Constant Factor Approximation of Vertex-Cuts in Planar Graphs

Eyal Amir
Computer Science Division
University of California
Berkeley, CA 94720.
eyal@cs.berkeley.edu

Robert Krauthgamer[*]
Computer Science Division
University of California
Berkeley, CA 94720.
robi@cs.berkeley.edu

Satish Rao
Computer Science Division
University of California
Berkeley, CA 94720.
satishr@cs.berkeley.edu

ABSTRACT

We devise the first constant factor approximation algorithm for minimum quotient vertex-cuts in planar graphs. Our algorithm achieves approximation ratio $3.5 \cdot (1+\epsilon)^2$ with running time $O(W \cdot n^{3+2/\epsilon})$. The approximation ratio improves to $4/3 \cdot (1+\epsilon)$ if there is an optimal quotient vertex-cut (A^*, B^*, C^*) where C^* has relatively small weight compared to A^* and B^*; this holds, for example, when the input graph has uniform (or close to uniform) weight and cost. The approximation ratio further improves to $1+\epsilon$ if, in addition, $\min\{w(A^*), w(B^*)\} \le \frac{1}{3}W$.

We use our quotient cut algorithm to find the first constant-factor pseudo-approximation for vertex separators in planar graphs.

Our technical contribution is two-fold. First, we prove a structural theorem for vertex-cuts in planar graphs, showing the existence of a near-optimal vertex-cut whose high-level structure is that of a bounded-depth tree. Second, we develop an algorithm for optimizing over such complex structures, whose running time depends (exponentially) not on the size of the structure, but rather only on its depth. These techniques may be applicable in other problems.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Delphi theory

[*]Supported in part by NSF grants CCR-9820951 and CCR-0121555 and DARPA cooperative agreement F30602-00-2-0601.

Keywords

ACM proceedings, LaTeX, text tagging

1. INTRODUCTION

Graph partitioning is extensively used in many areas, including scientific computing, VLSI design, task scheduling, machine vision, and automated reasoning, see e.g. [27, 17, 28, 3, 1]. One important graph partitioning problem that emerges naturally in a variety of applications is the following vertex-separator problem. Let $G(V, E)$ be a graph with vertex costs $c : V \to \mathbb{N}$ and vertex weights $w : V \to \mathbb{N}$. Throughout, let $w(S) = \sum_{v \in S} w(v)$ and similarly $c(S) = \sum_{v \in S} c(v)$ for $S \subseteq V$, let $n = |V|$ denote the number of vertices, and let $W = w(V)$ denote the total weight of the vertices. A *vertex-cut* of G is a partition of V into three disjoint sets A, B, C such that no edge in E has one endpoint in A and one in B. The cost of the vertex-cut is $c(C)$.

A vertex-cut is called a *vertex-separator* (aka $2/3$-*balanced*) if $\max\{w(A), w(B)\} \le \frac{2}{3}w(V)$. The vertex-separator problem is to find a minimum-cost vertex separator in an input graph G. This problem was shown to be NP-hard by Bui and Jones [5]. Leighton and Rao [18, 19] give an $O(\log |V|)$ pseudo-approximation algorithm for this problem in general graphs.[1] A closely related problem is the quotient vertex-cut problem. The *quotient* of the vertex-cut is defined as[2]

$$q(A, B, C) = \frac{c(C)}{\min\{w(A), w(B)\} + w(C)}.$$

The minimum quotient vertex-cut problem is to find a vertex-cut with minimum quotient in an input graph. Also this

[1]A polynomial-time algorithm is called a *ρ-approximation algorithm* for a minimization problem if for any instance of the problem A outputs a solution whose cost is at most ρ times that of a minimum-cost solution for the instance. A *pseudo-approximation* algorithm for 2/3-balanced cuts is allowed to output, say, a 3/4-balanced cut (and its cost is still compared to that of a 2/3-balanced cut).

[2]One difficulty for vertex separators is the question of how one counts the weight of the vertices on the separator C. In general, we can expect the weight of C to be small compared to $w(A), w(B)$ in cuts with small quotient. This is a natural assumption and is provably true for unweighted planar graphs. This suggests including $w(C)$ in the denominator. Indeed, it turns out that for many applications (including finding a separator) this is useful. We discuss this further in Section 2.2, where we define a more general notion of separator and quotient cost of a vertex-cut.

problem is NP-hard (see below), and the best approximation ratio known for this problem is $O(\log n)$, due to [16, 19].

For planar graphs, the celebrated Planar Separator Theorem, due to Lipton and Tarjan [21] (see also [2, 22, 29]), shows that every planar graph has a vertex-separator (A, B, C) with $|C| = O(\sqrt{n})$. (This corresponds to the case $c(v) = 1$ for all $v \in V$). Furthermore, they give algorithms for finding such a vertex-separator. However, these methods are only guaranteed to find separators of size $O(\sqrt{n})$. Many planar graphs have much smaller separators, so for any particular graph these algorithms may find separators that are far from optimal. We address this issue by devising a constant factor pseudo-approximation for the vertex-separator problem in planar graphs. No approximation better than that for general graphs was previously known for the vertex-separators problem in planar graphs; Interestingly, this problem is not known to be NP-hard.

1.1 Related work

Vertex-cut variants of graph partitioning problems are usually closely related to that of the edge-cut variants of these problems. For instance, an edge-cut variant (either directed or undirected) of a problem frequently reduces to a vertex-cut variant of the same problem by replacing every edge by a path of length two, letting the new vertex have zero weight and the same cost as the edge it replaces. Notice that this reduction preserve planarity of the input graph. There is also a well-known reduction from the (say undirected) vertex-variant to the directed edge-cut variant, see [19]. However, this reduction does not preserve planarity.

There is a significant amount of research on approximating various graph partitioning problems, and many of these results extend from one variant of the problem to another by using the aforementioned general reductions. An $O(\log n)$–approximation for the minimum quotient edge-cut problem (aka sparse cut) in undirected graphs was first devised in [18, 19]. Their results and techniques were extended and expanded (e.g., to other graph partitioning problems) in [16, 12, 11, 15, 20, 4, 7, 9].

For planar graphs, significantly better approximation ratios are usually known. Park and Phillips [24] devise a pseudo-polynomial time algorithm for solving (exactly) the minimum quotient edge-cut problem (improving over a constant factor approximation in [25, 26]). Klein, Plotkin and Rao [14] give a constant factor approximation For the directed version of this problem. These results yield a constant factor pseudo-approximation for edge-separators. Garg, Saran and Vazirani [10] give factor 2 (true) approximation for edge-separators (namely, 2/3-balanced cuts). Feige and Krauthgamer [9] give an $O(\log n)$ (true) approximation for minimum-bisection. As mentioned before, these results do not translate to similar approximation ratios for quotient vertex-cuts, since the well-known reduction of vertex-cut problems to edge-cut problems in directed graphs might create nonplanar graphs even if the original graph is planar.

1.2 Results

We devise the first constant factor approximation algorithm for minimum quotient vertex-cuts in planar graphs. Our algorithm achieves approximation ratio $3.5 \cdot (1+\epsilon)^2$ with running time $O(W \cdot n^{3+\frac{2}{\epsilon}})$. The approximation ratio improves to $4/3 \cdot (1+\epsilon)$ if in a vertex-cut (A^*, B^*, C^*) of optimal quotient cost, C^* has relatively small weight compared to A^* and B^* (namely, $w(C^*) \leq \frac{\epsilon}{2} \min\{w(A^*), w(B^*)\}$); this holds, for example, when the input graph has uniform (or close to uniform) weight and cost. The approximation ratio further improves to $1+\epsilon$ if, in addition, $\min\{w(A^*), w(B^*)\} \leq \frac{1}{3}W$.

We also achieve the first constant factor pseudo-approximation for vertex-separators in planar graphs. This follows from the approximation for quotient vertex-cuts by extending a well-known technique (see [25, 19]) from edge-cuts to vertex-cuts.

Techniques.. No constant-factor approximation was known for minimum-quotient vertex-cuts previously. This is, perhaps, due to subtle but fundamental differences between minimum quotient edge-cuts and vertex-cuts. The results for finding optimal quotient edge-cuts rely on a well-known lemma (see [23, 26]) that the optimal quotient edge-cut divides a graph into two connected graphs. For planar graphs, this implies that there is a simple Jordan curve in the plane that corresponds to an optimal quotient edge-cut. The algorithms proceed by searching for simple cycles in the dual of the planar graph. In particular, Park and Phillips [24] define a cost and weight for each edge in the dual graph such that the cost of any cycle in the dual graph corresponds to the cost of a cut in the input graph, and the weight of the cycle corresponds to the weight that the cut separates. Then, they can modify methods for finding minimum mean cycle, due to [13], to find the best such cycle.

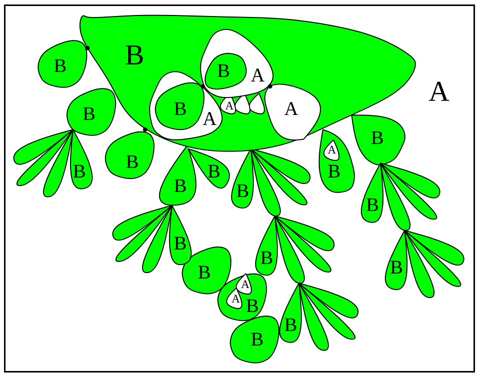

Figure 1: Jordan curve in the plane corresponding to an optimal quotient vertex-cut

In contrast, an optimal quotient vertex-cut in a graph might divide the graph into an arbitrary number of connected components. In fact, both sides of the cut can be an arbitrary set of components (consider, e.g., a star). For planar graphs, the "Jordan" curve in the plane corresponding to an optimal or near-optimal quotient vertex-cut might be quite complex. See, for example, Figure 1.

We partially address this difference by showing that for plane graphs there exists a vertex-cut whose quotient cost is within 4/3 of optimal, and in which *one* side is connected in the plane. It can be seen that such a separator must correspond to a "Jordan" type curve that is essentially a tree of cycles, see e.g. Figure 2.

Still, this structure is more complex than a simple cycle; any walk along the structure visits many vertices numerous times where any "revisits" are "free" in terms of the cost of the walk. Thus, standard reductions to the minimum mean cycle problem do not work. To address this, we show that there is a near-optimal quotient vertex-cut with an associated tree of cycles that has constant depth.

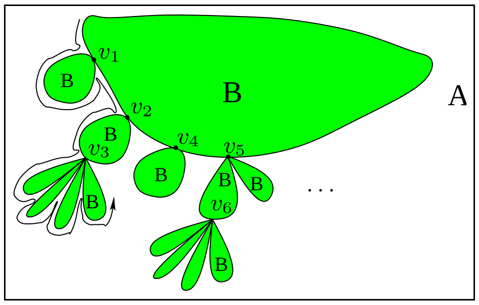

Figure 2: A tree of cycles corresponding to a near-optimal quotient vertex-cut. Midway through a walk around it, v_1 is completely visited, v_2, v_3 are partially visited, and v_5, v_6 have not been visited yet.

Of course, one can easily extend the mean cycle methods to "guess" a constant number of vertices that are allowed to be revisited. Thus, if a near-optimal walk has a simple structure and only revisits a constant number of vertices, we could find it. Unfortunately, this is not true even for a bounded-depth tree of cycles. For example, any near-optimal walk separating the graph in Figure 2 revisits many vertices.

Hence, we need to work a bit more. We consider a walk around a tree of cycles, and notice that at any point in the walk each vertex is either completely visited, has yet to be visited at all, or has been visited and will be visited again. However, the number of vertices in this partially visited state is bounded by the depth of the tree of cycles (see Figure 2). Thus, we devise an algorithm that optimizes over walks of this form. The algorithm needs only "remember" a constant number of vertices at a time (the partially visited ones). Hence, it has a polynomial running time.

This outline is a simplification of our methods. We need to also enforce some topological conditions due to the relationship between the edge weight of our walk and the vertex weight separated. Also, in the technical sections, we discuss the case where the separator vertices themselves have a significant amount of weight. We also discuss some generalizations of our structure theorems which will hopefully lead to the removal 4/3 factor in our approximation.

Organization. Section 2 presents background notions and concept; Section **??** shows that restricting the search for minimum quotient vertex-cuts to trees of cycles in an appropriately defined dual of the graph guarantees constant factor approximation; Section 4 presents a dynamic programming algorithm for computing a near-optimal tree of cycles; finally, Section 5 uses the quotient vertex-cut to find a approximate vertex separator.

2. PRELIMINARIES

Throughout this paper we allow parallel edges. A (directed) *cycle* in a graph G is a sequence of edges $(u_0, v_0), \dots, (u_{k-1}, v_{k-1})$ such that $v_i = u_{(i+1) \bmod k}$ for all $i < k$. A cycle is *simple* if it contains k distinct vertices and k distinct edges. Note that two parallel edges define a simple cycle of length 2.

2.1 Planar Graphs

A graph G is *planar* if it can be drawn in the plane $\mathbb{R}^2$ such that no two edges intersect, except possibly, at their endpoints. A *plane graph* is a planar graph together with a particular drawing of it. A plane graph G partitions $\mathbb{R}^2 \setminus G$ into *faces*; a face is a region that can be linked by an arc in $\mathbb{R}^2 \setminus G$. The *boundary* of a face is the subgraph whose point set in the drawing is the frontier of the face. The boundary of a face is not necessarily a simple cycle. For example, if G is a tree then the boundary of the outer face (the only face of G) is a cycle that contains every edge of G twice (once in each direction).

Adjacency and connectivity of faces.. Two faces of a plane graph G are *adjacent* if their boundaries have a common edge. Note that non-adjacent faces may contain a common vertex in their boundaries.

Definition 2.1. *Let R be a set of faces of a plane graph G. A* faces-path *in R between two faces $f, f' \in R$ is a sequence of faces $f = f_1, \dots, f_k = f'$ all from R such that for all $i < k$ the faces f_i and f_{i+1} are adjacent. We say that the set of faces R is* connected *if there is a faces-path in R between every two faces $f, f' \in R$. A connected set of faces is also called a* connected region*.*

Notice that every two faces in a connected region R are linked by a path in the plane $\mathbb{R}^2$ that goes only through faces of R and edges of G (that are on the boundary of two distinct faces of R), without going through any vertex of G. A maximal connected subset of a set of faces F is called a *connected component* of F. Clearly, any set of faces F can be partitioned (uniquely) into its connected components $F_1, \dots, F_t$; we let $\mathrm{CC}(F) := \{F_1, \dots, F_t\}$ be the set of connected components of F.

Edge-Dual Graphs.. Let $G(V, E)$ be a plane graph. The *edge-dual*[3] of G is the planar graph $G_{D_e}(F, E_e)$, where F is the set of faces of G on the plane, and E_e is the set of edges that connects two vertices in F iff the two corresponding faces in G have a common edge in their boundaries. G is an edge-dual of G_{D_e} (see example in Figure 2.1(A),(B)).

An important property of edge-duals is that a simple cycle in G_{D_e} disconnects the faces of G_{D_e} and corresponds to a minimal set of edges disconnecting the vertices in G. The converse is also true, namely, every cutset defines a simple cycle in the edge-dual, see e.g. [8, 6]. This property may suggest that some cutsets (such as minimum quotient-cuts) can be found by looking for certain cycles in the dual graph. Indeed, the algorithms of [25, 26, 24] use this approach to find approximate (or even optimal) minimum quotient edge-cuts and edge-separators in planar graphs. As suggested in [26], a different notion of a dual graph is a (partial) analogue for vertex cuts.

Face-Vertex-Dual Graphs.. Let $G_0(V_0, E_0)$ be a plane graph. The *face-vertex dual* (in short *fv-dual*) of G_0 is a graph

[3]The property called *edge-dual* here is traditionally called the *dual* of a planar graph [8]. We call it *edge-dual* here to distinguish it from a different notion of a dual graph that we shall focus on later.

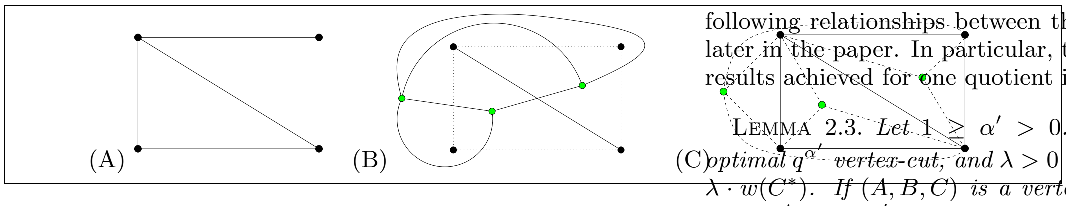

Figure 3: (A) A planar graph, (B) its edge-dual, and (C) its face-vertex-dual.

$G_D(V_D, E_D)$ where $V_D = V_0 \cup V_f$, each vertex of V_f corresponds to a distinct face of G_0, and E_D includes all the edges of E_0 and additional edges that connect every new vertex $v \in V_f$ to all the vertices (of V_0) in the boundary of the face corresponding to v. More precisely, every face f of G_0 has a vertex $v_f \in V_f$ positioned in f with edges to the boundary vertices of f such that for every edge $e_{u,v}$ on the boundary of f, there are edges between u, v, v_f that are the boundary of a face of G_D (thus, there may be parallel edges in G_D). Note that G_D is also a planar graph. An example of a plane graph and its fv-dual is shown in Figure 2.1(A),(C).

The fv-dual graph is not a *dual* graph in the classic sense [8, 6]. In particular, it has more vertices and edges than G_0, its dual is not G, and there is no exact correspondence between simple cycles in G_D and cutsets in G.

In the rest of this paper we use the following conventions. Let $G_0 = (V_0, E_0)$ be the input planar graph. Let $G_D = (V_D, E_D)$ be the face-vertex-dual of G_0. ≪Does not look like the right place for this? –Robi≫

By the definition of the fv-dual graph, we have the following properties.

PROPOSITION 2.2. *If G_0 is a plane graph without self-loops then G_D is a plane graph with the following properties. (1) The boundary of each face (is a cycle that) has length 3 and contains one edge from E_0 and two edges from $E_D \setminus E_0$. (2) One endpoint that belongs to V_0. (In particular, this edge is not a self-loop.)*

2.2 Vertex-Cuts

In the rest of this paper we allow a vertex-cut to be *trivial*, i.e., to have either A or B empty. This is convenient for our purposes, and does not affect applications, as will be seen later in the paper.

Quotient vertex-cuts.. Let $G(V, E)$ be a graph with a vertex weight function $w : V \to \mathbb{N}$ and a vertex cost function $c : V \to \mathbb{N}$. We define the quotient cost of a vertex-cut with respect to some parameter $\alpha \geq 0$. That is, the *α-quotient cost* of a vertex-cut (A, B, C) is

$$q^\alpha(A, B, C) = \frac{c(C)}{min\{w(A), w(B)\} + \alpha \cdot w(C)}$$

Note that the case $\alpha = 0$ matches the traditional definition of a vertex separator. We define the special case of $c(C) = 0$ and $min\{w(A), w(B)\} + \alpha \cdot w(C) = 0$ to have quotient 0. The *b-limited α-quotient cost* of a vertex-cut (A, B, C) is

$$q_b^\alpha(A, B, C) = \frac{c(C)}{min\{w(A) + \alpha w(C), w(B) + \alpha w(C), bW\}}$$

where $W = w(V)$ is the total weight of the vertices. Notice that $\forall \alpha \leq 1$, $q_1^\alpha(A, B, C) = q^\alpha(A, B, C)$. We will use the following relationships between the different quotient costs later in the paper. In particular, these allow us to translate results achieved for one quotient into another.

LEMMA 2.3. *Let $1 \geq \alpha' > 0$. Let (A^*, B^*, C^*) be an optimal $q^{\alpha'}$ vertex-cut, and $\lambda > 0$ such that $w(A^*), w(B^*) \geq \lambda \cdot w(C^*)$. If (A, B, C) is a vertex-cut with $q^0(A, B, C) \leq \rho \cdot (1 + \frac{\alpha'}{\lambda}) \cdot OPT^{\alpha'}$, then $w(A), w(B) \geq \Omega(\min(\lambda, \frac{1}{\rho - 1})) \cdot w(C)$ (precisely, $w(A), w(B) \geq w(C) \cdot \frac{\alpha'}{(\rho-1) + \frac{\rho}{\lambda}}$).*

LEMMA 2.4. *Let $\alpha' > \alpha \geq 0$, $b \leq 1$, (A^*, B^*, C^*) an optimal $q_b^{\alpha'}$ vertex-cut, and $\lambda > 0$ such that $w(A^*), w(B^*) \geq \lambda \cdot w(C^*)$. If (A, B, C) is a vertex-cut with $q_b^\alpha(A, B, C) \leq \rho \cdot OPT_b^\alpha$, then*

$$q_b^{\alpha'}(A, B, C) \leq \rho \cdot (1 + \frac{\alpha' - \alpha}{\lambda + \alpha}) \cdot OPT_b^{\alpha'}.$$

If $\alpha = 0$, then also, $w(A), w(B) \geq w(C) \cdot \frac{\alpha'}{(\rho-1) + \frac{\rho}{\lambda}}$.

One consequence of the last lemma is that for $\alpha = 0$, $\alpha' = 1$ we get $q_b^1(A, B, C) \leq \rho \cdot (1 + \frac{1}{\lambda}) \cdot OPT_b^1$. An interesting special case is when the cost and weight are uniform in planar graphs. We get that $\lambda \geq \Omega(\sqrt{|V|})$ from the Lipton-Tarjan theorem (there is always a separator of size $O(\sqrt{|V|})$). In that case, there is negligible difference between the quotients q_b^0 and q_b^1.

LEMMA 2.5. *Let $\alpha' > \alpha \geq 0$. Given a vertex-cut (A, B, C) whose b-limited α-quotient cost is within factor ρ of the minimum, one can find a vertex-cut (A', B', C') whose b-limited α'-quotient cost is within factor $\rho + 1$ of the minimum. (In fact, (A', B', C') is either the given cut (A, B, C) or some trivial cut with $|C'| = 1$.)*

≪What about the converse, i.e. going from α' to α? –Robi≫

Balanced Vertex-Cuts.. We define a balanced vertex-cut with respect to two parameters $b, \alpha \geq 0$. That is, a vertex-cut (A, B, C) is *(b, α)-balanced* if

$$\min\{w(A), w(B)\} + \alpha w(C) \geq bW.$$

2.3 Where is the problem

≪We now justify why our approach/problem is more complicated than Parks-Phillips. Shouldn't this be in the introduction? –Robi≫ ≪Motivation by figure with a cycle of cycles; explain why it is difficult to define in terms of the original graph using a star as an example. –Robi≫

First, notice that limiting ourselves to vertex-cuts that correspond to simple cycles leads to vertex-cuts that are arbitrarily far from the optimal. Figure 4 presents an example of a graph in which a simple cycle will not identify the minimum quotient cut. Divide the vertices in this graph into four types, t_1, t_2, t_3, t_4. If our graph has d cycles touching the center vertex, then we set $w(t_1) = \frac{bW}{d}$, $w(t_2) = 0$, $w(t_3) = (1 - b)W$, $w(t_4) = 0$, $c(t_1) = \infty$, $c(t_2) = 10$, $c(t_3) = \infty$, and $c(t_4) = 0$. This graph is triangular, so taking the face-vertex graph of this graph does not add any new cycle-induced vertex-cuts. Clearly, the optimal weight is given when vertices of type t_1 are in one side of the cut and that of type t_3 is in the other side. Vertices of types t_1, t_3 cannot be in the separator because of their large cost.

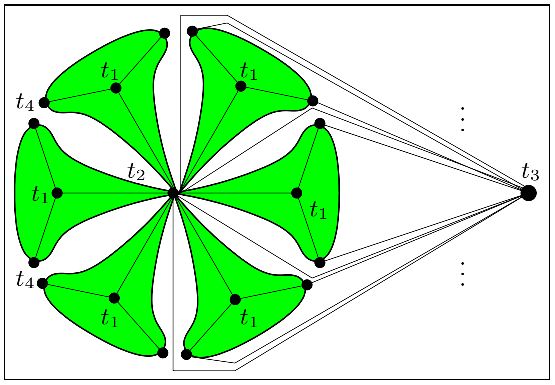

Figure 4: An example where any vertex-cut defined by a simple cycle has quotient cost that is arbitrarily far from the optimum.

The optimal quotient is achieved by selecting the vertices on the the cycles touching the center vertex. Any simple cycle in this graph separates at most one of the vertices of type t_1 from the rest, thus separating a factor d less weight at the same cost as the optimal quotient vertex-cut.

The main text below outlines the full proof of our arguments, breaking it into several relatively simple propositions, whose formal proof is given in Appendix **??**. Unless stated otherwise, all vertices, edges, faces and cycles are with respect to G_D (and not, say, G_0).

3. STRUCTURAL THEOREM

The main result of this section is that in every plane graph G_0 there is a vertex-cut that is nearly optimal in terms of quotient cost and also has a certain "nice" structure. Informally, this structure means that the vertex-cut corresponds to a collection of directed cycles in G_D that are arranged in a tree-like structure of constant depth; in short, we shall call it CAST (Cycles Arranged in a Shallow Tree). The intuition behind this structure is given in Section 2.3.

The precise statement of our basic structural theorem and the relevant definitions (including CAST) are given in Section 3.1. Its proof is then given in Section 3.2. Finally, we devise various extensions in Section 3.3.

3.1 Definitions and basic structural theorem

A face f (vertex v) of G_D is said to be *enclosed* by a directed simple cycle D (in G_D) if f (respectively, v) is contained in the region of $\mathbb{R}^2 \setminus D$ that is to the left of one (and thus all) of the edges in D. A set $\mathcal{D}$ of directed cycles (in G_D) is *proper* if every face of G_D is enclosed by at most one cycle in $\mathcal{D}$. Such a proper set $\mathcal{D} = \{D_1, \dots, D_k\}$ defines a vertex-cut (A, B, C) of G_0, as follows. Let C_i be the vertices of V_0 that appear in the cycle D_i, and let B_i be the vertices of V_0 that are enclosed by D_i. Then let $C := \cup_{i=1}^k C_i$, $B := (\cup_{i=1}^k B_i) \setminus C$ and $A := V_0 \setminus (C \cup B)$. To see that (A, B, C) is a vertex-cut, observe that an edge in G_0 between a vertex of A and a vertex of B must have its two endpoints in different regions of $\mathbb{R}^2 \setminus D_i$ for some i. Also, $B \cap C = \emptyset$ because $\mathcal{D}$ is proper.

The *auxiliary graph* of a proper set $\mathcal{D}$ (of directed simple cycles) is the bipartite graph $\tilde{G}(\tilde{V}_1, \tilde{V}_2, \tilde{E})$ where $\tilde{V}_1 := S$, $\tilde{V}_2 := V_D$, and $\tilde{E} := \{(d, v) : v \in V_D$ belongs to the cycle $d \in S\}$. Such a proper set $\mathcal{D}$ is called a *d-CAST* if its auxiliary graph $\tilde{G}$ is a forest (i.e. contains no cycles) and each of its connected components is a tree that can be rooted at some vertex $v \in \tilde{V}_2 = V_D$ so that its depth is at most $2d$. For example, the cycles drawn in Figure 4 form a 1-CAST. An $O(1)$-CAST is called in short a CAST.

THEOREM 3.1 (BASIC STRUCTURAL THEOREM). *Let $b \geq 0$ and $\alpha \geq 0$. Then every planar graph G_0 has a vertex-cut that is defined by a d-CAST in G_D, such that its b-limited α-quotient cost is within a factor of $\frac{4}{3}\frac{d}{d-1}+1$ for $\alpha \geq 0$ from the correspoding minimum quotient cost (over all vertex-cuts) in G_0. For $\alpha = 0$ the factor improves to $\frac{4}{3}\frac{d}{d-1}$.*

≪I think it is not necessary to have simultaneous guarantee? –Robi≫ ≪Is a trivial cut alwyas a 1-CAST? –Robi≫

We prove this theorem in Section 3.2. We note that the constant $4/3$ is tight, as shown by Figure **??**. ≪Add a figure of 3 nested cycles all touching same vertex with weights $1/4, 3/8, 1/4, 1/8$. –Robi≫ However, we can improve over this constant by allowing a more complicated structure than CAST. We can also show that the constant loss in the α-quotient cost for $\alpha > 0$ can be improved by relying on certain assumptions that include the unit-weight case. In fact, in the unit-cost case the CAST structure can be replaced with just one simple cycle (in G_D) at the expense of a larger constant than $4/3$. These extensions are described in Section 3.3.

3.2 Proof of basic structural theorem

The proof of Theorem 3.1 consists of three steps, as follows. Let $(\hat{A}, \hat{B}, \hat{C})$ be any vertex-cut in a plane graph G_0 (e.g., a minimum quotient cut). We first show in Section 3.2.1 that there exists in G_0 a vertex-cut $(\hat{A}', \hat{B}', \hat{C}')$ that has nearly the same quotient cost and for which $\hat{A}'$ corresponds to some "connected region" in the plane. We then prove in Section 3.2.2 that $\hat{B}'$ must correspond to a union of "connected regions" in the plane, each described by a simple directed cycle (in G_D), with the important property that these simple cycles form a forest (in a sense to be defined). By averaging arguments, we show in Section 3.2.3 that every such forest must contain a shallow subtree (namely, subtree of constant depth) that defines a vertex-cut $(\hat{A}'', \hat{B}'', \hat{C}'')$ whose quotient cost is nearly the same.

Letting $(\hat{A}, \hat{B}, \hat{C})$ be a minimum quotient cut in G_0, the three steps above yield a cut $(\hat{A}'', \hat{B}'', \hat{C}'')$ that has a nearly optimal quotient cost and also has a CAST structure, thus proving the main result of this section. The three steps above apply for 0-quotient cost (i.e. $\alpha = 0$), but the results easily extend to 1-quotient cost (i.e., $\alpha = 1$) by applying Lemma 2.5.

3.2.1 A connected region in the plane

The first step in the proof of the basic structural theorem is to exhibit in G_0 a vertex-cut $(\hat{A}', \hat{B}', \hat{C}')$ that has nearly the same quotient cost as $(\hat{A}, \hat{B}, \hat{C})$ and for which $\hat{A}'$ corresponds to some "connected region" in the plane. Intuitively, the vertex-cut $(\hat{A}, \hat{B}, \hat{C})$ partitions the plane into connected regions, each associated with one of $\hat{A}, \hat{B}, \hat{C}$; a careful rearrangement of the connected regions associated with $\hat{A}$ and $\hat{B}$ shall yield the desired vertex-cut $(\hat{A}', \hat{B}', \hat{C}')$. Below we give the formal argument.

We start by labeling each face of G_D by $\hat{A}$ if its boundary contains a vertex of $\hat{A}$, by $\hat{B}$ if its boundary contains a vertex of $\hat{B}$, and by $\hat{C}$ if neither event happens. It is easy to verify (see Appendix B) that every face of G_D has exactly one label.

Let $\mathrm{CC}(\hat{A})$ denote the set of connected components of the faces labeled $\hat{A}$ (recall Definition 2.1), and similarly for $\hat{B}$ and $\hat{C}$. Let $\mathrm{CC}(G_D) := \mathrm{CC}(\hat{A}) \cup \mathrm{CC}(\hat{B}) \cup \mathrm{CC}(\hat{C})$ be the set of all these connected regions. Each connected region $R \in \mathrm{CC}(G_D)$ corresponds to a set of vertices in G_0, namely,

$$V(R) := \{v \in V_0 : v \text{ is in the boundary of a face in } R\}. \tag{1}$$

Define the *weight of a connected region* R as $\tilde{w}(R) := \sum_{v \in V(R) \setminus C} w(v)$. Notice that ignoring the weight of vertices from C corresponds to 0-quotient cost (i.e., $\alpha = 0$). For a set S of connected regions, define $V(S) := \cup_{R \in S} V(R)$ and $\tilde{w}(S) := \sum_{R \in S} \tilde{w}(R)$. We say that a face f is in S if $f \in R$ for some connected region $R \in S$. It is easy to verify (see Appendix B that

$$\tilde{w}(\mathrm{CC}(\hat{A})) = w(\hat{A}),\ \tilde{w}(\mathrm{CC}(\hat{B})) = w(\hat{B}),\ \tilde{w}(\mathrm{CC}(\hat{C})) = 0,$$

and it then immediately follows that $\tilde{w}(\mathrm{CC}(G_D)) = \tilde{w}(\mathrm{CC}(\hat{A})) + \tilde{w}(\mathrm{CC}(\hat{B})) + \tilde{w}(\mathrm{CC}(\hat{C})) = w(\hat{A} \cup \hat{B})$.

PROPOSITION 3.2. *There exists a partition* $\mathrm{CC}(G_D) = S \cup \bar{S}$ *such that the faces of S form a connected region and* $\min\{\tilde{w}(S), \tilde{w}(\bar{S})\} \geq \frac{3}{4} \min\{w(\hat{A}), w(\hat{B})\}$.

PROOF. Without loss of generality assume that $w(\hat{A}) \geq w(\hat{B})$ and let $b := w(\hat{B})/w(\hat{A} \cup \hat{B}) \leq \frac{1}{2}$. ≪$b$ is confusing - why not use β? –Robi≫ We say that two connected regions R_1, R_2 are *adjacent* if they contain two faces $f_1 \in R_1$, $f_2 \in R_2$ that are adjacent in G_D. Apply the following procedure on $\mathrm{CC}(G_D)$. Start with $S_1 = \{R_{\max}\}$ where $R_{\max} := \mathrm{argmax}_{R \in \mathrm{CC}(G_D)} \tilde{w}(R)\}$. If $b > \frac{3}{8} w(\hat{A} \cup \hat{B})$ then let $S_2 = \{R'_{\max}\}$ where $R'_{\max} := \mathrm{argmax}_{R \in \mathrm{CC}(G_D) \setminus S_1} \tilde{w}(R)\}$; otherwise, set $S_2 = \emptyset$, thus ignoring it throughout this procedure. Now repeatedly add to either S_1 or S_2 a connected region $R \in \mathrm{CC}(G_D) \setminus (S_1 \cup S_2)$ that is adjacent to at least one connected region in S_1 or S_2, respectively. Stop this iterative process when $\tilde{w}(S_i) \geq \max\{\frac{1-b}{2}, \frac{1}{8} + \frac{b}{2}\} w(\hat{A} \cup \hat{B})$ for some $i \in \{1, 2\}$, letting $S := S_i$ and $\bar{S} := \mathrm{CC}(G_D) \setminus S$. By construction, the faces in S forms a connected region.

We first claim that at every iteration there exists a connected region $R \in \mathrm{CC} \setminus (S_1 \cup S_2)$ that is adjacent to some connected region in either S_1 or S_2. To this end, observe that $0 \leq b \leq \frac{1}{2}$, so whenever $\tilde{w}(S_1), \tilde{w}(S_2) < \max\{\frac{1-b}{2}, \frac{1}{8} + \frac{b}{2}\} w(\hat{A} \cup \hat{B})$, we also have that $\tilde{w}(S_1) + \tilde{w}(S_2) < \max\{1 - b, \frac{1}{4} + b\} w(\hat{A} \cup \hat{B}) \leq \tilde{w}(\mathrm{CC}(G_D))$, and thus $\mathrm{CC}(G_D) \setminus (S_1 \cup S_2)$ is not empty, i.e., contains at least one connected region. Let f be a face in such a connected region of $\mathrm{CC} \setminus S$, and let f' be a face in a connected region of S_1. Since the set of all faces in G_D is connected (in the sense of Definition 2.1), there is a sequence of faces $f = f_1, \ldots, f_k = f'$ such that f_i and f_{i+1} are adjacent for all $i < k$. It follows that there must exist two successive faces f_i in $\mathrm{CC} \setminus (S_1 \cup S_2)$ and f_{i+1} in either S_1 or S_2. The claim follows.

The above procedure always terminates since the number of iterations is bounded by the number of connected regions in $\mathrm{CC}(G_D)$, and in turn, by the (finite) number of faces in G_D.

Finally, some calculations show (see Appendix B) that $\min\{\tilde{w}(S), \tilde{w}(\bar{S})\} \geq \frac{3}{4} \min\{w(\hat{A}), w(\hat{B})\}$. In fact, if $b \leq \frac{1}{3}$ then $\min\{\tilde{w}(S), \tilde{w}(\bar{S})\} \geq \min\{w(\hat{A}), w(\hat{B})\}$. □

Finally, let $S, \bar{S}$ be as in Proposition 3.2. We use (1) to define the following subsets of V_0:

$$\hat{A}' := V(S) \setminus V(\bar{S}),\ \ \hat{B}' := V(\bar{S}) \setminus V(S),\ \ \hat{C}' := V(S) \cap V(\bar{S}). \tag{2}$$

The next easy to verify proposition (see Appendix B) summarizes the first step in the proof of the basic structural theorem.

PROPOSITION 3.3. $(\hat{A}', \hat{B}', \hat{C}')$ *is a vertex-cut of G_0 with* $\hat{C}' \subseteq \hat{C}$ *and* $\min\{w(\hat{A}'), w(\hat{B}')\} \geq \frac{3}{4} \min\{w(\hat{A}), w(\hat{B})\}$, *where $\hat{A}'$ corresponds to the connected region S.*

3.2.2 A forest of simple cycles

The second step in the proof of the basic structural theorem is to show that $\hat{B}'$ corresponds to a union of connected regions in the plane, each described by a simple cycle (in G_D), with the important property that these simple cycles form a "forest". Recall that $\hat{A}'$ and $\hat{B}'$ are defined in (2) so that they essentially correspond to S and $\bar{S}$, respectively. The point is that since S a connected region (by construction), the connected regions of $\bar{S}$ are arranged in a "forest". Below we give the formal argument. (Recall the definitions in Section 3.1.)

Slightly abusing notation, we denote the set of connected components of all the faces in the connected regions in $\bar{S}$ by $\mathrm{CC}(\bar{S}) := \mathrm{CC}(\cup_{R \in \bar{S}} \mathrm{CC}(R))$. It can be shown (see Appendix B that for every connected region $R \in \mathrm{CC}(\bar{S})$ there is a directed simple cycle D_R such that R is exactly the set of faces (of G_D) that are enclosed by D_R. (We remark that the fact that each cycle is simple relies on S being a connected region.) Let $\mathcal{D}(\bar{S})$ be the set of directed simple cycles D_R that enclose the connected regions $R \in \mathrm{CC}(\bar{S})$. It is easy to verify (see Appendix B) that $\mathcal{D}(\bar{S})$ is a proper set of cycles that defines (in the sense of Section 3.1) the vertex-cut $(\hat{A}', \hat{B}', \hat{C}')$ described in (2).

Once we describe the cut $(\hat{A}', \hat{B}', \hat{C}')$ using the directed cycles $\mathcal{D}(\bar{S})$ we can investigate the arrangement of these cycles in the plane, and prove the following proposition (see Appendix B). The main idea is that any "cycle of cycles" implies that S contains at least two connected regions, see Figure **??**.

PROPOSITION 3.4. *Let $\tilde{G}(\tilde{V}_1, \tilde{V}_2, \tilde{E})$ be the auxiliary graph of $\mathcal{D}(\bar{S})$. Then $\tilde{G}$ is a forest (i.e., contains no cycles).*

≪Add here the cycle-of-cycles figure. –Robi≫

3.2.3 A shallow tree of simple cycles

The third step in the proof of the basic structural theorem is to show that the forest $\tilde{G}$ must contain a subtree that is shallow (i.e., has small depth) and its cycles define a vertex-cut whose quotient cost is nearly the same (as that of $(\hat{A}', \hat{B}', \hat{C}')$). The main idea here is to remove every $2d$-th level in the forest $\tilde{G}$ (so that we lose at most $1/d$ of the cut's weight and do not increase its cost) and then apply averaging arguments on the resulting connected components. We can thus prove the following proposition (see Appendix B).

Proposition 3.5. *Let $d \geq 2$ be an integer and let $b \geq 0$. Then there is a d-CAST in $\mathcal{D}(\bar{S})$ that defines a vertex-cut $(\hat{A}'', \hat{B}'', \hat{C}'')$ with quotient $q_b^0(\hat{A}'', \hat{B}'', \hat{C}'') \leq \frac{d}{d-1} \cdot q_b^0(\hat{A}', \hat{B}', \hat{C}')$.*

Finally, we can prove Theorem 3.1. By applying Lemma 2.5 we can then extend the result to α-quotient cost for arbitrary α, with a slightly larger constant factor.

Proof of Theorem 3.1. Let $(\hat{A}, \hat{B}, \hat{C})$ be a vertex-cut of minimum b-limited 0-quotient cost (i.e., $\alpha = 0$). For $\alpha = 0$, the claimed result then follows from Propositions 3.3 and 3.5. For any $\alpha > 0$ we have by Lemma 2.5 that either $(\hat{A}'', \hat{B}'', \hat{C}'')$ or some trivial cut (A', B', C') with $|C'| = 1$ (which is clearly a CAST) has b-limited α-quotient cost within a factor of $\frac{4}{3}\frac{d}{d-1} + 1$ from optimal. □

3.3 Extensions

We can extend Theorem 3.1 in various directions. First, we show that the loss in the quotient cost is actually smaller if there exists an optimal quotient cut (A^*, B^*, C^*) with $w(C^*) \ll w(A^*), w(B^*)$; in particular, in the unit-weight case there is always an optimal quotient cut (A^*, B^*, C^*) with $w(C^*) \leq O(1/\sqrt{n}) \leq \min\{w(A^*), w(B^*)\}$, and thus the loss in the 1-quotient cost (i.e., for $\alpha = 1$) becomes arbitrarily close to 4/3. This result follows by combining Theorem 3.1 and Lemma 2.4 (see Appendix B).

Second, we reduce the loss in the quotient cost (for general weights) to $1 + \epsilon$, for arbitrary fixed $\epsilon > 0$, by somewhat relaxing the structural requirement (i.e., allowing a more complicated structure than CAST). Technically, we generalize Proposition 3.3 by allowing $\hat{A}'$ to correspond to $O(1/\epsilon)$ connected regions. The proof is a modification of that of Proposition 3.2 that distinguishes between "big" and "small" connected regions (see Appendix B).

Third, we show for the unit-cost case that the CAST structure can be replaced with just one simple cycle (in G_D) at the expense of a larger constant factor loss in the quotient cost. Technically, we replace a 1-CAST structure with one of its simple cycles (see Appendix B.

Proposition 3.6. *If G_0 has an optimal b-limited α-quotient cut (A^*, B^*, C^*) with $\lambda w(C^*) \leq w(A^*), w(B^*)$ for some $\lambda > 0$, then the factor in Theorem 3.1 can be improved to $\frac{4}{3}\frac{d}{d-1}(1 + \frac{\alpha}{\lambda})$.*

Proposition 3.7. *There exists a vertex-cut $(\hat{A}', \hat{B}', \hat{C}')$ of G_0 where $\hat{A}'$ corresponds to $O(1/\epsilon)$ connected regions, $\hat{C}' \subseteq \hat{C}$ and $\min\{w(\hat{A}'), w(\hat{B}')\} \geq (1-\epsilon)\min\{w(\hat{A}), w(\hat{B})\}$.*

Proposition 3.8. *Let $b \geq 0$ and $\alpha \geq 0$. Then every unit-cost planar graph G_0 has a vertex-cut that is either defined by a directed cycle or is trivial such that its b-limited α-quotient cost is within a factor of $6\frac{1}{3}$ for $\alpha \geq 0$ from the minimum.*

4. ALGORITHM FOR QUOTIENT-CUT

In this section we use the structural theorems of Section 3 to devise an algorithm for finding in planar graphs a cut with near-optimal 1-quotient cost (i.e., $\alpha = 1$). Given a planar input graph G, we fix an embedding G_0 of it in the plane, compute its face-vertex graph G_D, and then apply the algorithm described described below to search, roughly speaking, for a d-CAST structure in G_D (namely, cycles arranged in a shallow tree) with minimum α-quotient. The structural theorems guarantee that a search restricted to d-CASTs will find a vertex-cut whose 1-quotient is within a small constant factor of the minimum.

The search algorithm for d-CASTs is described in Section 4.1 and its correctness proofs are given in Section 4.2. Our algorithm is inspired by those of [25, 26, 24] for finding (or approximating) the minimum quotient edge-cut, and we point out some of the similarities and differences throughout the description. Proofs omitted from this section appear in Appendix C.

4.1 Algorithm and related structures

Our algorithm goes roughly as follows. We start by translating vertex weights of G_D to face weights in G_D, so that for any simple directed cycle, the total weight of the enclosed faces approximates the total weight of the enclosed vertices. Now, similarly to [24], we construct a search graph G_s, which is just a directed version of G_D with edge weights; these edge weights are carefully defined so that for every simple directed cycle, the total weight of the edges on the cycle is equal to the total weight of the faces it encloses. Finally, our main routine finds among a certain family of closed walks (i.e., cycles, but not necessarily simple ones) in G_s, a walk that has minimum cost to weight quotient; here, the weight of a walk is the sum of the weights of its edges and the cost of a walk is (in principle) the sum of the costs of the vertices it visits. This family of walks has three important properties. First, for any d-CAST there is in this family a corresponding walk whose weight and cost are equal to that of the d-CAST. However, this family contains also walks that do not correspond to a CAST, so the second important property is that from any walk in this family we can extract (efficiently) a vertex-cut (that need not be a d-CAST) whose quotient cost is no larger than that of the whole walk. Third, we have the algorithmic property that one can efficiently optimize the cost to weight quotient over this family (using dynamic programmming).

Section 4.1.1 describes the way we assign weight to faces, Section 4.1.2 presents the search graph G_s, Section 4.1.3 defines a certain kind of walks in G_s, and Section 4.1.4 presents our dynamic programming algorithm for optimizing over these walks.

4.1.1 *Assigning Weight to Faces*

We define the cost of vertices $v \in V_D$ to be the same as their cost in V_0 (if $v \in V_0$) or 0 (if $v \notin V_0$). We define weights for faces in G_D by distributing the weight of every vertex from V_0 equally among the faces of G_D incident to it. Formally, the weight of face f of G_D is $\sum_{v \in f} w(v)/\deg(v)$, where $\deg(v)$ is the degree of v in G_D. (Note that in contrast, the edge-quotient case [25, 26, 24] is simpler, with each face in the dual associated with a vertex whose weight that face receives.) The next proposition shows that the weight of faces enclosed in a cycle (not necessarily simple) is close to the weight of vertices enclosed by that cycle.

Proposition 4.1. *Let D be a cycle in G_D and let C denote the set of vertices on it. Let l denote the number of regions in $\mathbb{R}^2 \setminus D$, and let V_i and F_i denote the sets of vertices and faces, respectively, in the ith region. Then, for every $I \subseteq \{1, \ldots, l\}$, we have $w(\cup_{i \in I} V_i) \leq w(\cup_{i \in I} F_i) \leq w(\cup_{i \in I} V_i) + w(C)$.*

4.1.2 *The Search Graph*

Similar to [24], we define the *search graph* G_s to be the digraph obtained from G_D by replacing every undirected edge (u, v) by a pair of directed edges (u, v) and (v, u). Let the vertices of G_s have the same cost they have in G_D. In the rest of this subsection we associate weight with the (directed) edges of G_s.

Let T be any spanning tree of the graph G_D. Choosing any vertex r on the outer face as a root, orient the tree according to this embedding. That is, order the children of any vertex in a counter-clockwise direction; for the root vertex r start with the outer face, and for any other vertex start with the edge leading to its parent.

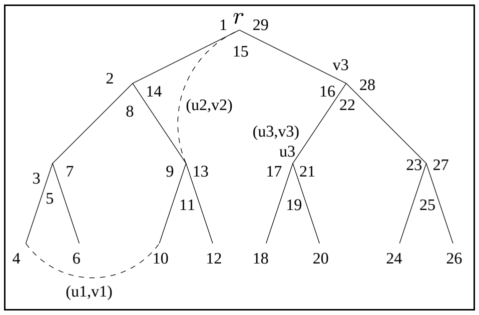

Figure 5: Labeling the regions of the tree T.

The tree edges incident to a vertex v divide the plane immediately surrounding v into $\deg(v)$ regions. We give each such region a unique labeling as follows: Starting at the root, traverse the tree in a Depth First Search manner, using the order on children defined above. Each step in this traversal corresponds to visiting a vertex from either its parent or from a child. Thus step 1 is visiting the root, step 2 is visiting the root's first child and so on. Each time a vertex is visited in this traversal, one immediate region around v, namely, the one between the edges used to enter and to exit v, is labeled with the traversal's step number. Formally, for a vertex v let e_1 be the edge to its parent and let $e_2, \ldots, e_{\deg_T(v)}$ be edges to its children, where $\deg_T(v)$ is the degree of v in the tree T. We then label the region between e_i and $e_{(i+1) \bmod \deg_T(v)}$ by the step of traversal at which vertex v is visited for the ith time, denoted $t(v, i)$. See for example Figure 5.

We give weight to every directed edge (u, v) in G_s as follows. If the underlying undirected edge belongs to the tree T, we define $w(u, v) = 0$; otherwise, (u, v) is embedded in some region relative to each of its endpoints. Let $t(u, i)$ and $t(v, j)$ be the labels of these regions. Assume without loss that $t(u, i) < t(v, j)$ and let $w(u, v)$ be the total weight of the faces enclosed by the simple cycle that (u, v) closes in the tree T (i.e., take the unique path in T between u and v and add to it the edge (u, v)); let $w(v, u) = -w(u, v)$. Thus, edges going left-to-right have positive weight and their reverse edges have negative weight.

4.1.3 Walks With Pebbles

In this subsection we define the walk in the basis for the dynamic program. A *walk* of *length* k in G_s is a sequence of (directed) edges $(u_1, v_1), \ldots, (u_k, v_k)$ such that $v_i = u_{i+1}$ for all $1 \le i < k$. The walk is *closed* if $v_k = u_1$. (Notice that a closed walk is just a directed cycle.) A closed walk is *simple* if it contains k distinct edges and k distinct vertices; A non-closed walk is *simple* if it contains k distinct edges and $k+1$ distinct vertices. For a walk π, let π_{e_i} (for $i = 1, \ldots, k$) denote the ith edge in the walk (namely, (u_i, v_i)), and let π_i (for $i = 1, \ldots, k+1$) denote the ith vertex in the walk π (namely, $u_i = v_{i-1}$). The *total weight* of a walk π of length k is $w(\pi) = \sum_{i=1}^{k} w(\pi_{e_i})$, and its *total cost* is $\sum_{i=1}^{k} c(\pi_i)$. Notice that in these definitions, if the walk is not simple then some costs and weights may be counted more than once. We define the *actual cost* of a walk π of length k as the sum of costs of all the *distinct* vertices in $\{\pi_1, \ldots, \pi_k\}$. For edge quotient cuts [24] it suffices to count the total cost of a walk because there it suffices to seek a simple walk in G_s. However, in our case the walk is a tree-of-cycles, and may visit some vertices more than once. Thus, our total cost might be a poor upper bound on the actual cost of the walk. We need a walk that allows, for a limited number of vertices, additional visits to these vertices at no additional cost; however, we can limit these "free" visits to be in a counter-clockwise order, without completing a full cycle. We define this walk in what follows.

A *d-pebble walk* is a walk that uses at most d pebbles as follows. In the beginning of the walk all d pebbles are unassigned, and during the walk pebbles change their assignment only according to these rules: (i) when the walk enters a vertex v that has no pebble, it must assign to v one of the unassigned pebbles; (ii) when the walk leaves a vertex v that has a pebble, it may (but does not have to) unassign the pebble from v. Note that a "simple visit" to v can be achieved by assigning a pebble to v upon entrance and then unassigning it upon leaving v, (in the next edge of the walk). The pebbles allow for a tighter upper bound of the actual cost of the walk, as follows. Let us call a visit to a vertex that already has a pebble a *free visit*, and let the *counted cost* of a walk π be $\sum c(\pi_i)$, where the summation is over all the non-free visits $1 \le i \le k$. Clearly, the counted cost of a walk upper bounds its actual cost.

In d-pebble walks we restrict the free visits to have a certain order in the plane. Specifically, between the moment that a pebble is assigned to v and the moment in which it is unassigned from v, the walk uses edges that are incident to v only in a counter-clockwise order around v without completing a full cycle. Formally, every pebble is represented by a *tuple* $\langle v, e_{\text{first}}, e_{\text{last}} \rangle$, where v is the vertex to which the pebble is assigned, e_{first} is the incoming edge to v that was used when the pebble was assigned to v, and e_{last} is the last edge that was used by the walk among all edges incident to v.

4.1.4 A Dynamic Programming Algorithm

Dynamic programming allows us to find a walk of minimum counted cost among all d-pebble walks of length $3n$ and total weight $w \in \{-W, \ldots, W\}$. Formally, let T be the following dynamic programming table. Let $T(l, s, t, \{p_1, \ldots, p_d\}, w)$ be the minimum cost of d-pebbled walk of length $0 \le l \le 3n$ that starts at s, ends at t, its pebbles are assigned (at the end of the walk) to $p_1, \ldots, p_d$, and has weight exactly w. Each p_i is either empty (in which case the ith pebble is not assigned) or a tuple $\langle v_i, e_{i,1}, e_{i,\text{last}} \rangle$ where v_i is a vertex (to which the ith pebble is assigned), $e_{i,1}$ is the incoming edge to v_i that was used when the pebble was assigned to v_i, and $e_{i,\text{last}}$ is the last edge, among all edges incident to v, that was used by the walk. We remark that a slightly faster algorithm can be devised using Dijkstra's shortest path algorithm. However, the dependency of the running time on d remains the same.

PROPOSITION 4.2. *Every value of T for $l+1$ can be computed from (some of the) values of T for l in time $O(n)$. Thus, The values $T(l, v, v, \emptyset, w)$ for all $l = O(n)$ and $w \in \{-W, \dots, W\}$ can be computed in time $W \cdot n^{2d+3}$.*

When the table T is complete, we choose the entry that minimizes $T(l, v, v, \emptyset, w)/w$ for $v \in V_D$, $w \leq W/2$.

≪Need to put the rest of the algorithm here. –Eyal≫

4.2 Proof of Algorithm Correctness

The cycle found by the dynamic program may or may not correspond to a CAST. In this section we prove that every CAST corresponds to some pebble walk of the same $T(l, v, v, \emptyset, w)/w$ ratio (thus, the minimum found by the dynamic program is at most that ratio), and also that every pebble walk that has a minimal $T(l, v, v, \emptyset, w)/w$ has a subwalk that is a pebble walk and that corresponds to a CAST whose quotient is at most $T(l, v, v, \emptyset, w)/w$ (thus, an minimal solution found by the dynamic program can be used to extract an optimal CAST).

4.2.1 A CAST Yields a Pebble Walk

The following proposition shows that every d-CAST D of face weight w_0 and cost c_0 yields a d-pebble walk of total weight w_0 and counted cost c_0. This implies that there is a vertex v on D such that $T(l, v, v, \emptyset, w_0) \leq c_0$ for some $l \leq 3n$.

PROPOSITION 4.3. *Let D be a d-CAST with weight w_0 (i.e., w_0 is the weight of the faces surrounded by the cycles in D) and cost c_0. Then, there exists a d-pebble walk of* total weight w_0 *and* counted cost c_0.

4.2.2 A Pebble Walk Yields a Vertex-Cut

We now prove that for every v, l and w_0 there exists a walk that defines a vertex-cut whose quotient cost is at most $T(l, v, v, \emptyset, w_0)/w_0$. The main hurdle here is that a d-pebble walk may not correspond to any vertex cut (e.g., when the interior and exterior coincide), and that even if it does, the total weight of the walk need not correspond to the weight separated by the vertex-cut it defines (because the weight of any face can contribute to it an arbitrary number of times). We now show that the walk can be decomposed so that the cost and weight decomposes appropriately. The main difficulty is in showing that the accumulated cost can be decomposed properly (Park and Phillips [24] showed that this can be done for walks in G_s when the total cost equals the actual cost for π).

The *net-count* of a face f by a walk π, denoted $\mathrm{nc}_\pi(f)$, is the (net) number of times that the weight of the face f contributes to $w(\pi)$. (See Section 4.1.3 for definition of $w(\pi)$.) The *number of visits* of a walk π in a directed edge e (of G_s), denoted $\mathrm{nv}_\pi(e)$ is the number of times that e appears in the walk π.

For a set A of integers, let $\Delta(A) := \max A - \min A$. For a closed walk π, let $\Delta_\pi(G_s) := \Delta(\{\mathrm{nc}_\pi(f) : f \text{ a face in } G_s\})$. For a vertex v, let F_v be the set of faces of G_s incident on v, and let $\Delta_\pi(v) = \Delta(\{\mathrm{nc}_\pi(f) : f \in F_v\})$. Thus, $\Delta_\pi(v)$ is the difference between the least-counted face and the most-counted face among those incident on v. The following proposition shows that every vertex's cost is counted in π at least $\Delta_\pi(v)$ times. Thus, $\Delta_\pi(v)$ is a lower bound on the cost accumulated for v in π.

PROPOSITION 4.4. *Let π be a closed pebble walk, and let v be a vertex in G_s. Then, v contributes its cost to the counted cost of π at least $\Delta_\pi(v)$ times.*

Now, we can decompose a d-pebble closed walk π into closed subwalks, each associated with its own cost and weight, in a way that is consistent with that of π. In particular, for every vertex that appears in more than one subwalk we can decompose the cost accumulated by the pebble-walk π such that this cost is contributed to every subwalk at least once. One of those subwalks is going to define our chosen vertex cut.

PROPOSITION 4.5. *Every walk π with $\Delta_\pi(G_s) \leq 1$ defines a vertex-cut (A, B, C) with $\mathrm{bal}^1(A, B, C) \geq \min\{w(\pi), W - w(\pi)\} \geq \mathrm{bal}^0(A, B, C)$ and $c(C)$ is at most the actual cost of π.*

COROLLARY 4.6. *Let A, B, C be a d-CAST cut. If $\lambda \cdot w(C) \leq \min\{w(A), w(B)\}$, then one can find a vertex-cut whose q^1 quotient is at most $(1 + \frac{1}{\lambda})q^1(A, B, C)$.*

PROOF. By Proposition 4.3, there exists a d-pebble walk π for (A, B, C) with cost $c_0 = c(C)$ and faces-weight w_0 such that $w(B) \leq w_0 \leq w(B) + w(C)$ and $w(A) \leq W - w_0 \leq w(A)+w(C)$. Thus, $\min\{w_0, W-w_0\} \geq \min\{w(B), w(A)\} \geq \frac{1}{1+\frac{1}{\lambda}} \min\{w(B) + w(C), w(A) + w(C)\}$.

By the definition of the dynamic programming table T, there is $v \in V_D$ and $l \leq 3n$ such that $T(l, v, v, \emptyset, w_0) \leq c(C)$. Let $T(w_0)$ be a shorthand for this $T(l, v, v, \emptyset, w_0)$. The algorithm finds a walk π for this table entry and breaks it into walks $\{\pi_i\}_{i \leq l}$ as promised by Proposition 4.6. By the averaging arguments of Park and Phillips [24, Theorem 2.2], there is one such walk π_i such that π_i defines a cut (A_i, B_i, C_i) with $c(C_i)$ is at most the actual cost of π_i, $w(A_i) + w(C_i) \geq w_i$ and $w(B_i) + w(C_i) \geq W - w_i$. For this walk π_i,

$$\frac{c(C_i)}{\min(w_i, W - w_i)} \leq \frac{c(\pi)}{\min(w(\pi), W - w(\pi))} = \frac{T(w_0)}{\min(w_0, W - w_0)} \leq (1+\lambda)\cdot$$

Thus, $\mathrm{bal}^1(A_i, B_i, C_i) \geq \min(w_i, W-w_i)$. Finally, $q^1(A_i, B_i, C_i) = \frac{c(C_i)}{bal^1(A_i,B_i,C_i)} \leq (1 + \lambda) \cdot \frac{c(C)}{bal^1(A,B,C)} = (1 + \lambda) \cdot q^1(A, B, C)$. This is the vertex-cut that the algorithm returns. □

Combining Theorem **??**, Lemma 2.4 and Corollary 4.8 we obtain the following result.

THEOREM 4.7. *Let (A^*, B^*, C^*) be an optimal q^1-quotient vertex-cut. Using d pebbles, our algorithm runs in time $O(W \cdot n^{3+2d})$ and finds a vertex-cut (A, B, C) with $q^1(A, B, C) \leq 3.5 \cdot (\frac{d}{d-1})^2 \cdot q^1(A^*, B^*, C^*)$, or there is a trivial vertex-cut with the same quotient bound. Furthermore, if $w(A^*), w(B^*) \geq \lambda \cdot w(C^*)$ and also $w(A), w(B) \geq \lambda \cdot w(C)$, for some $\lambda > 1$, then $q^1(A, B, C) \leq \frac{4}{3} \cdot \frac{d}{d-1} \cdot (1 + \frac{1}{\lambda})^2 \cdot q^1(A^*, B^*, C^*)$.*

≪State the result that we have for b-limited, and also for $\alpha < 1$. –Eyal≫

5. FROM QUOTIENT TO SEPARATOR

The following theorem shows that we can find a pseudo-approximation to a vertex separator using an approximation algorithm for a minimum-quotient vertex-cut.

THEOREM 5.1. *Let $1 \geq b \geq b' > 0$ and $0 < \alpha \leq 1$ such that $b'/\alpha \leq \min\{\frac{1}{3}, b\}$. Given a ρ-approximation algorithm for minimum b-limited α-quotient cost vertex-cuts one can find a (b', α)-balanced cut that is within a factor of $\frac{\rho}{b-b'/\alpha}$ from the minimum (b, α)-balanced cut.*

If G_0 is a graph with uniform cost and uniform weight, then $w(A') + \alpha w(C') \leq (1 + \frac{1}{\sqrt{n}})w(A') \leq (1 + \frac{1}{\sqrt{n}})(\frac{1}{2} + \frac{1}{2}\frac{b'}{\alpha})W$, and the last part of the proof shows that we get an approximation factor of $(1+\frac{1}{\sqrt{n}})\frac{\rho(1+b'/\alpha)}{2(b-b'/\alpha)}$. For $\alpha = 1$ we get an approximation factor of $(1 + \frac{1}{\sqrt{n}})\frac{\rho(1+b')}{2(b-b')}$. Finally, if also $b = \frac{1}{2}$ and $b' = \frac{1}{\beta}$, for some $\beta > 3$ we get an approximation factor of $(1 + \frac{1}{\sqrt{n}})\frac{\rho(\beta+1)}{\beta-2}$.

6. REFERENCES

[1] C. Ababei, N. Selvakkumaran, K. Bazargan, and G. Karypis. Multi-objective circuit partitioning for cutsize and path-based delay minimization. In *International Conference on Computer-Aided Design*, pages 181–186, 2002.

[2] N. Alon, P. Seymour, and R. Thomas. Planar separators. *SIAM J. Discrete Math.*, 7(2):184–193, 1994.

[3] E. Amir and S. McIlraith. Paritition-based logical reasoning. In *Proc. 7th Int'l Conference on Principles of Knowledge Representation and Reasoning*, pages 389–400. Morgan Kaufmann, 2000.

[4] Y. Aumann and Y. Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Comput.*, 27(1):291–301, 1998.

[5] T. N. Bui and C. Jones. Finding good approximate vertex and edge partitions is NP-hard. *Inform. Process. Lett.*, 42(3):153–159, 1992.

[6] R. Diestel. *Graph theory.* Springer-Verlag, New York, second edition, 2000.

[7] G. Even, J. Naor, S. Rao, and B. Schieber. Fast approximate graph partitioning algorithms. *SIAM J. Comput.*, 28(6):2187–2214, 1999.

[8] S. Even. *Graph algorithms.* Computer Science Press Inc., 1979.

[9] U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM J. Comput.*, 31(4):1090–1118, 2002.

[10] N. Garg, H. Saran, and V. V. Vazirani. Finding separator cuts in planar graphs within twice the optimal. *SIAM J. Comput.*, 29(1):159–179, 1999.

[11] N. Garg, V. V. Vazirani, and M. Yannakakis. Multiway cuts in directed and node weighted graphs. In *Automata, Languages and Programming, 21st ICALP*, volume 820 of *Lecture Notes in Computer Science*, pages 487–498. Springer-Verlag, 1994.

[12] N. Garg, V. V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM J. Comput.*, 25(2):235–251, 1996.

[13] R. M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23:309–311, 1978.

[14] P. Klein, S. A. Plotkin, and S. Rao. Excluded minors, network decomposition, and multicommodity flow. In *25th Annual ACM Symposium on Theory of Computing*, pages 682–690, May 1993.

[15] P. Klein, S. Rao, A. Agrawal, and R. Ravi. An approximate max-flow min-cut relation for undirected multicommodity flow, with applications. *Combinatorica*, 15(2):187–202, 1995.

[16] P. N. Klein, Se. A. Plotkin, S. Rao, and E. Tardos. Bounds on the max-flow min-cut ratio for directed multicommodity flows. Technical Report CS-93-30, Brown University, 1993.

[17] S. L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statistical Society B*, 50(2):157–224, 1988.

[18] F. T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *29th Annual Symposium on Foundations of Computer Science*, pages 422–431, October 1988.

[19] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

[20] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[21] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36(2):177–189, 1979.

[22] G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *J. ACM*, 44(1):1–29, 1997.

[23] B. Mohar. Isoperimetric numbers of graphs. *J. Combin. Theory Ser. B*, 47(3):274–291, 1989.

[24] J. K. Park and C. A. Phillips. Finding minimum-quotient cuts in planar graphs. In *25th Annual ACM Symposium on Theory of Computing*, pages 766–775, May 1993.

[25] S. Rao. Finding near optimal separators in planar graphs. In *28th Annual Symposium on Foundations of Computer Science*, pages 225–237. IEEE, 1987.

[26] S. Rao. Faster algorithms for finding small edge cuts in planar graphs. In *24th ACM Symp. on Theory of Computing*, pages 229–240. ACM, 1992.

[27] N. Robertson and P. D. Seymour. Graph minors. II: algorithmic aspects of treewidth. *J. Algorithms*, 7:309–322, 1986.

[28] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):888–905, 1997.

[29] D. A. Spielman and S.-H. Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *37th Annual Symposium on Foundations of Computer Science*, pages 96–105. IEEE, 1996.

APPENDIX

A. PROOFS FOR THE PRELIMINARIES

Proof of Lemma 2.3. Let $\alpha = 0$ and let $\beta = (1 + \frac{\alpha'-\alpha}{\lambda+\alpha}) \cdot \rho$. From the assumption, $q^\alpha(A,B,C) \leq \beta \cdot OPT^{\alpha'} \leq \beta \cdot q^{\alpha'}(A,B,C)$. Without loss of generality, assume that $w(A) \leq w(B)$. Then, $\frac{c(C)}{w(A)+\alpha \cdot w(C)} \leq \beta \cdot \frac{c(C)}{w(A)+\alpha' \cdot w(C)}$. Thus, $w(A) + \alpha' \cdot w(C) \leq \beta \cdot (w(A) + \alpha \cdot w(C))$. We get that

$$\begin{aligned} w(A) \geq \tfrac{\alpha'-\alpha\cdot\beta}{\beta-1} \cdot w(C) = \\ \frac{\alpha'-\alpha\cdot\rho\cdot(1+\frac{\alpha'-\alpha}{\lambda+\alpha})}{\rho\cdot(1+\frac{\alpha'-\alpha}{\lambda+\alpha})-1} \cdot w(C) = \\ \frac{\alpha'-\alpha\cdot\rho\cdot(1+\frac{\alpha'-\alpha}{\lambda+\alpha})}{(\rho-1)+\frac{\rho(\alpha'-\alpha)}{\lambda+\alpha})} \cdot w(C) \geq \\ \frac{\alpha'-\alpha\cdot\rho\cdot(1+\frac{\alpha'-\alpha}{\lambda+\alpha})}{(\rho-1)+\frac{\rho}{\lambda+\alpha})} \cdot w(C) \end{aligned}$$

In particular, for $\alpha = 0$ we get $w(A) \geq \frac{\alpha'}{(\rho-1)+\frac{\rho}{\lambda}} \cdot w(C)$. □

Proof of Lemma 2.4. From $\alpha' > \alpha$ we get that

$$\frac{q^\alpha(A^*,B^*,C^*)}{q^{\alpha'}(A^*,B^*,C^*)} = 1 + \frac{(\alpha'-\alpha)w(C^*)}{\min\{w(A^*),w(B^*)\}+\alpha w(C^*)} \leq 1 + \frac{\alpha'-\alpha}{\lambda+\alpha} \tag{3}$$

It follows that a similar bound holds also for the b-limited quotients, namely,

$$\frac{q_b^\alpha(A^*,B^*,C^*)}{q_b^{\alpha'}(A^*,B^*,C^*)} = \frac{\min\{w(A^*)+\alpha' w(C^*), w(B^*)+\alpha' w(C^*), bW\}}{\min\{w(A^*)+\alpha w(C^*), w(B^*)+\alpha w(C^*), bW\}} \leq 1 + \frac{\alpha'-\alpha}{\lambda+\alpha} \tag{4}$$

Indeed, consider the quotient in the middle. If the denominator equals bW then so does the numerator (since it is at least as large), and thus their quotient is 1; otherwise this quotient is upper bounded by $\frac{w(A^*)+\alpha' w(C^*), w(B^*)+\alpha' w(C^*)}{w(A^*)+\alpha w(C^*), w(B^*)+\alpha w(C^*)}$, and thus reduces to the one in (3). Therefore,

$$q_b^{\alpha'}(A,B,C) \leq q_b^\alpha(A,B,C) \leq \rho \cdot q_b^\alpha(A^*,B^*,C^*) \leq \rho \left(1 + \frac{\alpha'-\alpha}{\lambda+\alpha}\right) \cdot q_b^{\alpha'}(A^*,B^*,C^*).$$

(The first inequality is immediate; the second one follows from the assumption that (A,B,C) is near-optimal; the third one is by (4)).

Now, if $\alpha' = 0$, then using Lemma 2.3 and the inequality above ($q_b^0(A,B,C) \leq \rho \cdot (1 + \frac{\alpha'}{\lambda}) \cdot OPT^{\alpha'}$) we get that $w(A), w(B) \geq w(C) \cdot \frac{\alpha'}{(\rho-1)+\frac{\rho}{\lambda}}$. □

Proof of Lemma 2.5. Let (A^*,B^*,C^*) be a vertex-cut whose whose b-limited α'-quotient cost is minimal, and let $\lambda > 0$ be a parameter that we will later choose as $\lambda = \alpha'\rho$.

Assume first that $\min\{w(A^*), w(B^*)\} \leq \lambda w(C^*)$. Then

$$q_b^{\alpha'}(A^*,B^*,C^*) \geq \frac{c(C^*)}{\min\{(\lambda+\alpha')w(C^*), bW\}} \geq \frac{c(C^*)}{(\lambda+\alpha')w(C^*)}. \tag{5}$$

We claim that there always exists $v^* \in C^*$ such that $q_b^{\alpha'}(\emptyset, V \setminus \{v^*\}, \{v^*\}) \leq (\frac{\lambda}{\alpha'}+1) \cdot q_b^{\alpha'}(A^*,B^*,C^*)$. Indeed, if C^* contains a vertex of weight larger than bW then for such a vertex v^* we have that $q_b^{\alpha'}(\emptyset, V \setminus \{v^*\}, \{v^*\}) = \frac{c(v^*)}{bW} \leq \frac{c(C^*)}{bW} = q_b^{\alpha'}(A^*,B^*,C^*)$. Otherwise, we have by (5) that $v^* = \operatorname{argmin}_{v \in C} \frac{c(v)}{w(v)}$ satisfies $q_b^{\alpha'}(\emptyset, V\setminus\{v^*\}, \{v^*\}) = \frac{c(v^*)}{\alpha' w(v^*)} \leq \frac{c(C^*)}{\alpha' w(C^*)} \leq \frac{\lambda+\alpha'}{\alpha'} \cdot q_b^{\alpha'}(A^*,B^*,C^*)$. The claim follows.

By the claim above we have that if we find $v' = \operatorname{argmin}_{v\in V} \frac{c(v)}{\min\{w(v),bW\}}$ then the vertex-cut $(\emptyset, V \setminus \{v'\}, \{v'\})$ satisfies

$$q_b^{\alpha'}(\emptyset, V\setminus\{v'\}, \{v'\}) \leq q_b^{\alpha'}(\emptyset, V\setminus\{v^*\}, \{v^*\}) \leq \frac{\lambda+\alpha'}{\alpha'} \cdot q_b^{\alpha'}(A^*,B^*,C^*).$$

In other words, the b-limited α'-quotient cost of this cut is within a factor of $\frac{\lambda}{\alpha'} + 1 = \rho + 1$ from the minimum.

Assume now that $\min\{w(A^*), w(B^*)\} > \lambda w(C^*)$. Lemma 2.4 shows that

$$q_b^{\alpha'}(A,B,C) \leq \rho \left(1 + \frac{\alpha'-\alpha}{\lambda+\alpha}\right) \cdot q_b^{\alpha'}(A^*,B^*,C^*).$$

In other words, the b-limited α'-quotient cost of (A,B,C) is within a factor of $\rho(1 + \frac{\alpha'-\alpha}{\lambda+\alpha}) = \rho + 1 - \frac{\alpha\rho+\alpha}{\alpha'\rho+\alpha} \leq \rho + 1$ from the minimum.

Therefore, taking (A',B',C') to be the vertex-cut with the smaller b-limited α'-quotient cost among $(\emptyset, V \setminus \{v'\}, \{v'\})$ and (A^*,B^*,C^*) always yields a vertex-cut whose quotient cost is within factor $\rho + 1$ of the minimum. □

B. PROOFS FOR THE STRUCTURAL THEOREM

Proposition B.1. *Every face of G_D has exactly one label.*

Proof of Proposition B.1. It suffices to show that no face is labeled both by $\hat{A}$ and by $\hat{B}$, so assume to the contrary that a face is labeled both by $\hat{A}$ and $\hat{B}$. Then the boundary of this face contains both a vertex $a \in \hat{A}$ and a vertex $b \in \hat{B}$. By Proposition 2.2, this boundary is a 3-cycle and thus (a,b) is an edge in G_D, and furthermore this edge belongs to G_0. This contradicts $(\hat{A}, \hat{B}, \hat{C})$ being a vertex-cut. □

Proposition B.2. *$\tilde{w}(\mathrm{CC}(\hat{A})) = w(\hat{A})$, $\tilde{w}(\mathrm{CC}(\hat{B})) = w(\hat{B})$, and $\tilde{w}(\mathrm{CC}(\hat{C})) = 0$. It immediately follows that $\tilde{w}(\mathrm{CC}) = \tilde{w}(\mathrm{CC}(\hat{A})) + \tilde{w}(\mathrm{CC}(\hat{B})) + \tilde{w}(\mathrm{CC}(\hat{C})) = w(\hat{A} \cup \hat{B})$.*

Proof of Proposition B.2. By definition $\tilde{w}(\mathrm{CC}(\hat{A})) = \sum_{R\in \mathrm{CC}(\hat{A})} \sum_{v\in V(R)\setminus \hat{C}} w(v)$. Observe that all the faces of G_D around a vertex $a \in \hat{A}$ are labeled $\hat{A}$ and are thus all in the same connected component of $\mathrm{CC}(\hat{A})$. Hence, every vertex $a \in \hat{A}$ belongs to $V(R)$ for exactly one connected region $R_a \in \mathrm{CC}(\hat{A})$. Similarly, every vertex $b \in \hat{B}$ belongs to $V(R)$ for no $R \in \mathrm{CC}(\hat{A})$ It follows that every vertex of $\hat{A}$ contributes its weight (exactly once) to the aforementioned summation, while vertices of $\hat{B}$ contribute nothing to this summation, and vertices of $\hat{C}$ are explicitly excluded from it. We conclude that this summation is equal to $\sum_{v\in\hat{A}} w(v) = w(\hat{A})$, as claimed. The proof for $\hat{B}$ and $\hat{C}$ is similar. □

Proposition B.3. *If $b \leq \frac{1}{3}$, then $\min\{\tilde{w}(S), \tilde{w}(\bar{S})\} \geq \min\{w(\hat{A}), w(\hat{B})\}$.*

Proof of Proposition B.3. Proposition B.2 implies that $\tilde{w}(S) + \tilde{w}(\bar{S}) = w(\hat{A} \cup \hat{B})$; we will use this fact throughout the proof. Also, notice that $\max\{\frac{1-b}{2}, \frac{1}{8} + \frac{b}{2}\} = \frac{1-b}{2}$ whenever $b \leq \frac{3}{8}$.

First, when the algorithm stops, $\tilde{w}(S) \geq \frac{1-b}{2} w(\hat{A} \cup \hat{B}) \geq \frac{2/3}{2} w(\hat{A} \cup \hat{B}) = \frac{1}{3} w(\hat{A} \cup \hat{B}) \geq w(\hat{B})$. Now, if $S = \{R^1_{\max}\}$, then $\tilde{w}(S) \leq w(\hat{A})$ (recall that $w(\hat{A}) \geq w(\hat{B})$). Thus,

$\tilde{w}(\bar{S}) \geq w(\hat{B})$, and we get that $\min\{\tilde{w}(S), \tilde{w}(\bar{S})\} \geq \min\{w(\hat{A}), w(\hat{B})\}$. On the other hand, if $S \neq \{R^1_{\max}\}$, then $\tilde{w}(R^1_{\max}) \leq \frac{1-b}{2} w(\hat{A} \cup \hat{B})$. We claim that $\tilde{w}(S) \leq w(\hat{A})$. If $\tilde{w}(S) < \frac{2}{3} w(\hat{A} \cup \hat{B})$, then this is true because $w(\hat{A}) = (1-b)w(\hat{A} \cup \hat{B}) \geq \frac{2}{3} w(\hat{A} \cup \hat{B})$. On the other hand, if $\tilde{w}(S) > \frac{2}{3} w(\hat{A} \cup \hat{B})$, then the weight of S after the last iteration is at least $\frac{2}{3} w(\hat{A} \cup \hat{B})$ and prior to the last iteration it must have been less than $\frac{1-b}{2} w(\hat{A} \cup \hat{B})$ (since the last iteration was performed and $b \leq \frac{2}{3}$). Any successive step adds at most $\tilde{w}(R^1_{\max})$ weight, so $\tilde{w}(S) < 2\frac{1-b}{2} w(\hat{A} \cup \hat{B}) = (1-b)w(\hat{A} \cup \hat{B}) = w(\hat{A})$.

Thus, we proved that in both cases $\tilde{w}(S) \leq w(\hat{A})$ and $\tilde{w}(S) \geq b \cdot w(\hat{A} \cup \hat{B})$ implies that $\min\{w(\hat{A}), w(\hat{B})\} \leq \min\{\tilde{w}(S), \tilde{w}(\bar{S})\}$, and the proof follows. $\square$

Proposition B.4. $\min\{\tilde{w}(S), \tilde{w}(\bar{S})\} \geq \frac{3}{4} \min\{w(\hat{A}), w(\hat{B})\}$.

Proof of Proposition B.4. The case of $b \leq \frac{1}{3}$ is provided by Proposition B.3, so for the rest of this proof we assume $b > \frac{1}{3}$. First, notice that there cannot be three (or more) connected regions with more than $\frac{1-b}{2} w(\hat{A} \cup \hat{B})$ weight because then two of those regions must be together in one of A, B, contradicting $w(\hat{A}), w(\hat{B}) \leq (1-b)w(\hat{A} \cup \hat{B})$.

Thus, there must be at most two regions with weight greater than $\frac{1-b}{2} w(\hat{A} \cup \hat{B})$. Thus, the algorithm must stop with $\tilde{w}(S) \leq (\max\{\frac{1}{8}+\frac{b}{2}, \frac{1-b}{2}\} + \frac{1-b}{2}) w(\hat{A} \cup \hat{B}) \leq \max\{\frac{5}{8}, 1-b\} w(\hat{A} \cup \hat{B})$. We get that $\tilde{w}(S) \leq w(\hat{A})$ because if $\frac{5}{8} > 1-b$, then $b > \frac{3}{8}$ and $\tilde{w}(S) \leq \frac{5}{8} w(\hat{A} \cup \hat{B}) \leq w(\hat{A})$, and otherwise if $\frac{5}{8} \leq 1-b$, then $\tilde{w}(S) \leq (1-b)w(\hat{A} \cup \hat{B}) = w(\hat{A})$. This shows that S is not *too big*. Now we show that it is not too small either.

If $b \leq \frac{3}{8}$, then $\tilde{w}(S) \geq \max\{\frac{1}{8}+\frac{b}{2}, \frac{1-b}{2}\} w(\hat{A} \cup \hat{B}) = \frac{1-b}{2} w(\hat{A} \cup \hat{B})$. We get that

$$\frac{\tilde{w}(S)}{w(\hat{B})} \geq \frac{\frac{1-b}{2}}{b} = \frac{1}{2b} - \frac{1}{2} \geq \frac{1}{2 \cdot \frac{3}{8}} - \frac{1}{2} = \frac{5}{6}$$

Thus, in this case $\tilde{w}(S) \geq \frac{5}{6} w(\hat{B})$ and we get that $min(\tilde{w}(S), \tilde{w}(\bar{S})) \geq \frac{5}{6} \min\{w(\hat{A}), w(\hat{B})\}$. If $b > \frac{3}{8}$, then $\tilde{w}(S) \geq (\frac{1}{8}+\frac{b}{2}) w(\hat{A} \cup \hat{B})$. We get that since $b \leq \frac{1}{2}$,

$$\frac{\tilde{w}(S)}{w(\hat{B})} \geq \frac{\frac{1}{8}+\frac{b}{2}}{b} = \frac{1}{8b} + \frac{1}{2} \geq \frac{1}{8 \cdot \frac{1}{2}} + \frac{1}{2} = \frac{3}{4}$$

Thus, in this case $\tilde{w}(S) \geq \frac{3}{4} w(\hat{B})$ and we get that $min(\tilde{w}(S), \tilde{w}(\bar{S})) \geq \frac{3}{4} \min\{w(\hat{A}), w(\hat{B})\}$. $\square$

Proof of Proposition 3.3. We first show that $(\hat{A}', \hat{B}', \hat{C}')$ is a vertex-cut. $V_0 = \hat{A}' \cup \hat{B}' \cup \hat{C}'$ since every vertex $v \in V_0$ appears in at least one face, which by Prop. B.1 must have a label, and thus the face is in $V(\mathrm{CC})$. Note that $\hat{A}', \hat{B}', \hat{C}'$ are disjoint by definition.

Assume now to the contrary that G_0 contains an edge between $a \in \hat{A}'$ and $b \in \hat{B}'$. By definition, $a \in V(S) \setminus V(\bar{S})$ and $b \in V(\bar{S}) \setminus V(S)$. Consequently, a is in no face in (the connected regions in) $\bar{S}$ and b is in no face in (the connected regions in) S. But just like any edge in G_0, the edge (a, b) is on the boundary of a face in G_D. But by the above, this face can be neither in S nor in $\bar{S}$, which contradicts the fact that CC contains all the faces of G_D (i.e., Proposition B.1). We thus conclude that $(\hat{A}', \hat{B}', \hat{C}')$ is a vertex-cut of G_0.

We next show that $\hat{C}' \subseteq C$. Consider any $v \in \hat{C}'$ and assume for contradiction that $v \notin C$, i.e., $v \in A \cup B$. Then the faces of G_D containing v are either all labeled $\hat{A}$ or all labeled $\hat{B}$. Furthermore, all these faces are in the same connected region in $\mathrm{CC}(\hat{A}) \cup \mathrm{CC}(\hat{B})$ (because by ordering all the faces around a vertex clockwise, then every two successive faces are adjacent). This connected region is either in S or in $\bar{S}$ (but not both), implying that v is in exactly one of $V(S)$, $V(\bar{S})$. This contradicts the assumption that $v \in \hat{C}'$.

Finally, we show the lower bound on $\min\{w(\hat{A}'), w(\hat{B}')\}$. Observe that $v \in V(S) \setminus C$ implies that $v \notin V(\bar{S})$, as otherwise we would have that $v \in V(S) \cap V(\bar{S}) = \hat{C}' \subseteq C$. Then, by definition,

$$\tilde{w}(S) = \sum_{v \in V(S), v \notin C} w(v) \leq \sum_{v \in V(S), v \notin V(\bar{S})} w(v) = \sum_{v \in \hat{A}'} w(v) = w(\hat{A}')$$

where the first inequality follows from the fact that every vertex of V_0 appears in exactly one connected component in CC ≪make it a separate Proposition? –Robi≫. A similar argument shows that $\tilde{w}(\bar{S}) \leq w(\hat{B}')$. We then conclude using Proposition B.4 that $\min\{w(\hat{A}'), w(\hat{B}')\} \geq \min\{w(S), w(\bar{S})\} \geq \frac{3}{4} \min\{w(\hat{A}), w(\hat{B})\}$, as claimed. $\square$

We will need the following technical lemma regarding plane graphs.

Proposition B.5. *Let $G(V, E)$ be a plane graph, and suppose that four of the regions around a vertex $v \in V$, when considered in, say, a counter-clockwise order, belong to the faces f_1, f_2, f_3, f_4 of G, respectively. If $f_2 \cup f_4$ is disjoint from $f_1 \cup f_3$ then any arc in $\mathbb{R}^2 \setminus \{v\}$ between f_1 and f_3 intersects any arc in $\mathbb{R}^2 \setminus \{v\}$ between f_2, f_4. (See Figure 6 for illustration.)*

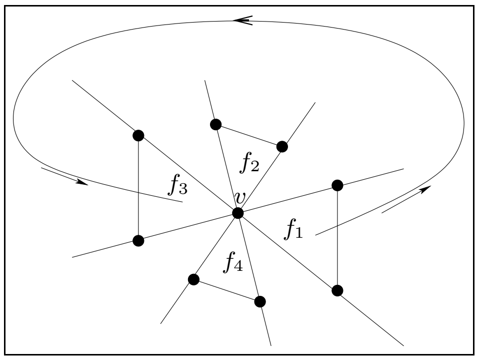

Figure 6: Four faces touching v

Proof of Proposition B.5. Assume for contradiction that two such arcs exist. Extend the arc between f_1 and f_3 into a polygon P in $\mathbb{R}^2$ by going through f_1, v and f_3. By the Jordan Curve Theorem, the polygon P divides $\mathbb{R}^2 \setminus P$ into exactly two regions, each having v on its frontier. Observe that f_2 and f_4 must be in different regions of $\mathbb{R}^2 \setminus P$, as otherwise v is on the frontier of only one region. Since $f_2 \cup f_4$ is disjoint from $f_1 \cup f_3$, the arc between f_2 and f_4 does not intersect the polygon P, which contradicts the fact that f_2 and f_4 are in different regions of $\mathbb{R}^2 \setminus P$. $\square$

Proposition B.6. *For every connected region $R \in \mathrm{CC}(\bar{S})$ there is a directed simple cycle D_R such that R is exactly the set of faces (of G_D) that are enclosed by D_R.*

Proof of Proposition B.6. Consider a connected region $R \in \mathrm{CC}(\bar{S})$ and let $\bar{R} := \cup_{R' \in S \cup \bar{S} \setminus R} R'\}$ denote the set of all other faces in G_D. Let D_R be the collection of directed edges in G_D whose face to the left is in R and their face to the right is not in R. We will show that D_R is the desired cycle.

Let us first show that if a vertex v of G_D appears in the cycle D_R then its in-degree in D_R is equal to its out-degree in D_R. Each immediate regions around v is contained in a face of G_D, that must be either in R or not in R. Considering the regions around v in, say, a counter-clockwise order, the edge between two successive regions belongs to D_R if one of these regions is in R and the other one is in $\bar{R}$. Specifically, when going in the above order, a region of R followed by one of $\bar{R}$ corresponds to an edge of the cycle D_R that is directed toward v, and a region of $\bar{R}$ followed by one of R corresponds to an edge of the cycle D_R that is directed away from v. Since switches from R to $\bar{R}$ alternate (when considered in the above order) with switches from $\bar{R}$ to R, edges of D_R that are incoming to v alternate with those edges of D_R that are outgoing from v. In particular, the in-degree and out-degree of v in D_R are equal. (See Figure 7 for illustration.)

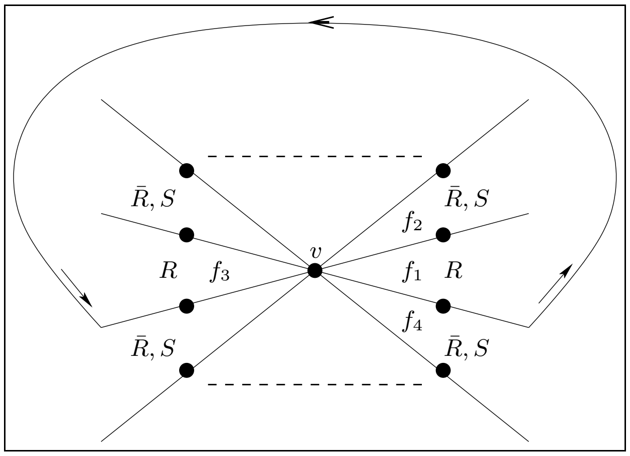

Figure 7: The edges separating between R and $\bar{R}$

Assume now for contradiction that the in-degree in D_R of v is at least two, and let then e_1, e_2 be two edges of D_R that are incoming to v. Consider the regions around v in a counter-clockwise order. The region just before e_1 belongs to a face $f_1 \in R$, and the region that is just after e_1 belongs to a face $f_2 \in \bar{R}$. In fact, f_2 is in S as otherwise R is not a connected component of $\bar{S}$. Similarly, the region just before e_2 belongs to a face $f_3 \in R$, and the region just after e_2 belongs to a face f_4 of S. We thus get that there are around v four regions that when considered in a counter-clockwise order belong to faces f_1, f_2, f_3, f_4 of G_D, respectively, with $f_1, f_3 \in R$ and $f_2, f_4 \in S$. In particular, $f_2 \cup f_4$ is disjoint from $f_1 \cup f_3$. Since R is a connected region, it contains a faces-path between f_1 and f_3 (trivially, if $f_1 = f_3$). This defines an arc between f_1 and f_3 that goes only through faces in R and edges that are on the boundary of two distinct faces of R. In particular, this arc does not go through v. Similarly, S is also a connected region and thus it defines an arc between f_2 and f_4 that goes only through faces in S and edges that are on the boundary of two distinct faces of S. In particular, this arc does not go through v. It follows that these two arcs do not intersect each other, in contradiction with Proposition B.5; thus the in-degree and out-degree of vertices in D_R are exactly 1.

We now show that D_R is not a union of disjoint cycles. Assume to the contrary that it contains $k \geq 2$ disjoint cycles. Thus, $\mathbb{R}^2 \setminus C_R$ has at least $k+1 \geq 3$ regions in the plane. Each such region contains at least one face of $R \cup S$, since the frontier of the region is an edge of D_R, namely, an edge that separates between a face of R and a face of S. All the faces of R are in the same region of $\mathbb{R}^2 \setminus D_R$ since R is a connected region (note that edges between two faces in R have are not in D_R), and similarly all the faces of S are in the same region of $\mathbb{R}^2 \setminus D_R$. It follows that $\mathbb{R}^2 \setminus D_R$ can have at most 2 regions, in contradiction with the above assumption. We furthermore conclude that R is exactly the set of faces of G_D that are contained in one region of $\mathbb{R}^2 \setminus D_R$, namely to the left of D_R, as claimed. $\square$

Proposition B.7. *$\mathcal{D}(\bar{S})$ defines the vertex-cut $(\hat{A}', \hat{B}', \hat{C}')$ that is described in (2). Furthermore, the sets of vertices of V_0 that are enclosed by these cycles are disjoint.*

Proof of Proposition B.7. Let (A, B, C) be the vertex-cut defined by $\mathcal{D}(\bar{S})$. Every vertex $v \in C$ appears in a directed edge in some D_i. This edge is on the boundary of a face of $\bar{S}$ and a face of S. Then by the definitions (1) and (2) we have that $v \in \hat{C}'$. Every vertex $v \in B$ is in the region of the plane that is enclosed by (i.e., to the left of) some D_i, and is thus on the boundary only of faces in $\bar{S}$. It follows that $v \in \hat{B}'$. Similarly, every vertex $v \in A$ is on the boundary only of faces in S (as otherwise it would have been in $\hat{A}$ or $\hat{B}$), and thus $v \in \hat{A}'$.

Finally, let v be a vertex enclosed by two cycles $D_i, D_j \in \mathcal{D}(\bar{S})$. Then at least one face around v is contained in both of these regions, and so D_i and D_j enclose (the faces of) the same connected component of $\bar{S}$. Therefore, $i = j$. $\square$

Proof of Proposition 3.4. Assume to the contrary that $\tilde{G}$ contains a cycle $\tilde{D} =< d_0, v_0, \ldots, d_{k-1}, v_{k-1} >$ where $v_i \in \tilde{V}_1$ and $d_i \in \tilde{V}_2$ for all i and $k \geq 2$. Without loss of generality this cycle is simple. By the definition of $\tilde{E}$, each v_i appears in the cycles d_i and $d_{(i+1)\mathrm{mod}k}$. Thus, there is in the plane a polygon P that goes through the vertices $v_0, \ldots, v_{k-1}$ and through the connected regions that correspond to $d_0, \ldots, d_{k-1}$ (and edges between faces in each connected region). This polygon does not intersect itself because the cycle $\tilde{D}$ is simple and the connected regions in $\mathrm{CC}(\bar{S})$ are disjoint.

Consider the faces around v_0 in a counter-clockwise order. The cycle d_0 contains an edge e_0 that is incoming to v_0, and thus the region just before e_0 belongs to a face f_1 in a connected region R_0 that is defined by d_0, and the region just after e_0 belongs to a face $f_2 \in \bar{R}$. In fact, f_2 is in S as otherwise R_0 is not a connected component of $\bar{S}$. Similarly, the cycle d_1 contains an edge e_2 that is incoming to v_0 and thus the region just before e_1 belongs to a face f_3 in a connection region R_1 that is defined d_1, and the region just after e_1 belongs to a face f_4 of S. To see that these four regions are distinct, observe that the two edges e_0, e_1 are distinct, as otherwise they have the same face to their left, and then the cycles d_0, d_1 are the boundaries of the same connected component of $\mathrm{CC}(\bar{S})$. Note that S defines an arc between

f_2 and f_4 that goes only through faces in S and edges that are on the boundary of two distinct faces of S, and that the polygon P defines an arc between f_1 and f_3 that goes only through $v_1, \ldots, v_{k-1}$ (but not through v_0), faces in $\bar{S}$ and edges that are on the boundary of two distinct faces in $\bar{S}$. It follows that these two arcs do not intersect each other, in contradiction with Proposition B.5; thus, there is no cycle $\tilde{D}$ in $\tilde{G}$. □

PROOF OF PROPOSITION 3.5. In each connected component of $\tilde{G}$, choose arbitrarily some vertex of $\tilde{V}_1$ to be the *root* of this connected component. (Note that there must be such a vertex.) Select T to be the connected component of $\tilde{G}$ with the best q_b^0 quotient. Using averaging arguments we know that this quotient is at least as good as that of $(\hat{A}', \hat{B}', \hat{C}')$. Let the *level* of a vertex $v \in V_1$ in T be the distance (in $\tilde{G}$) between v and the root of T plus 1, not including vertices of V_2 in the distance count (for example, the level of the root cycle is 1, and the level of its children cycles is 2).

Choose $i \leq d$ arbitrarily. Define T_1^i to be the (disjoint) union of d-tree's rooted in vertices of V_1 on levels $l \cong i($ mod $d+1)$ of T. Let T_2^i be the union of the rest of the vertices in T (i.e., those vertices of V_D that appear in levels $i-1($ mod $d+1)$ of T and nowhere else). Thus, $w(T) = w(T_1^i) + w(T_2^i)$. If for all $i \leq d$, $w(T_1^i) < \frac{d-1}{d} w(T)$, then for all $i \leq d$ $w(T_2^i) > \frac{1}{d} w(T)$. We get that $w(T) \geq \sum_{i \leq d} w(T_2^i) > d\frac{1}{d} w(T) = w(T)$. Thus, there must be some $i \leq d$ such that $w(T_1^i) \geq \frac{d-1}{d} w(T)$. For this i we get

$$\frac{c(T_1^i)}{w(T_1^i)} \leq \frac{c(T)}{w(T_1^i)} \leq \frac{c(T)}{\frac{d-1}{d} w(T)} = \frac{d}{d-1} \frac{c(T)}{w(T)}.$$

It follows that there exists a vertex-cut $(\hat{A}'', \hat{B}'', \hat{C}'')$ such that

$$q_b^0(\hat{A}'', \hat{B}'', \hat{C}'') = \frac{c(\hat{C}'')}{\min\{w(\hat{B}''), w(\hat{A}''), bW\}} \leq \frac{\frac{d}{d-1} c(\hat{C}')}{\min\{w(\hat{B}'), w(\hat{A}'), bW\}}.$$

By Proposition 3.3 we have that $c(\hat{C}') \leq c(\hat{C})$ and that $\min\{w(\hat{A}'), w(\hat{B}'), bW\} \geq \frac{3}{4} \min\{w(\hat{A}), w(\hat{B}), bW\}$. Thus,

$$\frac{\frac{d}{d-1} c(\hat{C}')}{\min\{w(\hat{B}'), w(\hat{A}'), bW\}} \leq \frac{\frac{4}{3}\frac{d}{d-1} c(\hat{C})}{\min\{w(\hat{B}), w(\hat{A}), bW\}} = \frac{4}{3}\frac{d}{d-1} \cdot q_b^0(\hat{A}, \hat{B}, \hat{C}).$$

□

PROOF OF PROPOSITION 3.6. Now, if $\lambda w(C^*) < w(A^*), w(B^*)$, for some $\lambda > 0$, then Theorem 3.1 guarantees a vertex cut whose b-limited 0-quotient cost (i.e., $\alpha = 0$) is within a factor of $\frac{4}{3}\frac{d}{d-1}$ from optimal. We then have from Lemma 2.4 that also its b-limited α-quotient cost (for any $\alpha > 0$) is within a factor of $\frac{4}{3}\frac{d}{d-1}(1 + \frac{\alpha}{\lambda})$ from optimal. □

PROOF OF PROPOSITION 3.7. Let $b = \min\{w(\hat{A}), w(\hat{B})\}/w(\hat{A} \cup \hat{B})$ be as above. The case of $b \leq \frac{1}{3}$ follows from Proposition B.3 (with $\hat{A}'$ corresponding to only one connected region), so for the rest of the proof we assume $\frac{1}{3} < b \leq \frac{1}{2}$. We say that a connected region $R \in \text{CC}$ is *small* if $\tilde{w}(R) \leq \epsilon b \tilde{w}(\text{CC})$, and *big* otherwise. The number of big connected regions is clearly at most $1/(\epsilon b) = O(1/\epsilon)$.

Every big connected region belongs to either $\text{CC}(\hat{A})$ or $\text{CC}(\hat{B})$, since $\tilde{w}(R) = 0$ for all $R \in \text{CC}(\hat{C})$ by Proposition B.2. Thus, the $O(1/\epsilon)$ big connected regions can be partitioned into two groups corresponding to A and B, respectively. Now iteratively pick a small connected region that is adjacent to a big connected region and add the former to the latter, until either one of the two groups of big connected regions R has weight $\tilde{w}(R) \geq (1-\epsilon)b$. Notice that the small connected regions cannot be exhausted, since if both groups have weight at most $(1-\epsilon)b\tilde{w}(\text{CC})$ then the remaining $1 - 2(1-\epsilon)b\tilde{w}(\text{CC})$ must be used by small connected regions. Furthermore, it is straightforward that there always exists a small connected region that is adjacent to some big connected region. By definition, when these iterations terminate, both groups consist of $O(1/\epsilon)$ connected regions. We can thus add all the remaining small connected regions to the group of smaller weight. Thus, the group of larger weight has total weight at least $(1-\epsilon)b\tilde{w}(\text{CC})$ and at most $b\tilde{w}(\text{CC})$, and still consists of $O(1/\epsilon)$ connected regions. □

PROOF OF PROPOSITION 3.8. Theoerm 3.1 guarantees the existence of a vertex-cut (A, B, C) defined by a 1-CAST, such that its b-limited 0-quotient cost is within a factor of $\frac{8}{3}$ from the minimum. We claim that there is either a trivial cut or a cut defined by one of the cycles in this CAST, that has b-limited 0-quotient cost that is within a factor of 2 from that of the whole CAST. Indeed, let $D_1, \ldots, D_k$ denote the cycles in the 1-CASTwhose cost is at least 2. For each such cycle D_i let w_i and c_i denote its weight and cost, respectively. Similarly, the set of cycles in the 1-CAST that has cost 1 defines a vertex-cut that has some weight w_0 and cost $c_0 = 1$. By definition, $w(B) = \sum_{i=0}^k w_i$ and $c(C) = 1 + \sum_{i=1}^k (c_i - 1)$. Thus, $\sum_{i=0}^k c_i = c(C) + k \leq 2c(C)$. By averaging arguments, it follows that at least one of these $k+1$ vertex-cuts has 0-quotient cost that is at most $2c(C)/\min\{w(A), w(B), bW\}$, as claimed.

For $\alpha = 0$ we thus obtain the desired result (with factor $5\frac{1}{3}$). This can be extended to any $\alpha > 0$ (with factor $6\frac{1}{3}$) by using Lemma 2.5. □

C. PROOFS FOR FINDING A NEAR OPTIMAL VERTEX-CUT

PROOF OF PROPOSITION 4.1. Notice that the weight of a face in F_i originates from weights of vertices in V_i and/or weights of vertices in C. Formally, for a face f let $w^0(f)$ be the portion of f's weight coming from vertices not in C, and let $w^1(f)$ be the portion of f's weight coming from vertices in C. Observe that the weight of a vertex in V_i is (completely) distributed among faces that are all in F_i. Thus, $w(V_i) = \sum_{f \in F_i} w^0(f)$ for all i.

Since $F_1, \ldots, F_l$ are disjoint, $w(\cup_{i \in I} F_i) = \sum_{i \in I} \sum_{f \in F_i} w^0(f) + \sum_{i \in I} \sum_{f \in F_i} w^1(f))$. Observe that the weight of a vertex in V_i is (completely) distributed among faces that are all in F_i, and that also $V_1, \ldots, V_l$ are disjoint, and thus the first term on the right hand side is equal to $\sum_{i \in I} w(V_i) = w(\cup_{i \in I} V_i)$. The second term on that right hand side is similarly bounded by $0 \leq \sum_{i \in I} \sum_{f \in F_i} w^1(f) \leq w(C)$, which proves the proposition.

□

PROOF OF PROPOSITION 4.2. We compute $T(l+1, s, t, \{p_1, \ldots, p_d\}, w)$ by finding the minimal cost of extending a length l walk by one additional edge. Formally, if t' is in the pebble p'_i then

$$T(l+1, s, t, \{p_1, ..., p'_i, ..., p_d\}, w) =$$
$$min_{(t_0,t)\in E}\begin{cases} T(l, s, t_0, \{p_1, ..., p_i, ..., p_d\}, w - w((t_0, t))) & t_0 \in p_i \ \& \ t \in p_{i-1} \ \& \ p'_i = nil \\ T(l, s, t_0, \{p_1, ..., p_i, ..., p_d\}, w - w((t_0, t))) + c(t) & t_0 \in p_i \ \& \ \forall o \leq d \ t \notin p_o \ \& \ (t \in p_i \text{ or } t \in p_{i+1} \ \& \ t_0 \in p_i) \end{cases} \quad (6)$$

The intuition behind this computation is that at every step we either remove the pebble from the current vertex and go to a vertex that already has a pebble (this is the first case above), or we move to a vertex that has no pebble and either remove the current vertex's pebble or remove it (this is the second case above). In the first case we do not add the cost of the new vertex t, while in the second case we do add this cost.

The values of T for $l = 1$ are trivial to compute: $T(1, u, v, \{< v, (u, v), (u, v) >\}, w(u, v)) = c(u, v)$ for every edge $e = (u, v)$, and are undefined (or say ∞) otherwise.

We know that always either the last vertex equals the first, or it holds a pebble. Also, every pebble can be encoded as two choices over $O(n)$ edges. These two observations show that the size of T is $W \cdot n^{2d+2}$. Proposition 4.2 shows that we can fill the table in time $W \cdot n^{2d+3}$. □

Proof of Proposition 4.3. The intuition for this theorem is that D's embedding in the plane (as part of the embedding of G_D) defines a closed walk on the *outside* of D. Regarding D as a rooted tree, and using this walk, whenever we get to a vertex that connects two cycles of D we leave a pebble on that vertex for the duration of the walk in the descendant cycles of that vertex. When we leave the subtree of that vertex and continue onward we take the pebble with us.

Formally, let $\tilde{G}$ be as in Proposition 3.4, with D corresponding to a tree in $\tilde{G}$. Recall that $\tilde{G}$ is planar with an embedding in $\mathbb{R}^2$ that corresponds to the planar embedding of G_D. A left-to-right DFS traversal of $\tilde{G}$ defines an order over the visits in the cycles and the vertices connecting them. This ordering over the cycles in D can be extended to a total order over the edges in D that is a closed walk and that surrounds each of the cycles of D from the left (i.e., the interior of each cycle is to the left of the directed edge). For every vertex v in G_D that is also in $\tilde{G}$, we leave a pebble in v from the first time that we enter v until the last time that we exit the vertex. We take the pebble when we exit v for the last time.

The total weight surrounded by our walk is the sum of weight surrounded by cycles in D, i.e., w_0, because the edges of our walk surround the cycles of D (having them on the left), and the cycles of D have disjoint faces. The *counted cost* of our walk is exactly the sum of costs of the vertices in the walk, counting every vertex exactly once because every vertex that is visited more than once is a vertex in $\tilde{G}$, so a pebble is left there between the first and last visit to it. □

Proof of Proposition 4.4. First, we mention an equivalent form of a proposition proved in [24].

Proposition C.1 (Park and Phillips [24]). *Let π be a simple closed walk in G_s. Let F be the set of faces of G_D that are separated by π from the outer face. If the faces of F are to the left of π then $w(\pi) = w(F)$, and* $\text{nc}_\pi(f) = \begin{cases} 1 & \textit{if } f \in F \\ 0 & \textit{if } f \notin F \end{cases}$. *Otherwise, $w(\pi) = -w(F)$ and* $\text{nc}_\pi(f) = \begin{cases} -1 & \textit{if } f \in F \\ 0 & \textit{if } f \notin F \end{cases}$.

Decompose the walk π into simple cycles $\pi_1, \pi_2, \ldots$ (arbitrarily). If π_i contains neither of e, e' then f_1 and f_2 are in the same region of $\mathbb{R}^2 \setminus \pi_i$, so $\text{nc}_{\pi_i}(f_1) = \text{nc}_{\pi_i}(f_2)$ by Proposition C.1. If π_i contains e then f_1 and f_2 are in different connected regions of $\mathbb{R}^2 \setminus \pi_i$, and in particular $\text{nc}_{\pi_i}(f_1) = \text{nc}_{\pi_i}(f_2) + 1$. Similarly, if π_i contains e' then $\text{nc}_{\pi_i}(f_1) = \text{nc}_{\pi_i}(f_2) - 1$. The proposition follows by adding these equations for all i, since $\text{nc}_\pi(f_1) = \sum_i \text{nc}_{\pi_i}(f_1)$ and similarly for f_2. □

Proof of Proposition 4.5. Let t_v be the number of times that the cost of v contributes to the counted cost of π. This is the number of times that we put a pebble in v. We shall show that $t_v \geq \Delta_\pi(v)$.

For a walk w, define $\Delta_w(v)$

$$\Delta_w(v) = \max_{i_0 < i_1 \leq 2d(v)} \sum_{i=i_0}^{i_1} (nv_w(e_{(i \ mod \ d(v))}) - nv_w(e'_{(i \ mod \ d(v))}))$$

where $e_1, ..., e_{d(v)}$ enumerate the edges (incoming) incident on v and e'_i is the opposite direction of e_i. Proposition 4.4 implies that this definition applied to closed walk π coincides with our original definition.

Now, we claim that for every walk w with a sequence of visits to v, if we put a pebble in v initially and pick it up only at the last exist from v, then $\Delta_w(v) = 1$. Consider the part of that walk that is the directed edges incoming and outgoing to/from v. The incoming and outgoing edges alternate counter-clockwise from v, never completing a full cycle, by the way our dynamic program is constructed. Thus, $\Delta_w(v) = 1$.

Now, break π into walks $w_1, ..., w_l$ such that every walk puts a single pebble in v and later picks it up. The definition implies that $\Delta_\pi(v) \leq \sum_{i\leq l} \Delta_{w_i}(v)$. We get that $\Delta_\pi(v) \leq l$. However, each one of the walks $w_1, ..., w_l$ counts the cost of v exactly once, so the overall count of cost for v is l. Thus, the cost of π is at least $\Delta_\pi(v)$ and the proposition is proved. □

Proof of Proposition 4.6. Proceed by induction on $\Delta_\pi(G_s)$. If $\Delta_\pi(G_s) = 1$ then we're trivially done.

Assume now that $\Delta_\pi(G_s) > 1$. Let $H = \max\{\text{nc}_\pi(f) : f \text{ is a face of } G_s\}$ and $L = \min\{\text{nc}_\pi(f) : f \text{ is a face of } G_s\}$. Let $F_H = \{f \in G_s : \text{nc}_\pi(f) = H\}$. Let E' be the set of directed edges in G_s that have a face from F_H to their left and a face not from F_H to their right. Note that $E' \subseteq \pi$, since by Proposition 4.4 any edge in E' must occur at least once in π. We claim that E' is a union of walks, i.e., that *in-deg*(v, E') = *out-deg*(v, E') for every vertex v. This is easy to see by considering the faces around v in a counter-clockwise order. As we enumerate the edges incident on v in a counter-clockwise order, the face between an edge and its successor alternates from F_H to non-F_H and back. Every such alternation gives an incoming or outgoing edge to v that is in E' (by Proposition 4.4). Thus, E' is a union of disjoint closed walks $\pi' \subseteq \pi$. Let $\pi'' = \pi - \pi'$.

We claim that $\Delta(\pi'') \leq \Delta_\pi(G_s) - 1$. Consider any path $\psi = \langle f_1, ..., f_l \rangle$ from a face in G_s to another face in G_s. Proposition 4.4 implies that

$$nc_\pi(f_1) - nc_\pi(f_l) = \sum_{i<l} nc_\pi(f_i) - nc_\pi(f_{i+1}) = \sum_{i<l} (nv_\pi(e^i_1) - nv_\pi(e^i_2))$$

for e_1^i being the edge with face f_i to its left and face f_{i+1} to its right, and e_2^i being the opposite edge. Let ψ be such a path that connects two faces f_1, f_l with $nc_{\pi''}(f_1) - nc_{\pi''}(f_l) = \Delta(\pi'')$. We show that $nc_{\pi''}(f_1) - nc_{\pi''}(f_l) \leq \Delta_\pi(G_s) - 1$. Let $e_1 = \sum_{i<l} nv_\pi(e_1^i)$, $e_2 = \sum_{i<l} nv_\pi(e_2^i)$. Similarly, define e_1', e_2' for π' and e_1'', e_2'' for π''. Then, $e_1 = e_1' + e_1''$ and $e_2 = e_2' + e_2''$. If f_1, f_l both belong to F_H, then $e_1 - e_2 = nc_\pi(f_1) - nc_\pi(f_l) = 0$ and $e_1' - e_2' = nc_{\pi'}(f_1) - nc_{\pi'}(f_l) = 0$. Thus,

$$e_1'' - e_2'' = (e_1 - e_1') - (e_2 - e_2') = (e_1 - e_2) - (e_1' - e_2') = 0.$$

If f_1, f_l both do not belong to F_H, then $e_1 - e_2 = nc_\pi(f_1) - nc_\pi(f_l) \leq \Delta_\pi(G_s) - 1$ and $e_1' - e_2' = nc_{\pi'}(f_1) - nc_{\pi'}(f_l) = 0$. Thus,

$$e_1'' - e_2'' = (e_1 - e_1') - (e_2 - e_2') = (e_1 - e_2) - (e_1' - e_2') \leq \Delta_\pi(G_s) - 1.$$

Finally, if f_1 belongs to F_H and f_l does not belong to F_H (or vice versa), then $e_1 - e_2 = nc_\pi(f_1) - nc_\pi(f_l) \leq \Delta_\pi(G_s)$ and $e_1' - e_2' = nc_{\pi'}(f_1) - nc_{\pi'}(f_l) \geq 1$. Thus,

$$e_1'' - e_2'' = (e_1 - e_1') - (e_2 - e_2') = (e_1 - e_2) - (e_1' - e_2') \leq \Delta_\pi(G_s) - 1.$$

Now, consider a vertex v. If v does not appear in π' then $\Delta_{\pi''}(v) = \Delta_\pi(v)$, by Proposition 4.4 and going over the faces around v in counter-clockwise order. Thus, the induction hypothesis implies that v appears in at most $\Delta_{\pi''}(v)$ walks. If v does appear in π', then $\Delta_{\pi''}(v) \leq \Delta_\pi(v) - 1$ by a similar argument to the one for $\Delta(\pi'')$. Thus, v appears in at most $\Delta_{\pi''}(v) + 1 \leq \Delta_\pi(v)$ walks.

Finally, notice that since π' is a disjoint union of closed walks $\pi_1', ..., \pi_j'$ we can consider each of these walks separately in the group of walks that we generate for this proposition. □

Proof of Proposition 4.7. $\Delta_\pi(G_s) \leq 1$ implies that for every face f either $\mathrm{nc}_\pi(f) \in \{0, 1\}$ for all $f \in G_s$ or $\mathrm{nc}_\pi(f) \in \{0, -1\}$ for all $f \in G_s$, since $\mathrm{nc}_\pi(f) = 0$ for the outer face f. Similar to Section 4.1.1, the vertices of π separate the faces/vertices with nc=0 from those with nc=-1/1. First, the vertices of π are put in C. Then, the set of vertices that are on the boundary of some f with $nc_\pi(f) \neq 1$ and not on π are put in A, and the rest of the vertices (those not in C) are put in B. This is a vertex-cut with cost at most $c(\pi)$ because every vertex in π is counted at least once in the cost computation (by definition of $c(\pi)$ for a walk π). For the weight, Proposition 4.1 shows that $w(A) \leq w(\{f \mid \mathrm{nc}_\pi(f) \neq 0\}) = w(\pi)$ (the last equality follows from $\Delta_\pi(G_s) \leq 1$) and similarly that $w(B) \leq w(\{f \mid \mathrm{nc}_\pi(f) = 0\}) = W - w(\pi)$. Thus, $w(A) + w(C) = W - w(B) \geq W - w(\{f \mid \mathrm{nc}_\pi(f) = 0\}) = W - (W - w(\pi)) = w(\pi)$ and similarly $w(B) + w(C) \geq W - w(\pi)$. Thus, $\mathrm{bal}^1(A, B, C) = min(w(A) + w(C), w(B) + w(C)) \geq min(W - w(\pi), w(\pi))$. We already showed $w(B) \leq W - w(\pi)$ and $w(A) \leq w(\pi)$, so we get $\mathrm{bal}^0(A, B, C) = min(w(A), w(B)) \leq min(W - w(\pi), w(\pi))$. □

Proof of Theorem 4.9. Let $\mu \geq 1$ such that $\mu = \frac{4}{3}\frac{d}{d-1}(1 + \frac{1}{\mu^2 - 1})$. Let $\beta \geq 1$ be a parameter that we will select later to be $\mu^2 - 1$.

If $w(A^*) \leq w(B^*)$ and $w(A^*) \leq \beta \cdot w(C^*)$, then there is a trivial vertex-cut (A, B, C^*) with $A = V_0 \setminus C^*$, $B = \emptyset$ and

$$q^1(A, B, C^*) = \frac{c(C^*)}{w(C^*)} \leq \frac{c(C^*)}{\frac{1}{\beta+1}(w(A^*) + w(C^*))} = (\beta+1) q^1(A^*, B^*, C^*).$$

On the other hand, Theorem **??** shows that there is a d-tree in G_D that defines a vertex-cut (A, B, C) whose quotient cost satisfies $q^0(A, B, C) \leq \frac{4}{3}\frac{d}{d-1} \cdot OPT^0$. If $w(A^*) \geq \lambda \cdot w(C^*)$, for some $\lambda > \beta$, then Lemma 2.4 shows that $q^1(A, B, C) \leq \frac{4}{3}\frac{d}{d-1}(1 + \frac{1}{\lambda}) \cdot q^1(A^*, B^*, C^*)$. and also that $w(A), w(B) \geq w(C) \cdot \frac{1}{(\frac{4}{3}\frac{d}{d-1} - 1)(1 + \frac{1}{\lambda}) - \frac{1}{\lambda}}$. Let $\mu = \frac{4}{3}\frac{d}{d-1}(1 + \frac{1}{\lambda})$, and let $\delta = \mu - (1 + \frac{1}{\lambda}) - \frac{1}{\lambda}$. Using $w(A), w(B) \geq \frac{1}{\delta} w(C)$ in Corollary 4.8 shows that our algorithm finds a vertex cut (A', B', C') such that

$$\begin{aligned} q^1(A', B', C') &\leq (1 + \delta) q^1(A, B, C) \leq \\ &(1 + \delta)\mu \cdot q^1(A^*, B^*, C^*) \leq \\ &(1 + \mu - (1 + \tfrac{1}{\lambda}) - \tfrac{1}{\lambda})\mu \cdot q^1(A^*, B^*, C^*) = \\ &(\mu - \tfrac{1}{\lambda} - \tfrac{1}{\lambda})\mu \cdot q^1(A^*, B^*, C^*) \leq \\ &\mu^2 \cdot q^1(A^*, B^*, C^*) \end{aligned}$$

Finally, choose $\beta = \mu^2 - 1$. This shows that our resulting vertex-cut has q^1 quotient at most $\mu^2 \cdot q^1(A^*, B^*, C^*)$. Now, let $\mu_\infty = \lim_{d \to \infty} \mu_d$, for μ_d the μ we chose in the beginning of the proof. Then, $\mu_\infty^2 - \frac{4}{3}\mu_\infty - 1 = 0$, and $\mu_\infty = \frac{\frac{4}{3} + \sqrt{\frac{16}{9} + 4}}{2}$. Similarly, we can compute for an arbitrary d that $\mu_d = \frac{\frac{4}{3}\frac{d}{d-1} + \sqrt{\frac{16}{9}(\frac{d}{d-1})^2 + 4}}{2}$. Thus,

$$\frac{\mu_d}{\mu_\infty} = \frac{\frac{4}{3}\frac{d}{d-1} + \sqrt{\frac{16}{9}(\frac{d}{d-1})^2 + 4}}{\frac{4}{3} + \sqrt{\frac{16}{9} + 4}} \leq \frac{\frac{4}{3}\frac{d}{d-1} + \sqrt{\frac{16}{9} + 4}\frac{d}{d-1}}{\frac{4}{3} + \sqrt{\frac{16}{9} + 4}} = \frac{d}{d-1}$$

Since $\mu_\infty^2 \leq 3.5$ we get that $\mu_d^2 \leq 3.5(\frac{d}{d-1})^2$, so $q^1(A, B, C) \leq 3.5 \cdot (\frac{d}{d-1})^2 \cdot q^1(A^*, B^*, C^*)$.

For the second part of the theorem, notice that the vertex-cut that Theorem **??** selected for us (in the previous paragraph) satisfies $q^1(A, B, C) \leq \frac{4}{3}\frac{d}{d-1}(1 + \frac{1}{\lambda}) q^1(A^*, B^*, C^*)$. Corollary 4.8 says that our algorithm finds a vertex-cut (A', B', C') such that $q^1(A', B', C') \leq (1 + \frac{1}{\lambda}) q^1(A, B, C)$. Thus, $q^1(A', B', C') \leq \frac{4}{3} \cdot \frac{d}{d-1} \cdot (1 + \frac{1}{\lambda})^2 \cdot q^1(A^*, B^*, C^*)$. □

D. PROOF OF FINDING A NEAR OPTIMAL SEPARATOR

Proof of Theorem 5.1. The algorithm works as follows. Iteratively, find a ρ-approximate b-limited α-quotient vertex-cut; let (A_i, B_i, C_i) denote the vertex-cut found at iteration i, and assume without loss of generality that $w(A_i) \leq w(B_i)$; then remove from the graph the vertices of $A_i \cup C_i$. Repeat these iterations until the total weight of the (remaining) graph vertices is at most $(1 - \lambda)W$, where $0 < \lambda < 1$ is a parameter that we will choose later to be b'/α (much of the analysis that follows is independent of this choice of λ). Finally, output the vertex-cut (A', B', C') where $A' = \cup_i A_i$, $C' = \cup_i C_i$ and $B' = V \setminus (A' \cup C')$.

We first show that the output cut (A', B', C') is (b', α)-balanced. Let $k \geq 1$ be the number of iterations performed. Then $B' = B_k$. It follows that $w(B') + w(C') \geq w(B_k) + w(C_k) = w(B_{k-1}) - w(A_k)$. Since $w(A_k) = \min\{w(A_k), w(B_k)\} \leq \frac{1}{2} w(B_{k-1})$ we get that $w(B') + w(C') \geq \frac{1}{2} w(B_{k-1}) > \frac{1}{2}(1 - \lambda)W$. We also have that $w(B') \leq (1 - \lambda)W$ and thus $w(A') + w(C') \geq \lambda W$. We conclude, using $\alpha \leq 1$, that

$$\min\{w(A'), w(B')\} + \alpha w(C') \geq \alpha \cdot \min\{w(A'), w(B')\} + \alpha w(C') \geq \alpha \cdot \min\{\frac{1}{2}$$

Now, let us choose $\lambda := b'/\alpha \leq \frac{1}{3}$. Then $\min\{\frac{1}{2}(1 - \lambda), \lambda\} \geq \lambda$ and we get, as desired, that $\min\{w(A'), w(B')\} + \alpha w(C') \geq$

$\alpha\lambda W = b'W$.

We next bound the cost of the cut (A', B', C'). Let (A^*, B^*, C^*) be a minimum cost (b, α)-balanced cut, and let $\mathrm{OPT} = c(C^*)$ denote its cost. Iteration i is performed on the subgraph induced on B_{i-1}; this graph contains the vertex-cut that is induced by (A^*, B^*, C^*), namely, $(B_{i-1} \cap A^*, B_{i-1} \cap B^*, B_{i-1} \cap C^*)$. Without loss of generality assume that $w(B_{i-1} \cap A^*) \leq w(B_{i-1} \cap B^*)$. We know that $w(B_{i-1}) \geq (1-\lambda)W$, for W the initial total weight of the graph. Thus, $w(B_{i-1} \cap A^*) + w(B_{i-1} \cap B^*) + w(B_{i-1} \cap C^*) \geq (1-\lambda)W$. Since (A^*, B^*, C^*) is (b, α) balanced we get that $w(A^*) + \alpha \cdot w(C^*) \geq bW$ we also get that

$$w(B_{i-1} \cap B^*) + (1-\alpha) \cdot w(B_{i-1} \cap C^*) \leq w(B^*) + (1-\alpha) w(C^*) = W - (w(A^*) + \alpha \cdot w(C^*)) \leq W - bW = (1-b)W.$$

Thus,

$$w(B_{i-1} \cap A^*) + \alpha \cdot w(B_{i-1} \cap C^*) \geq (1-\lambda)W - \big(w(B_{i-1} \cap B^*) + (1-\alpha) \cdot w(B_{i-1} \cap C^*)\big) \geq (1-\lambda)W - (1-b)W = (b-\lambda)W.$$

Thus, the b-limited α-quotient cost of this cut is

$$q_b^\alpha(B_{i-1} \cap A^*, B_{i-1} \cap B^*, B_{i-1} \cap C^*) \leq \frac{\mathrm{OPT}}{(b-\lambda)W}$$

It follows that the ρ-approximate b-limited α-quotient vertex-cut (A_i, B_i, C_i) that the algorithm finds has quotient cost

$$q_b^\alpha(A_i, B_i, C_i) \leq \rho \cdot \frac{\mathrm{OPT}}{(b-\lambda)W}$$

Thus, the total cost of the output cut (A', B', C') is

$$c(C') = \sum_i c(C_i) \leq \rho \cdot \frac{\mathrm{OPT}}{(b-\lambda)W} \sum_i \min\{w(A_i) + \alpha w(C_i), bW\} \leq \rho \cdot \frac{\mathrm{OPT}}{(b-\lambda)W} [w(A') + \alpha w(C')].$$

Using $\alpha \leq 1$ we get that $w(A') + \alpha w(C') \leq w(A') + w(C') \leq W$, and thus conclude that

$$c(C') \leq \frac{\rho}{b-\lambda} \cdot \mathrm{OPT} = \frac{\rho}{b - b'/\alpha} \cdot \mathrm{OPT}.$$

□