Bisection of Bounded Treewidth Graphs by Convolutions*

Eduard Eiben[a], Daniel Lokshtanov[b], Amer E. Mouawad[c]

[a]Department of Computer Science, Royal Holloway, University of London, UK
[b]Department of Computer Science, UC Santa Barbara, US
[c]Department of Computer Science, American University of Beirut, Lebanon

---

Abstract

In the Bisection problem, we are given as input an edge-weighted graph G. The task is to determine whether there exists 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 impossi-

---

*A preliminary version of this paper was accepted for publication at the 27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany.

Email addresses: eduard.eiben@rhul.ac.uk (Eduard Eiben), daniello@ucsb.edu (Daniel Lokshtanov), aa368@aub.edu.lb (Amer E. Mouawad)

ble. 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 $O(8^t t^{O(1)} n^{1.864} \log n)$.

Keywords: bisection, treewidth, convolution, fine-grained complexity

---

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 [1]. Bisection has been studied extensively from the perspective of approximation algorithms [2, 3, 4, 5], parameterized algorithms [6, 7, 8] heuristics [9, 10] and average case complexity [11].

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 [12] 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 [13]. 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. [14] 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 [15]). Thus, it is quite natural to ask whether the dependence on n in the algorithm of Jansen et al. [14] 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].

---

[1]Note that the Goldberg and Miller's algorithm [13] is parallel, while ours is sequential.

Theorem 1.1. *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 $c[k] = \min_{i=0,\dots,k}(a[i] + b[k-i])$. 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 1.1 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* [16] analysis, Cygan et al. [17] 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 1.1 to a sub-quadratic algorithm without disproving the conjecture of Cygan et al. [17], 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 1.2. *If* Bisection *on weighted trees can be solved in time $\mathcal{O}(n^{2-\epsilon})$ for $\epsilon > 0$, then* $(\min,+)$-Convolution *can be solved in $\mathcal{O}(n^{2-\delta})$ time for $\delta > 0$.*

Theorem 1.2 together with Theorem 1.1 (or rather its re-statement in terms of convolutions), puts Bisection on weighted trees in the class of problems equivalent to $(\min,+)$-Convolution [17].

In light of Theorem 1.2, the BISECTION problem on *unweighted* graphs (where all weights are 1) 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 1.3. *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 1.3 is that the $(\min, +)$-convolution steps in the algorithm of Theorem 1.1 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 one 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 [18] 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 1.1 and 1.3. Theorem 1.2 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 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 2.1 ($(\min, +)$-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 sometime 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 [18]. Moreover, we can relax these requirements [19] 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 $G - S = G[V(G) \setminus S]$.

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 2.2 (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 2.3. *A* tree decomposition *of a graph G is a pair $(\{X_i \mid i \in V(T)\}, T)$, where $\{X_i \mid i \in V(T)\}$ is a collection of subsets of V(G), T is a rooted tree such that the following conditions hold:*

- $\bigcup_{i \in V(T)} X_i = V(G)$;
- *For all edges $\{u,v\} \in E(G)$, there exists $i \in V(T)$ with $u,v \in X_i$;*

- *For every vertex $v \in V(G)$, the subgraph of T induced by $\{i \in V(T) \mid v \in X_i\}$ is connected.*

The width *of a tree decomposition $(\{X_i \mid i \in V(T)\}, T)$ is $max_{i \in V(T)}(|X_i| - 1)$. The* treewidth *of a graph G,* $\text{tw}(G)$, *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 family of graphs where each graph has treewidth at most some fixed constant t is called a bounded treewidth family of graphs. A graph within a bounded treewidth family is called a bounded treewidth graph.*

Given a tree decomposition $(\{X_i \mid i \in V(T)\}, T)$ 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 2.1 (Bodlaender et al. [20])**.** *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 2.2 below with standard arguments (we refer the reader to [20] for more details), we arrive at Proposition 2.1, which is the form that will be required to obtain our algorithms.

Theorem 2.2 (Bodlaender and Hagerup [21])**.** *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 2.1. *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 $\mathcal{O}(2^t \cdot n^3)$-time algorithm for solving the Bisection problem on graphs of treewidth at most t by Jansen et al. [14]. The algorithm is a standard dynamic programming algorithm over a tree decomposition. Given a graph G together with its nice tree decomposition $(\{X_i | i \in V(T)\}, T)$ of width t the algorithm works as follows.

For each node $i \in V(T)$, 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 $i \in V(T)$, 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 $\mathrm{mwp}_i(\ell, S)$ is set to $\min_{S' \subseteq Y_i, |S'| = \ell, S' \cap X_i = S}(\sum_{e \in E(S', Y_i \setminus S')} \mathrm{w}(e))$. That is, $\mathrm{mwp}_i(\ell, S)$ is equal to the minimum possible weight of a partition where S and $X_i \setminus S$ are in different parts of the partition and the side including S is of cardinality exactly ℓ. When such a partition is not possible, we set $\mathrm{mwp}_i(\ell, S)$ 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 $\mathrm{mwp}_i(0, \emptyset) = 0$.

- 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

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

  Note that there are exactly two subsets S' that satisfy the condition $S' \subseteq X_j$ and $S' \cap X_i = S$. Those subsets are S and $S \cup \{v\}$, where v is the forgotten vertex.

- 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

$$\mathrm{mwp}_i(\ell, S) = \mathrm{mwp}_j(\ell - 1, S \setminus \{v\}) + \sum_{e \in \{\{v,s\} | s \in X_i \setminus S\}} \mathrm{w}(e).$$

Otherwise, we set

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

- 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

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

We omit the proof of correctness and refer the reader to [14] 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 3.1. *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 $\mathrm{mwp}_i(\ell, S)$ for all $\ell \in [|Y_i|]$ and a fixed S in a join node.*

Proof. Let $(\{X_i | i \in V(T)\}, T)$ 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 $Y_i \cap Y_j \subseteq X_i \cap X_j$ and $(Y_i \setminus X_i) \cap (Y_j \setminus X_j) = \emptyset$. Summing over all $i \in L_k$, we get $\sum_{i \in L_k} |Y_i| \leq \sum_{i \in L_k} |X_i| + n \leq \sum_{i \in V(T)} |X_i| + n \leq \mathcal{O}(t^2 n)$. 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 3.2. *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 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

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

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,

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

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 the Lemmas 3.1 and 3.2 with Theorem 2.2 we conclude the proof of Theorem 1.1. We remark that if a tree decomposition is not given then we can compute it, using the algorithm of Theorem 2.1, at the cost of a worse dependence on t.

Proof of Theorem 1.1. We assume that $(\min,+)$-Convolution can be solved in $\mathcal{O}(\tau(n))$ time. Using Proposition 2.1, we can compute in $\mathcal{O}(tn)$ time a nice tree decomposition $(\{X_i | i \in V(T)\}, T)$ 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 3.1 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, more generally, $\mathcal{O}(8^t \cdot (tW)^{\mathcal{O}(1)} \cdot n^{1.864+\epsilon})$, for $\epsilon > 0$.

Lemma 3.3. *Let G be an edge-weighted graph with maximum weight of an edge W with a tree decomposition $(\{X_i \mid i \in V(T)\}, T)$ of width t. Then for every node $i \in V(T)$, every $S \subseteq X_i$ and every $\ell \in \{|S|, \dots |Y_i| - |X_i \setminus S| - 1\}$ it holds that $|\operatorname{mwp}_i(\ell, S) - \operatorname{mwp}_i(\ell + 1, S)| \leq (2t + 1) \cdot W$.*

Proof. It is easy to see that $\operatorname{mwp}_i(\ell, S) = \operatorname{mwp}_i(|Y_i| - \ell, X_i \setminus S)$. Hence, without loss of generality, we can assume that $\operatorname{mwp}_i(\ell, S) \leq \operatorname{mwp}_i(\ell + 1, S)$. Now let A be a set of size ℓ such that $S = A \cap X_i$ and $\operatorname{mwp}_i(\ell, S) = \sum_{e \in E(A, \overline{A})} \mathrm{w}(e)$. It is well-known that we can order the vertices of graph G such that every vertex has at most $\operatorname{tw}(G)$ neighbors earlier in the ordering [22]. Let us denote such an ordering by σ and let v be the last vertex from $Y_i \setminus (A \cup X_i)$ in σ. We show that "switching the side" of v allows us to bound $\operatorname{mwp}_i(\ell + 1, S)$. Note that we pick a vertex $Y_i \setminus (A \cup X_i)$ since S is fixed and, consequently, $X_i \setminus S$ is also fixed. We have $E(A \cup \{v\}, \overline{A \cup \{v\}}) = (E(A, \overline{A}) \setminus E(\{v\}, A)) \cup E(\{v\}, \overline{A \cup \{v\}})$. It follows that $\operatorname{mwp}_i(\ell + 1, S) \leq \operatorname{mwp}_i(\ell, S) + |E(\{v\}, \overline{A \cup \{v\}})| \cdot W$. 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 $\operatorname{tw}(G)$ many neighbors that are earlier in σ than v and there are at most $t+1$ vertices in X_i, hence $|E(\{v\}, \overline{A \cup \{v\}})| \leq \operatorname{tw}(G) + t + 1$. Since $\operatorname{tw}(G) \leq t$, the lemma follows. $\square$

Observe, that the bound of Lemma 3.3 is tight up to a multiplicative constant. As an example achieving difference $|\operatorname{mwp}_i(\ell, S) - \operatorname{mwp}_i(\ell+1, S)| \leq (t+1) \cdot W$ 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 3.3 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 3.2 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 [18]. To overcome this issue, we use a standard trick to change these sequence 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 [18] but never formally stated, we repeat it here for completeness.

Theorem 3.1 ([18])**.** Monotone $(\min,+)$-Convolution *with all entries in* $\{0,\dots,nD\}$ *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 [18] 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 3.4. *Let n_1, n_2 be two integers such that $n_1 \le n_2$ and let sequences $(a[p])_{p=0}^{n_1}$ and $(b[p])_{p=0}^{n_2}$ be two sequences with the difference bounded by an integer D and all entries in $\{0,\dots,n_2D'\}$, for some integer 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 sequences $(a''[p])_{p=0}^{n_2}$ 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 $k \in \{0,\dots,n_1+n_2\}$. 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 $C \le 2(D \cdot n_2 + n_2D') + 1$ and the proof follows due to Theorem 3.1. $\square$

We are now in position to prove Theorem 1.3.

Proof of Theorem 1.3. Same as in the proof of Theorem 1.1, we start by using Proposition 2.1 to compute a nice tree decomposition $(\{X_i | i \in V(T)\}, T)$ 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 3.1 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 3.2, 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 3.3 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 suffice 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 3.3, $(a'[p])_{p=0}^{n_{j_1}}$ 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 3.4, finishing the proof. $\square$

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

We complement Theorem 1.3 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 then 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 [23] used for proving a lower bound on the Tree Sparsity problem.

Definition 4.1 (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 4.1 ([23, 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 4.1, 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 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 4.1. *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 $3W$. First, note that such a bisection cannot contain any of the pendant edges because they are too heavy, i.e., have weight larger than $3W$. 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 $3W + \frac{6W}{10}$,

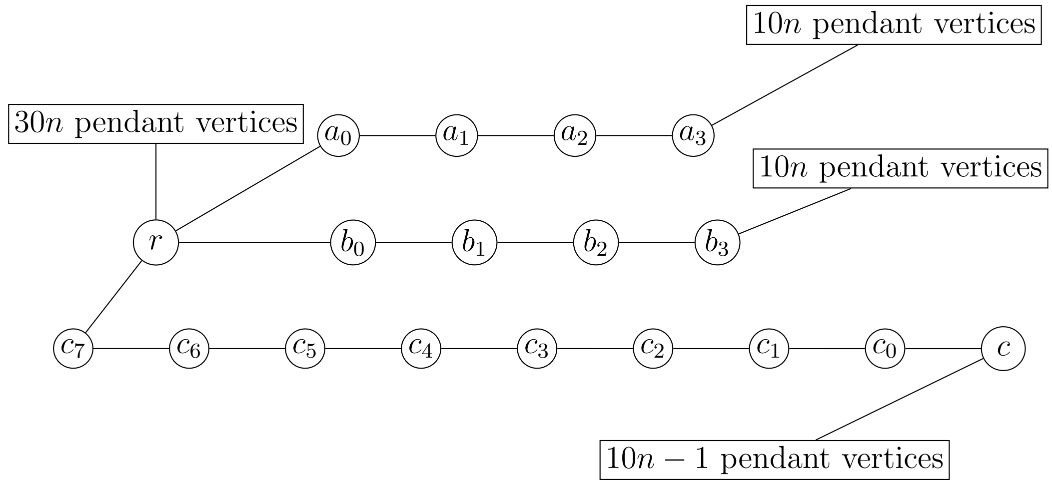

Figure 1: The reduction from SUM3 (for $n = 4$) to the BISECTION problem on weighted trees.

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 4.1 and Lemma 4.1 concludes the proof of Theorem 1.2.

References

[1] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, 1979.

[2] U. Feige, R. Krauthgamer, K. Nissim, Approximating the minimum bisection size (extended abstract), in: STOC, 2000, pp. 530–536.

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

[4] S. Khot, N. K. Vishnoi, The unique games conjecture, integrality gap for cut problems and embeddability of negative-type metrics into $_1$, J. ACM 62 (1) (2015) 8:1–8:39.

[5] H. Räcke, Optimal hierarchical decompositions for congestion minimization in networks, in: STOC, 2008, pp. 255–264.

[6] T. N. Bui, A. Peck, Partitioning planar graphs, SIAM J. Comput. 21 (2) (1992) 203–215.

[7] M. Cygan, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Minimum bisection is fixed parameter tractable, in: STOC, ACM, 2014, pp. 323–332.

[8] R. van Bevern, A. E. Feldmann, M. Sorge, O. Suchý, On the parameterized complexity of computing graph bisections, in: WG, 2013, pp. 76–87.

[9] T. N. Bui, C. Heigham, C. Jones, F. T. Leighton, Improving the performance of the kernighan-lin and simulated annealing graph bisection algorithms, in: D. E. Thomas (Ed.), Proceedings of the 26th ACM/IEEE Design Automation Conference, Las Vegas, Nevada, USA, June 25-29, 1989., ACM Press, 1989, pp. 775–778. `doi:10.1145/74382.74527`. URL `https://doi.org/10.1145/74382.74527`

[10] T. N. Bui, L. C. Strite, An ant system algorithm for graph bisection, in: GECCO, Morgan Kaufmann, 2002, pp. 43–51.

[11] T. N. Bui, S. Chaudhuri, F. T. Leighton, M. Sipser, Graph bisection algorithms with good average case behavior, Combinatorica 7 (2) (1987) 171–191.

[12] R. M. Macgregor, On partitioning a graph: a theoretical and empirical study. (1979).

[13] M. Goldberg, Z. Miller, A parallel algorithm for bisection width in trees, Computers & Mathematics with Applications 15 (4) (1988) 259–266.

[14] K. Jansen, M. Karpinski, A. Lingas, E. 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, Springer-Verlag, Berlin, Heidelberg, 2001, pp. 365–375.
URL http://dl.acm.org/citation.cfm?id=646515.759237

[15] B. Courcelle, The monadic second-order logic of graphs. i. recognizable sets of finite graphs, Inf. Comput. 85 (1) (1990) 12–75.

[16] V. V. Williams, On some fine-grained questions in algorithms and complexity, in: Proceedings of the ICM, 2018.

[17] M. Cygan, M. Mucha, K. Wegrzycki, M. Wlodarczyk, On problems equivalent to (min, +)-convolution, ACM Trans. Algorithms 15 (1) (2019) 14:1–14:25.
URL https://dl.acm.org/citation.cfm?id=3293465

[18] T. M. Chan, M. Lewenstein, Clustered integer 3sum via additive combinatorics, in: R. A. Servedio, R. Rubinfeld (Eds.), Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015, ACM, 2015, pp. 31–40. doi:10.1145/2746539.2746568.
URL https://doi.org/10.1145/2746539.2746568

[19] K. Bringmann, F. Grandoni, B. Saha, V. V. Williams, Truly sub-cubic algorithms for language edit distance and rna-folding via fast bounded-difference min-plus product, in: I. Dinur (Ed.), IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA, IEEE Computer Society, 2016, pp. 375–384. doi:10.1109/FOCS.2016.48.
URL https://doi.org/10.1109/FOCS.2016.48

[20] H. L. Bodlaender, P. G. Drange, M. S. Dregi, F. V. Fomin, D. Lokshtanov, M. Pilipczuk, A c^{k} n 5-approximation algorithm for treewidth, SIAM J. Comput. 45 (2) (2016) 317–378. doi:10.1137/130947374.
URL https://doi.org/10.1137/130947374

[21] H. L. Bodlaender, T. Hagerup, Parallel algorithms with optimal speedup for bounded treewidth, SIAM J. Comput. 27 (6) (1998) 1725–1746. doi:10.1137/S0097539795289859.
URL https://doi.org/10.1137/S0097539795289859

[22] T. Kloks, Treewidth, Computations and Approximations, Vol. 842 of Lecture Notes in Computer Science, Springer, 1994. doi:10.1007/BFb0045375.
URL https://doi.org/10.1007/BFb0045375

[23] A. Backurs, P. Indyk, L. 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, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2017, pp. 2215–2229.
URL http://dl.acm.org/citation.cfm?id=3039686.3039831

[24] V. V. Williams, R. R. Williams, Subcubic equivalences between path, matrix, and triangle problems, J. ACM 65 (5) (2018) 27:1–27:38. doi:10.1145/3186893.
URL http://doi.acm.org/10.1145/3186893