $(\text{weight}(t)\epsilon\ell/3, \text{weight}(t)\epsilon(\ell+1)/3]$ (where $0 \leq \ell \leq 3\epsilon^{-3}$), for each weight range $(\text{weight}(t)\epsilon j/3, \text{weight}(t)\epsilon(j+1)/3]$ (where $0 \leq j \leq 3\epsilon^{-2}$), apply Theorem 3 to find a maximally enclosing cycle C enclosing t and with its length and weight in the respective ranges, and add it to our first edge set H_1.

The skeleton at this point, i.e., H_1, is a 2-connected graph. An *ear* E with respect to a 2-connected subgraph H of G is a path in G that starts and ends on a connected component K of H and that does not properly cross H, i.e., all edges of ear E are enclosed by or on the boundary of only one face of H. We say E *separates* a pair of faces f, f'' of G if the faces are in the same face of K but not of $K \cup E$. Refer to Figure 4 for an illustration.

Algorithm `BuildSkeleton`
Input: (dual) instance $(G, \hat{v}, T)$ such that Corollary 8 holds
Output: skeleton H_3 satisfying Theorems 10, and 15, Lemma 28, and Corollary 17

Step 1 (cycles):
$H_1 := \emptyset$
For each terminal t,
  for $j = 0, 1, 2, 3, \ldots, 3\epsilon^{-3}$ and $\ell = 0, 1, \ldots, 3\epsilon^{-2}$,
    find a maximally enclosing cycle C with the following properties:
      C encloses t,
      C has length in the range $(\text{weight}(t)\epsilon\ell/3, \text{weight}(t)\epsilon(\ell+1)/3]$, and
      C encloses weight in the range $(\text{weight}(t)\epsilon j/3, \text{weight}(t)\epsilon(j+1)/3]$
    if there is such a C then $H_1 := H_1 \cup C$

Step 2 (ears):
$H_2 := H_1$
While there is an ear E such that
      E separates terminals t, t' and $\text{length}(E) \leq 3\epsilon^{-3} \min\{\text{weight}(t), \text{weight}(t')\}$, or
      E separates terminal t from $\hat{v}$, and $\text{length}(E) \leq 3\epsilon^{-3}\text{weight}(t)$,
    add E to H_2

Step 3 (clustering):
Let $G' := G/H_2$, obtained from G by contracting each connected component K of H_2.
For each vertex $v \in G'$, assign to v a potential
  $\phi(v) := \epsilon^{-2}\text{length}(G)$ if v is an *outer* blob (i.e., one that contains the infinite face, if it exists),
  $\phi(v) := \epsilon^{-2}\text{length}(K)$ if v is obtained by contracting non-outer blob K,
  $\phi(v) := 0$ otherwise.
Run the `PC-Clustering` algorithm on (G', ϕ). Let H_3' be the resulting set of edges.
$H_3 := H_2 \cup H_3'$, where the edges of H_3' are viewed as the corresponding edges of G.

Step 2: Ears. The algorithm repeatedly applies Theorem 3 to find and add ears that are short (in terms of sum of their edge lengths) in comparison to the weights of terminals they separate[2]. A connected component of H_2 after completion of the ears step is called a *blob*.

Step 3: Clustering. A blob that contains the infinite face is called the *outer* blob, if it exists. Let $G' := G/H_2$ be obtained from G by contracting each blob K of H_2. For each vertex $v \in G'$, assign to v a potential $\phi(v) := \epsilon^{-2}\text{length}(K)$ if v is obtained by contracting a non-outer blob K, $\phi(v) := \epsilon^{-2}\text{length}(G)$ if v is obtained by the outer blob (if it exists), and 0 otherwise. Run the `PC-Clustering` algorithm on (G', ϕ). Let H_3' be the resulting set of edges. Set $H_3 := H_2 \cup H_3'$, where the edges of H_3' are viewed as the corresponding edges of G.

A connected component of H_3 after completion of the clustering step is called a *cluster*. The edges selected in this step are called *cluster edges*. We say a blob is *of* a cluster or *belongs* to a cluster if the blob is a subset of the cluster.

7 Structure of the skeleton and a near-optimum solution

7.1 Structural consequences of the *cycle* step and the *cluster* step

Given a set L of edges (possibly a feasible multiway cut), and given a terminal t, $L(t)$ denotes the minimally enclosing simple cycle enclosing t (if such a cycle exists). The connected component of L containing $L(t)$ is denoted $K_L(t)$.

In G, given a set S of edges, the *region* of a terminal t with respect to S is the set of vertices connected to t in the graph obtained from G by removing edge set S. In G^*, the region of a terminal t is a set of faces;

[2]Beware that those terminals might already be separated, but the ear is attached to a particular connected component, and "separates" means that those terminals were not separated by that component until the ear was added.

if t_∞ is not among these faces, the outer boundary of the region is a simple cycle called the *cycle of* t with respect to S, and denoted by $C(S,t)$. The connected component of S containing $C(S,t)$ is denoted $K(S,t)$.

Theorem 10. *There exist a set of blobs we call* special*, a set $\hat{L}$ of edges, and a set R of terminals such that*

1. $\hat{L} \cup \bigcup\{\text{mincut}(t) \,:\, t \in R\}$ *is a feasible multicut.*
2. $\text{length}(\hat{L}) \leq (1+\epsilon^2)\mathsf{OPT} + \epsilon W$.
3. $\text{weight}(R) \leq 3\epsilon\,\mathsf{OPT} + \epsilon W$.
4. *If two blobs B_1, B_2 are connected by $\hat{L}$ and are not special, then they are in the same cluster.*
5. *No edge of $\hat{L}$ is strictly enclosed by a special blob.*
6. *For each terminal $t \notin R$, a cycle (henceforth denoted $C(t)$) added for t in the* cycles *step intersects $K_{\hat{L}}(t)$.*

Before proving the theorem, we establish the following claims that allow us to invoke Corollary 8 in certain situations without worrying about whether the necessary conditions hold. Indeed, in the first two cases the corollary can be applied, which immediates gives the results, however, the proof of Claim 13 indirectly applies the corollary on a different cycle.

Invoke Theorem 9 on H_3 and $\hat{L}$ to obtain the subset Q of special blobs. The proof of Claim 11 requires that the outer blob, if it exists, should not be special. A simple modification guarantees this condition as follows. If the outer blob exists and is special, place all the terminals in R. Property 1 of Theorem 9 implies that the potential of the outer blob is at most $\text{length}(\hat{L})$. The construction gives $\text{weight}(G) \leq \epsilon^2\text{length}(\hat{L})$. Therefore, the total weight of R is small, and this case is trivial. In what follows, we assume that the outer blob is not special.

Claim 11. *Let ∂B denote the boundary of a special blob B with respect to $\hat{L}$. Then,* $\text{length}(\partial B) \geq \epsilon\text{weight}(B)$.

Claim 12. *For a terminal t, let $\hat{C}(t) = C(\hat{L}, t)$ denote the cycle in $\hat{L}$ that minimally encloses t. We have:* $\text{length}(\hat{C}(t)) \geq \epsilon\text{weight}(\hat{C}(t))$.

Claim 13. *For a terminal t, we have* $\text{length}(C(t)) \geq \frac{\epsilon}{3}\text{weight}(C(t))$.

The upper bound in the following lemma is essential in the construction of our spanner. Let F be a finite face of some cluster. Define $\tilde{T}_F$ as the set of terminals t in F such that some cycle C chosen for t encloses F, and $\text{length}(C) \geq \frac{\epsilon}{3}\text{weight}(C)$. Observe that a terminal t enclosed by F and whose cycle $C(t)$ encloses F falls into this definition, since Claim 13 guarantees $\text{length}(C(t)) \geq \frac{\epsilon}{3}\text{weight}(C(t))$. However, although $C(t)$ is not known during the execution of the algorithm, $\tilde{T}_F$ can indeed be computed.

Lemma 14. $|\tilde{T}_F| \leq 3\epsilon^{-3}$.

Proof of Theorem 10. We first define the special blobs. Let $\hat{L}_0$ be an optimal solution to the multiway cut instance (G,T) and let $\hat{L}'_0$ be obtained from $\hat{L}_0$ by contracting each blob B of H_2. We apply part 2 of Theorem 9 to the subgraph $\hat{L}'_0$ of G'; notice that, as required by statement of that theorem, we ran `PC-Clustering` on (G', ϕ), and obtained the subgraph H'_3. This part asserts the existence of a set Q of vertices of G', which correspond to blobs. We designate a blob B as *special* if the corresponding supernode of G' belongs to Q. Part 2 of Theorem 9 implies that $\hat{L}_0$ satisfies part 4 of the present theorem.

We eventually derive a set $\hat{L}$ of edges from $\hat{L}_0$, and define three sets R_1, R_2, R_3 of terminals. Finally we will define $R = R_1 \cup R_2 \cup R_3$.

We obtain a solution $\hat{L}_1$ from $\hat{L}_0$ as follows. For each special blob B, remove all edges strictly enclosed in B, and add the boundary of B. We define R_1 to be the set of terminals enclosed by special blobs. By construction, Property 5 holds for $(\hat{L}_1, R_1)$. Let us look at the other properties.

Note that $\hat{L}_1$ separates all terminals not in R_1 from each other. Hence $\hat{L}_1 \cup \bigcup\{\text{mincut}(t) \,:\, t \in R_1\}$ is still a feasible multicut. The length of $\hat{L}_1$ can be bounded by $\text{length}(\hat{L}_1) \leq \text{length}(\hat{L}_0) + \sum_{B\ special} \text{length}(B) \leq \text{length}(\hat{L}_0) + \epsilon^2\mathsf{OPT}$ since $\sum_{B\ special} \text{length}(B) \leq \epsilon^2 \sum_{v\in Q} \phi(v) \leq \epsilon^2\text{length}(\hat{L}'_0) \leq \epsilon^2\mathsf{OPT}$,

where we used the definition of ϕ, Part 1 of Theorem 9, and the optimality of $\hat{L}_0$. The weight of R_1 can be bounded as follows. For each special blob B, Claim 11 implies that the total weight enclosed by B is less than $\epsilon^{-1}\,\text{length}(B)$. Summing yields $\text{weight}(R_1) \leq \epsilon^{-1} \sum_{B\ special} \text{length}(B) \leq \epsilon\mathsf{OPT}$. Moreover, it is not hard to see that property 4 still holds for $\hat{L}_1$.

Next, define $R_2 = \{t \notin R_1 \ : \ \text{length}(C(\hat{L}_1, t)) > \epsilon^{-1}\text{weight}(t)\}$ and use it to define $C(t)$ for $t \notin R_1 \cup R_2$ as follows. We claim that for each terminal $t \notin R_1 \cup R_2$, when in Step 1 the algorithm adds cycles for t, there is one length range that includes $\text{length}(C(\hat{L}_1, t))$ and there is one weight range that includes $\mathsf{weight\text{-}enclosed}(C(\hat{L}_1, t))$. Indeed, the first part of the statement follows by definition of R_2, and the second part follows from Claim 12 and the definition of R_2:

$$\text{weight}(t) \leq \mathsf{weight\text{-}enclosed}(C(\hat{L}_1, t)) < \text{length}(C(\hat{L}_1, t))\epsilon^{-1} < \text{weight}(t)\epsilon^{-2}.$$

Since $C(\hat{L}_1, t)$ then satisfies all three constraints, there exists a maximally enclosing cycle that is added by the algorithm for that weight range and length range. Call it $C(t)$.

Finally we derive a solution $\hat{L}$ from $\hat{L}_1$ and a set R_3 of terminals. Because $C(t)$ is maximally enclosing, $C(t)$ cannot be strictly enclosed by $C(\hat{L}_1, t)$, so $C(t)$ either intersects or strictly encloses $C(\hat{L}_1, t)$. If $C(t)$ encloses $K(\hat{L}_1, t)$, then replace $K(\hat{L}_1, t)$, the connected component of $\hat{L}_1$ containing $C(\hat{L}_1, t)$, with $C(t)$, and add to R_3 all the terminals enclosed by $C(t)$ and not by $C(\hat{L}_1, t)$. This defines $\hat{L}$ and R_3. By construction, property 6 is satisfied. Indeed, for each terminal $t \notin R_1 \cup R_2 \cup R_3$, if $C(t)$ did not intersect $K(\hat{L}_1, t)$ then the transformation ensured $C(t) = K(\hat{L}, t)$. It is not hard to check that $\hat{L}$ still satisfies property 5. It only remains to check the other properties.

We show that two terminals $t_1, t_2 \notin R_1 \cup R_2 \cup R_3$ that were separated before this transformation are still separated afterwards. There are three cases. If both are enclosed by $C(\hat{L}_1, t)$ then they were separated by some component of $\hat{L}_1$ that is itself enclosed by $C(\hat{L}_1, t)$, and that component was not removed by the transformation. If one is enclosed by $C(\hat{L}_1, t)$ and the other is completely outside $C(t)$, then they were separated by $C(\hat{L}_1, t)$ before the transformation and are separated by $C(t)$ after the transformation. Finally, if both are outside $C(t)$, then they are outside $K(\hat{L}_1, t)$, so they were separated by some component of $\hat{L}_1$ that is itself outside $K(\hat{L}_1, t)$, and that component was not removed by the transformation. (In all other situations, at least one of t_1, t_2 must be in R_3.) Since $\hat{L}_1$ separated all terminals not in $R_1 \cup R_2$, we have shown that $\hat{L}$ separates all terminals not in $R_1 \cup R_2 \cup R_3$. This establishes Property 1.

Since $C(t)$ is in the same range as $C(\hat{L}_1, t)$, $\text{length}(C(t)) \leq \text{length}(C(\hat{L}_1, t)) + \epsilon\,\text{weight}(t)$. Summing over all terminals $t \notin R_1 \cup R_2$, we conclude that $\text{length}(\hat{L}) \leq \text{length}(\hat{L}_1) + \epsilon\, W$. This shows Property 2.

Clearly $\text{weight}(R_2) < 2\epsilon\,\mathsf{OPT}$. Since $C(t)$ is in the same range as $C(\hat{L}_1, t)$, $\mathsf{weight\text{-}enclosed}(C(t)) \leq \mathsf{weight\text{-}enclosed}(C(\hat{L}_1, t)) + \epsilon\,\text{weight}(t)$. When terminals are added to R_3, $C(t)$ encloses $C(\hat{L}_1, t)$ so we also have $\mathsf{weight\text{-}enclosed}(C(\hat{L}_1, t)) \leq \mathsf{weight\text{-}enclosed}(C(t))$, and the weight of terminals then added to R_3 is not counted towards $\mathsf{weight\text{-}enclosed}(C(\hat{L}_1, t))$, so it is at most $\epsilon\,\text{weight}(t)$. Summing over all terminals $t \notin R_1 \cup R_2$, we conclude that $\text{weight}(R_3) \leq \epsilon\, W$. This gives Property 3.

Moreover, $\hat{L}$ still satisfies Property 4. Indeed, in each transformation one component of $\hat{L}_1$ was removed and replaced with a set of edges that belong to a single blob and that therefore does not connect any two blobs. □

7.2 Structural consequences of the *ears* step

Theorem 15. *There is a set R_4 of terminals of total weight at most $\epsilon\,\mathsf{OPT}$ with the following property. Let B be a blob and F be a finite face of B. For every terminal t enclosed by F and not in $R \cup R_4$, except for at most one terminal, there is a blob B' enclosed by F that intersects $K(\hat{L}, t)$.*

7.3 Structure of a near-optimal solution

The clusters and enclosure relation between clusters naturally induce a nesting forest of clusters. Let $\texttt{Enclosed}(K)$ denote the terminals enclosed by cluster K. A *structured solution* with respect to a subset R of terminals

is a multiset of edges S that can be partitioned into sets, $S = \cup_{K \text{ cluster}} S_K$, in such a way that for every K, $\cup\{S_{K'} : K' \text{ enclosed by } K\}$ is a feasible multiway cut for $\texttt{Enclosed}(K) - R$. Moreover, if no cluster strictly encloses K, then in addition $\cup\{S_{K'} : K' \text{ enclosed by } K\}$ also separates $\texttt{Enclosed}(K) - R$ from t_∞. The following structural Theorem is proved using the technical properties listed in Theorems 10 and 15.

Theorem 16. *Using the definitions and notations of Theorem 10, for each cluster K, let S_K be the union of $\hat{K}$ for every component $\hat{K}$ of $\hat{L}$ that intersects a nonspecial blob of K. Then $S = \cup_K S_K$ has the following properties:*

1. *S is a structured solution with respect to $\hat{R}$.*
2. *For each K, we have:*
   1. *Every edge of S_K is reachable from cluster K without crossing into a blob of any other cluster. (Or equivalently: S_K is in a single face of $H - K$, the skeleton deprived of K.)*
   2. *If some terminal t in $\texttt{Enclosed}(K) - \hat{R}$ is not separated from t_∞ by $\cup_{K'}$ strictly enclosed by $_K S_{K'}$, then $C(t)$ encloses the face F of K that encloses t.*
   3. *Let $\mathsf{Mandate}(S_K)$ be the set of pairs $\{t, t'\}$ of terminals in $\{t_\infty\} \cup \texttt{Enclosed}(K) - \hat{R}$ that are separated by S_K and that are not already separated by $S_{K'}$ for any K' strictly enclosed by K. Then $\{t : t \text{ appears in } \mathsf{Mandate}(S_K)\}$ has at most one terminal per face of K.*

Corollary 17. *There is a constant c such that* $\text{length}(H_3) \leq \epsilon^{-c}\mathsf{OPT}$.

8 Spanner

An outline of the spanner algorithm is presented by Algorithm `BuildSpanner`. The details of the implementation of the algorithm are similar to [6] and are omitted in this submission.

In Sections 8.1 and 8.2 we modify the near-optimal solution further to make it consistent with the portals. Finally in Section 8.3 we show our construction is indeed a spanner for the multiway cut instance.

8.1 Making the near-optimal solution portal-respecting

We now introduce the notions of brick decomposition and portals (from previous works [7, 26]), and restate the relevant theorems that allow us to transform the current solution S_K into the portal-respecting, yet almost as good, solution S''_K. The new solution S''_K respects the mandates defined for S_K. See Theorem 20.

Algorithm `BuildSpanner`
Input: (dual) instance $(G, \hat{v}, T)$ equipped with skeleton H
Output: spanner

Include in the spanner the min-cuts for all terminals.
For each cluster K of H,
  define a subgraph G_K of G as follows:
    retain an edge iff it is on a path that starts on K and that does not cross H.
  find a brick decomposition B_K of G_K with respect to K
  for each brick b,
    select portals on ∂b
    let $T_b := \{t \in b : t \text{ enclosed by } K, \text{ some cycle } C \text{ selected for } t \text{ encloses } b \text{ and } \text{length}(C) \geq \frac{\epsilon}{3}\text{weight}(C)\}$
    include in the spanner the edges of ∂b
    for each configuration of b,
      for each terminal $t \in T_b$ and for the no-terminal case,
        include in the spanner a near-optimal solution for that brick
        that is consistent with the configuration (see Appendix A.2 for the definition),
        using terminal t, if we are not in the no-terminal case

The proof of the theorem relies on the facts that (1) each brick of the decomposition contains at most one terminal of $\mathsf{Mandate}(S_K)$; (2) each connected component of S_K is connected to K (and therefore to brick boundaries: i.e., there is no component floating unattached inside a brick); and (3) the skeleton has length $O(\mathsf{OPT})$. Given these, the subsection is an adaptation of [7].

The planar graph algorithm of [7] uses a *brick decomposition* based on the spanner construction of [26]. Given a connected subgraph K of a planar embedded graph G and given a parameter $\epsilon' > 0$, there is an $O(n \log n)$ algorithm, outlined in the appendix, to compute a connected subgraph M of G, called *graph of bricks*. For each face f of M, the subgraph of G enclosed by ∂f is called a *brick*. M includes all edges of K, and $\text{length}(M) \leq 3\epsilon'^{-1} \cdot \text{length}(K)$. The boundary of each brick B consists of four paths $W_B \cup S_B \cup E_B \cup N_B$.

To analyze the brick decomposition, we use the following slight generalization of Theorem 10.7 from [7].

Theorem 18. *Let B be a brick with boundary $N \cup E \cup S \cup W$, let F be a set of edges in B, and let $U = \{u_0, u_1\}$ be a set of at most two nodes of F. Then there exists a forest F' of B with the following properties:*

- *If two vertices of $N \cup S \cup U$ are connected in F, then they are connected in F' too.*
- *The number of leaves of F' is at most $\alpha(\epsilon')$.*
- $\text{length}(F') \leq (1 + c\epsilon')\text{length}(F)$.
- *All edges of F' are in the subgraph of B enclosed by $F \cup N_F \cup S_F$, where N_F (resp. S_F) denotes the subpath of N (resp. S) spanned by F.*

The proof, omitted, closely mirrors the proof of [7]. We also use the following observation from [7] (Lemma 10.1 therein).

Lemma 19. *Let T be a tree in B whose leaves lie all on N_B (resp. all on S_B). Then there is a subpath of N_B (resp. of S_B) spanning the vertices of $T \cap N_B$ (resp. $T \cap S_B$) whose length is at most $(1 + \epsilon')\text{length}(T)$.*

Then portals are spread out[3] so as to achieve the following *coverage property:* For any vertex x on $\partial\mathcal{B}$, there is a portal y such that the x-to-y subpath of $\partial\mathcal{B}$ has length at most $\text{length}(\partial B)/\theta$.

Theorem 20. *For each cluster K, there exists a set of edges S''_K that*

1. *has length at most $(1 + \epsilon')\text{length}(S_K) + \epsilon'\,\text{length}(B_K)$,*
2. *still satisfies Properties 1 and 2 of Theorem 16,*
3. *still intersects the nonspecial blobs intersected by S_K,*
4. *still separates every pair $\{t, t'\}$ in $\mathsf{Mandate}(S_K)$, and*
5. *is portal-respecting.*

8.2 Feasibility of the portal-respecting solution

A new problem may arise: $S'' = (S''_K)$ still respects the mandates of each S_K, and by construction it is in the spanner, but it is not necessarily a feasible solution overall. The following theorem patches the solution.

Theorem 21. *There exist sets R_5 and R_6 of weight at most $\epsilon\mathsf{OPT}$ such that S'' is a structured solution with respect to $\hat{R} \cup R_5 \cup R_6$.*

8.3 Proof of Theorem 2

Consider the set of edges formed by $S^{(3)} = S'' \cup \bigcup\{\text{mincut}(t) : t \in \hat{R} \cup R_5 \cup R_6\}$. From Theorem 21, it follows that $S^{(3)}$ is a feasible solution. Let us prove that it is near-optimal. The length of S'' is at most $(1 + O(\epsilon'))\text{length}(S) + \epsilon'\text{length}(H)$ by Theorem 20. By Theorem 16, S has length $(1 + O(\epsilon))\mathsf{OPT}$. Corollary 17 bounds length of H by $O(\epsilon^{-c}\mathsf{OPT})$. Theorem 21 gives $\sum_{R_5 \cup R_6} \text{mincut}(t) = O(\epsilon)\mathsf{OPT}$. By Theorem 10 part 3, $\sum_{\hat{R}} \text{mincut}(t) = O(\epsilon)\mathsf{OPT}$. Therefore, overall $S^{(3)}$ has length $(1 + O(\epsilon))\mathsf{OPT}$ if $\epsilon' = O(\epsilon^{c+1})$.

Finally, we show that the length of the spanner itself is not too long. Corollary 17 bounds the length of the skeleton by $\epsilon^{-c}\mathsf{OPT}$. Due to the brick decomposition property stated in Subsection 8.1, the brick boundaries have total length at most $3\epsilon^{-1}$ times the skeleton length. Fix a brick B. By Lemma 14, there are at most $3\epsilon^{-2}$ terminals considered by the spanner algorithm when dealing with B. By Theorem 20, the number of portals is bounded, therefore the number of configurations is bounded. Each solution has length at most the length of the boundary of the brick. Therefore, the sum of lengths of all solutions for a brick B is $O(1)$ times the length of the brick boundary. Altogether the length of the spanner is bounded by some function $f(\epsilon)$ times OPT.

[3] A simple greedy algorithm suffices; see [8].

References

[1] S. ARORA, D. KARGER, AND M. KARPINSKI, *Polynomial time approximation schemes for dense instances of np-hard problems*, in Proceedings of the twenty-seventh annual ACM symposium on Theory of computing (STOC'95), New York, NY, USA, 1995, ACM, pp. 284–293.

[2] B. BAKER, *Approximation algorithms for NP-complete problems on planar graphs*, JACM, 41 (1994), pp. 153–180.

[3] M. BATENI, M. HAJIAGHAYI, AND D. MARX, *Approximation schemes for Steiner forest on planar graphs and graphs of bounded treewidth*, in Proceedings of the fourty-second annual ACM Symposium on Theory of computing (STOC'10), New York, NY, USA, 2010, ACM.

[4] C. BENTZ, *A simple algorithm for multicuts in planar graphs with outer terminals*, Discrete Appl. Math., 157 (2009), pp. 1959–1964.

[5] G. BORRADAILE, C. KENYON-MATHIEU, AND P. KLEIN, *A polynomial-time approximation scheme for Steiner tree in planar graphs*, in 18th SODA, 2007, pp. 1285–1294.

[6] G. BORRADAILE, P. KLEIN, AND C. MATHIEU, *Steiner tree in planar graphs: An $O(n \log n)$ approximation scheme with singly exponential dependence on epsilon*, in 10th WADS, vol. 4619 of LNCS, 2007, pp. 275–286.

[7] ——, *A polynomial-time approximation scheme for Euclidean Steiner forest*, in 49th FOCS, 2008.

[8] G. BORRADAILE, P. N. KLEIN, AND C. MATHIEU, *An $o(n \log n)$ approximation scheme for Steiner tree in planar graphs*, ACM Transactions on Algorithms, 5 (2009).

[9] G. CALINESCU, H. KARLOFF, AND Y. RABANI, *An improved approximation algorithm for multiway cut*, in 30th STOC, 1998, pp. 48–52.

[10] D. Z. CHEN AND X. WU, *Efficient algorithms for k-terminal cuts on planar graphs*, Algorithmica, 38 (2004), pp. 299–316. Twelfth Annual International Symposium of Algorithms and Computation.

[11] S. CHOPRA AND J. H. OWEN, *Extended formulations for the A-cut problem*, Math. Programming, 73 (1996), pp. 7–30.

[12] W. CUNNINGHAM AND L. TANG, *Optimal 3-terminal cuts and linear programming*, in 7th IPCO, no. 1610 in LNCS, 1999, pp. 114–125.

[13] W. H. CUNNINGHAM, *The optimal multiterminal cut problem*, in DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 5, 1991, pp. 105–120.

[14] E. DAHLHAUS, D. S. JOHNSON, C. H. PAPADIMITRIOU, P. D. SEYMOUR, AND M. YANNAKAKIS, *The complexity of multiterminal cuts*, SIAM Journal on Computing, 23 (1994), pp. 864–894.

[15] E. D. DEMAINE AND M. HAJIAGHAYI, *Bidimensionality: New connections between FPT algorithms and PTASs*, in 16th SODA, 2005, pp. 367–376.

[16] E. D. DEMAINE, M. T. HAJIAGHAYI, AND K. KAWARABAYASHI, *Algorithmic graph minor theory: decomposition, approximation, and coloring*, in Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, 2005, pp. 637–646.

[17] R. ERICKSON, C. MONMA, AND A. VEINOTT, *Send-and-split method for minimum-concave-cost network flows*, Math. Op. Res., 12 (1987), pp. 634–664.

[18] A. FRIEZE, *The regularity lemma and approximation schemes for dense problems*, in Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS'96), Washington, DC, USA, 1996, IEEE Computer Society, p. 12.

[19] N. GARG, V. V. VAZIRANI, AND M. YANNAKAKIS, *Approximate max-flow min-(multi)cut theorems and their applications*, SIAM J. Comput., 25 (1996), pp. 235–251.

[20] ——, *Primal-dual approximation algorithms for integral flow and multicut in trees*, Algorithmica, 18 (1997), pp. 3–20.

[21] J. GUO, F. HFFNER, E. KENAR, R. NIEDERMEIER, AND J. UHLMANN, *Complexity and exact algorithms for vertex multicut in interval and bounded treewidth graphs*, European Journal of Operational Research, 186 (2008), pp. 542 – 553.

[22] D. HARTVIGSEN, *The planar multiterminal cut problem*, Discrete Appl. Math., 85 (1998), pp. 203–222.

[23] T. C. HU, *Integer Programming and Network Flows*, Addison-Wesley, 1969.

[24] Y. KAMIDOI, N. YOSHIDA, AND H. NAGAMOCHI, *A deterministic algorithm for finding all minimum k-way cuts*, SIAM J. Comput., 36 (2006/07), pp. 1329–1341 (electronic).

[25] D. R. KARGER, P. KLEIN, C. STEIN, M. THORUP, AND N. E. YOUNG, *Rounding algorithms for a geometric embedding of minimum multiway cut*, in Proceedings of the thirty-first annual ACM Symposium on Theory of Computing, ACM, ed., New York, NY, USA, 1999, ACM Press, pp. 668–678.

[26] P. KLEIN, *A subset spanner for planar graphs, with application to subset TSP*, in 38th STOC, 2006, pp. 749–756.

[27] P. N. KLEIN, *Multiple-source shortest paths in planar graphs*, in 16th SODA, 2005, pp. 146–155.

[28] J. K. PARK AND C. A. PHILLIPS, *Finding minimum-quotient cuts in planar graphs*, in Proceedings of the twenty-fifth annual ACM symposium on Theory of computing (STOC'93), New York, NY, USA, 1993, ACM, pp. 766–775.

[29] É. TARDOS AND V. V. VAZIRANI, *Improved bounds for the max-flow min-multicut ratio for planar and $k_{r,r}$-free graphs*, Inf. Process. Lett., 47 (1993), pp. 77–80.

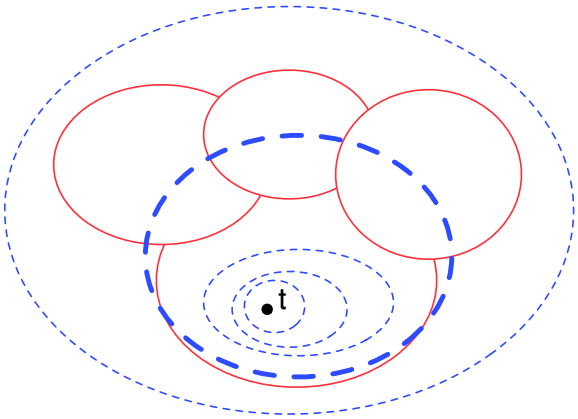

Figure 3: Property 6 of Theorem 10. Consider a terminal t, and the corresponding component $K(\hat{L}, t)$, depicted by red solid edges. The cycle step of the algorithm finds sets of nested cycles for t; each set consists of cycles that have roughly the same length and are such that the weight of terminals enclosed increases smoothly. At least one of these cycles, whose length and weight is suitably chosen, intersects $K(\hat{L}, t)$.

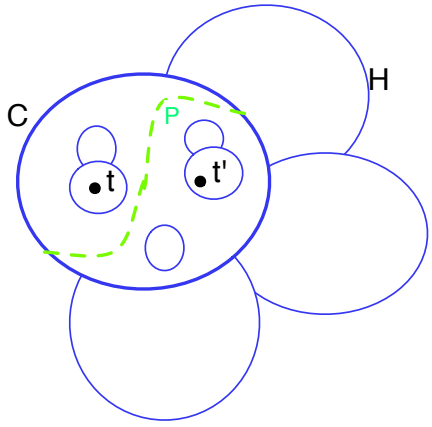

Figure 4: Step 2 of the algorithm adds "ears" to H_1 in order to obtain H_2. The bold blue cycle bounds the face of component H that contains terminals t and t'. The dashed green path P is an ear that separates t from t' with respect to H. Note that t, t' are already separated by H_1, but not by the connected component H.

A Missing proofs and figures

Proof of Lemma 5. The first inequality holds since $\text{weight}(t) \geq \text{length}(\text{mincut}(t))$. The second inequality follows since for each t, $\text{weight}(t) \leq \text{length}(\text{mincut}(t)) + \epsilon \sum_{t \in T_{in}} \text{length}(\text{mincut}(t))/|T_{in}|$. Summing over all $t \in T_{in}$ gives the result. □

Proof of Lemma 6. The cost of the cuts $\delta(S)$ can be charged to $\text{weight}(S)$, and S, which is then deleted, gets charged to only once. The claim then follows from Lemmas 4 and 5. □

Proof of Lemma 7. There are two cases. Suppose the invocation occurs in Step *. In this case, there is a minimal nonempty set $S \not\ni \hat{v}$ of vertices for which $\text{length}(\delta(S)) \leq \epsilon\, \text{weight}(S)$, and $G' = G/(V(G) - S)$ and $T' = T \cap S$ and $\hat{v}'$ is the supervertex obtained by coalescing the vertices in $V(G) - S$. Let X be any proper subset of $S = V(G') - \{\hat{v}'\}$. Minimality of S gives $\text{length}(\delta(X)) > \epsilon \text{weight}(X)$.

Suppose the invocation occurs in Step ‡. In this case, there is no subset S of $V(G') - \{\hat{v}'\}$ such that $\text{length}(\delta(S)) \leq \epsilon\, \text{weight}(S)$. □

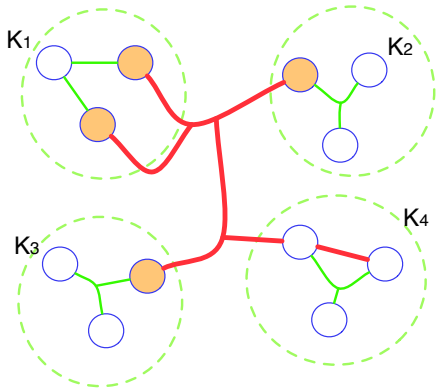

Figure 5: The small circles show blobs defined in the cycles and ears steps. The green edges are the cluster edges added during the clustering step. The larger dashed circles are the resulting clusters. Apply Theorem 9 to the graph where blobs are contracted to nodes and to L equal to the (contracted) optimum solution, part of which is shown in red here: the blobs of Q (special blobs) are orange. Except for orange blobs, the optimum solution does not connect blobs from different clusters. To understand property 4 of Theorem 10, note that all terminals enclosed by the orange blobs are in R_1. The two non-special blobs of that part of the optimum are in the same cluster, cluster K_4 here.

Proof of Theorem 9. (See Figure 5 for an example application). The reader is referred to [3] for details of the algorithm itself.

The `PC-Clustering` algorithm builds a forest F, and produces a vector y satisfying

$$\sum_{S:e\in\delta(S)} \sum_{v\in S} y_{S,v} \le c_e \qquad \forall e \in E \tag{1}$$

$$\sum_{S\ni v} y_{S,v} = \phi(v) \qquad \forall v \in V \tag{2}$$

$$y_{S,v} \ge 0 \qquad \forall v \in S \subseteq V. \tag{3}$$

The analysis takes advantage of the connection between F and y. Consider a topological structure in which vertices of the graph are represented by points, and each edge is a curve connecting its endpoints whose length is equal to the weight of the edge. We assume that each vertex v has a unique color. The algorithm paints by color v a connected portion with length $y_{S,v}$ of all the edges in $\delta(S)$. In particular, each edge e gets exactly $\sum_{C:e\in\delta(S)} y_{S,v}$ units of color v.

Property 1 of the statement of the theorem follows directly from the following lemma.

Lemma 22 ([3]). *The length of F is at most $2\sum_v \phi(v)$.*

In the rest of the proof, we establish the second property of the statement. We say a graph $G'(V, E')$ *exhausts* a color u if and only if $E' \cap \delta(S) \neq \emptyset$ for any $S : y_{S,u} > 0$. Note that this does not imply that all edges with color u are part of E'.

Lemma 23 ([3]). *If a subgraph L of G connects two vertices u_1, u_2 from different components of F, then L exhausts the color corresponding to at least one of u_1 and u_2.*

We can also relate the length of a subgraph to the potential value of the colors it exhausts.

Lemma 24 ([3]). *Let X be the set of colors exhausted by a subgraph L of G. Then* length(L) *is at least $\sum_{v\in X} \phi(v)$.*

We add to Q any vertex whose color is exhausted by L. Lemma 24 gives Property 1. For Property 2, suppose L connects two vertices u_1, u_2 that are in different connected components of H. By Lemma 23, L exhausts the color of at least one of u_1, u_2, so it is placed in Q.

□

Proof of Claim 11. Corollary 8 applies since B cannot be the outer blob. □

Proof of Claim 12. Notice that $\hat{C}$ cannot be the boundary of the infinite face since otherwise it can be removed from $\hat{L}$. Corollary 8 applies. □

Proof of Claim 13. The choice of $C(t)$ ensures

$$\text{length}(C(t)) \geq \text{length}(\hat{C}(t)) - \frac{\epsilon}{3}\,\text{weight}(t), \tag{4}$$

$$\text{weight}(C(t)) \leq \text{weight}(\hat{C}(t)) + \frac{\epsilon}{3}\,\text{weight}(t). \tag{5}$$

We have

$$\begin{aligned}
\text{length}(C(t)) &\geq \text{length}(\hat{C}(t)) - \frac{\epsilon}{3}\,\text{weight}(t) && \text{by (4)}\\
&\geq \epsilon\text{weight}(\hat{C}(t)) - \frac{\epsilon}{3}\,\text{weight}(t) && \text{by Claim 12}\\
&\geq \epsilon\text{weight}(C(t)) - \frac{\epsilon+\epsilon^2}{3}\,\text{weight}(t) && \text{by (4)}\\
&\geq \frac{\epsilon}{3}\text{weight}(C(t)),
\end{aligned}$$

where the last derivation assumes $\epsilon \leq 1$, and notes that $\text{weight}(C(t)) \geq \text{weight}(t)$. □

Proof of Lemma 14. Write $T_F = \{t_1, \ldots, t_k\}$, and assume without loss of generality that t_1 is the terminal with smallest weight. Then $k \leq \sum_{1\leq i\leq k} \text{weight}(t_i)/\text{weight}(t_1)$. Let C_1 be the cycle chosen for t_1 that encloses F for which we have $\text{length}(C_1) \geq \frac{\epsilon}{3}\text{weight}(C_1)$. Then C_1 encloses $t_1, \ldots, t_k$, so $\sum_{1\leq i\leq k} \text{weight}(t_i) \leq \text{weight}(C_1)$. The construction implies that $\text{length}(C_1) \leq \epsilon^{-2}\text{weight}(t_1)$. Combining yields the lemma. □

Proof of Claim 15. For each blob B and each finite face F of B, define

$$T_F = \{t : t \text{ enclosed by } F \text{ and is not in } R,\ C(t) \text{ encloses } F, \textit{and}$$

$$\text{there is no blob } B' \text{ in } F \text{ that encloses an edge of } K(\hat{L}, t)\}.$$

The proof of the following claim follows immediately from the definition of T_F and $\tilde{T}_F$.

Claim 25. $T_F \subseteq \tilde{T}_F$.

Define

$$R_4 = \bigcup\{T_F \ :\ B \text{ a blob}, F \text{ a finite face of } B, |T_F| \geq 2\}.$$

Let $\hat{R} = R \cup R_4$. Lemmas 26 and 27 together prove Theorem 15.

Lemma 26. *Let B be a blob and F be a finite face of B. If $t \notin R$ is enclosed by F, and there is no blob B' in F that intersects $K(\hat{L}, t)$, then $t \in T_F$.*

Proof. Since $t \notin R$, by property 6 of Theorem 10 cycle $C(t)$ intersects $K(\hat{L}, t)$. Since no blob B' in F intersects $K(\hat{L}, t)$, $C(t)$ belongs to none of those blobs. Since $C(t)$ must nevertheless enclose t and belong to some blob, it follows that $C(t)$ encloses F. Now assume, for a contradiction, that some blob B' of F encloses an edge of $K(\hat{L}, t)$. Since B' does not intersect $K(\hat{L}, t)$, B' must enclose $K(\hat{L}, t)$ entirely. So $K(\hat{L}, t)$ is enclosed in B' which is (strictly) enclosed in F which is enclosed in $C(t)$: this contradicts the fact that $K(\hat{L}, t)$ intersects $C(t)$. □

Lemma 27. weight$(R_4) \leq 2\epsilon$OPT

Proof. Let B be a blob and F be a finite face of B. Let $\hat{L}_{B,F} = \bigcup\{K(\hat{L}, t) \,:\, t \in T_F\} \cap F$.

Let $t_{\min}$ be the terminal in T_F of minimum weight. By definition of T_F, the cycle $C(t_{\min})$ encloses F and therefore encloses all terminals in T_F, so weight(T_F) is at most the weight enclosed by $C(t_{\min})$. Thus, Claim 13 implies weight$(T_F) \leq 3\epsilon^{-1}$length$(C(t_{\min}))$. By the construction in Step 1 of the algorithm, length$(C(t_{\min})) \leq \epsilon^{-1}$weight$(t_{\min})$. Thus weight$(T_F) \leq 3\epsilon^{-2}$weight$(t_{\min})$.

Suppose $|T_F| \geq 2$, and let t, t' be two distinct terminals in T_F. Assume without loss of generality that $C(\hat{L}, t)$ does not enclose $C(\hat{L}, t')$. Then $C(\hat{L}, t)$ separates t from t'. Since both t and t' are in F, at least some edges of $C(\hat{L}, t)$ are in F. Since $C(t)$ encloses F and $C(t)$ intersects $K(\hat{L}, t)$, it follows that $K(\hat{L}, t)$ intersects the boundary of F. Hence $K(\hat{L}, t) \cap F$ includes an ear E that separates t from t'. Since this ear was not added in the *ears* step, length$(K(\hat{L}, t) \cap F) \geq$ length$(E) > 3\epsilon^{-3}$weight$(t_{\min})$. Together with the previous paragraph, this implies weight$(T_F) < \epsilon$ length$(\hat{L}_{B,F})$.

By definition of T_F, no blob in F encloses any edge of $\hat{L}_{B,F}$. Thus the sets $\hat{L}_{B,F}$ are disjoint. Therefore, summing over all B and F, we obtain weight$(R_4) \leq 2\epsilon$ length$(\hat{L})$. □

□

Next we state one simpler consequence of the *ears* step.

Lemma 28. *Suppose B and B' are blobs such that there is no blob that strictly encloses one but not the other (i.e., B and B' are siblings). Suppose E is an ear that starts and ends on B such that E together with a path in B between its endpoints forms a cycle that encloses B'. Then, for any terminal t enclosed by B',* length$(E) > 3\epsilon^{-3}$weight(t).

Proof. Otherwise, E would have been added to H_2 in the *ears* step in the special case about separation from $\hat{v}$. □

Proof of Theorem 16. To prove that we have a structured solution, consider two terminals $t_1, t_2 \notin \hat{R}$ and enclosed by some cluster K. By Property 1 of Theorem 10 S is feasible, so at least one of $K_{\hat{L}}(t_1), K_{\hat{L}}(t_2)$ separates t_1 from t_2. Say that $K_{\hat{L}}(t_1)$ separates t_1 from t_2. By Property 6 of Theorem 10, $K_{\hat{L}}(t_1)$ intersects a cycle $C(t)$ that is part of a blob of some cluster K_1 of the skeleton (a nonspecial blob, since $t_1 \notin R$ and we remember that all terminals enclosed by special blobs are in $\hat{R}$). K_1 could be enclosed by K, or equal to K, in which cases $K_{\hat{L}}(t_1)$ is in the union of $\{S_{K'} \,:\, K'$ enclosed by K and thus t_1 and t_2 are separated by that union, as desired. Else assume, for a contradiction, that K_1 encloses K. Then $K_{\hat{L}}(t_1)$ simultaneously intersects a nonspecial blob of K_1 but also separates two terminals enclosed by K, hence must also intersect a nonspecial blob of K: that contradicts Property 4 of Theorem 10.

In terms of separation from t_∞, we know that S is feasible, so every $t \notin \hat{R}$ must be separated from t_∞ by some $K_{\hat{L}}(t)$, that (by Property 6 of Theorem 10) intersects the nonspecial blob of $C(t)$, so if K denotes the cluster of that blob, then K encloses t and $K_{\hat{L}}(t)$ is in S_K, hence S_K separates t from t_∞, as desired. This proves that S is a structured solution with respect to $\hat{R}$.

To prove Property 1, consider a connected component $\hat{K}$ of S_K. By definition it intersects a nonspecial blob of K. By Property 4 of Theorem 10 it does not intersect any other nonspecial blob, and by Property 5 of Theorem 10 it does not enter into any special blob. This implies the desired property.

To prove Property 2, let t be in $\texttt{Enclosed}(K) - \hat{R}$ and not separated from t_∞ by $\cup\{S_{K'} \,:\, K'$ strictly enclosed by $K\}$. In other words, $K_{\hat{L}}(t)$ is not in $S_{K'}$ for K' enclosed by K. By Property 6 of Theorem 10, $K_{\hat{L}}(t)$ intersects a cycle $C(t)$ of the skeleton. This cycle is part of a blob of a cluster, call it K_1, and by definition of S_{K_1}, $K_{\hat{L}}(t)$ is in S_{K_1}. So K_1 is not any K' enclosed in K, and so $C(t)$ must enclose the face F of K that encloses t.

To prove Property 3, consider a cluster K and a face F of K and study the terminals pairs of $\mathsf{Mandate}(S_K)$ that involve at least one terminals in F. By Theorem 15, every terminal t' in F except at most 1 (call that

special terminal t_F) is in a blob B' (part of some K') enclosed in F and that intersects $K_{\hat{L}}(t')$, so $K_{\hat{L}}(t')$ is in $S_{K'}$, and it encloses t'; moreover by Theorem 10 Part 4, $K_{\hat{L}}(t')$ cannot intersect any blob of K, so it has to stay enclosed by F, so $S_{K'}$ separates t' from everyone outside F, and so $\mathsf{Mandate}(S_K)$ does not contain any pair $(t', t")$ with $t"$ outside F. Any two terminals $t'_1, t'_2 \neq t_F$ that are in F are separated by at least one of $K(\hat{L}, t'_1)$ and $K(\hat{L}, t'_2)$, hence separated by $S_{K'_1} \cup S_{K'_2}$, so that pair also cannot appear in $\mathsf{Mandate}(S_K)$. For t' and t_F, since we already have $K(\hat{L}, t')$ in $S_{K'}$, the only way in which t_F and t' could be not separated would be if $K(\hat{L}, t')$ enclosed both t' and t_F, and then $K(\hat{L}, t_F)$ would have to be enclosed in $K(\hat{L}, t')$. But that would contradict the fact that $K(\hat{L}, t')$ is enclosed in F whereas $K(\hat{L}, t_F)$ must intersect a blob by Theorem 10 Part 6, and that has to be F or outside F. This proves the Theorem. □

Lemma 29. $\text{length}(H_1) \leq 9\epsilon^{-7}W$.

Proof. In Step 1, for each terminal t we add cycles for at most $3\epsilon^{-3}$ weight ranges, and for $3\epsilon^{-2}$ length ranges, and each cycle has length at most $\epsilon^{-2}\text{weight}(t)$. □

Lemma 30. $\text{length}(H_2) \leq \text{length}(H_1) + 3\epsilon^{-3}W$.

Proof. Just for for this proof, to avoid the special case of E separating a terminal from $\hat{v}$, we hallucinate that $\hat{v}$ is a terminal and that it has the largest weight.

Whenever the algorithm adds an ear E to a component K, consider the terminals t, t' separated by E that have maximum weight, and charge the length of E to whichever of the two terminals has minimum weight, resolving ties in a consistent manner, assuming for example, up to an infinitesimal perturbation, that all weights are distinct.

We claim that each terminal gets charged at most once. To see this, assume, for a contradiction, that t_0 gets charged, first by an ear E (in face F of component K) separating t_0 from t_1, and then later by an ear E' (in face F' of component K') separating t_0 from t_2. By the definition of charging, $\text{weight}(t_0) < \text{weight}(t_1)$ and $\text{weight}(t_0) < \text{weight}(t_2)$. Ear E splits F into two faces, F_0 and F_1, containing t_0 and t_1 respectively. Ear E' splits F' into two faces, F'_0 and F'_1, containing t_0 and t_2 respectively. Face F' either is enclosed in F_0 (possibly with equality), or encloses K. In the first case, t_2 is in face F_0, contradicting the maximality of $\text{weight}(t_0)$ among terminals in F_0; in the second case, t_1 is in face F'_0, contradicting the maximality of $\text{weight}(t_0)$ among terminals in F'_0. Thus the claim holds. Finally, since $\hat{v}$ has the largest weight, it never gets charged, and so the total length of the ears added is at most $3\epsilon^{-3}\sum_{t\neq\hat{v}}\text{weight}(t) = 3\epsilon^{-3}W$. □

Lemma 31. $\text{length}(H_3) \leq (4\epsilon^{-2}+1)\text{length}(H_2)$.

Proof. By construction $\text{length}(H_3) \leq \text{length}(H_2) + \text{length}(H'_3)$. Using Theorem 9 and the definition of potential:

$$\text{length}(H'_3) \leq 2\sum_v \phi(v) \leq 4\sum \epsilon^{-2}\{\text{length}(K) \ : \ K \text{ a blob}\} = 4\epsilon^{-2}\text{length}(H_2),$$

where the second inequality follows since each blob B is counted once for its own potential and possibly once for the outer blob. □

By combining Lemmas 29, 30, 31, 5 and 4. we obtain Corollary 17.

Proof of Theorem 20. The proof uses the following definition. Given a forest F in the brick B, by N_F (resp. S_F) we denote the subpath of N (resp. S) spanning the vertices of $F \cap N$ (resp. $F \cap S$). Given a terminal t, we say that t is outside F if t is not enclosed by $F \cup N_F \cup S_F$. We say that F is em north of t (resp. *south* of t) if t is outside F and $S_F = \emptyset$. We say that F is *east* of t (resp. *west* of t) if t is outside F and F separates t from E (resp. from W).

First, S_K is transformed into S'_K that, as we will show, only crosses the boundary ∂B of each brick B a bounded number of times; then to build S''_K from S'_K, we simply take, for each crossing, a detour to the

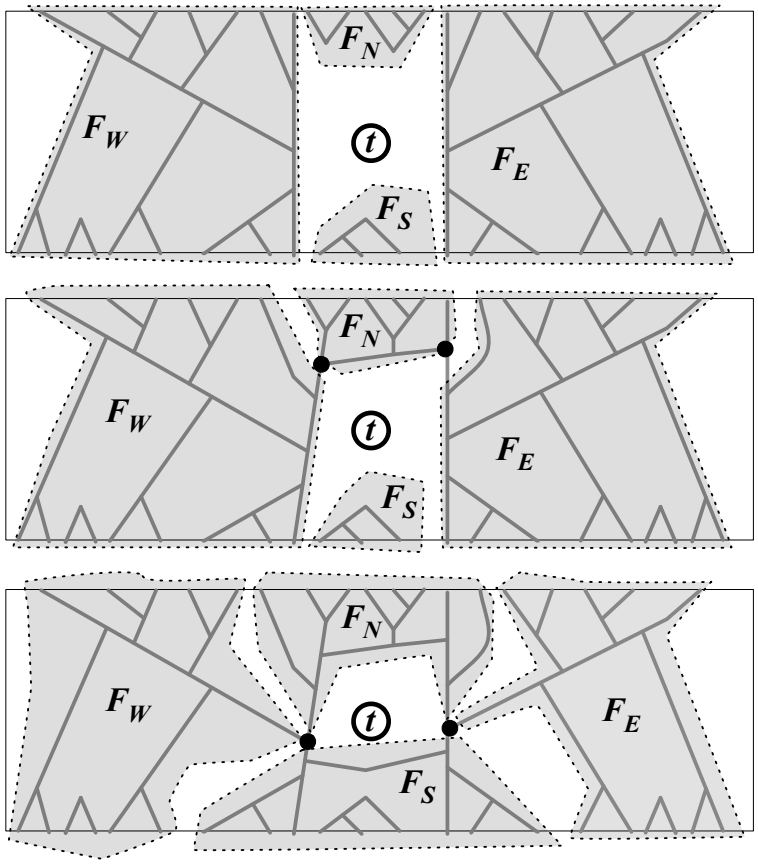

Figure 6: The three cases in the proof of Theorem 20.

nearest portal. The latter operation is straightforward, and is omitted in this discussion. Hence, we focus on the first transformation.

There are two cases to consider. In the first case, there is no terminal of $\mathsf{Mandate}(S_K)$ inside the brick. Inside the brick, the near-optimal solution is a forest F. We apply Theorem 18 to F with $U = \emptyset$. Since the new forest F' preserves connectivity, it also preserves separation: if some interior path from some $x \in \partial B$ to some $y \in \partial B$ crosses F, then it also crosses F'. Thus, replacing F by F' preserves feasibility of our multiway cut solution.

In the second case, there is a single terminal t of $\mathsf{Mandate}(S_K)$ inside the brick. Then we can decompose F into at most four forests, F_N, F_S, F_W, F_E, respectively North, South, West and East of t. More precisely, there are three subcases. In the first subcase, the cell of t intersects both N and S. In the second subcase, the cell of t intersects just one side, S for example. Then there exist two vertices u_0 and u_1 in B such that u_0 is a leaf of F_W and of F_N, and u_1 is a leaf of F_E and of F_N. In the third subcase, the cell of t intersects neither S nor N, and then there exist two vertices u_0 and u_1 in B such that u_0 is a leaf of F_W, F_N and F_S, and u_1 is a leaf of F_E, F_N and F_S. See Figure 6. It only remains to apply Theorem 18 to each of the four forests with $U \subseteq \{u_0, u_1\}$, defined appropriately for each forest. Thanks to the fourth property of Theorem 18, we can check feasibility of our multiway cut solution is preserved.

The portal vertices are defined as in Step 5 of [5], S'_K is modified by adding detours to portals in the same way, and the cost analysis is identical. □

Proof of Theorem 21. We apply Theorem 9 part 2 with $L = S''$, to define *extra-special* blobs enclosing a set of vertices $Q = R_5$, of total weight at most $\epsilon^2\mathsf{OPT}$, and such that, if two blobs are connected in S'' and are not extra-special, then they are in the same cluster of the skeleton.

Consider a terminal pair $\{t_1, t_2\}$ not in $\hat{R} \cup R_5$ and that are not separated by S''. First, assume that $t_2 = t_\infty$. By Theorem 16, S is feasible. Thus t_1 is separated from t_∞. Let K be chosen minimally enclosing so that S_K separates t_1 from t_∞: by definition of mandates, $\{t_1, t_\infty\}$ is in $\mathsf{Mandate}(K)$. Then, Theorem 20 ensures t_1, t_∞ are still separated by S''_K and hence by S'', contradicting our assumption.

Second, assume that neither t_1 nor t_2 is t_∞. From Theorem 10 Part 6, a cycle $C(t_1)$ intersects the component of S minimally enclosing t_1. Let K_1 be the cluster that $C(t_1)$ is part of: K_1 has a blob enclosing t_1. Thus, the component of S minimally enclosing t_1 intersects the blob B_1 of K_1 enclosing t_1, hence is in S_{K_1}. We define B_2 and K_2 similarly.

If $K_1 = K_2$ then, since S_{K_1} encloses both t_1 and t_2, by definition of structured solution $\cup\{S_{K'} : K' \text{ enclosed by } K\}$ must separate $\{t_1, t_2\}$. Since S_{K_1} minimally encloses both t_1 and t_2, it follows that $\{t_1, t_2\}$ must be in $\mathsf{Mandate}(K_1)$ and so they are separated in S'', contradicting our assumption.

If K_1 and K_2 do not enclose each other, then let K be the parent of K_1 and F be the face of K in which K_1 lies. If K_2 is in a different face, then Property 1 implies that S''_{K_1}, just like S_{K_1}, cannot cross any skeleton cycles other than those in K_1, and therefore stays in F, so it separates t_1 from t_2, contradicting our assumption. Thus, K_1 and K_2 must be sibling clusters. Since S_{K_1} minimally separates t_1 from t_∞, we have $\{t_1, t_\infty\} \in \mathsf{Mandate}(S_{K_1})$. Hence, S''_{K_1} also separates them due to Theorem 20. Therefore, the only way in which t_1 and t_2 are not separated is if S''_{K_1} encloses both t_1 and t_2, and similarly S''_{K_2} encloses both t_1 and t_2. However, by Theorem 20 S''_{K_1} still intersects B_1, and S''_{K_2} still intersects B_2, so S''_{K_1} and S''_{K_2} must intersect each other, and together they define a path in S'' connecting B_1 to B_2. Since K_1 and K_2 are different clusters, it follows that one of the two blobs must be extra-special, hence t_1 or t_2 is in R_5, contradicting our assumption.

If K_1 contains K_2 in one of its faces F, then S_{K_2}, as it does not cross any skeleton cycle other than K_2, must be contained in F. But S''_{K_2} also stays in F by Theorem 20. The only possibility for it not to separate t_1 from t_2 is if t_1 is in F. Then, S''_{K_2} still intersects B_2, a blob of K_2, but it also encloses t_1. Notice that part of S''_{K_2} or S_{K_2} was not added in the ear step as a K_2-to-K_2 path separating t_1 from t_∞. By definition of K_1, we know that $\cup\{S_{K'} : K' \text{ enclosed by } K_2\}$ does not separate t_1 from t_∞, so S''_{K_2} would have given a candidate ear. Since it was not selected, it must be that $\text{weight}(t_1)$ is less than $(1/3)\epsilon^3\text{length}(S''_{K_2})$, and then we put t_1 in R_6. We bound as follows the number of such terminals added to R_6 for each K_2. Notice that K_1 is the parent of K_2 in the nesting forest, t_1 is a terminal in the same face F of K_1 as K_2, and $C(t_1)$ encloses F. All terminals added to R_6 due to K_2 are part of $\tilde{T}_F$ by definition. Lemma 14 bounds their number to be at most $3\epsilon^{-3}$, hence the total weight of R_6 is at most $\epsilon\text{length}(S'')$. This concludes the theorem. □

A.1 Reduction to bounded-treewidth graph after the spanner construction

Here we present more details about proving Theorem 1 given Theorem 2. Starting from the spanner we incorporate a Baker-like technique [2] to reduce the problem to a bounded-treewidth graph that can then be solved using standard techniques. More precisely we start with a planar embedding of the spanner graph. We partition edges into layers by running a breadth-first search (BFS) algorithm from an arbitrary root vertex r: layer j consists of the edges whose nearest endpoints are distance j from r. Let p be a certain function of ϵ^{-1}. The approximation algorithm computes, for each choice of $i < p$, the union of the edges in layers congruent to i modulo p, and of the optimal solution for the residual instance (the algorithm will then output the best among these p solutions). The residual instance is defined by removing the layers congruent to i modulo p; each of its connected components is p-outerplanar, hence has bounded-treewidth. Guo et al. [21, Corollary 1] show that vertex-weighted multiway cut (a generalization of our (edge-weighted) multiway cut since we can put weight ∞ on the original vertices and a vertex v_e of weight w_e on top of an original edge e of weight w_e) with a terminal set S on a graph G of treewidth p can be solved in $O((|S|+1)^{p+1}p^2n)$ time by a standard dynamic program. Though this running time is sufficient for us to obtain a PTAS, it is not hard to see that the running time can be easily improved to $p^{O(p)}n$[4]. The key argument in the analysis is that the removal

[4]Guo et al. [21] in their dynamic program, for each bag of the (nice) tree decomposition, keep all $|S|$ colorings of the vertices of the bag. Due to this they have a factor $(|S|+1)^{p+1}$ in their running time. Here each color class is corresponding to the portal of a connected component in the optimum solution and its color is the color of the only terminal vertex in this component. However, we

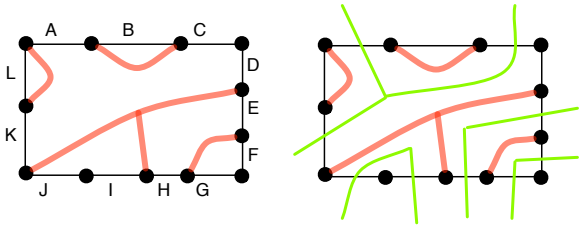

Figure 7: The figure on the left represents a brick b. The big dots are the portals, and the thick red lines consist of edges in E. The subpaths of ∂b are labeled by the letters A through L. The corresponding partition π is $\{\{A, C, D, K\}, \{B\}, \{E, H\}, \{F, G\}, \{I, J\}, \{L\}\}$. The figure on the righ illustrates the definition of consistency. The thinner green lines represent paths in the dual. (The outgoing edges all lead to the vertex representing the infinite face but that vertex is not shown.)

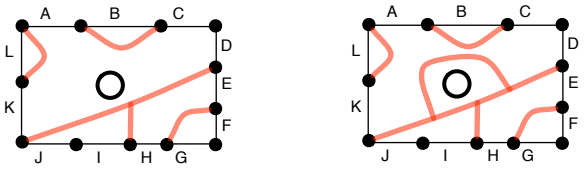

Figure 8: A terminal is indicated by the circle. The figure on the left is consistent with the pair $(\pi, \{A, C, D, K\})$ where π is the partition of Figure 7. The second element of the pair is the part that corresponds to the region containing the terminal. The figure on the right is consistent with the pair $(\pi, \emptyset)$ since there is no way to get from an edge of ∂b to the terminal while avoiding E.

of one of the p congruence classes of layers removes at most a $1/p$ fraction of the original spanner, which is at most ϵ^{-c}/p fraction of the optimum solution. For sufficiently large p (i.e., $p = \epsilon^{-c-1}$) this part is at most $\epsilon\mathsf{OPT}(G_{in}, T_{in})$. When we remove these edges the solution to the multiway cut problem on the residual graph can be only smaller than $\mathsf{OPT}(G_{in}, T_{in})$. So the returned solution must be within a $1 + \epsilon$ factor of an optimum solution. The running time is thus $\epsilon^{-O(\epsilon^{-c})}n$.

A.2 Configurations for the dynamic program

In this section we define the configurations needed for the dynamic program. In fact, `BuildSpanner` of Section 8 constructs a set of partial solutions that are consistent to all possible configurations for each brick. Here we describe what these configurations are and what the notion of consistency means.

Consider a particular brick b. The portals divide ∂b into a set $\mathcal{P}_b$ of θ subpaths. Each subpath starts and ends at a portal and has no internal vertices that are portals. We define a *configuration* for a brick b to be a pair (π, S) where π is a partition of $\mathcal{P}_b$ and S is one of the parts or is $\emptyset$.

We say a set E of edges of b is *consistent with* π if the following condition holds: if two edges e_1, e_2 of ∂b are in different parts of π, then in the planar dual b^* they are separated by E (i.e., there is no path connecting them that avoids edges of E and avoids the infinite face of b^*.) We say E and a terminal t are *consistent with* (π, S) if, in addition, the following condition holds: if $e \in \partial b$ is not in a subpath of S, then in b^*, e and t are separated by E (i.e., there is no path P connecting e with t that avoids edges of E and avoids the infinite face of b^*.) Figures 7 and 8 illustrate the definitions.

For each configuration of the brick b, we are going to find the optimal consistent solution. This is done via a dynamic programming inside the brick. The details are omitted here since a similar DP has been described in [8, 17].

B Finding the minimum-weight cycle enclosing a given amount of weight

Here, we prove Theorem 3.

can replace this $|S|$ by $p + 1$ since in the dynamic programming we only need $p + 1$ colors to represent at most $p + 1$ sets in the partition of vertices of each bag and not the real identity of the terminal which lies in this connected component. By this replacement we obtain the aforementioned running time.

Algorithm `BrickDecomposition`
Input: graph G_K, equipped with connected set of edges K
Output: brick decomposition M of G_K

Include K in the brick decomposition M.
For each face F of K, consider the subgraph G_F of G_K with boundary F.
  Decompose G_F into strips:
    initially $G_1 = G_F$;
    find two vertices x and y of the boundary ∂G_1 of G_1, such that
      the shortest path S from x to y along ∂G_1 is more than
      $(1+\epsilon)$ longer than the shortest path N from x to y in G_1
      (breaking ties by taking x, y closest along ∂G_1;
      then add N to the brick decomposition M;
      the subgraph enclosed by $N \cup S$ is a strip.
    Recursively decompose the subgraph of G_1 bounded by $N \cup (\partial G_1 - S)$ into strips.
  For each strip, bounded by $N \cup S$, decompose into bricks:
    Starting on S with the point s_0 common to S and N, go along S to define (s_i):
      s_i is the first vertex x such that the shortest path from x to N going through s_{i-1}
      is more than $(1+\epsilon)$ longer than the shortest path from x to N.
    Let $\kappa = 4\epsilon^{-2}(1+\epsilon^{-1})$.
    Let $j_0 \in [0, \kappa - 1]$ be such that $\sum_{j : j \equiv j_0 (mod\ \kappa)} \text{dist}_{G_1}(r_j, N)$ is minimum.
    The supercolumns are the shortest paths from r_j to N for $j \equiv j_0 (\mod \kappa)$.
    Add the supercolumns to the brick decomposition M, decomposing the strip into bricks.

Let G be an undirected planar embedded graph with edge-lengths and face-weights. Let T be a spanning tree of G. Park and Phillips [28] give a technique for finding minimum-length cycles enclosing a given amount of weight. Root T at a node r. Each edge of G corresponds to two oppositely directed *darts*. Assign weights to the darts as follows. The weight of a dart belonging to an edge of T is zero. For a dart uv not in T, there is a unique simple cycle consisting of uv and the u-to-v path in T, called the *elementary cycle* of uv with respect to T. If the elementary cycle of uv is clockwise, the weight of uv is defined to be the sum of the weights of the faces enclosed by the elementary cycle, and the weight of vu, the oppositely directed dart, is the negative of this sum.

Fact 32. *For any simple clockwise cycle C, the weight of the darts forming C equals the weight enclosed by C.*

This fact can be used to solve a variety of problems involving minimum-cost cycles as long as the weights are integers of bounded magnitude, using the dynamic-programming algorithm for weight-constrained shortest paths.

Suppose the lengths are nonnegative numbers and the weights are nonnegative integers summing to less than W. There is a simple dynamic program to compute a table that, for each $w \in [0, W)$, for each pair u, v of nodes, stores the shortest path of weight w. This table can be used to find, for each $w \in [0, W)$, the shortest simple cycle of weight w.

A similar technique can be used to find, for a given face f and for each $w \in [0, W)$, the shortest simple cycle of weight w that encloses f. (For this, introduce a second kind of weight, which is 1 on face f and zero on all other faces.)

A similar technique can be used for the following problem: given a simple cycle C, and given two faces f_1 and f_2 enclosed by C , find the shortest path between nodes of C that (i) is enclosed by C and that (ii) separates f_1 and f_2 within C. (For this, start with a spanning tree T that includes all but one of the edges of C, and assign weight 1 to each of f_1 and f_1.)