# New Hardness Results for Planar Graph Problems in P and an Algorithm for Sparsest Cut

Amir Abboud
IBM Almaden Research Center
Almaden, U.S.A.
amir.abboud@gmail.com

Vincent Cohen-Addad
Sorbonne Universités, UPMC Univ
Paris 06, CNRS, LIP6
Paris, France
vcohenad@gmail.com

Philip N. Klein
Brown University
Providence, U.S.A.
klein@brown.edu

## ABSTRACT

The Sparsest Cut is a fundamental optimization problem that has been extensively studied. For planar inputs the problem is in $P$ and can be solved in $\tilde{O}(n^3)$ time if all vertex weights are 1. Despite a significant amount of effort, the best algorithms date back to the early 90's and can only achieve $O(\log n)$-approximation in $\tilde{O}(n)$ time or 3.5-approximation in $\tilde{O}(n^2)$ time [Rao, STOC92]. Our main result is an $\Omega(n^{2-\varepsilon})$ lower bound for Sparsest Cut even in planar graphs with unit vertex weights, under the $(min, +)$-Convolution conjecture, showing that approximations are inevitable in the near-linear time regime. To complement the lower bound, we provide a 3.3-approximation in near-linear time, improving upon the 25-year old result of Rao in both time and accuracy.

Our lower bound accomplishes a repeatedly raised challenge by being *the first fine-grained lower bound* for a natural planar graph problem in P. Moreover, we prove near-quadratic lower bounds under SETH for variants of the closest pair problem in planar graphs, and use them to show that the popular Average-Linkage procedure for Hierarchical Clustering cannot be simulated in truly subquadratic time.

At the core of our constructions is a diamond-like gadget that also settles the complexity of Diameter in *distributed* planar networks. We prove an $\Omega(n/\log n)$ lower bound on the number of communication rounds required to compute the weighted diameter of a network in the CONGEST model, even when the underlying graph is planar and all nodes are $D = 4$ hops away from each other. This is the first $poly(n)+\omega(D)$ lower bound in the planar-distributed setting, and it complements the recent $poly(D, \log n)$ upper bounds of Li and Parter [STOC 2019] for (exact) unweighted diameter and for $(1 + \varepsilon)$ approximate weighted diameter.

## CCS CONCEPTS

• **Theory of computation → Graph algorithms analysis**.

## KEYWORDS

Algorithm, Planar graphs, Sparsest cut


*STOC '20, June 22–26, 2020, Chicago, IL, USA*

ACM ISBN 978-1-4503-6979-4/20/06…$15.00
https://doi.org/10.1145/3357713.3384310

**ACM Reference Format:**
Amir Abboud, Vincent Cohen-Addad, and Philip N. Klein. 2020. New Hardness Results for Planar Graph Problems in P and an Algorithm for Sparsest Cut. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC '20), June 22–26, 2020, Chicago, IL, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3357713.3384310

## 1 INTRODUCTION

*Cuts in Planar Graphs.* The Sparsest Cut problem is among the most fundamental optimization problems. It is NP-hard and one of the most important problems in the field of approximation algorithms; through the years, it has led to the design of new, powerful algorithmic techniques (e.g. the $O(\sqrt{\log n})$-approximation of Arora, Rao, and Vazirani [12]), and is also increasingly becoming a keystone of divide-and-conquer strategies for a variety of problems arising in graph compression [31], clustering [30], and beyond. The goal is to cut the graph into two roughly balanced parts while cutting as few edges as possible.

This paper studies this problem in *planar graphs* where it is solvable in (weakly) polynomial time, in part because the cut can be shown to be a cycle in the dual of the graph. Let us mention two motivating reasons. First, finding sparse cuts in planar graphs is of high interest in applications such as network evaluation [74], VLSI design [18, 55], and more [59]. For instance, sparse cuts are used to identify portions of road networks that may suffer from congestion, or to design good VLSI layouts. A second motivation comes from finding *optimal* separators, a ubiquitous subtask in planar graph algorithms. The classic result of Lipton and Tarjan [60] shows that any bounded-degree planar graph has a balanced vertex separator of size $O(\sqrt{n})$, but this bound may be suboptimal in many cases (see for example [74]). In such non-worst-case instances, algorithms for finding better separators could speed up many algorithms.

There are two common ways to define the value (or sparsity) of a cut: we divide its cost by either the weight of the smaller of the two sides, or their product. The former definition is more standard and easier to work with in planar graphs; we will refer to it as MQC, defined below. We will refer to the other one, that asks to find asks to find the cut $S$ that minimizes $\frac{\text{cost}(S)}{w(S)\cdot w(V-S)}$ simply as Sparsest Cut.

Definition 1 (The Minimum Quotient Cut Problem (MQC)). *Given a graph $G = (V, E)$ with edge costs $c : E \mapsto \mathbb{R}^+$ and vertex weights $w : V \mapsto \mathbb{R}^+$, find the cut $(S, V - S), S \subseteq V$ minimizing the quotient:*

$$quotient(S) := \frac{cost(S)}{\min\{w(S), w(V - S)\}}$$

*where* $cost(X) := \sum_{\substack{(u,v)\in E \\ u\in X, v\notin X}} c(u,v)$ *and* $w(X) = \sum_{u\in X} w(u)$.

The study of Sparsest Cut in planar graphs dates back to Rao's first paper in the 80's [70], and to subsequent works by Rao [71] and by Park and Phillips [66]. The first *exact* algorithm was by Park and Phillips and had a running time of $\tilde{O}(n^2W)$ where $W$ is the sum of the vertex weights. Here the $\tilde{O}(\cdot)$ notation hides logarithmic factors in $n$, $W$, and $C$ the sum of edge costs. Note that in the "unweighted" case of unit vertex weights, $W = n$ and this upper bound is $\tilde{O}(n^3)$. They also showed that the problem is weakly NP-Hard, and therefore cannot be solved exactly in $\tilde{O}(poly(n))$ time. Rao's work gave a 3.5-approximation for MQC in $\tilde{O}(n^2)$ time, and an $O(\log n)$ approximation in $\tilde{O}(n)$ time.

Since then, there has been progress on related problems such as the Minimum Bisection problem where we want to find the cut of minimum cost that is balanced, i.e. $w(S) = w(V - S) = W/2$. Rao [70] showed that an approximation algorithm for MQC can be used to approximate Minimum Bisection in the following bicriteria way: we return a cut with at most two-thirds of the vertex weight on each side and with cost that is $O(\log n)$ times the cost of the minimum bisection. Garg et al. [37] gave a different algorithm with similar bicriteria guarantees where the cost is only a factor of 2 away from the optimal. This involves iterative application of the exact algorithm of Park and Phillips. More recently, Fox et al. [33] gave a polynomial-time bicriteria approximation scheme for Minimum Bisection but the algorithm runs in time $n^{\text{poly}(1/\varepsilon)}$.

Still, almost no progress has been made on Sparsest Cut since the early 90's. In the most interesting regime of near-linear running times, Rao's $O(\log n)$-approximation is the best known, and there is no exact algorithm running in time $\tilde{o}(n^2W)$. The gaps are large, but the most pressing question is:

Open Question 2. *Can Sparsest Cut in planar graphs be solved* exactly *in near-linear time?*

Given that the upper bound is longstanding, it is natural to try to use the recent tools of fine-grained complexity in order to resolve this question negatively. Can we show that a linear time algorithm would refute SETH or one of the other popular conjectures? This is challenging because this field has not been successful in proving *any conditional lower bound for a planar graph problem in P*, not to mention a natural and important problem like Sparsest Cut. Nonetheless, our main result is a quadratic conditional lower bound even for the unit-vertex-weight version of Sparsest Cut where the upper bound is cubic, and it also applies for MQC and Minimum Bisection. The lower bound is based on the hypothesis that the basic $(min, +)$-Convolution problem requires quadratic time. This hardness assumption was recently highlighted by Cygan et al. [28] after being used in other papers [15, 17, 52, 53]. It is particularly appealing because it implies both the 3-SUM and the All-Pairs Shortest Paths conjectures, and therefore also all the dozens of lower bounds that follow from them (see [28, 77]).

Theorem 3. *If for some $\varepsilon > 0$, the Sparsest Cut, the Minimum Quotient Cut, or the Minimum Bisection problems can be solved in $O(n^{2-\varepsilon})$ time in planar graphs of treewidth 3 on $n$ vertices with unit vertex-weights and total edge cost $C = n^{O(1)}$, then the $(min, +)$-Convolution problem can be solved in $O(n^{2-\varepsilon})$ time.*

After settling the high-order question it is easier to direct our energies into decreasing the gaps. A natural next question is whether there could be a cubic lower bound, which would completely settle the exact case. We show that this is not the case; a natural use of $O(\sqrt{n})$-size separators in the right way inside the Park and Phillips algorithm reduces the running time to $n^{2.5}$. Figuring out the exact exponent remains an important open question. It seems that new algorithmic techniques will be required to bring the upper bound down to $O(nW)$, yet we do not know of hard instances that seem to require super-quadratic time.

Theorem 4. *The Sparsest Cut and Minimum Quotient Cut problems in planar graphs on $n$ vertices with total vertex-weight $W$ and total edge costs $C$ can be solved in $O(n^{3/2}W\log(CW))$ time.*

Since near-linear time algorithms are the most desirable, perhaps the next most pressing question is whether the $O(\log n)$ approximation of Rao is the best possible:

Open Question 5. *Is there an $O(1)$-approximation algorithm for MQC in near-linear time?*

Our main algorithmic result in this work is a near-linear time 3.3-approximation algorithm for MQC, showing that constant factors are indeed possible. Surprisingly, we also beat the previous 3.5-approximation in $\tilde{O}(n^2)$ time both in time and accuracy. Our algorithm combines several techniques with new ideas; the main advantage comes from finding and utilizing a node that is guaranteed to be close to the optimal cycle rather than on it.

Theorem 6. *The Minimum Quotient Cut problem in planar graphs on $n$ vertices with total vertex-weight $W$ and total edge cost $C$ can be approximated to within a factor of 3.3 in time $n\log^{O(1)}(CWn)$.*

*New Hardness Results in Planar Graphs.* Theorem 3 finally resolves a repeatedly raised challenge in fine-grained complexity: *Are there natural planar graph problems in P for which we can prove a conditional $\omega(n)$ lower bound?* The list of problems with such lower bounds under SETH or other conjectures is long, exhibiting problems on graphs [78], strings [9, 14], geometric data [13, 19, 36], trees [2, 20], dynamic graphs [7, 48], compressed strings [1], and more[1]. Perhaps the most related results are the lower bounds for problems on *dynamic* planar graphs [6] but those techniques do not seem to carry over to the more restricted setting of (static) graphs. Indeed, the above question has been raised repeatedly, even after [6], including in the best paper talk of Cabello at SODA 2017 [22]. The search for an answer to this question has been remarkably fruitful from the viewpoint of upper bounds; Cabello's breakthrough (a subquadratic time algorithm for computing the diameter of a planar graph) came following attempts at proving a quadratic lower bound (such a lower bound holds in sparse but non-planar graphs [73]), and the techniques introduced in his work (mainly Abstract Voronoi Diagrams) have led to major breakthroughs in planar graph algorithms (see [25, 27, 38, 39]).

Strong lower bounds were found for some restricted graph classes such as graphs with logarithmic treewidth [8] (e.g. a quadratic lower bound for Diameter); but these are incomparable with planar graphs. For some problems such as subgraph isomorphism there are lower

[1] For a more extensive list see the survey in [77].

bounds even for trees [2], a restricted kind of planar graphs; however these problems are not in P when the graphs are planar but not trees. Many hardness results are known for geometric problems on points in the plane (e.g. [13, 16, 36]); while related in flavor, the techniques are specific to the euclidean nature of the data and it is not clear how to extract lower bounds for natural graph problems out of these results.

The main challenge, of course, is in designing *planar* gadgets and constructions that are capable of encoding fine-grained reductions. While this has already been accomplished in other contexts such as NP-hardness proofs or in parameterized complexity, those techniques do not work under the more strict efficiency requirements that are needed for fine-grained reductions. From the perspective of lower bounds, the main contribution of this paper is in coming up with a planar construction that exhibits the super-linear complexity of basic problems like Sparsest Cut. By extracting the core gadget from this construction and building up on it we are able to prove lower bounds for other, seemingly unrelated problems on planar graphs.

Notably, our constructions are not only planar but also have very small treewidth of two or three, but crucially not one since our problems become easy on trees. This might be of independent interest.

*Closest Pair of Sets and Hierarchical Clustering.* Hierarchical Clustering (HC) is a ubiquitous task in data science and machine learning. Given a data set of $n$ points with some similarity or distance function over them (e.g. points in Euclidean space, or the nodes of a planar graph with the shortest path metric), the goal is to group similar points together into clusters, and then recursively group similar clusters into larger clusters. Perhaps the two most popular procedures for HC are Single-Linkage and Average-Linkage. Both are so-called *agglomerative* HC algorithms (as opposed to *divisive*) since they proceed in a bottom-up fashion: In the beginning, each data point is in its own cluster, and then the most similar clusters are iteratively merged - creating a larger cluster that contains the union of the points from the two smaller clusters - until all points are in the same, final cluster.

The difference between the different procedures is in their notion of similarity between clusters, which determines the choice of clusters to be merged. In Single-Linkage the distance (or *dissimilarity*) is defined as the minimum distance between any two points, one from each cluster. While in Average-Linkage we take the average instead of the minimum. It is widely accepted that Single-Linkage is sometimes simpler and faster, but the results of Average-Linkage are often more meaningful. Extensive discussions of these two procedures (and a few others, such as Complete-Linkage where we take the max, rather than min or average) can be found in many books (e.g. [10, 34, 56, 75]), surveys (e.g. [23, 63, 64]), and experimental studies (e.g. [68]).

Both of these procedures can be performed in nearly quadratic time and a faster, subquadratic implementation is highly desirable. Some subquadratic algorithms that try to approximate the performance of these procedures have been proposed, e.g. [5, 26]. However, it is often observed that an exact implementation is at least as hard as finding the closest pair (of data points), since they are the first pair to be merged. Indeed, if the points are in Euclidean space with $\omega(\log n)$ dimensions, the Closest Pair problem requires quadratic time under SETH [11, 49], and therefore these procedures cannot be sped up without a loss.

But what if we are in the planar graph metric? This argument breaks down because the Closest Pair problem is trivial in planar graphs (the minimum weight edge is the answer). Moreover, the Single-Linkage procedure can be implemented to run in near-linear time in this setting, since it reduces to the computation of a minimum spanning tree [43]. In fact, subquadratic algorithms are known for many other metrics that have subquadratic closest pair algorithms such as spaces with bounded doubling dimension [61], and efficient approximations are known when the closest pair can be approximated efficiently [5]. This naturally leads to the question:

Open Question 7. *Can Average-Linkage be computed in subquadratic time in any metric where the closest pair can be computed in subquadratic time?*

Surprisingly to us, it turns out that the answer is no. In this paper we prove a near-quadratic lower bound under SETH for simulating the Average-Linkage and Complete-Linkage procedures in planar graphs, by proving a lower bound for variants of the closest *pair of sets* problem which are natural problems of independent interest: We are given a planar graph on $n$ nodes that are partitioned into $O(n)$ sets and the goal is to find the pair of sets that minimizes the sum (or max) of pairwise distances. An $O(n^2)$ upper bound is easy to obtain from an all-pairs shortest paths computation.

Theorem 8. *If for some $\varepsilon > 0$, the Closest Pair of Sets problem, with sum-distance or max-distance, in unweighted planar graphs on $n$ nodes can be solved in $O(n^{2-\varepsilon})$ time, then SETH is false. Moreover, if for some $\varepsilon > 0$ the Average-Linkage or Complete-Linkage algorithms on $n$ node planar graphs with edge weights in $[O(\log n)]$ can be simulated in $O(n^{2-\varepsilon})$ time, then SETH is false.*

*Diameter in Distributed Graphs.* Our final result is on the complexity of diameter in planar graphs in the CONGEST model. This is the central theoretical model for *distributed computation*, where the input graph defines the communication topology: in each round, each of the $n$ nodes can send an $(\log n)$-bit message to each one of their neighbors. The complexity of a problem is the worst case number of rounds until all nodes know the answer.

In the CONGEST, a problem is considered tractable if it can be solved in $poly(D, \log n)$ time, where $D$ is the diameter of the underlying unweighted network[2] (i.e. the hop-diameter). Many basic problems such as finding a Minimum Spanning Tree (MST) and distance computations have been shown to be intractable [4, 21, 24, 29, 32, 35, 65, 69]. For example, no algorithm can decide whether the diameter of the network is 3 or 4 in $n^{o(1)}$ rounds [35]. That is, the Diameter problem itself cannot be solved in $poly$(diameter) time.

While (sequential[3]) algorithms for planar graphs have been an extensively studied subject for the past three decades, only recently have they been considered in the distributed setting [42, 44–47, 57]. This study was initiated by Ghaffari and Hauepler [40, 41] who also demonstrated its potential: While MST has an $\Omega(\sqrt{n})$ lower bound in general graphs [29], the problem is tractable on planar graphs.

[2]Note that $\Omega(D)$ rounds are usually required; some nodes cannot exchange any information otherwise.

[3]This seems to be the standard term for *not distributed* algorithms.

All previous lower bound constructions are far from being planar, and it is natural to wonder: Do *all* problems[4] become tractable in the CONGEST when the network is planar?[5]

In this paper, we provide a negative answer with a simple argument ruling out any $f(D) \cdot n^{o(1)}$ distributed algorithms even in planar graphs. A very recent breakthrough of Li and Pater showed that the diameter problem in *unweighted planar graphs* is tractable in the CONGEST [58]. We show that the *weighted* case is intractable. Our lower bound is only against exact algorithms which is best-possible since Li and Parter achieve a $(1+\varepsilon)$-approximation in the weighted case with $\tilde{O}(D^6)$ rounds.

Theorem 9. *The number of rounds needed for any protocol to compute the diameter of a weighted planar network of constant hop-diameter $D = O(1)$ on $n$ nodes in the CONGEST model is $\Omega(\frac{n}{\log n})$.*

Our technique for showing lower bounds in the CONGEST model is by reduction from two-party communication complexity and is similar to the one in previous works. For general graphs, there are strong $\Omega(n/\log^2 n)$ lower bounds for computing the diameter even in unweighted, sparse graphs of constant diameter [4, 21, 35]. Our high-level approach is similar, but a substantially different implementation is needed in order to keep the graph planar. In fact, we design a simple but subtle, diamond-like gadget for this purpose (see Section 3). The other lower bounds in the paper were obtained by building on top of this simple construction and they show that this gadget may really be capturing the difficulty in many planar graph problems. In particular, the lower bounds for closest pair of sets, which are the most complicated in this paper, are achieved by combining $O(n)$ copies of this gadget together in an "outer construction" that also has the same structure of this gadget.

## 2 SPARSEST CUT, MINIMUM QUOTIENT CUT AND MINIMUM BISECTION

We now provide formal definitions of the Sparsest Cut, Minimum Quotient Cut and Minimum Bisection problems. Consider a planar graph $G = (V, E)$ with edge costs $c : E \mapsto \mathbb{R}^+$ and vertex weights $w : V \mapsto \mathbb{R}^+$. Given a subset of vertices $S$, we define the *cut induced by $S$* as the set of edges with one extremity in $S$ and the other in $V - S$. We will slightly abuse notation by referring to the cut induced by $S$ as the cut of $S$. We let

$$\text{cost}(S) = \sum_{\substack{(u,v)\in E \\ u\in S, v\notin S}} c(u,v)$$

and, with a slight abuse of notation, $w(S) = \sum_{v\in S} w(v)$. Given a subset of vertices $S$, we define the *sparsity of the cut induced by $S$* as the ratio $\text{cost}(S)/(w(S)\cdot w(V-S))$. The *Sparsest Cut problem* asks for a subset $S$ of $V$ that has minimum sparsity over all cuts induced by a subset $S \subset V$. This is not to be confused with the General Sparsest Cut which is APX-Hard in planar graphs[6].

[4]Here, we mean *decision* problems. It is easy to show that problems with a large output such as All-Pairs-Shortest-Paths are not tractable even in trivial networks.

[5]Note that even NP-Hard problems might become tractable in this model, since the only measure is the number of rounds, not the computation time at the nodes. For example, in the LOCAL model where we do not restrict the messages to be short, all problems can be solved in $O(D)$ rounds.

[6]There, there is a weighted demand between pair of vertices and the goal is to find a subset $S$ such that the cut induced by $S$ minimizes the ratio of $\text{cut}(S)$ to the amount of demand between pairs of vertices in $S$ and $V - S$.

The *quotient* of a cut $S$ is defined to be $\text{cost}(S)/(\min\{w(S), w(V-S)\})$. The *Minimum Quotient Cut* problem asks for a cut with minimum quotient. The *Minimum Bisection problem* asks for a subset $S$ such that $w(S) = w(V)/2$ and that minimizes $\text{cost}(S)$.

### 2.1 Proof of Theorem 3: Lower Bounds

In this section, we aim prove a conditional lower bound of $\Omega(n^{2-\varepsilon})$ for the unit vertex-weight case of all three problems: the Sparsest Cut, the Minimum Quotient Cut, and the Minimum Bisection problems. We will first provide a reduction for the case of non-unit vertex-weight and then show how to adapt it to the unit vertex-weight case.

Our lower bounds are based on the hardness of the $(min, +)$-Convolution Problem, defined as follows, which is conjectured to require $\Omega(n^{2-\varepsilon}$ time, for all $\varepsilon > 0$.

Definition 10 (The $(min, +)$-Convolution Problem). *Given two sequences of integers $A = \langle a_1, \ldots, a_n\rangle$, $B = \langle b_1, \ldots, b_n\rangle$, the output is a sequence $C = \langle c_1, c_2, \ldots, c_n\rangle$ such that $c_k = \min_{0\le i\le k} a_{k-i+1} + b_i$.*

To prove our conditional lower bounds we will show reductions from the following variant called $(min, +)$-Convolution Upper Bound, which was shown to be *subquadratic-equivalent* to $(min, +)$-Convolution by Cygan et al. [28]. Namely, there is an $O(n^{2-\varepsilon})$ algorithm for some constant $\varepsilon > 0$ for the $(min, +)$-Convolution Upper Bound problem if and only if there is an $O(n^{2-\varepsilon'})$ for the $(min, +)$-Convolution problem, for some $\varepsilon' > 0$.

Definition 11 (The $(min, +)$-Convolution Upper Bound Problem). *Given three sequences of positive integers $A = \langle a_1, a_2, \ldots, a_n\rangle$, $B = \langle b_1, b_2, \ldots, b_n\rangle$, and $C = \langle c_1, c_2, \ldots, c_n\rangle$, verify that for all $k \in [n]$, there is no pair $i, j$ such that $i + j = k$ and $a_i + b_j < c_k$.*

*The Reduction.* The construction in each of our three reductions is the same, and the analysis is a little different in each case. Therefore, we will present all three in parallel. Given an instance $A, B, C$ of the $(min, +)$-Convolution Upper Bound problem, we build an instance $G$ for the Sparsest Cut, Minimum Quotient Cut or Minimum Bisection problems as follows.

Let $T = \sum_{i=1}^{n} a_i + b_i + c_i$ and $\beta = 4Tn^2$. The graph $G$ will have two special vertices $u$ and $v$ of weights $10n$ and $11n$ respectively. It will also have three paths $P_A, P_B, P_C$ that connect $u$ and $v$ and will encode the three sequences as follows.

- The path $P_A$ has a vertex $v_{a_i}$ for each $a_i$ in $A$, of weight 1. We connect $v_{a_i}$ to $v_{a_{i+1}}$ with an edge of cost $\beta + a_i$ for each $1 \le i < n$. Moreover, we connect $v_{a_n}$ to $u$ with an edge of cost $\beta + a_n$ and $v$ to $v_{a_1}$ with an edge $e_A$ of cost $1210n^2(2\beta + T)$.
- The path $P_B$ is defined in an analogous way. We create a vertex $v_{b_i}$ of weight 1 for each $b_i$ in $B$ and connect it with an edge of cost $\beta + b_i$ to $v_{b_{i+1}}$ for each $1 \le i < n$, and we connect $v_{b_n}$ to $u$ with an edge of cost $\beta + b_n$ and $v$ to $v_{b_1}$ with an edge $e_B$ of cost $1210n^2(2\beta + T)$.
- The path $P_C$ is defined differently: the indices are ordered in the opposite direction and the numbers are flipped. We create a vertex $v_{c_i}$ of weight 1 for each $c_i$ in $C$ but connect each $v_{c_i}$ to $v_{c_{i+1}}$ with an edge of cost $\beta + T - c_i$ for each

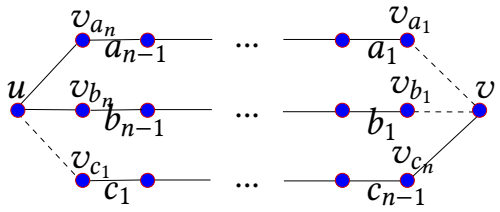


**Figure 1: The graph generated in our reduction. Dashed edges are edges of weight $1210n^2(2\beta + T)$.**

$1 \le i < n$. And this time we connect $v_{c_1}$ to $u$ with an edge $e_C$ of cost $1210n^2(2\beta + T)$ and $v_{c_n}$ to $v$ with an edge of cost $\beta + c_n$.

It is easy to see that the resulting graph is planar and has treewidth at most 3. See also Figure 1. The total weight in our construction is $W = 24n$ because there are $3n$ vertices of weight 1 and the two special vertices $u, v$ have weight $21n$.

*Correctness of the Reductions.* To analyze the reduction, we start by proving two lemmas about the structure of the optimal solution in each of the three problems in the instances we generate. To build intuition, observe that in our construction any cut that does not separate $u$ and $v$ is far from being balanced and therefore will not be an optimal solution. Another observation is that the weights of the edges $\{e_A, e_B, e_C\}$ is practically infinite and therefore they will not be cut by an optimal cut.

Lemma 12. *The Minimum Quotient Cut, the Sparsest Cut, and the Minimum Bisection cut intersect each of $P_A, P_B, P_C$ exactly once and do not intersect any edge of $\{e_A, e_B, e_C\}$.*

Proof. We start with the Minimum Bisection, which is the simplest case since the cut is forced to have exactly $W/2$ weight on each side. By picking edges $(v_{a_1}, v_{a_2})$, $(v_{b_1}, v_{b_2})$, and $(v_{c_2}, v_{c_3})$, we indeed obtain a cut that breaks the graph into two connected components of the same weight. The value of the cut is then at most $\beta + a_1 + \beta + b_1 + \beta + T - c_2 \le 3\beta + 2T$. However, any cut intersecting $\{e_A, e_B, e_C\}$ has cost at least $1210n^2(2\beta + T)$ and so the (optimal) Minimum Bisection does not intersect $\{e_A, e_B, e_C\}$. Moreover, it is easy to see that the Minimum Bisection Cut must separate $u$ from $v$ as otherwise, the cut is not balanced. Thus the Minimum Bisection intersects each of $P_A, P_B, P_C$ at least once. Finally, suppose it intersects them more than once. The cost is thus at least $4\beta$, while by picking edges $(v_{a_1}, v_{a_2})$, $(v_{b_1}, v_{b_2})$, and $(v_{c_2}, v_{c_3})$, the cost achieved is at most $2T + 3\beta$. By the choice of $\beta$, we have $2T + 3\beta < 4\beta$ and so the Minimum Bisection intersects each of $P_A, P_B, P_C$ exactly once.

We then argue that the Minimum Quotient cut $Q$ and the Sparsest Cut $S$ do not intersect any edge of $\{e_A, e_B, e_C\}$. Indeed, any cut $U$ that intersect an edge $\{e_A, e_B, e_C\}$ has cost at least $1210n^2(2\beta + T)$ and so induces a Quotient Cut of value at least $1210n^2(2\beta + T)/(12n) = 110n(2\beta + T)$ and a cut of Sparsity at least $1210n^2(2\beta + T)/(12n)^2 = 10(2\beta + T)$. Now, consider the cut separating $a_1$ from the rest of the graph. This cut has cost at most $2\beta + T$. Thus, it forms a quotient cut of value at most $2\beta + T$ and a cut of sparsity at most $(2\beta + T)/12n$. This induces a cut that is both of smaller sparsity and of smaller quotient value than any cut involving any of $\{e_A, e_B, e_C\}$. It follows that $Q$ and $S$ do not intersect $\{e_A, e_B, e_C\}$.

We now show that both $Q$ and $S$ separate $u$ from $v$. Consider a cut $U$ that has both $u$ and $v$ on one side. This cut needs to contain at least two edges and so has cost at least $2(\beta + 1)$. It thus induces a quotient cut of value at least $2(\beta + 1)/3n$ and a cut of sparsity at least $2(\beta + 1)/(22n \cdot 3n)$. On the other hand, consider a cut $Y$ obtained by picking an edge from each of $P_A, P_B, P_C$. The cost of this cut is at most $3\beta + 2T$, which induces a quotient cut of value at most $(3\beta + 2T)/(10n)$ and of sparsity $(3\beta + 2T)/(10n)^2$. Since $\beta = 4Tn^2$, we have that $(3\beta + 2T)/(10n) < (2\beta + 1)/(3n)$ and $(3\beta + 2T)/(10n)^2 < 2(\beta + 1)/(20n \cdot 3n)$, as long as $n > 1$. Therefore, $Q$ and $S$ separate $u$ from $v$ and so intersect at least one edge from each of $P_A, P_B, P_C$.

Finally, by Theorem 2.2 in [66] and Proposition 2.3 in [67], we have that the minimum quotient cut and the sparsest cut are simple cycles in the dual of the graph. Picking two edges of $P_A$ (or of $P_B$, or $P_C$) together with at least one edge of $P_B$ and of $P_C$ would induce a non-simple cycle in the dual of the graph and so a non-optimal cut. Therefore, we conclude that the minimum quotient cut and sparsest cut uses exactly one edge of $P_A$, one edge of $P_B$, and one edge of $P_C$. □

Lemma 13. *If the Minimum Quotient Cut, the Sparsest Cut, or the Minimum Bisection intersects edges $(v_{a_i}, v_{a_{i+1}})$, $(v_{b_j}, v_{b_{j+1}})$ and $(v_{c_k}, v_{c_{k+1}})$, then $v$ and the vertices in $\{v_{a_1}, \ldots, v_{a_i}\}$, $\{v_{b_1}, \ldots, v_{b_j}\}$, and $\{v_{c_{k+1}}, \ldots, v_{c_n}\}$, are on one side of the cut while the remaining vertices are on the other side.*

Proof. By Lemma 12, the Minimum Quotient Cut, the Sparsest Cut and the Minimum Bisection intersect each of $P_A, P_B, P_C$ exactly once. Thus, if one of them intersect edges $(v_{a_i}, v_{a_{i+1}})$, $(v_{b_j}, v_{b_{j+1}})$ and $(v_{c_k}, v_{c_{k+1}})$, then $v_{a_i}$ remains connected to $v$ through the path $\{v_{a_1}, \ldots, v_{a_i}\}$ and so all the vertices in $\{v, v_{a_1}, \ldots, v_{a_i}\}$ are in the same connected component. The remaining vertices of $P_A$ remains connected to $u$. A similar reasoning applies to $v_b$ and $v_c$ and yields the lemma. □

From these two lemmas it follows that the only way that an optimal cut can be completely balanced (i.e. has weight $W/2 = 12n$ on each side) is by cutting three edges $(v_{a_i}, v_{a_{i+1}})$, $(v_{b_j}, v_{b_{j+1}})$ and $(v_{c_k}, v_{c_{k+1}})$, where $i + j = k$. This is the crucial property of our construction. To see why it is true, note that $i + j + (n - k)$ vertices go to the side of $v$ while $(n - i) + (n - j) + k$ vertices go to the side of $u$, and so to achieve balance it must be that:

$$i + j - k + n + w(v) = k - i - j + 2n + w(u)$$

which simplifies to $i + j = k$ because of our choice of $w(u) = 10n$ and $w(v) = 11n$. Moreover, the cost of this cut is exactly $(3\beta + T) + (a_i + b_j - c_k)$ which is less than $(3\beta + T)$ if and only if $a_i + b_j < c_k$. The correctness of the reductions follows from the following claim.

Claim 14. *There is no $k \in [n]$ and a pair $i, j$ such that $i + j = k$ and $a_i + b_j < c_k$, if and only if either of the following statements is true:*

- *the Minimum Quotient Cut has value at least $(3\beta + T)/12n$,*
- *the Sparsest Cut has value at least $(3\beta + T)/(12n^2)$, or*
- *the Minimum Bisection has value at least $(3\beta + T)$.*

Proof. Consider first the Minimum Bisection. By Lemma 12, the Minimum Bisection intersects each of $P_A, P_B, P_C$ exactly once.

Thus, combined with Lemma 13, we have that the if the Minimum Bisection intersects an edge $(v_{c_k}, v_{c_{k+1}})$ for some $k$, then it must intersect $(v_{a_i}, v_{a_{i+1}})$, and $(v_{b_j}, v_{b_{j+1}})$ such that $j + i = k$ to achieve balance. Therefore, the cut has value $3\beta + a_i + b_{k-i} + T - c_k$ which is at least $3\beta + T$ if and only if there is no $i, j$ such that $i + j = k$ and $a_i + b_j < c_k$.

We now turn to the cases of Minimum Quotient Cut and Sparsest Cut. For the first direction, assume that there is a triple $i, j, k$ where $k = i + j$ such that $a_i + b_j < c_k$. In this case, we have a cut of quotient value less than $(3\beta+T)/(12n)$ and a cut of sparsity less than $(3\beta + T)/(12n)^2$ obtained by taking edges $(v_{a_i}, v_{a_{i+1}})$, $(v_{b_j}, v_{b_{j+1}})$ and $(v_{c_k}, v_{c_{k+1}})$.

For the other direction, let us first focus on the Minimum Quotient Cut $Q$. By Lemma 12, $Q$ contains one edge from each of $P_A, P_B, P_C$ say $(v_{a_i}, v_{a_{i+1}})$, $(v_{b_j}, v_{b_{j+1}})$ and $(v_{c_k}, v_{c_{k+1}})$. First, if $i + j \neq k$, by Lemma 13, we have that the cut has quotient value at least $(3\beta+a_i+b_j+T-c_k)/(12n-1)$ which is at least $(3\beta+1)/(12n-1)$. By the choice of $\beta$, we have that $3\beta/12n > 10T$ and so, $(3\beta + 1)/(12n - 1) \geq (3\beta + T)/12n$.

Thus, we may assume that $i + j = k$. By Lemma 13, we hence have that the quotient value of the cut is less than $(3\beta + T)/(12n)$ if and only if $a_i + b_j < c_k$. This follows from the fact that the quotient value of the cut is $(3\beta + a_i + b_i + T - c_k)/(12n)$ which is less than $(3\beta + T)/(12n)$ if and only if $a_i + b_j < c_k$.

The argument for the Sparsest Cut is similar. Again, by Lemma 13, the sparsest cut contains one edge from each of $P_A, P_B, P_C$, say $(v_{a_i}, v_{a_{i+1}})$, $(v_{b_j}, v_{b_{j+1}})$ and $(v_{c_k}, v_{c_{k+1}})$. Similarly, if $i + j = k$, we have that the sparsity of the cut is less than $(3\beta + T)/(12n)^2$ if and only if $a_i + b_j < c_k$, since the sparsity of the cut is $(3\beta + a_i + b_i + T - c_k)/(12n)^2$.

Finally, if $i+j \neq k$ then the sparsity of the cut is at least $(3\beta+a_i+b_j+T-c_k)/((12n-1)(12n+1))$ which is at least $(3\beta+1)/((12n-1)(12n+1))$. By the choice of $\beta$, we have that $3\beta/((12n)^2-1) > 10T$ and so, $(3\beta + 1)/(12n - 1) \geq (3\beta + T)/12n$. □

*A Unit-Vertex-Weight Reduction.* Intuitively, we are able to remove the weights because the total weight $W$ is $O(n)$. To show this more precisely, we note that the above reduction makes use of vertices of weight 1, except for $u$ and $v$ which are of weight $10n$ and weight $11n$ respectively. Now, place a weight of 1 on $u$ and $v$ and add vertices $u^1, \ldots, u^{10n-1}$ and connect them with edges of length $1210n^2(2\beta + T)$ to $u$ and add vertices $v^1, \ldots, v^{11n-1}$ and connect them with edges of length $1210n^2(2\beta + T)$ to $v$. For the same argument used in Lemma 12, the Minimum Quotient cut, the Sparsest Cut, and the Minimum Bisection do not intersect any of these edges and so the above proof can be applied unchanged.

## 3 LOWER BOUND FOR DIAMETER IN CONGEST

In this section we prove Theorem 9 and present the simple gadget that is at the core of our lower bounds.

Proof of Theorem 9. The proof is by reduction from the two-party communication complexity of Disjointness: There are two players, Alice and Bob, each has a private string of $n$ bits, $A, B \in \{0,1\}^n$ and their goal is to determine whether the strings are disjoint, i.e. for all $i \in [n]$ either $A[i] = 0$ or $B[i] = 0$ (or both). It is known that the two players must exchange $\Omega(n)$ bits of communication in order to solve this problem [72], even with randomness, and we will use this lower bound to derive our lower bound for distributed diameter.

Let $A, B \in \{0,1\}^n$ be the two private input strings in an instance $(A, B)$ of Disjointness. We will construct a planar graph $G$ on $O(n)$ nodes based on these strings and show that a CONGEST algorithm that can compute the diameter of $G$ in $T(n)$ rounds implies a communication protocol solving the instance $(A, B)$ in $O(T(n) \log n)$ rounds. This is enough to deduce our theorem.

The nodes $V$ of $G$ are partitioned into two types: nodes $V_A$ that "belong to Alice" and nodes $V_B$ that "belong to Bob". For each coordinate $i \in [n]$ we have two nodes $a_i \in V_A$ and $b_i \in V_B$. In addition, there are four special nodes: $\ell, r \in V_A$ and $\ell', r' \in V_B$. In total, there are $|V_A| + |V_B| = 2n + 4$ nodes in $G$.

Let us first describe the edges $E$ of $G$ before defining their weights $w : E \to G$. The edges are independent of the instance $(A, B)$ but their weights will be defined based on the strings. Every coordinate node $a_i$, for all $i \in [n]$, has two edges: one left-edge (which will be drawn to the left of $a_i$ in a planar embedding) connecting it to $\ell$, and one right-edge connecting it to $r$. Similarly for Bob's part of the graph, every coordinate $b_i$ has a left-edge to $\ell'$ and a right-edge to $r'$. Finally, there is an edge connecting $\ell$ with $\ell'$ and an edge connecting $r$ with $r'$.

One way to embed $G$ in the plane is as follows: The nodes $a_1, \ldots, a_n, b_1, \ldots, b_n$ are ordered in a vertical line with $a_1$ at the top. In between $a_n$ and $b_1$ we add some empty space in which we place the other four nodes in $G$ such that $\ell, \ell'$ are to the left of the vertical line and $r, r'$ are to the right, and the four nodes are placed in a rectangle-like shape with $\ell, r$ on top and $\ell', r'$ on the bottom.

The final shape (see Figure2) looks like a diamond (especially if we rotate it by 90 degrees) with $\ell, \ell'$ on top and $r, r'$ on the bottom. It is important to observe that the hop-diameter $D$ of this graph is a small constant, $D = 3$. A crucial property of $G$ for the purposes of reductions from two-party communication problems is that there is a very small cut between Alice's and Bob's parts of the graph: there are only two edges that go from one part to the other ($(\ell, \ell')$ and $(r, r')$).

The main power of this gadget comes from the weights, defined next. Set $M = 4$ (but it will be useful to think of $M$ as a large weight).

$$w(\ell, \ell') = M$$

$$w(r, r') = M$$

$$w(a_i, \ell) = \begin{cases} i \cdot M, & \text{if } A[i] = 0 \\ i \cdot M + 1, & \text{if } A[i] = 1 \end{cases}$$

$$w(a_i, r) = \begin{cases} (n + 1 - i) \cdot M, & \text{if } A[i] = 0 \\ (n + 1 - i) \cdot M + 1, & \text{if } A[i] = 1 \end{cases}$$

$$w(b_j, \ell') = \begin{cases} (n + 1 - j) \cdot M, & \text{if } B[j] = 0 \\ (n + 1 - j) \cdot M + 1, & \text{if } B[j] = 1 \end{cases}$$

$$w(b_j, r') = \begin{cases} j \cdot M, & \text{if } B[j] = 0 \\ j \cdot M + 1, & \text{if } B[j] = 1 \end{cases}$$

The key property of this construction is that every pair of nodes in $G$ will have distance less than $(n+2) \cdot M$ *except for pairs* $a_i, b_j$ *with*

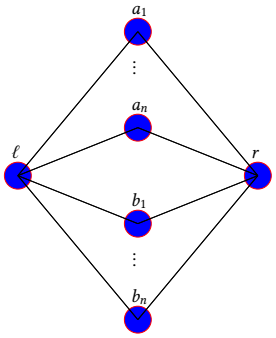


**Figure 2: Our basic construction. For the Diameter CONGEST lower bound, the nodes $\ell$ and $r$ are each split into an edge. The complexity of handling this gadget comes from a careful choice of the weights that makes $a_i$ and $b_i$ "interact" for all $i \in [n]$, while the other pairs are not effective.**

$i = j$. And for these special pairs $a_i, b_i$ the distance will be exactly $(n+2) \cdot M$ plus 0,1, or 2, depending on $A[i], B[i]$; thus the diameter of $G$ will be affected by whether $A, B$ are disjoint. Achieving this kind of property is the crux of most reductions from Disjointness to graph problems. Next we formally show such bounds on the distances in $G$.

Claim 15. *The weighted diameter of $G$ is $(n+2)\cdot M+2$ if there exists an $i \in [n]$ such that $A[i] = B[i] = 1$ and it is at most $(n+2) \cdot M + 1$ otherwise.*

Proof. The proof is by a case analysis on all pairs of nodes $x, y$ in $G$. We start with the less interesting cases, and the final case is the interesting one (which will depend on $A, B$).

- If $x = a_i$ and $y \in \{\ell, \ell', r, r'\}$ then the path of length one or two from $x$ to $y$ has weight $d(x,y) \leq n \cdot M + 1 + M = (n+1) \cdot M + 1$.
- Similarly for Bob's side, if $x = b_j$ and $y \in \{\ell, \ell', r, r'\}$ then $d(x,y) \leq n \cdot M + 1 + M = (n+1) \cdot M + 1$.
- If $x = a_i$ and $y = b_j$ but $i \neq j$ then the shortest path goes through the cheaper of the two ways (left or right). Specifically, the left path $(a_i, \ell, \ell', b_j)$ has weight $(i - j + n + 2) \cdot M + \alpha$ for some $\alpha\{0,1,2\}$ (that depends on the strings: $\alpha = A[i] + B[j]$), and the right $(a_i, r, r', b_j)$ path has weight $(j - i + n + 2) \cdot M + \alpha$. Thus, if $i < j$ we choose the left path, and if $i > j$ we choose the right path. In either case, $d(x,y) \leq (n+1) \cdot M + 2$.
- If $x = a_i$ and $y = a_j$ then we again have that $i \neq j$ (or else $x = y$) and the shortest path goes through the cheaper of the two ways (left or right). Specifically, the left path $(a_i, \ell, a_j)$ has weight $(i + j) \cdot M + \alpha$ for some $\alpha\{0,1,2\}$, and the right $(a_i, r, a_j)$ path has weight $(2n + 2 - i - j) \cdot M + \alpha$. Thus, if $i + j < n + 1$ we choose the left path, if $i + j > n + 1$ we choose the right path, and if $i + j = n + 1$ then both options are equally good. In either case, $d(x,y) \leq (n+1) \cdot M + 2$.
- The case that $x = b_i$ and $y = b_j$ is analogous.
- Now comes the final case of $x = a_i$ and $y = b_i$. These are the special pairs corresponding to the coordinates and their distances are larger than all the other distances in the graph. This happens because the two paths (left or right) have the same weight and are equally "bad". This weight is $(n+2) \cdot M + \alpha$ where $\alpha \in \{0,1,2\}$ is equal to $A[i] + B[i]$. Therefore, if $A, B$ are disjoint, then for all $i \in [n]$ we have $A[i] + B[i] \leq 1$ and so $d(a_i, b_i) \leq (n+2) \cdot M + 1$. Otherwise, if there is an $i \in [n]$ such that $A[i] = B[i] = 1$ then $d(a_i, b_i) = (n+2) \cdot M + 2$ which will be the furthest pair in the graph. Finally, observe that any path from $x$ to $y$ that uses more than three edges cannot be shortest, since its weights will be at least $(n+3) \cdot M$ and $M > 3$.

□

Thus we have constructed a graph $G$ from the strings $(A, B)$ such that diameter of $G$ is at most $(n+2) \cdot M + 1$ if and only if $(A, B)$ are disjoint. To conclude the proof we describe how a CONGEST algorithm for Diameter leads to a two-party communication protocol. Assume there is such an algorithm for Diameter with a $T(n)$ upper bound on the number of rounds. To use this algorithm for their two-party protocol, Alice and Bob look at their private inputs and construct the graph $G$ from our reduction. Note that all edges in Alice's part are known to Alice and all edges in Bob's part are known to Bob. The "common" edges which have one endpoint in each side are known to both players since they do not depend on the private inputs. Then, they can start simulating the algorithm. In each round, each node $x$ sends an $O(\log n)$-bit message to each one of its neighbors $y$. For the messages sent on "internal" edges $(x, y)$, having both endpoints belong to Alice or to Bob, the players can readily simulate the message on their own without any interaction. This is because all information known to $x$ during the CONGEST algorithm will be known to the player who is simulating $x$. For the two non-internal edges $(\ell, \ell'), (r, r')$ the two players must exchange information in order to continue simulating the nodes. This can be done by exchanging four messages of length $O(\log n)$ at each round. At the end of the simulation of the algorithm, some node will know the diameter of $G$ and will therefore know whether $(A, B)$ are disjoint. At the cost of another bit, both players will know the answer. The total communication cost is $T(n) \cdot O(\log n)$. □

# 4 LOWER BOUNDS FOR CLOSEST PAIR OF SETS AND HIERARCHICAL CLUSTERING

In this section we prove a lower bound on the time it takes to simulate the output of the Average-Linkage algorithm, perhaps the most popular procedure for Hierarchical Clustering, in planar graphs, thus proving Theorem 8. We build on the diamond-like gadget from the simple lower bound for diameter. The constructions will combine many copies of these gadgets into one big graph that is also diamond-like.

## 4.1 Preliminaries for the Reductions

The starting point for the reductions in this section is the Orthogonal Vectors problem, which is known to be hard under SETH [76] and the Weighted Clique conjecture [3].

Definition 16 (Orthogonal Vectors). *Given a set of binary vectors, decide if there are two that are orthogonal, i.e. disjoint.*

We consider two variants of the closest pair problem.

Definition 17 (Closest Pair of Sets with Max-distance). *Given a graph $G = (V, E)$, a parameter $\Delta$, and disjoint subsets of the nodes $S_1, \ldots, S_m \subseteq V$, decide if there is a pair of sets $S_i, S_j$ such that*

$$\textit{Max-Dist}(S_i, S_j) = \max_{u \in S_i, v \in S_j} d(u, v) \leq \Delta.$$

In the second variant we look at the sum of all pairs within two sets, rather than just the max. This definition is used in the Average-Linkage heuristic and it is important for its success.

Definition 18 (Closest Pair of Sets with Sum-distance). *Given a graph $G = (V, E)$, a parameter $\Delta$, and disjoint subsets of the nodes $S_1, \ldots, S_m \subseteq V$, decide if there is a pair of sets $S_i, S_j$ such that*

$$\textit{Sum-Dist}(S_i, S_j) = \Sigma_{u \in S_i, v \in S_j} d(u, v) \leq \Delta.$$

We could also look at the Min-distance. However, it is easy to observe that the corresponding closest pair of sets problem is solvable in near-linear time. It is enough to sort all the edges and scan them once until a non-internal edge is found. Interestingly, there is also a popular heuristic for hierarchical clustering based on Min-distance, called Single Linkage, and it known that Single-Linkage can be computed in near-linear time in planar graphs.

## 4.2 Reduction with Max-Distance and Complete Linkage

We start with a simpler reduction which works only in the Max-distance case. The reduction to Sum-distance will be similar in structure but more details will be required.

Theorem 19. *Orthogonal Vectors on $n$ vectors in $d$ dimensions can be reduced to Closest Pair of Sets with Max-distance in a planar graph on $O(nd)$ nodes with edge weights in $[O(d)]$. The graph can be made unweighted by increasing the number of nodes to $O(nd^2)$.*

Proof. Let $v_1, \ldots, v_n \in \{0,1\}^d$ be an input instance for Orthogonal Vectors and we will show how to construct a planar graph $G$ and certain subsets of its nodes from it. For each vector $v_k, k \in [n]$ we have a set of $2d$ nodes $S_k = \{u_{k,1}, \ldots, u_{k,d}\} \cup \{u'_{k,1}, \ldots, u'_{k,d}\}$ in $G$. Each coordinate $v_k[j]$ is represented by two nodes $u_{k,j}$ and $u'_{k,j}$. In addition, there are two extra nodes in $G$ that we denote $\ell$ and $r$. Thus, $G$ contains the $2nd + 2$ nodes $S_1 \cup \cdots \cup S_n \cup \{\ell, r\}$. The edges of $G$ are defined in a diamond-like way as follows. Every node $u_{k,j}$ or $u'_{k,j}$ is connected with a left-edge to $\ell$ and with a right-edge to $r$. Thus, $G$ is planar.

The crux of the construction is defining the weights, and it will be done in the spirit of our gadget from the diameter lower bound. Set $M = 4$ as before, and for each $k \in [n]$ and $j \in [d]$ we define:

$$w(u_{k,j}, \ell) = \begin{cases} j \cdot M, & \text{if } v_k[j] = 0 \\ j \cdot M + 1, & \text{if } v_k[j] = 1 \end{cases}$$

$$w(u'_{k,j}, \ell) = \begin{cases} (2d + 1 - j) \cdot M, & \text{if } v_k[j] = 0 \\ (2d + 1 - j) \cdot M + 1, & \text{if } v_k[j] = 1 \end{cases}$$

$$w(u_{k,j}, r) = \begin{cases} (2d + 1 - j) \cdot M, & \text{if } v_k[j] = 0 \\ (2d + 1 - j) \cdot M + 1, & \text{if } v_k[j] = 1 \end{cases}$$

$$w(u'_{k,j}, r) = \begin{cases} j \cdot M, & \text{if } v_k[j] = 0 \\ j \cdot M + 1, & \text{if } v_k[j] = 1 \end{cases}$$

Note that all weights are positive integers up to $O(\log n)$.

Claim 20. *For any two sets $S_a, S_b$ we have that*

$$\textit{Max-Dist}(S_a, S_b) = \begin{cases} \leq (2d + 1) \cdot M + 1, & \textit{if } v_a, v_b \textit{ are orthogonal} \\ (2d + 1) \cdot M + 2, & \textit{otherwise.} \end{cases}$$

Proof. The proof is similar to Claim 15 since the subgraph of $G$ induced by two sets $S_a, S_b$ (and the shortest paths between them) is similar to our construction for the diameter lower bound (with $2d$ nodes instead of $n$). The details are deferred to the full version. □

Thus, solving the closest pair problem on $G$ with $\Delta = (2d + 1) \cdot M + 1$ gives us the solution to Orthogonal Vectors.

The reduction can be made to produce an unweighted graph by subdividing each edge of weight $w$ into a path of length $w$. The created nodes do not belong to any of the sets. The total number of nodes is $O(nd^2)$.

□

Next, we present an argument based on this reduction showing that the Complete-Linkage algorithm for hierarchical clustering cannot be sped up even if the data is embedded in a planar graph. We give a reduction only to the weighted case; the unweighted case remains open (and seems doable but challenging).

Theorem 21. *If for some $\varepsilon > 0$ the Complete-Linkage algorithm on $n$ node planar graphs with edge weights in $[O(\log n)]$ can be simulated in $O(n^{2-\varepsilon})$ time, then SETH is false.*

Proof. To refute SETH it is enough to solve OV on $n$ vectors of $d = O(\log n)$ dimensions in $O(n^{2-\varepsilon})$ time, for some $\varepsilon > 0$. Given such an instance of OV, we construct a planar graph $G$ such that the solution to the OV instance can be inferred from a simulation of the Complete-Linkage algorithm on $G$.

The graph $G$ is similar to the one produced in the reduction of Theorem 19 with a few additions described next. First, we add $M' = 11d$ to all the edge weights in $G$. This does not change any of the shortest paths, because for all pairs $s, t$ the shortest path has length exactly one if they are adjacent and exactly two otherwise. Then, we connect the nodes of each set $S_i$ with a path such that $u_{i,j}$ is connected to $u_{i,j+1}$ for all $j \in [d-1]$, $u_{i,d}$ is connected to $u'_{i,1}$, and $u'_{i,j}$ is connected to $u'_{i,j+1}$ for all $j \in [d-1]$. All these new edges have weight $M + 1 = 5$. As a result, all nodes within $S_i$ are at distance up to $5 \cdot 2d = 10d$ from each other, but the distance from any $u_{i,j}$ or $u'_{i,j}$ to $\ell$ or $r$ does not decrease (since the new edges are at least as costly as the difference between, e.g., $w(u_{i,j}, \ell)$ and $w(u_{i,j+1}, \ell)$).

Next, we analyze the clusters generated by an execution of the Complete-Linkage algorithm on $G$: we argue that at some point in the execution, each $S_i$ will be its own cluster (except that the nodes $\ell, r$ will be included in one of these clusters), and that the next pair to be merged is exactly the closest pair of sets (in max-distance). This is because the algorithm starts with each node in its own cluster, and at each stage, the pair of clusters of minimum Max-distance are merged into a new cluster. Let the *merge-value* of a stage be the distance of the merged cluster, and observe that this value does not decrease throughout the stages. The first few merges will involve pairs of adjacent nodes on the new paths we

added, in some order (that depends on the tie-breaking rule of the implementation, which we do not make any assumptions about), and the merge value will be 5. After all adjacent pairs are merged, two adjacent clusters will be merged, increasing the merge-value to 10. This continues until the merge value gets to $10d$, and at this point, each $S_i$ is its own cluster (since their inner distance is at most $10d$ and their distance to any other node is larger), plus the two clusters $\{\ell\}, \{r\}$. Next, the merge value becomes $M' + 2dM$ and each of the latter two clusters will get merged into one of the $S_i$'s (could be any of them). At this point, the max-distance between any pair of clusters is exactly the max-distance between the corresponding two sets $S_a, S_b$. This is because the nodes $\ell, r$ will not affect the max-distance. And so if we know the next pair to be merged, we will know the closest pair and can therefore deduce the solution to OV.

□

### 4.3 Reduction with Sum-Distance and Average Linkage

The issue with extending the previous reductions to the Sum-distance case is that pairs $i, j$ with $i \neq j$ will contribute to the score (even though their distance is designed to be smaller than that of the pairs with $i = j$). Indeed, if we look at Sum-Dist$(S_a, S_b)$ instead of Max-Dist$(S_a, S_b)$ for two vectors $a, b$ we will just get some fixed value that depends on $d$ plus $|a| + |b|$ (the hamming weight of the two vectors, i.e. the number of ones). Finding a pair of vectors with minimum number of ones is a trivial problem, since the objective function does not depend on any interaction between the pair. To overcome this issue, we utilize a degree of freedom in our diamond-like gadget that we have not used yet: so far, the left and right edges both have a $+v[i]$ term, but now we will gain extra hardness by choosing two distinct values for the two edges. The key property of the special pairs $i, j, i = j$ that we will utilize is not that their distance is larger, but that their left and right paths are equally long. Thus the shortest path can choose either path depending on the lower order terms of the weights, whereas for the non-special pairs the shortest path is constrained by the high order terms.

The starting point for the reduction will be the Closest Pair problem on binary vectors with hamming weight. Alman and Williams [11] gave a reduction from OV to the bichromatic version of this problem, and very recently a surprising result of C.S. and Manurangasi [49] showed that the monochromatic version (which is often easier to use in reductions, as we will do) is also hard.

Definition 22 (Hamming Closest Pair). *Given a set of binary vectors, output the minimum hamming distance between a pair of them.*

Theorem 23 ([49]). *Assuming OVH, for every $\varepsilon > 0$, there exists $s_\varepsilon > 0$ such that no algorithm running in time $O(n^{2-\varepsilon})$ can solve Hamming Closest Pair on $n$ binary vectors in $d = (\log n)^{s_\varepsilon}$ dimensions.*

Next we adapt the reduction from Theorem 19 to the sum-distance case.

Theorem 24. *Hamming Closest Pair on $n$ vectors in $d$ dimensions can be reduced to Closest Pair of Sets with Sum-distance in a planar graph on $O(nd)$ nodes with edge weights in $[O(d)]$. The graph can be made unweighted by increasing the number of nodes to $O(nd^2)$.*

Proof. The construction of the planar graph $G$ from the set of vectors will be similar, with one modification in the weights, to the one in Theorem 19 but the analysis will be quite different.

As before, for each vector $v_k, k \in [n]$ we have a set of $2d$ nodes $S_k = \{u_{k,1}, \ldots, u_{k,d}\} \cup \{u'_{k,1}, \ldots, u'_{k,d}\}$ in $G$, and we have two additional nodes $\ell, r$. Each node $u_{k,j}$ or $u'_{k,j}$ is connected to both $\ell$ and $r$.

Set $M = 4$ as before and for each $i \in [n], j \in [d]$ we define the edge weights of $G$ as follows. The difference to the previous reduction is that in the edges to $r$ we add the complement of $v_k[j]$ rather than $v_k[j]$ itself.

The proof of the following claim is deferred to the full version.

Claim 25. *For any two vectors $a, b$:*

$$\textit{Sum-Dist}(S_a, S_b) = f(d, M) + 2 \cdot \textit{Ham-Dist}(v_a, v_b)$$

*where $f(d, M) = O(Md^3)$ depends only on $d$ and $M$.*

Thus, the closest pair of sets $S_a, S_b$ in $G$ will correspond to the pair of vectors $a, b$ that minimize Ham-Dist$(v_a, v_b)$. This completes the reduction. As before, the graph can be made unweighted by subdividing the edges into paths. □

Finally, we present a lower bound argument for the Average-Linkage algorithm in planar graphs. As before, the unweighted case remains open.

Theorem 26. *If for some $\varepsilon > 0$ the Average-Linkage algorithm on $n$ node planar graphs with edge weights in $[O(\log n)]$ can be simulated in $O(n^{2-\varepsilon})$ time, then SETH is false.*

The proof is similar in structure to the proof of Theorem 21, and is deferred to the full version.

## 5 ALGORITHMS FOR SPARSEST CUT AND MINIMUM QUOTIENT CUT

In this section we present our algorithms.

### 5.1 Proof of Theorem 6: An $O(1)$-Approximation for Minimum Quotient Cut in Near-Linear Time

We will describe the algorithm in the dual graph, where cuts are cycles. Thus the input is a connected undirected planar graph $G$ with positive integral edge-costs $\text{cost}(e)$ and integral face-weights $\text{w}(f)$. Unless otherwise specified, $n$ denotes the size of $G$. We denote the sum of (finite) costs by $P$ and we denote the sum of weights by $W$. Given a cycle $C$, the total cost of the edges of $C$ is denoted $\text{cost}(C)$, and the total weight enclosed by $C$ is denoted $\text{w}(C)$, while the total weight outside $C$ is denoted by $\overline{\text{w}}(C)$. We denote by $\lambda(C)$ the ratio $\text{cost}(C)/\min\{\text{w}(C), \overline{\text{w}}(C)\}$. The goal is to find a cycle $C$ that minimizes $\lambda(C)$. We give a constant-factor approximation algorithm. For an overview of the algorithm, please see Section **??**.

*Overview of the Algorithm.* Let $C$ be the cycle $C$ that achieves the optimal cut. Our algorithm has two main parts, both of which combine previously known techniques with a novel idea. Roughly speaking, the goal of the first one is to find a node $s$ that is close to $C$, i.e. there is a path of small cost from $s$ to some node in $C$. Then, the second part will find an approximately optimal cycle $\hat{C}$ by starting from a reasonable candidate that can be computed in near-linear time and then iteratively fixing it using the node $s$. This idea of finding a nearby node (rather than insisting on a node that is on the optimal cycle, which incurs an extra $O(n)$ factor) and then using it to fix a candidate cycle is the crucial one that lets us improve on the quadratic-time 3.5-approximation Rao [71] both in terms of time *and* accuracy (as our fixing strategy turns out to be more effective).

In more detail, the first part will utilize a careful recursive decomposition of the graph with shortest-path cycle separators, in order to divide the graph into subgraphs such that: the total size of all subgraphs is $O(n \log n)$ and we are guaranteed that $C$ will be in one of them, and moreover, for each subgraph there are $O(1/\epsilon)$ candidate portals $s$ such that one of them is guaranteed to be close to $C$ (if it is there). In the second part, we build on the construction of Park and Phillips [66] that uses a spanning tree to define a directed graph with cleverly chosen edge weights so that the sum of weights of any fundamental cycle of the tree (if all edges have the same direction) is exactly the total weight of faces enclosed by the cycle. Then, using ideas from Rao's algorithm [71] we can modify the weights further so that any negative cycle $\hat{C}$ in the new graph (which can be found in near-linear time) is *almost* what we are looking for. The quotient of $\hat{C}$ is defined to be the cost of $C$ divided by the minimum between the weight inside and outside $\hat{C}$, while the construction so far is only looking at the weight *inside*. Therefore, our candidate $\hat{C}$ could have too much weight inside it, which makes it far from optimal. The final step, which is also the most novel, will use a portal $s$ to perform *weight reduction* steps on $\hat{C}$ without adding much to the cost; in fact, the increase in cost will depend on the distance from $s$ to the cycle.

*5.1.1 Outermost loop.* The outermost loop of the algorithm is a binary search for the (approximately) smallest value $\lambda$ such that there is a cycle $C$ for which $\lambda(C) \leq \lambda$. The body of this loop is a procedure $\textsc{Find}(\lambda)$ that for a given value of $\lambda$ either (1) finds a cycle $C$ such that $\lambda(C) \leq 3.29\lambda$ or (2) determines that there is no cycle $C$ such that $\lambda(C) \leq \lambda$. The binary search seeks to determine the smallest $\lambda$ (to within a 1.003 factor, or any $1+\varepsilon$) for which $\textsc{Find}(\lambda)$ returns a cycle. Because the optimal value (if finite) is between $1/W$ and $P$, the number of iterations of binary search is $O(\log WP)$.

*5.1.2 Cost loop.* The loop of the $\textsc{Find}(\lambda)$ procedure is a search for the (approximately) smallest number $\tau$ such that there is a cycle $C$ of cost at most $2\tau$ with $\lambda(C)$ not much more than $\lambda$. The body of this loop is a procedure $\textsc{Find}(\lambda, \tau)$ that either (1) finds a cycle $C$ such that $\lambda(C) \leq (1+\epsilon)3.29\lambda$ (in which case we say the procedure *succeeds*) or (2) determines that there is no cycle $C$ such that $\lambda(C) \leq \lambda$ and $\text{cost}(C) \leq 2\tau$. The outer loop tries $\tau = 1, \tau = 1+\epsilon, \tau = (1+\epsilon)^2$ and so on, until $\textsc{Find}(\lambda, \tau)$ succeeds. The number of iterations is $O(\log P)$ where $\epsilon$ is a constant to be determined. In proving the correctness of $\textsc{Find}(\lambda, \tau)$, we can assume that calls corresponding to smaller values of $\tau$ have failed.

*5.1.3 Recursive decomposition using shortest-path separators.* The procedure $\textsc{Find}(\lambda, \tau)$ first finds a shortest-path tree (with respect to edge-costs) rooted at an arbitrary vertex $r$. The procedure then finds a recursive decomposition of $G$ using balanced cycle separators with respect to that tree. Each separator is a non-self-crossing (but not necessarily simple) cycle $S = P_1P_2P_3$, where $P_1$ and $P_2$ are shortest paths in the shortest-path tree, and every edge $e$ not enclosed by $S$ but adjacent to $S$ is adjacent to $P_1$ or $P_2$. This property ensures that any cycle that is partially but not fully enclosed by $S$ intersects $P_1$ or $P_2$.

The recursive decomposition is a binary tree. Each node of the tree corresponds to a subgraph $H$ of $G$, and each internal node is labeled with a cycle separator $S$ of that subgraph. The children of a node corresponding to $H$ and labeled $S$ correspond to the subgraph $H_1$ consisting of the interior of $S$ and the subgraph $H_2$ consisting of the exterior. (Each subgraph includes the cycle $S$ itself.) In $H_1$ and $H_2$, the cycle $S$ is the boundary of a new face, which is called a *scar*. The scar is assigned a weight equal to the sum of the weights of the faces it replaced. Each leaf of the binary tree corresponds to a subgraph with at most a constant number of faces. We refer to the subgraphs corresponding to nodes as *clusters*.

One modification: for the purpose of efficiency, each vertex $v$ on the cycle $S$ that has degree exactly two after scar formation is *spliced out*: the two edges $e_1, e_2$ incident to $v$ are replaced with a single edge whose cost is the sum of the costs of $e_1$ and $e_2$. Clearly there is a correspondence between cycles before splicing out and cycles after splicing out, and costs are preserved. For the sake of simplicity of presentation, we identify each post-splicing-out cycle with the corresponding pre-splicing-out cycle.

Consecutive iterations of separator-finding alternate balancing number of faces with balancing number of scars. As a consequence, the depth of recursion is bounded by $O(\log n)$ and each cluster has at most six scars. (This is a standard technique.) Because of the splicing out, the sum of the sizes of graphs at each level of recursion is $O(n)$. Therefore the sum of sizes of all clusters is $O(n \log n)$.

Let $H$ be a cluster. Because $H$ has at most six scars, there are at most twelve paths in the shortest-path tree such that any cycle in the original graph that is only partially in the cluster must intersect at least one of these paths (these are the two paths $P_1, P_2$ from above). We call this the *intersection property*, and we refer to these paths as the *intersection* paths.

Because each scar is assigned weight equal to the sum of the weights of the faces it replaced, for any cluster $H$ and any simple cycle $C$ within $H$, the cost-to-weight ratio for $C$ in $H$ is the same as the ratio for $C$ in the original graph $G$.

*5.1.4 Decompositions into annuli.* The procedure also finds $1/\epsilon$ decompositions into *annuli*, based on the distance from $r$. The *annulus* $A[a, b)$ consists of every vertex whose distance from $r$ lies in the interval $[a, b)$. The *width* of the annulus is $b - a$. Let $\delta = \epsilon\tau$ and let $\sigma = (1+2\epsilon)\tau$. For each integer $i$ in the interval $[0, 1/\epsilon]$, the decomposition $D_i$ consists of the annuli $A[i\delta, i\delta+\sigma), A[i\delta+\sigma, i\delta+2\sigma), A[i\delta+2\sigma, i\delta+3\sigma)$ and so on. Thus the decomposition $D_i$ consists of disjoint annuli of width $\sigma$.

*5.1.5 Using the decompositions.* The procedure Find$(\lambda, \tau)$ is as follows:

- Search for a solution in each leaf cluster
- For each integer $i \in [0, 1/\epsilon]$
  - For each annulus $A[a, b)$ in $D_i$
    - For each non-root cluster $Q$
      - For each $P$ that is the intersection of the annulus with one of the twelve intersection paths of $Q$
        - Form an $\epsilon\tau$-net $S$ of $P$ (take nodes that are $\epsilon\tau$ apart)
        - For each vertex $s$ of $S$
          - Call subprocedure RootedFind$(\lambda, \tau, s, R)$ where $R$ = intersection of $A[a, b)$ with the parent of cluster $Q$

Here RootedFind$(\lambda, \tau, s, R)$ is a procedure such that if there is a cycle $C$ in $R$ with the properties listed below then the procedure finds a cycle $C$ such that $\lambda(C) \leq 3.29\lambda$ (in which case we say that the call *succeeds*).

The properties are:

(1) $\lambda(C) \leq \lambda$, and
(2) $(1+\epsilon)^{-1}\tau < \text{cost}(C) \leq 2\tau$, and
(3) $C$ contains a vertex $v$ such that the minimum cost of a $v$-to-$s$ path is at most $\epsilon\tau$.

In the last step of Find, the procedure takes the intersection of an annulus with a cluster. Let us elaborate on how this is done. Taking the intersection with an annulus involves deleting vertices outside the annulus. Deleting a vertex involves deleting its incident edges, which leads to faces merging; when two faces merge, the weight of the resulting face is defined to be the sum of weights of the two faces. This ensures that the cost-to-weight ratio of a cycle is the same in the subgraph as it is in the original graph.

We show that Find$(\lambda, \tau)$ is correct as follows. If the search for a solution in a leaf cluster succeeds or one of the calls to RootedFind succeeds, it follows from the construction that the cycle found meets Find's criterion for success. Conversely, suppose that there is a cycle $C$ in $G$ such that $\lambda(C) \leq \lambda$ and $(1+\epsilon)^{-1}\tau < \text{cost}(C) \leq 2\tau$. Our goal is to show that Find$(\lambda, \tau)$ succeeds. Let $Q_0$ be the smallest cluster that contains $C$. If $Q_0$ is a leaf cluster then the first line ensures that Find$(\lambda, \tau)$ succeeds. Otherwise, $Q_0$ has a child cluster $Q$ such that $C$ is only partially in $Q$. Therefore by the intersection property $C$ intersects one of the intersection paths $P$ of $Q$. Let $v$ be a vertex at which $C$ intersects $P$. Let $s$ be the point in the $\epsilon\tau$-net of $P$ closest to $v$.

Let $\alpha$ be the minimum distance from $r$ of a vertex of $C$, and let $\beta$ be the maximum distance. Because $\text{cost}(C) \leq 2\tau$, we have $\beta - \alpha \leq \tau$. Let $\alpha' = \min\{\alpha, \text{distance of } s \text{ from } r\}$ and let $\beta' = \max\{\beta, \text{distance of } s \text{ from } r\}$. Then $\beta' - \alpha' \leq \tau + \epsilon\tau$, so there exists an integer $i \in [0, 1/\epsilon]$ and an integer $j \geq 0$ such that the interval $[i\delta + j\sigma, i\delta + (j+1)\sigma)$ contains both $\alpha'$ and $\beta$, and therefore the annulus $A[i\delta + j\sigma, i\delta + (j+1)\sigma)$ contains $C$ together with the $v$-to-$s$ subpath of $P$. The specification of RootedFind$(\lambda, \tau, s, R)$ therefore shows that the procedure succeeds.

Now we consider the run-time analysis. The sum of sizes of all leaf clusters is $O(n)$. Because each leaf cluster has at most a constant number of faces, therefore, solutions can be sought in each of the leaf clusters in a total of $O(n)$ time.

For each integer $i \in [0, 1/\epsilon]$, the annuli of decomposition $D_i$ are disjoint. Because the sum of sizes of clusters is $O(n \log n)$, the sum of sizes of intersections of clusters with annuli of $D_i$ is $O(n \log n)$. Moreover, note that the total size of the $\epsilon\tau$-nets we pick within any annulus of width $O(\tau)$ is $O(1/\epsilon)$. Therefore $O(\epsilon^{-1} n \log n)$ is a bound on the sum of sizes of all intersections $R$ on which RootedFind is called. Therefore in order to obtain a near-linear time bound for Find, it suffices to prove a near-linear time bound for RootedFind.

*5.1.6* RootedFind. It remains to describe and analyze the procedure RootedFind$(\lambda, \tau, s, R)$. We use a construction of Park and Phillips [66] together with approximation techniques of Rao [71].

Let $T$ be a shortest-path tree of $R$, rooted at $s$. Delete from the graph every vertex whose distance from $s$ exceeds $(1+\epsilon)\tau$, and all incident edges, merging faces as before. This includes deleting vertices that cannot be reached from $s$ in $R$. Let $\hat{R}$ denote the resulting graph. Note that a cycle $C$ in $R$ that satisfies Properties 2 and 3 (see Section 5.1.5) must also be in $\hat{R}$.

According to a basic fact about planar embeddings (see e.g. [50]), in the planar dual $\hat{R}^*$ of $\hat{R}$, the set of edges not in $T$ form a spanning tree $T^*$. Each vertex of $T^*$ corresponds to a face in $\hat{R}$ and therefore has an associated weight. The procedure arbitrarily roots $T^*$, and finds the leafmost vertex $f_\infty$ such that the combined weight of all the descendants of $f_\infty$ is greater than $W/2$. The procedure then designates $f_\infty$ as the infinite face of the embedding of $\hat{R}$.

Lemma 27. *For any nontree edge $e$, the fundamental cycle of $e$ with respect to $T$ encloses (with respect to $f_\infty$) at most weight $W/2$.*

Park and Phillips describe a construction, which applies to any spanning tree of a planar graph with edge-costs and face-weights, and this construction is used in RootedFind. Each undirected edge of $\hat{R}$ corresponds to two *darts*, one in each direction. Each dart is assigned the cost of the corresponding edge. A dart corresponding to an edge of $T$ is assigned zero weight. Let $d$ be a nontree dart. Define $w_d$ to be the weight enclosed (with respect to $f_\infty$) by the fundamental cycle of $d$ with respect to $T$. Define the weight of $d$, denoted $\text{w}(d)$, to be $w_d$ if the orientation of $d$ in the the fundamental cycle is counterclockwise, and $-w_d$ otherwise. We refer to this graph as the *weight-transfer graph.*

Lemma 28 (Park and Phillips). *The sum of weights of darts of a counterclockwise cycle $C$ is the amount of weight enclosed by the cycle.*

We adapt an approximation technique of Rao [71]. (His method differs slightly.) The procedure selects a collection of candidate cycles; if any candidate cycle has quotient at most $3.29\lambda$, the procedure is considered to have succeeded. We will show that if $\hat{R}$ contains a cycle with properties 1-3 (see Section 5.1.5) then one of the candidate cycles has quotient at most $3.29\lambda$.

Let $\alpha, \beta$ be two parameters in $[0, 1]$ to be determined. Recall that $W$ is the sum of weights. We say a dart $d$ is *heavy* if $\text{w}(d) \geq \beta W$. For each heavy dart, the procedure considers as a candidate the fundamental cycle of $d$.

We next describe the search for a cycle in the weight-transfer graph minus heavy darts. Following a basic technique (see [54, 62]), we define a modified cost per dart as $\widehat{\text{cost}}(d) = \text{cost}(d) - \lambda\text{w}(d)$. A cycle has negative cost (under this cost assignment) if and only

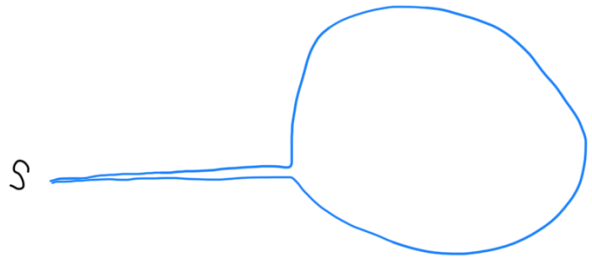


**Figure 3: This diagram illustrates the structure of a cycle arising in the algorithm. The cycle is nearly simple but includes a path and its reverse, where one endpoint of the path is the root $s$.**

if its ratio of cost to enclosed weight is less than $\lambda$. Note that the actual quotient of such a cycle may be much larger than $\lambda$ since we must divide by the *min* of the weight inside and the weight outside the cycle. Still, the information we get from such a cycle will be sufficient for getting a cycle that has quotient not much larger than $\lambda$.

The procedure seeks a negative-cost cycle in this graph. Using the algorithm of Klein, Mozes, and Weimann [51], this can be done in $O(n \log^2 n)$ time on a planar graph of size $n$.

Suppose the algorithm does find a negative-cost cycle $\hat{C}$. If $\hat{C}$ encloses at most $\alpha W$ weight then $\hat{C}$ is a candidate cycle. (In this case, the denominator in the actual quotient of $\hat{C}$ is not much smaller.)

Otherwise, the procedure proceeds as follows. (Here, the denominator is much smaller, so we would like to fix $\hat{C}$ so that it encloses much less weight, but the cost does not increase by much.) It first modifies the cycle to obtain a cycle $C_0$ that encloses the same amount of weight and that includes the vertex $s$. This step consists in adding to $\hat{C}$ the shortest path from $s$ to $\hat{C}$ and the reverse of this shortest path. Because the shortest path is in $\hat{R}$, this increases the cost of the cycle by at most $2(1+\epsilon)\tau$. (The new cycle will be easier to fix. This is the main idea that allows us to improve the previous approximation factor of 3.5 to below 3.3.)

The new cycle $C_0$ might not be a simple cycle: it has the form illustrated in Figure 3: it is mostly a simple cycle but contains a path and its reverse, such that $s$ is an endpoint of the path. We refer to such a cycle as a *near-simple cycle*.

Next the algorithm iteratively modifies the cycle so as to reduce the weight enclosed without increasing the cost. In each iteration, the algorithm considers the current cycle $C_i$ as a path starting and ending at $s$, and identifies the last dart $xy$ with $\mathrm{w}(xy) > 0$ in this path. The algorithm then finds the closest ancestor $u$ of $x$ in $T$ among vertices occurring after $y$ in the current path. The algorithm replaces the $x$-to-$u$ subpath of the current path with the $x$-to-$u$ path in $T$. Because the $x$-to-$u$ path in $T$ is a shortest path, this does not increase the cost of the current path. It reduces the enclosed weight by at most the weight of $xy$. This process repeats until the enclosed weight is at most $\alpha W$.

Here we restate the process, which we call *weight reduction*:

while $C_i$ encloses weight more than $\alpha W$
  write $C_i = sP_1xyP_2s$
  where $xy$ is a positive-weight dart and $P_2$ contains no such dart
  let $u$ be the closest ancestor of $x$ in $T$ among vertices in $P_2$
  let $C_{i+1} = sP_1P_3s$ where $P_3$ is the $x$-to-$u$ path in $T$
  let $i = i + 1$

Lemma 29. *The result of each iteration is a near-simple cycle. The enclosed weight is reduced by less than $\beta W$.*

*5.1.7 Analysis.* We will show that if $\hat{R}$ contains a cycle $C$ with properties 1-3 (see Section 5.1.5) then one of the candidate cycles considered by the procedure has quotient at most $3.29\lambda$. The details are deferred to the full version.

## 5.2 Proof of Theorem 4: An Exact Algorithm for Sparsest Cut with Running Time $O(n^{3/2}W)$

In this section, we provide an exact algorithm for Sparsest Cut and the Minimum Quotient problems running in time $O(n^{3/2}W\log(C))$. This improves upon the algorithm of Park and Phillips [66] running in time $O(n^2W\log(C))$. We first need to recall their approach (see [66] for all details).

The approach of Park and Phillips [66] works as follows. It works in the dual of the input graph and thus looks for a cycle $C$ minimizing $\ell(C)/\min(w(\text{Inside}(C)), w(\text{Outside}(C)))$, where $\ell(C)$ is the sum of length of the dual of the edges of $C$ and $w(\text{Inside}(C))$ (resp. $w(\text{Outside}(C))$) is the total weight of the vertices of $G$ whose corresponding faces in $G^*$ are in Inside($C$) (resp. Outside($C$)). Park and Phillips show that the approach also works for the sparsest cut problem. Their algorithm is as follows:

> Step 1. Construct an arbitrary spanning tree $T$ and order the vertices with a preorder traversal of $T$ which is consistent with the cyclic ordering of edges around each vertex.
> Step 2. For each edge $(u,v)$ of $G^*$, create two directed edges $e_1 = \langle u,v\rangle$ and $e_2 = \langle v,u\rangle$ and assume $u$ is before $v$ in the ordering computed at Step 1. Define the length of $e_1$ and $e_2$ to be the length of the dual edge of $e$, define the weight of $e_1$ to be the total weight of vertices enclosed by the fundamental cycle induced by $e$ (for the edges of $T$ the weight is 0) and the weight of $e_2$ to be minus the weight of $e_1$.
> Step 3. Construct a graph $\mathcal{G}$ as follows: for each vertex $v$ of $G^*$, for each weight $y \in [W]$, create a vertex $(v,y)$. For each directed edge $\langle u,v\rangle \in G^*$, for each $y \in [W]$, create an edge between vertices $(u,y)$ and $(v, y + w(\langle u,v\rangle))$ where $w(\langle u,v\rangle)$ is the weight of the edge $\langle u,v\rangle$ as defined at Step 2. The length of the edge created is equal to the length of $\langle u,v\rangle$.

Let $\mathcal{P}$ be the set of all shortest paths $P_{u,y}$ from $(u,0)$ to $(u,y)$ in $\mathcal{G}$ for $y \in [W/2]$, for each vertex $u \in G^*$. Let $P^*$ be a shortest path of $\mathcal{P}$ that achieves $\min_{u\in G^*, y\in[W/2]} \ell(P_{u,y})/y$. Park and Phillips show that $P^*$ corresponds to a minimum quotient cut of $G$. The running time $O(n^2W\log C)$ of the algorithm follows from applying a single source shortest path (SSSP) algorithm for each vertex $(u,0)$ of $\mathcal{G}$. Since $\mathcal{G}$ has $O(nW)$ vertices, these $n$ SSSP computations can be done in time $O(n^2W\log C)$.

*The Improvement.* We now show how to speed up the above algorithm. Consider taking an $O(\sqrt{n})$-size balanced separator $S$ of

$G^*$. We make the following observation: either $P^*$ intersects $S$, in which case it is only needed to perform a single source shortest path computation from each vertex $(u, 0)$ of $\mathcal{G}$, where $u \in S$, or $P^*$ does not intersect $S$ and in which case we can simply focus on computing the minimum quotient cut on each side of the separator separately (treating the other side as a single face of weight equal to the sum of the weights of the faces in the side).

More precisely, our algorithm is as follows:

Step 1. Compute an $O(\sqrt{n})$-size balance separator $S$ of $G^*$, that separates $G^*$ into two components $S_1, S_2$ both having size $|S_1|, |S_2| \leq 2n/3$;

Step 2. Compute $\mathcal{G}$ and perform an SSSP computation from each vertex $(u, 0) \in \mathcal{G}$ where $u \in S$ and let $P_0^*$ be the shortest path $P_{u,y}$ from $(u, 0)$ to $(u, y)$ for $u \in S$, $y \in [W/2]$ that minimizes $\ell(P_{u,y})/y$.

Step 3. For $i \in 1, 2$, create the graph $G_i$ with vertex set $S_i$ and where the face containing $S_{3-i}$ has weight equal to the sum of the weights of the faces in $S_{3-i}$.

Step 4. Returns the minimum quotient cut among $P_0^*, P_1^*, P_2^*$.

The correctness follows from our observation: if $P^*$ intersects $S$ then, by [66], Step 2 ensures that $P_0^*$ corresponds to a minimum quotient cut, otherwise $P^*$ is strictly contained within $S_1$ or $S_2$ and in which case the following argument applies. Assuming $P^*$ lies completely within $S_1$, then the graph $G_1$ where the face containing $S_2$ has weight equal to the sum of the weight of faces in $S_2$ contains a minimum quotient cut of value at most the minimum quotient cut of $G$ since the cut places all the vertices of $S_2$ on one side. Hence, an immediate induction shows that the minimum quotient cut among the cuts induced by the paths $P_0^*, P_1^*, P_2^*$ is optimal. The running time follows from a direct application of the master theorem.

## ACKNOWLEDGMENTS

P. Klein is supported by NSF Grant CCF-1841954, and V. Cohen-Addad supported by ANR-18-CE40-0004-01.

## REFERENCES

[1] Amir Abboud, Arturs Backurs, Karl Bringmann, and Marvin Künnemann. 2017. Fine-Grained Complexity of Analyzing Compressed Data: Quantifying Improvements over Decompress-and-Solve. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*. 192–203.

[2] Amir Abboud, Arturs Backurs, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Or Zamir. 2016. Subtree Isomorphism Revisited. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. 1256–1271.

[3] Amir Abboud, Karl Bringmann, Holger Dell, and Jesper Nederlof. 2018. More consequences of falsifying SETH and the orthogonal vectors conjecture. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*. 253–266.

[4] Amir Abboud, Keren Censor-Hillel, and Seri Khoury. 2016. Near-Linear Lower Bounds for Distributed Distance Computations, Even in Sparse Networks. In *Distributed Computing - 30th International Symposium, DISC 2016, Paris, France, September 27-29, 2016. Proceedings*. 29–42.

[5] Amir Abboud, Vincent Cohen Addad, and Hussein Houdrouge. 2019. Subquadratic High-Dimensional Hierarchical Clustering. *NeurIPS* (2019).

[6] Amir Abboud and Søren Dahlgaard. 2016. Popular Conjectures as a Barrier for Dynamic Planar Graph Algorithms. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*. 477–486.

[7] Amir Abboud and Virginia Vassilevska Williams. 2014. Popular Conjectures Imply Strong Lower Bounds for Dynamic Problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*. 434–443.

[8] Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. 2016. Approximation and Fixed Parameter Subquadratic Algorithms for Radius and Diameter in Sparse Graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. 377–391.

[9] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. 2014. Consequences of Faster Alignment of Sequences. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*. 39–51.

[10] James Abello, Panos M Pardalos, and Mauricio GC Resende. 2013. *Handbook of massive data sets*. Vol. 4. Springer.

[11] Josh Alman and Ryan Williams. 2015. Probabilistic Polynomials and Hamming Nearest Neighbors. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*. 136–150.

[12] Sanjeev Arora, Satish Rao, and Umesh Vazirani. 2009. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)* 56, 2 (2009), 5.

[13] Arturs Backurs, Nishanth Dikkala, and Christos Tzamos. 2016. Tight Hardness Results for Maximum Weight Rectangles. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*. 81:1–81:13.

[14] Arturs Backurs and Piotr Indyk. 2015. Edit Distance Cannot Be Computed in Strongly Subquadratic Time (unless SETH is false). In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*. 51–58.

[15] Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. 2017. Better approximations for tree sparsity in nearly-linear time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2215–2229.

[16] Gill Barequet and Sariel Har-Peled. 2001. Polygon Containment and Translational Min-Hausdorff-Distance Between Segment Sets are 3SUM-Hard. *Int. J. Comput. Geometry Appl.* 11, 4 (2001), 465–474. https://doi.org/10.1142/S0218195901000596

[17] MohammadHossein Bateni, MohammadTaghi Hajiaghayi, Saeed Seddighin, and Cliff Stein. 2018. Fast algorithms for knapsack via convolution and prediction. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 1269–1282.

[18] Sandeep N Bhatt and Frank Thomson Leighton. 1984. A framework for solving VLSI graph layout problems. *J. Comput. System Sci.* 28, 2 (1984), 300–343.

[19] Karl Bringmann. 2014. Why Walking the Dog Takes Time: Frechet Distance Has No Strongly Subquadratic Algorithms Unless SETH Fails. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*. 661–670.

[20] Karl Bringmann, Pawel Gawrychowski, Shay Mozes, and Oren Weimann. 2018. Tree Edit Distance Cannot be Computed in Strongly Subcubic Time (unless APSP can). In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*. 1190–1206.

[21] Karl Bringmann and Sebastian Krinninger. 2018. A note on hardness of diameter approximation. *Inf. Process. Lett.* 133 (2018), 10–15. https://doi.org/10.1016/j.ipl.2017.12.010

[22] Sergio Cabello. 2017. Subquadratic Algorithms for the Diameter and the Sum of Pairwise Distances in Planar Graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*. 2143–2152.

[23] Gunnar Carlsson and Facundo Mémoli. 2010. Characterization, stability and convergence of hierarchical clustering methods. *Journal of machine learning research* 11, Apr (2010), 1425–1470.

[24] Keren Censor-Hillel, Seri Khoury, and Ami Paz. 2017. Quadratic and Near-Quadratic Lower Bounds for the CONGEST Model. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*. 10:1–10:16.

[25] Panagiotis Charalampopoulos, Pawe? Gawrychowski, Shay Mozes, and Oren Weimann. 2019. Almost Optimal Distance Oracles for Planar Graphs. In *STOC, to appear*.

[26] Michael Cochez and Hao Mou. 2015. Twister tries: Approximate hierarchical agglomerative clustering for average distance in linear time. In *Proceedings of the 2015 ACM SIGMOD international conference on Management of data*. ACM, 505–517.

[27] Vincent Cohen-Addad, Søren Dahlgaard, and Christian Wulff-Nilsen. 2017. Fast and Compact Exact Distance Oracle for Planar Graphs. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*. 962–973.

[28] Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michał Włodarczyk. 2019. On problems equivalent to (min,+)-convolution. *ACM Transactions on Algorithms (TALG)* 15, 1 (2019), 14.

[29] Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. 2012. Distributed Verification and Hardness of Distributed Approximation. *SIAM J. Comput.* 41, 5 (2012). https://doi.org/10.1137/11085178X

[30] Sanjoy Dasgupta. 2016. A Cost Function for Similarity-based Hierarchical Clustering. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory*

*of Computing* (Cambridge, MA, USA) *(STOC 2016)*. ACM, New York, NY, USA, 118–127. https://doi.org/10.1145/2897518.2897527

[31] Laxman Dhulipala, Igor Kabiljo, Brian Karrer, Giuseppe Ottaviano, Sergey Pupyrev, and Alon Shalita. 2016. Compressing graphs and indexes with recursive graph bisection. *arXiv preprint arXiv:1602.08820* (2016).

[32] Michael Elkin. 2006. An Unconditional Lower Bound on the Time-Approximation Trade-off for the Distributed Minimum Spanning Tree Problem. *SIAM J. Comput.* 36, 2 (2006). https://doi.org/10.1137/S0097539704441058

[33] Kyle Fox, Philip N. Klein, and Shay Mozes. 2015. A Polynomial-time Bicriteria Approximation Scheme for Planar Bisection. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*. 841–850. https://doi.org/10.1145/2746539.2746564

[34] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York, NY, USA:.

[35] Silvio Frischknecht, Stephan Holzer, and Roger Wattenhofer. [n.d.]. Networks cannot compute their diameter in sublinear time. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, 2012*. http://portal.acm.org/citation.cfm?id=2095207&CFID=63838676&CFTOKEN=79617016

[36] Anka Gajentaan and Mark H. Overmars. 2012. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.* 45, 4 (2012), 140–152. https://doi.org/10.1016/j.comgeo.2011.11.006

[37] Naveen Garg, Huzur Saran, and Vijay V. Vazirani. 1999. Finding Separator Cuts in Planar Graphs within Twice the Optimal. *SIAM J. Comput.* 29, 1 (1999), 159–179. https://doi.org/10.1137/S0097539794271692

[38] Pawel Gawrychowski, Haim Kaplan, Shay Mozes, Micha Sharir, and Oren Weimann. 2018. Voronoi Diagrams on Planar Graphs, and Computing the Diameter in Deterministic $\tilde{O}(n^{5/3})$ Time. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*. 495–514.

[39] Pawel Gawrychowski, Shay Mozes, Oren Weimann, and Christian Wulff-Nilsen. 2018. Better Tradeoffs for Exact Distance Oracles in Planar Graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*. 515–529.

[40] Mohsen Ghaffari and Bernhard Haeupler. [n.d.]. Distributed Algorithms for Planar Networks II: Low-Congestion Shortcuts, MST, and Min-Cut. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, 2016*. https://doi.org/10.1137/1.9781611974331.ch16

[41] Mohsen Ghaffari and Bernhard Haeupler. 2016. Distributed Algorithms for Planar Networks I: Planar Embedding. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*. 29–38.

[42] Mohsen Ghaffari and Merav Parter. 2017. Near-Optimal Distributed DFS in Planar Graphs. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*. 21:1–21:16.

[43] John C Gower and Gavin JS Ross. 1969. Minimum spanning trees and single linkage cluster analysis. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 18, 1 (1969), 54–64.

[44] Bernhard Haeupler, D. Ellis Hershkowitz, and David Wajc. 2018. Round- and Message-Optimal Distributed Graph Algorithms. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*. 119–128.

[45] Bernhard Haeupler, Taisuke Izumi, and Goran Zuzic. 2016. Low-Congestion Shortcuts without Embedding. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*. 451–460.

[46] Bernhard Haeupler, Taisuke Izumi, and Goran Zuzic. 2016. Near-Optimal Low-Congestion Shortcuts on Bounded Parameter Graphs. In *Distributed Computing - 30th International Symposium, DISC 2016, Paris, France, September 27-29, 2016. Proceedings*. 158–172.

[47] Bernhard Haeupler, Jason Li, and Goran Zuzic. 2018. Minor Excluded Network Families Admit Fast Distributed Algorithms. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*. 465–474.

[48] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. 2015. Unifying and Strengthening Hardness for Dynamic Problems via the Online Matrix-Vector Multiplication Conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*. 21–30.

[49] Karthik C. S. and Pasin Manurangsi. 2019. On Closest Pair in Euclidean Metric: Monochromatic is as Hard as Bichromatic. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*. 17:1–17:16.

[50] Philip N. Klein and Shay Mozes. [n.d.]. Optimization Algorithms for Planar Graphs. Draft chapters available at http://planarity.org.

[51] Philip N. Klein, Shay Mozes, and Oren Weimann. 2010. Shortest paths in directed planar graphs with negative lengths: A linear-space $O(n \log^2 n)$-time algorithm. *ACM Trans. Algorithms* 6, 2 (2010), 1–18. https://doi.org/10.1145/1721837.1721846

[52] Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. 2017. On the Fine-Grained Complexity of One-Dimensional Dynamic Programming. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*. 21:1–21:15. https://doi.org/10.4230/LIPIcs.ICALP.2017.21

[53] Eduardo S Laber, Wilfredo Bardales, and Ferdinando Cicalese. 2014. On lower bounds for the maximum consecutive subsums problem and the (min,+)-convolution. In *2014 IEEE International Symposium on Information Theory*. IEEE, 1807–1811.

[54] Eugene L Lawler. 2001. *Combinatorial optimization: networks and matroids*. Courier Corporation.

[55] Charles E Leiserson. 1980. Area-efficient graph layouts. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*. IEEE, 270–281.

[56] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. *Mining of massive datasets*. Cambridge university press.

[57] Jason Li. 2018. Distributed Treewidth Computation. *CoRR* abs/1805.10708 (2018). arXiv:1805.10708 http://arxiv.org/abs/1805.10708

[58] Jason Li and Merav Parter. 2019. Planar Diameter via Metric Compression. In *STOC, to appear*.

[59] Richard J Lipton and Robert Endre Tarjan. 1977. Applications of a planar separator theorem. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE, 162–170.

[60] Richard J Lipton and Robert Endre Tarjan. 1979. A separator theorem for planar graphs. *SIAM J. Appl. Math.* 36, 2 (1979), 177–189.

[61] William B March, Parikshit Ram, and Alexander G Gray. 2010. Fast euclidean minimum spanning tree: algorithm, analysis, and applications. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 603–612.

[62] Nimrod Megiddo. 1979. Combinatorial Optimization with Rational Objective Functions. *Math. Oper. Res.* 4, 4 (1979), 414–424. https://doi.org/10.1287/moor.4.4.414

[63] Fionn Murtagh. 1983. A survey of recent advances in hierarchical clustering algorithms. *Comput. J.* 26, 4 (1983), 354–359.

[64] Fionn Murtagh. 1992. Comments on 'Parallel Algorithms for Hierarchical Clustering and Cluster Validity'. *IEEE Trans. Pattern Anal. Mach. Intell.* 14, 10 (1992), 1056–1057. https://doi.org/10.1109/34.159908

[65] Danupon Nanongkai, Atish Das Sarma, and Gopal Pandurangan. [n.d.]. A tight unconditional lower bound on distributed randomwalk computation. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC, 2011*. https://doi.org/10.1145/1993806.1993853

[66] J. K. Park and C. A. Phillips. 1993. Finding minimum-quotient cuts in planar graphs. In *STOC*. 766–775.

[67] Viresh Patel. 2010. Determining edge expansion and other connectivity measures of graphs of bounded genus. In *European Symposium on Algorithms*. Springer, 561–572.

[68] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[69] David Peleg and Vitaly Rubinovich. [n.d.]. A Near-Tight Lower Bound on the Time Complexity of Distributed MST Construction. In *40th Annual Symposium on Foundations of Computer Science, FOCS, 1999*. https://doi.org/10.1109/SFFCS.1999.814597

[70] Satish Rao. 1987. Finding Near Optimal Separators in Planar Graphs. In *28th Annual Symposium on Foundations of Computer Science*. 225–237. https://doi.org/10.1109/SFCS.1987.26

[71] Satish Rao. 1992. Faster Algorithms for Finding Small Edge Cuts in Planar Graphs (Extended Abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*. 229–240. https://doi.org/10.1145/129712.129735

[72] Alexander A. Razborov. 1992. On the Distributional Complexity of Disjointness. *Theor. Comput. Sci.* 106, 2 (1992). https://doi.org/10.1016/0304-3975(92)90260-M

[73] Liam Roditty and Virginia Vassilevska Williams. 2013. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*. 515–524.

[74] Aaron Schild and Christian Sommer. 2015. On Balanced Separators in Road Networks. In *Experimental Algorithms - 14th International Symposium, SEA 2015, Paris, France, June 29 - July 1, 2015, Proceedings*. 286–297. https://doi.org/10.1007/978-3-319-20086-6_22

[75] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Vol. 39. Cambridge University Press.

[76] Ryan Williams. 2005. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.* 348, 2-3 (2005), 357–365. https://doi.org/10.1016/j.tcs.2005.09.023

[77] Virginia Vassilevska Williams. 2018. On some fine-grained questions in algorithms and complexity. In *Proceedings of the ICM*.

[78] Virginia Vassilevska Williams and R. Ryan Williams. 2018. Subcubic Equivalences Between Path, Matrix, and Triangle Problems. *J. ACM* 65, 5 (2018), 27:1–27:38. https://doi.org/10.1145/3186893