Bisection of Bounded Treewidth Graphs by Convolutions

Eduard Eiben
Department of Informatics, University of Bergen, Norway
eduard.eiben@uib.no

Daniel Lokshtanov
Department of Computer Science, US Santa Barbara, United States
daniello@ucsb.edu

Amer E. Mouawad
Department of Computer Science, American University of Beirut, Lebanon
aa368@aub.edu.lb

Abstract

In the Bisection problem, we are given as input an edge-weighted graph G. The task is to find a partition of $V(G)$ into two parts A and B such that $||A| - |B|| \leq 1$ and the sum of the weights of the edges with one endpoint in A and the other in B is minimized. We show that the complexity of the Bisection problem on trees, and more generally on graphs of bounded treewidth, is intimately linked to the $(\min, +)$-Convolution problem. Here the input consists of two sequences $(a[i])_{i=0}^{n-1}$ and $(b[i])_{i=0}^{n-1}$, the task is to compute the sequence $(c[i])_{i=0}^{n-1}$, where $c[k] = \min_{i=0,\ldots,k}(a[i] + b[k-i])$.

In particular, we prove that if $(\min, +)$-Convolution can be solved in $O(\tau(n))$ time, then Bisection of graphs of treewidth t can be solved in time $O(8^t t^{O(1)} \log n \cdot \tau(n))$, assuming a tree decomposition of width t is provided as input. Plugging in the naive $O(n^2)$ time algorithm for $(\min, +)$-Convolution yields a $O(8^t t^{O(1)} n^2 \log n)$ time algorithm for Bisection. This improves over the (dependence on n of the) $O(2^t n^3)$ time algorithm of Jansen et al. [SICOMP 2005] at the cost of a worse dependence on t. "Conversely", we show that if Bisection can be solved in time $O(\beta(n))$ on edge weighted trees, then $(\min, +)$-Convolution can be solved in $O(\beta(n))$ time as well. Thus, obtaining a sub-quadratic algorithm for Bisection on trees is extremely challenging, and could even be impossible. On the other hand, for *unweighted* graphs of treewidth t, by making use of a recent algorithm for Bounded Difference $(\min, +)$-Convolution of Chan and Lewenstein [STOC 2015], we obtain a sub-quadratic algorithm for Bisection with running time $O(8^t t^{O(1)} n^{1.864} \log n)$.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases bisection, convolution, treewidth, fine-grained analysis, hardness in P

Digital Object Identifier 10.4230/LIPIcs.ESA.2019.42

1 Introduction

A *bisection* of a graph G is a partition of $V(G)$ into two parts A and B such that $||A|-|B|| \leq 1$. The *weight* of a bisection (A, B) of an edge-weighted graph G is the sum of the weights of all edges with one endpoint in A and the other in B. In the Bisection problem the task is to find a minimum weight bisection in an edge-weighted graph G given as input. The problem can be seen as a version of Minimum Cut with a balance constraint on the sizes of two sides of the cut. While Minimum Cut is solvable in polynomial time, Bisection is one of the classic examples of NP-complete problems [15]. Bisection has been studied extensively from the perspective of approximation algorithms [14, 13, 18, 21], parameterized algorithms [7, 11, 22] heuristics [6, 8] and average case complexity [5].

In this paper we consider Bisection when the input graph is required to be a tree, or more generally a graph with treewidth at most t. For trees, an $O(n^3)$ time algorithm was given by MacGregor [20] already in 1978, this was improved to a parallel algorithm running in time $O(\log^2 n \log\log n)$ on $O(n^2)$ processors by Goldberg and Miller [16]. This corresponds to a sequential algorithm running in time $O(n^2 \log^2 n \log\log n)$. For graphs of bounded treewidth Jansen et al. [17] gave an algorithm that solves Bisection in time $O(2^t n^3)$ if a tree decomposition of width t is given as input.

The majority of natural graph problems are solvable in linear time on trees and bounded treewidth graphs (see e.g. Courcelle's theorem [10]). Thus, it is quite natural to ask whether the dependence on n in the algorithm of Jansen et al. [17] could be improved to linear. Our first result goes "half the way" from Jansen et al.'s cubic algorithm to a linear time one, and matches (in fact slightly improves) the fastest known algorithm for Bisection on trees[1].

▶ **Theorem 1.** *There is an algorithm that, given an edge-weighted graph G on n vertices together with a tree decomposition of G of width at most t, computes a minimum weight bisection of G in time $\mathcal{O}(8^t \cdot t^5 \cdot n^2 \cdot \log n)$.*

Our algorithm crucially uses the $(\min,+)$-*convolution* operation. The $(\min,+)$-convolution of two number sequences $(a[i])_{i=0}^{n-1}$ and $(b[i])_{i=0}^{n-1}$ is a sequence $(c[i])_{i=0}^{n-1}$, where $c[k] = \min_{i=0,\ldots,k}(a[i] + b[k-i])$. In the $(\min,+)$-Convolution problem the input consists of the two sequences $(a[i])_{i=0}^{n-1}$ and $(b[i])_{i=0}^{n-1}$, the task is to compute their convolution $(c[i])_{i=0}^{n-1}$. A direct application of the definition of $(\min,+)$-convolution yields a $\mathcal{O}(n^2)$ time algorithm to compute it. The bulk of the work of our algorithm consists of making a series of $(\min,+)$-convolution steps. In fact, the running time of our algorithm can be stated as $\mathcal{O}(8^t \cdot t \cdot \log n \cdot \tau(t^2 n))$, where $\tau(n)$ is the running time of an algorithm computing the $(\min,+)$-convolution of two sequences of length n. Therefore, there are two natural avenues for attempting to improve the algorithm of Theorem 1 to sub-quadratic. The first approach is to design a sub-quadratic algorithm for $(\min,+)$-convolution, the second is to design an entirely different algorithm avoiding convolution altogether.

It turns out that the first approach is quite challenging, perhaps even impossible. Indeed, in the spirit of *fine-grained complexity* [23] analysis, Cygan et al. [12] identified a number of problems that admit algorithms with running time $\mathcal{O}(n^{2-\epsilon})$ if and only if $(\min,+)$-Convolution does. With this background they conjecture that $(\min,+)$-Convolution does *not* admit a $\mathcal{O}(n^{2-\epsilon})$ time algorithm.

Thus, if we want to improve the algorithm of Theorem 1 to a sub-quadratic algorithm without disproving the conjecture of Cygan et al. [12], we need to avoid $(\min,+)$-convolution altogether. However, it turns out that $(\min,+)$-convolution is unavoidable! In particular, we prove that a sub-quadratic algorithm for Bisection on trees implies one for $(\min,+)$-Convolution as well.

▶ **Theorem 2.** *If there exists an $\epsilon > 0$ such that* Bisection *on weighted trees can be solved in time $\mathcal{O}(n^{2-\epsilon})$, then there exists $\delta > 0$ such that* $(\min,+)$-Convolution *can be solved in $\mathcal{O}(n^{2-\delta})$-time.*

Theorem 2 together with Theorem 1 (or rather its re-statement in terms of convolutions), puts Bisection on weighted trees in Cygan et al. [12]'s class of problems equivalent to $(\min,+)$-Convolution.

[1] Note that the Goldberg and Miller's algorithm [16] is parallel, while ours is sequential.

In light of Theorem 2, the Bisection problem on *unweighted* graphs[2] becomes a natural target. Our final contribution is a sub-quadratic algorithm for Bisection on unweighted graphs of bounded treewidth. Our algorithm also works for the case when all weights are bounded by a constant W.

▶ **Theorem 3.** *There is an algorithm that, given an edge-weighted graph G, where all edge weights are integers between 1 and W, together with a tree decomposition of G of width t, computes a minimum weight bisection of G in time $\mathcal{O}(8^t \cdot (tW)^{O(1)} \cdot n^{1.864} \log n)$.*

The key observation behind the algorithm of Theorem 3 is that the $(\min,+)$-convolution steps in the algorithm of Theorem 1 are applied to sequences $(a[i])_{i=0}^{n-1}$ and $(b[i])_{i=0}^{n-1}$ where $a[i]$ and $b[i]$ are both essentially equal to the minimum possible sum of weights of the edges between the two sides A and B of a partition (A,B) of $V(G)$ with $|A| = i$. Bounded treewidth graphs have many vertices of small degree, and moving one vertex of small degree from A to B or vice versa changes the number of edges between A and B by at most its degree. Thus, $a[i]$ and $a[i+1]$ cannot be too different. This allows us to use the faster algorithm for $(\min,+)$-Convolution of Chan and Lewenstein [9] for "bounded difference" sequences.

Organization of the paper. We start by setting up the needed notation in Section 2. Section 3 is devoted to proving our algorithmic results - namely Theorems 1 and 3. Theorem 2 is proved in Section 4.

2 Preliminaries

2.1 The $(\min,+)$-Convolution problem

For integer n, we let $[n] := \{0,1,\ldots,n\}$. Given a vector or a sequence $A \in \mathbb{Z}^n$ and an integer $i \in [n-1]$, we denote by A_i the i-th coordinate of A.

▶ **Definition 4** ($(\min,+)$-Convolution problem)**.** *Given two sequences $(a[i])_{i=0}^{n-1}$ and $(b[i])_{i=0}^{n-1}$, compute a third sequence $(c[i])_{i=0}^{n-1}$, where $c[k] = \min_{i=0,\ldots,k}(a[i]+b[k-i])$. Equivalently, we have $c[k] = \min_{i+j=k}(a[i]+b[j])$.*

In the $(\min,+)$-Convolution problem, we sometimes require the target sequence to be computed all the way up to $2n-2$, i.e., $(c[i])_{i=0}^{2n-2}$. In both cases, the problem is trivially solvabled in $\mathcal{O}(n^2)$ time. Recent breakthroughs have shown that computing the $(\min,+)$-Convolution for monotone non-decreasing sequences with integer values bounded by $\mathcal{O}(n)$ can be achieved in $\mathcal{O}(n^{1.864})$ deterministic time [9]. Moreover, we can relax these requirements [4] and simply require that the sequences have bounded differences, i.e., $|a[i]-a[i+1]|, |b[i]-b[i+1]| \in O(1)$.

2.2 Graphs and the Bisection problem

We assume that each graph G is finite, simple, and undirected. We let $V(G)$ and $E(G)$ denote the vertex set and edge set of G, respectively. The *open neighborhood* of a vertex v is denoted by $N_G(v) = \{u \mid \{u,v\} \in E(G)\}$ and the *closed neighborhood* by $N_G[v] = N_G(v) \cup \{v\}$. For a set of vertices $S \subseteq V(G)$, we define $N_G(S) = \{v \notin S \mid \{u,v\} \in E(G), u \in S\}$ and $N_G[S] = N_G(S) \cup S$. The subgraph of G induced by S is denoted by $G[S]$, where $G[S]$ has vertex set S and edge set $\{\{u,v\} \in E(G) \mid u,v \in S\}$. We let $G - S = G[V(G) \setminus S]$.

[2] where all weights are 1

Given a graph G and two disjoint sets $A, B \subseteq V(G)$, we denote by $E(A, B)$ the subset of edges of G with one endpoint in A and the other endpoint in B. Given an edge-weighted graph G and a weight function $\mathrm{w} : E(G) \to \mathbb{N}$ over the edges of G, a *bisection* of G is a partition of $V(G)$ into two disjoint sets $A, B \subseteq V(G)$ such that $||A| - |B|| \leq 1$ and the *weight of bisection* (A, B) is $\sum_{e \in E(A,B)} \mathrm{w}(e)$. Formally, the Bisection problem is defined as follows:

▶ **Definition 5** (Bisection problem)**.** *Given an edge-weighted graph G, find a bisection (A, B) of G of minimum weight.*

2.3 Treewidth and tree decompositions

▶ **Definition 6.** *A* tree decomposition *of a graph G is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, where $\{X_i \mid i \in I\}$ is a collection of subsets of $V(G)$, $T = (I, F)$ is a rooted tree such that the following conditions hold:*

- $\bigcup_{i \in I} X_i = V(G)$*;*
- *For all edges $\{u, v\} \in E(G)$, there exists $i \in I$ with $u, v \in X_i$;*
- *For every vertex $v \in V(G)$, the subgraph of T induced by $\{i \in I \mid v \in X_i\}$ is connected.*

The width *of a tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is $max_{i \in I}(|X_i| - 1)$. The* treewidth *of a graph G, $\mathrm{tw}(G)$, is the minimum width over all possible tree decompositions of the graph. We call the vertices of the tree T* nodes *and the sets X_i* bags*. A graph of treewidth $\mathcal{O}(1)$ is called a bounded treewidth graph.*

Given a tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of an n-vertex graph G of treewidth k, we can turn this decomposition in time in $\mathcal{O}(k^{\mathcal{O}(1)} \cdot n)$ into a *nice tree decomposition* with at most $\mathcal{O}(k|V(G)|)$ nodes, i.e., a decomposition of the same width and satisfying the following properties:

- The root bag as well as all leaf bags are empty;
- Every node of the tree decomposition is of one of four different types:
  - Leaf node: a node i with $X_i = \emptyset$ and no children;
  - Introduce node: a node i with exactly one child j such that $X_i = X_j \cup \{v\}$ for some vertex $v \in X_j$;
  - Forget node: a node i with exactly one child j such that $X_i = X_j \setminus \{v\}$ for some vertex $v \in X_j$;
  - Join node: a node i with two children j_1 and j_2 such that $X_i = X_{j_1} = X_{j_2}$.

▶ **Theorem 7** (Bodlaender et al. [2])**.** *There exists an algorithm, that given an n-vertex graph G and an integer k, in time $2^{\mathcal{O}(k)} n \log n$ either outputs that the treewidth of G is larger than k, or constructs a tree decomposition of G of width at most $3k + 4$.*

Combining Theorem 8 below with standard arguments (we refer the reader to [2] for more details), we arrive at Proposition 9, which is the form that will be required to obtain our algorithms.

▶ **Theorem 8** (Bodlaender and Hagerup [3])**.** *There is an algorithm that, given a tree decomposition of width k with $\mathcal{O}(n)$ nodes of a graph G, finds a rooted binary tree decomposition of G of width at most $3k + 2$ with depth $\mathcal{O}(\log n)$ in $\mathcal{O}(kn)$-time.*

▶ **Proposition 9.** *There is an algorithm that, given an n-vertex graph G and a tree decomposition of G of width k, runs in $\mathcal{O}(kn)$-time, and computes a nice tree decomposition of G of width $3k + 2$, height $\mathcal{O}(k \log n)$, and with $\mathcal{O}(kn)$ nodes.*

3 Algorithms for Bisection on Bounded Treewidth Graphs

We start by reviewing the $\mathcal{O}(2^{t+1} \cdot n^3)$-time algorithm for solving the Bisection problem on graphs of treewidth at most t by Jansen et al. [17]. The algorithm is a standard dynamic programming algorithm over a tree decomposition. Given a graph G together with its nice tree decomposition $(\{X_i | i \in I\}, T = (I, F))$ of width t the algorithm works as follows.

For each node $i \in I$, we let Y_i denote the set of all vertices in X_j, where either j is a descendant of i in T or $j = i$. The algorithm computes for each $i \in I$, an array mwp_i (which stands for minimum weight partition) containing $\mathcal{O}(2^t \cdot |Y_i|)$ entries. For each $\ell \in \{0, 1, \ldots, |Y_i|\}$ and each $S \subseteq X_i$, the entry $\text{mwp}_i(\ell, S)$ is set to $\min_{S' \subseteq Y_i, |S'| = \ell, S' \cap X_i = S} (\sum_{e \in E(S', Y_i \setminus S')} \text{w}(e))$. That is, $\text{mwp}_i(\ell, S)$ is equal to the minimum possible weight of a bisection where S and $X_i \setminus S$ are in different parts of the bisection and the side including S is of cardinality exactly ℓ. When such a partition is not possible, we set $\text{mwp}_i(\ell, S)$ to ∞.

We compute the entries of the array following the levels of the tree decomposition in a bottom-up manner as follows.

- Let i be a leaf in T. Note that $Y_i = X_i = \emptyset$. We set $\text{mwp}_i(0, \emptyset) = 0$.
- Let i be a forget node with one child j such that $X_i \subseteq X_j$. Then, for all $\ell \in \{0, 1, \ldots, |Y_i|\}$ and $S \subseteq X_i$, we set

$$\text{mwp}_i(\ell, S) = \min_{S' \subseteq X_j, S' \cap X_i = S} (\text{mwp}_j(\ell, S')).$$

- Let i be an introduce node with one child j such that $X_j \cup \{v\} = X_i$ and $v \notin X_j$. Then, for all $\ell \in \{0, 1, \ldots, |Y_i|\}$ and $S \subseteq X_i$, if $v \in S$ we set $\text{mwp}_i(\ell, S) = \text{mwp}_j(\ell - 1, S \setminus \{v\})$. Otherwise, we set

$$\text{mwp}_i(\ell, S) = \text{mwp}_j(\ell, S) + \sum_{e \in \{\{v,s\} | s \in S\}} \text{w}(e).$$

- Let i be a join node with two children j_1 and j_2, where $X_i = X_{j_1} = X_{j_2}$. For all $\ell \in \{0, 1, \ldots, |Y_i|\}$ and $S \subseteq X_i$, we set

$$\text{mwp}_i(\ell, S) = min_{\ell_1 + \ell_2 - |S| = \ell, \ell_1, \ell_2 \geq |S|} \left(\text{mwp}_{j_1}(\ell_1, S) + \text{mwp}_{j_2}(\ell_2, S) - \sum_{e \in E(S, X_i \setminus S)} \text{w}(e) \right).$$

We omit the proof of correctness and refer the reader to [17] for more details. We focus here on the runtime analysis. Analyzing the above algorithm on the tree decomposition of width t and height $\mathcal{O}(t \log n)$, we obtain the following lemma.

▶ **Lemma 10.** *There is an algorithm that, given an edge-weighted graph G on n vertices and a nice tree decomposition of width t, height $\mathcal{O}(t \log n)$, and $\mathcal{O}(tn)$ nodes, computes a minimum weight bisection of G in time $\mathcal{O}(2^{t+1} \cdot t \cdot \log n \cdot \tau(t^2 n))$, where $\tau(|Y_i|)$ is the time required to compute the entries $\text{mwp}_i(\ell, S)$ for all $\ell \in [|Y_i|]$ and a fixed S in a join node.*

Proof. Let $(\{X_i | i \in I\}, T = (I, F))$ be the nice tree decomposition of G given as input. The time spent at each leaf node, introduce node, or forget node i is bounded by $\mathcal{O}(2^{t+1} \cdot |Y_i|)$. Moreover, by our assumption the time spend in each join node is $\mathcal{O}(2^{t+1} \tau(|Y_i|))$.

Now let us split the nodes of T into $r = \mathcal{O}(t \log n)$ levels $L_0, \ldots, L_r$ depending on the distance of the node from the root of T. We analyze the running time on each level separately. Clearly, the running time at level k is at most $\mathcal{O}(\sum_{i \in L_k} 2^{t+1} \tau(|Y_i|))$. Moreover, given $i, j \in L_k$ the nodes i and j cannot be descendants of each other. Therefore, from the

definition of a tree decomposition and Y_i and Y_j respectively, it follows that $Y_i \cap Y_j \subseteq X_i \cap X_j$. Hence, $\sum_{i \in L_k} |Y_i| \leq \sum_{i \in L_k} |X_i| + n \leq \sum_{i \in I} |X_i| + n \leq \mathcal{O}(t^2 n)$. Clearly $\tau(|Y_i|) = \Omega(|Y_i|)$ and it follows that $\mathcal{O}(\sum_{i \in L_k} 2^{t+1} \tau(|Y_i|)) \leq \mathcal{O}(2^{t+1}(\sum_{i \in L_k} \tau(|Y_i|)) \leq \mathcal{O}(2^{t+1} \tau(t^2 n))$. Combined with the fact that the height of the tree decomposition is $\mathcal{O}(t \log n)$, we get the claimed running time of $\mathcal{O}(2^{t+1} \cdot t \cdot \log n \cdot \tau(t^2 n))$. ◀

▶ **Lemma 11.** *Let i be a join node with children j_1 and j_2, where $X_i = X_{j_1} = X_{j_2}$. There is an algorithm that, for a fixed $S \subseteq X_i$, computes all the entries $\mathrm{mwp}_i(\ell, S)$, for all $\ell \in [|Y_i|]$, in time $\mathcal{O}(\tau(|Y_i|))$ if and only if there is an $\mathcal{O}(\tau(|Y_i|))$ time algorithm solving an instance of $(\min, +)$-Convolution with two sequences $(a[p])_{p=0}^{|Y_i|}$ and $(b[p])_{p=0}^{|Y_i|}$, where $a[p] = \mathrm{mwp}_{j_1}(p, S)$ for $p \in [|Y_{j_1}|]$ and $a[p] = \infty$ otherwise and $b[p] = \mathrm{mwp}_{j_1}(p, S)$ for $p \in [|Y_{j_2}|]$ and $a[p] = \infty$ otherwise.*

Proof. Recall that

$$\mathrm{mwp}_i(\ell, S) = min_{\ell_1 + \ell_2 - |S| = \ell, \ell_1, \ell_2 \geq |S|} \left(\mathrm{mwp}_{j_1}(\ell_1, S) + \mathrm{mwp}_{j_2}(\ell_2, S) - \sum_{e \in E(S, X_i \setminus S)} \mathrm{w}(e) \right).$$

Let $W = \sum_{e \in E(S, X_i \setminus S)} \mathrm{w}(e)$. Note that for a fixed i and a fixed S, both $\sum_{e \in E(S, X_i \setminus S)} \mathrm{w}(e)$ and $|S|$ are fixed. Hence,

$$\mathrm{mwp}_i(\ell, S) = min_{\ell_1 + \ell_2 - |S| = \ell, \ell_1, \ell_2 \geq |S|} \left(\mathrm{mwp}_{j_1}(\ell_1, S) + \mathrm{mwp}_{j_2}(\ell_2, S) \right) - W.$$

Let $(c[p])_{p=0}^{2|Y_i|-1}$ be the $(\min, +)$-convolution of the sequences $(a[p])_{p=0}^{|Y_i|}$ and $(b[p])_{p=0}^{|Y_i|}$; that is $c[k] = \min_{q+r=k}(a[q] + b[r])$. Finally, we set $\mathrm{mwp}_i(p, S) = c[p - |S|] - W$, for $p \in \{|S|, |S|+1, \ldots, |Y_i|\}$. All other entries are set to ∞. ◀

Combining Lemmas 10 and 11 with Theorem 8 we conclude the proof of Theorem 1. We remark that if a tree decomposition is not given then we can compute it, using the algorithm of Theorem 7, at the cost of a worse dependence on t.

Proof of Theorem 1. We assume that $(\min, +)$-Convolution can be solved in $\mathcal{O}(\tau(n))$ time. Using Proposition 9, we can compute in $\mathcal{O}(tn)$ time a nice tree decomposition $(\{X_i | i \in I\}, T = (I, F))$ of G, such that the width of the decomposition is $3t + 2$, the height is $\mathcal{O}(t \log n)$, and the number of nodes of T is $\mathcal{O}(tn)$. Afterwards, we invoke the algorithm of Lemma 10 to compute the minimum weight bisection in time $\mathcal{O}(2^{3t+3} \cdot (3t+2) \cdot \log n \cdot \tau((3t+2)^2 n)) = \mathcal{O}(8^t \cdot t \cdot \log n \cdot \tau(t^2 n))$ using the $\mathcal{O}(\tau(|Y_i|))$ time algorithm to compute the $(\min, +)$-convolution needed in the join nodes. Plugging in the naive $\mathcal{O}(n^2)$ time algorithm for $(\min, +)$-Convolution gives $\tau(n) = \mathcal{O}(n^2)$, completing the proof. ◀

3.1 Bounded Edge Weights

We now turn our attention to the case when the maximum weight of every edge in the input graph is bounded by some constant W. We show that in this case, we can actually compute a minimum bisection of a bounded treewidth graph of size n in time $\mathcal{O}(8^t \cdot (tW)^{\mathcal{O}(1)} \cdot n^{1.864} \log n)$ or, equivalently, $\mathcal{O}(8^t \cdot (tW)^{\mathcal{O}(1)} \cdot n^{1.864+\epsilon})$, for $\epsilon > 0$.

▶ **Lemma 12.** *Let G be an edge-weighted graph with maximum weight of an edge W with a tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of width t. Then for every node $i \in I$, every $S \subseteq X_i$ and every $\ell \in \{|S|, \ldots |Y_i| - |X_i \setminus S| - 1\}$ it holds that $|\mathrm{mwp}_i(\ell, S) - \mathrm{mwp}_i(\ell+1, S)| \leq (2t+1) \cdot W$.*

Proof. It is easy to see that $\mathrm{mwp}_i(\ell, S) = \mathrm{mwp}_i(|Y_i| - \ell, X_i \setminus S)$. Hence, without loss of generality, we can assume that $\mathrm{mwp}_i(\ell, S) \geq \mathrm{mwp}_i(\ell + 1, S)$. Now let A be a set of size ℓ such that $S = A \cap X_i$ and $\mathrm{mwp}_i(\ell, S) = \sum_{e \in E(A, \overline{A})} \mathrm{w}(e)$. It is well-known that we can order the vertices of a graph G such that every vertex has at most $\mathrm{tw}(G)$ neighbors earlier in the ordering [19]. Let us denote such an ordering σ and let v be the last vertex from $Y_i \setminus (A \cup X_i)$ in σ. Now $E(A \cup \{v\}, \overline{A \cup \{v\}}) = (E(A, \overline{A}) \setminus E(\{v\}, A)) \cup E(\{v\}, \overline{A \cup \{v\}})$. It follows that $\mathrm{mwp}_i(\ell + 1, S) \leq \mathrm{mwp}_i(\ell, S) + |E(\{v\}, \overline{A \cup \{v\}})| \cdot W$. By the choice of v, all the vertices in $\overline{A \cup \{v\}}$ are either earlier in σ than v or in X_i. Moreover, v has only at most $\mathrm{tw}(G)$ many neighbors that are earlier in σ than v and there are at most $t + 1$ vertices in X_i, hence $|E(\{v\}, \overline{A \cup \{v\}})| \leq \mathrm{tw}(G) + t + 1$. Since $\mathrm{tw}(G) \leq t$, the lemma follows. ◀

Observe, that the bound of Lemma 12 is tight up to a multiplicative constant. As an example achieving difference $|\mathrm{mwp}_i(\ell, S) - \mathrm{mwp}_i(\ell + 1, S)| \leq (t + 1) \cdot W$ take $S = X_i$ and an instance where the edges in Y_i have all weight W and are precisely all the pairs with one endpoint in X_i and the other in $Y_i \setminus X_i$.

Lemma 12 tells us that the restriction of the sequences $(a[p])_{p=0}^{|Y_i|}$ and $(b[p])_{p=0}^{|Y_i|}$ for which we need to compute the $(\min, +)$-Convolution in Lemma 11 to entries that are not ∞ has bounded difference. However, these two restricted sequences might not have the same length and it is not straightforward how to adapt the algorithm by Chan and Lewenstein [9]. To overcome this issue, we use a standard trick to change these sequences to monotone non-decreasing sequences with integer values bounded by $\mathcal{O}(n)$ and pad the shorter sequence by some large value. This trick is outlined by Chan and Lewenstein [9] but never formally stated, we repeat it here for completeness.

▶ **Theorem 13** ([9]). Monotone $(\min, +)$-Convolution *with all entries in* $\{0, \ldots, nD\}$ *can be solved in time* $\mathcal{O}((nD)^{1.859})$ *by a randomized algorithm, or in time* $\mathcal{O}((nD)^{1.864})$ *deterministically.*

We remark that Chan and Lewenstein [9] do not explicitly state the dependence on D. It is easy to see from their arguments that the dependence on D is at most $\mathcal{O}(D^{1.864})$, but we suspect that it is much better.

▶ **Lemma 14.** *Let* n_1, n_2 *be two integers such that* $n_1 \leq n_2$ *and let sequences* $(a[p])_{p=0}^{n_1}$ *and* $(b[p])_{p=0}^{n_2}$ *be two sequences with the difference bounded by a constant* D *and all entries in* $\{0, \ldots, n_2 D'\}$*, for some constant* D'*. Then we can compute the sequence* $(c[p])_{p=0}^{n_1+n_2}$ *such that* $c[k] = \min_{i+j=k}(a[i] + b[j])$ *in time* $\mathcal{O}((2n_2(D + D'))^{1.864})$.

Proof. To compute $(c[p])_{p=0}^{n_1+n_2}$ we start by changing the sequences $(a[p])_{p=0}^{n_1}$ and $(b[p])_{p=0}^{n_2}$ to bounded monotone sequences $(a'[p])_{p=0}^{n_1}$ and $(b'[p])_{p=0}^{n_2}$ by adding $D \cdot i$ to $a'[i]$ and $b'[i]$, respectively. Note that $\min_{i+j=k}(a[i] + b[j]) = \min_{i+j=k}(a'[i] + b'[j]) - D \cdot k$. Now let $C = \max(a'[n_1], b'[n_2])$. Finally, we create sequence $(a''[p])_{p=0}^{n_2}$ by setting $a''[p] = a'[p]$ if $a'[p]$ is defined and $a''[p] = 2C + 1$ otherwise. It is easy to see that $\min_{i+j=k}(a'[i] + b'[j]) = \min_{i+j=k}(a''[i] + b'[j])$ for all $k \in \{0, \ldots, n_1 + n_2\}$. Therefore, to compute the $(\min, +)$-convolution of the sequences $(a[p])_0^{n_1}$ and $(b[p])_0^{n_2}$ it suffices to compute the $(\min, +)$-convolution of the sequences $(a''[p])_{p=0}^{n_2}$ and $(b'[p])_{p=0}^{n_2}$, which are both monotone with integer entries between 0 and $C \leq 2(D \cdot n_2 + n_2 D') + 1$ and the proof follows due to Theorem 13. ◀

We are now in position to prove Theorem 3.

Proof of Theorem 3. Same as in the proof of Theorem 1, we start by using Proposition 9 to compute a nice tree decomposition $(\{X_i | i \in I\}, T = (I, F))$ of G, such that the width of the decomposition is $3t + 2$, the height is $\mathcal{O}(t \log n)$, and the number of nodes of T is $\mathcal{O}(tn)$.

Afterwards, we invoke the algorithm of Lemma 10 to compute the minimum weight bisection in time $\mathcal{O}(8^t \cdot t \cdot \log n \cdot \tau(t^2 n))$, where $\mathcal{O}(\tau(|Y_i|))$ is the time required to compute the entries $\mathrm{mwp}_i(\ell, S)$ for all $\ell \in [|Y_i|]$ and a fixed S in a join node.

It remains to show that we can compute $\mathrm{mwp}_i(\ell, S)$ for all $\ell \in [|Y_i|]$ and a fixed S in time $\mathcal{O}((tW)^{\mathcal{O}(1)} \cdot |Y_i|^{1.864})$. By Lemma 11, this is equivalent to solving an instance of $(\min, +)$-convolution with two sequences $(a[p])_{p=0}^{|Y_i|}$ and $(b[p])_{p=0}^{|Y_i|}$, where $a[p] = \mathrm{mwp}_{j_1}(p, S)$ for $p \in [|Y_{j_1}|]$ and $a[p] = \infty$ otherwise and $b[p] = \mathrm{mwp}_{j_1}(p, S)$ for $p \in [|Y_{j_2}|]$ and $a[p] = \infty$ otherwise. Note that $\mathrm{mwp}_{j_1}(\ell, S)$ ($\mathrm{mwp}_{j_1}(\ell, S)$) is set to ∞ if $\ell < |S|$ or $\ell > |Y_{j_1}| - |X_{j_1} \setminus S|$ ($\ell > |Y_{j_2}| - |X_{j_2} \setminus S|$). Hence, from Lemma 12 it follows that if both $a[p]$ and $a[p+1]$ (respectively $b[p]$ and $b[p+1]$) are finite, then $|a[p+1] - a[p]|$ (respectively $|b[p+1] - b[p]|$) is bounded by $(2t+1) \cdot W$, where W is the maximum weight of an edge in G, and hence it is constant. To finish the proof, let $n_{j_1} = |Y_{j_1}| - |S| - |X_{j_1} \setminus S|$ and $n_{j_2} = |Y_{j_2}| - |S| - |X_{j_2} \setminus S|$ and let sequences $(a'[p])_{p=0}^{n_{j_1}}$ and $(b'[p])_{p=0}^{n_{j_2}}$ be such that $a'[p] = a[p+|S|]$ and $b'[p] = b[p+|S|]$. That is a' and b' are created from a and b by removing ∞ from the sequences. For all $k \in \{2|S|, \ldots, n_{j_1} + n_{j_2} + 2|S|\}$ (that is whenever $\min_{i+j=k}(a[i] + b[j]) \neq \infty$) it holds that $\min_{i+j=k}(a[i] + b[j]) = \min_{i+j=k}(a'[i' - |S|] + b'[j' - |S|]) = \min_{i'+j'=k-2|S|}(a'[i'] + b'[j'])$. Therefore, to compute the $(\min, +)$-convolution of the sequences $(a[p])_0^{|Y_i|}$ and $(b[p])_0^{|Y_i|}$, it suffices to compute the sequence $(c'[p])_0^{n_{j_1}+n_{j_2}}$ such that $c'[k] = \min_{i+j=k}(a'[i] + b'[j])$. Clearly, due to Lemma 12, $(a'[p])_{p=0}^{n_{j_1}}$ and $(b'[p])_{p=0}^{n_{j_2}}$ have difference bounded by $(6t+5) \cdot W$. Moreover, let $n' = \max(n_{j_1}, n_{j_2})$, then it is easy to see that both $a[|S|]$ and $b[|S|]$ are at most $|S| \cdot n' \cdot W \leq (3t+3) \cdot n' \cdot W$ and hence the entries in $(a'[p])_{p=0}^{n_{j_1}}$ and $(b'[p])_{p=0}^{n_{j_2}}$ are all integers between 0 and $(3t+3) \cdot n' \cdot W + (6t+5) \cdot W \cdot n' = (9t+8) \cdot W \cdot n'$. Therefore, we can compute the sequence $(c'[p])_0^{n_{j_1}+n_{j_2}}$ in $\mathcal{O}(((30t+26) \cdot W \cdot n')^{1.864})$ by Lemma 14, finishing the proof. ◀

4 Tree Bisection is as Hard as $(\min, +)$-Convolution

We complement Theorem 3 by showing that if the Bisection problem can be solved in subquadratic time, i.e., in time $\mathcal{O}(n^{2-\epsilon})$ for $\epsilon > 0$, on weighted trees than the $(\min, +)$-convolution problem can be solved in subquadratic time as well, i.e., in time $\mathcal{O}(n^{2-\delta})$ for $\delta > 0$. We follow a strategy similar to that of [1] used for proving a lower bound on the Tree Sparsity problem.

▶ **Definition 15** (SUM3 problem). *Given three sequences $A, B, C \in \mathbb{Z}^n$, decide if the following statement is true: $\exists i, j : A_i + B_j + C_{i+j} \leq 0$.*

▶ **Theorem 16** ([1, 24]). *The $(\min, +)$-Convolution problem can be solved in time $\mathcal{O}(n^{2-\epsilon})$, for $\epsilon > 0$, if and only if the SUM3 problem can be solved in $\mathcal{O}(n^{2-\delta})$ time, for $\delta > 0$.*

Hence, given Theorem 16, we prove the main theorem of this section by a reduction from SUM3 to the Bisection problem on weighted trees. We start by describing the construction.

Let W be equal to 10 times the largest absolute value of an entry in A, B, and C. We create a root vertex r. Consider $A \in \mathbb{Z}^n$. We first construct a path $P_A = \{r, a_0, a_1, \ldots, a_{n-1}\}$ of n vertices (excluding r) such that the weight of the ith edges is $W + A_i$, for $i = 0, 1, \ldots, n-1$. Similarly, for $B \in \mathbb{Z}^n$, we construct a path $P_B = \{r, b_0, b_1, \ldots, b_{n-1}\}$ of n vertices (excluding r) such that the weight of the ith edges is $W + B_i$, for $i = 0, 1, \ldots, n-1$. We then create a new vertex c and a path $P_C = \{c, c_0, c_1, \ldots, c_{n-1}, c_n, c_{n+1}, \ldots, c_{2n-1}, r\}$ of $2n+1$ vertices such that the weight of the ith edges is $W + C_i$, for $i = 0, 1, \ldots, n-1$ and the weight is nW otherwise ($i > n-1$). Finally, we attach $30n$ pendant vertices to r, $10n$ pendant vertices to

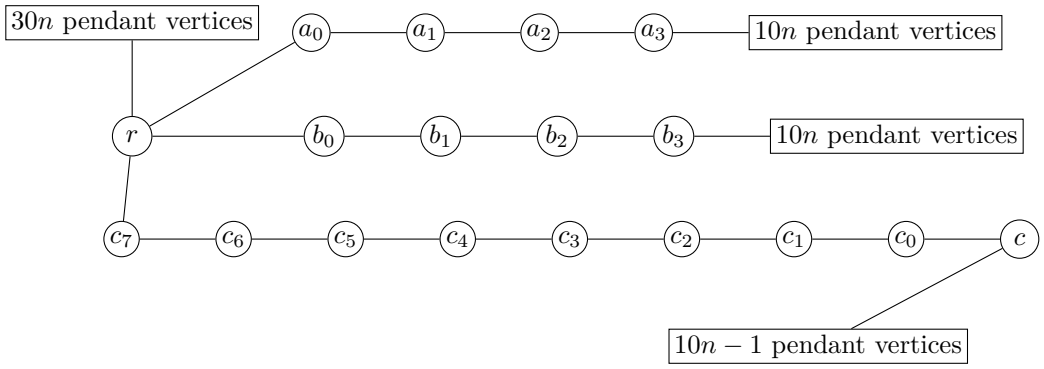

Figure 1 The reduction from SUM3 (for $n = 4$) to the BISECTION problem on weighted trees.

a_{n-1}, $10n$ pendant vertices to b_{n-1}, and $10n - 1$ pendant vertices to c. The weight of each of those edges is nW. We let T denote the resulting tree (see Figure 1). Note that the total number of vertices in T is $60n + 4n = 64n$.

▶ **Lemma 17.** *Let $A, B, C \in \mathbb{Z}^n$ be an instance of* SUM3 *and let T be the corresponding instance of* BISECTION. *Then $\exists i, j : A_i + B_j + C_{i+j} \leq 0$ if and only if T has a bisection of weight less than or equal $3W$.*

Proof. Assume that $\exists i, j : A_i + B_j + C_{i+j} \leq 0$. We claim that T admits a bisection whose weight is at most $3W$. We pick one edge from each of the three paths P_A, P_B, and P_C. In particular, we pick the i-th edge from P_A, the j-th edge from P_B, and the k-th edge from P_C, where $k = i + j$. The total weight is therefore $3W + A_i + B_j + C_{i+j} \leq 3W$. The total number of vertices in the r-partition is $30n + i + j + 2n - k = 32n$ and the total number of vertices in the abc-partition is $30n + 2n + k - (i + j) = 32n$, as needed.

For the other direction, assume that T admits a bisection (X, Y) whose weight is at most $3W$. Notice, that from the choice of W and the construction, it follows that the weight of any at least four edges is at least $3W + \frac{6W}{10}$, and consequently $|E(X, Y)| \leq 3$. We claim that $E(X, Y)$ contains exactly three edges from T, each edge from a different path. Assume otherwise, i.e., that at least one path remains untouched. Then, the corresponding partition will contain at least $40n$ vertices which is greater than $32n$ vertices. Now, let $E(X, Y)$ contain the i-th edge from P_A, the j-th edge from P_B, and the k-th edge from P_C. It remains to show that $k = i + j$. The size of the partition containing r is $30n + i + j + 2n - k$. Since the number of vertices in T is $64n$ and both partitions must have equal size, we get $30n + i + j + 2n - k = 32n$ and therefore $i + j = k$, as needed. ◀

The construction, together with Proposition 16 and Lemma 17 conclude the proof of Theorem 2.

References

1 Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. Better Approximations for Tree Sparsity in Nearly-linear Time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 2215–2229, Philadelphia, PA, USA, 2017. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=3039686.3039831`.

2 Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A c^k n 5-Approximation Algorithm for Treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016. `doi:10.1137/130947374`.

3 Hans L. Bodlaender and Torben Hagerup. Parallel Algorithms with Optimal Speedup for Bounded Treewidth. *SIAM J. Comput.*, 27(6):1725–1746, December 1998. doi:10.1137/S0097539795289859.

4 Karl Bringmann, Fabrizio Grandoni, Barna Saha, and Virginia Vassilevska Williams. Truly Subcubic Algorithms for Language Edit Distance and RNA-Folding via Fast Bounded-Difference Min-Plus Product. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 375–384. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.48.

5 Thang Nguyen Bui, Soma Chaudhuri, Frank Thomson Leighton, and Michael Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7(2):171–191, 1987.

6 Thang Nguyen Bui, C. Heigham, Curt Jones, and Frank Thomson Leighton. Improving the Performance of the Kernighan-Lin and Simulated Annealing Graph Bisection Algorithms. In Donald E. Thomas, editor, *Proceedings of the 26th ACM/IEEE Design Automation Conference, Las Vegas, Nevada, USA, June 25-29, 1989.*, pages 775–778. ACM Press, 1989. doi:10.1145/74382.74527.

7 Thang Nguyen Bui and Andrew Peck. Partitioning Planar Graphs. *SIAM J. Comput.*, 21(2):203–215, 1992.

8 Thang Nguyen Bui and Lisa C. Strite. An Ant System Algorithm For Graph Bisection. In *GECCO*, pages 43–51. Morgan Kaufmann, 2002.

9 Timothy M. Chan and Moshe Lewenstein. Clustered Integer 3SUM via Additive Combinatorics. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 31–40. ACM, 2015. doi:10.1145/2746539.2746568.

10 Bruno Courcelle. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Inf. Comput.*, 85(1):12–75, 1990.

11 Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Minimum bisection is fixed parameter tractable. In *STOC*, pages 323–332. ACM, 2014.

12 Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On Problems Equivalent to (min, +)-Convolution. *ACM Trans. Algorithms*, 15(1):14:1–14:25, 2019. URL: https://dl.acm.org/citation.cfm?id=3293465, doi:10.1145/3293465.

13 Uriel Feige and Robert Krauthgamer. A Polylogarithmic Approximation of the Minimum Bisection. *SIAM J. Comput.*, 31(4):1090–1118, 2002.

14 Uriel Feige, Robert Krauthgamer, and Kobbi Nissim. Approximating the minimum bisection size (extended abstract). In *STOC*, pages 530–536, 2000.

15 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

16 M Goldberg and Z Miller. A parallel algorithm for bisection width in trees. *Computers & Mathematics with Applications*, 15(4):259–266, 1988.

17 Klaus Jansen, Marek Karpinski, Andrzej Lingas, and Eike Seidel. Polynomial Time Approximation Schemes for MAX-BISECTION on Planar and Geometric Graphs. In *Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science*, STACS '01, pages 365–375, Berlin, Heidelberg, 2001. Springer-Verlag. URL: http://dl.acm.org/citation.cfm?id=646515.759237.

18 Subhash Khot and Nisheeth K. Vishnoi. The Unique Games Conjecture, Integrality Gap for Cut Problems and Embeddability of Negative-Type Metrics into $_1$. *J. ACM*, 62(1):8:1–8:39, 2015.

19 Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994. doi:10.1007/BFb0045375.

20 Robert Malcolm Macgregor. On partitioning a graph: a theoretical and empirical study. Technical report, UC Berkeley, 1979.

21 Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of*

Computing, Victoria, British Columbia, Canada, May 17-20, 2008, pages 255–264. ACM, 2008. doi:10.1145/1374376.1374415.

22 René van Bevern, Andreas Emil Feldmann, Manuel Sorge, and Ondrej Suchý. On the Parameterized Complexity of Computing Graph Bisections. In *WG*, pages 76–87, 2013.

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

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