On Minimum Bisection and Related Partition Problems in Graphs with Bounded Tree Width

Cristina G. Fernandes\textsuperscript{a,1,3}, Tina Janne Schmidt\textsuperscript{b,2,4}, and Anusch Taraz\textsuperscript{b,4}

\textsuperscript{a} Instituto de Matemática e Estatística, Universidade de São Paulo, 05508-090 São Paulo, Brazil
\textsuperscript{b} Institut für Mathematik (E-10), Technische Universität Hamburg-Harburg, 21073 Hamburg, Germany

Abstract

Minimum Bisection denotes the NP-hard problem to partition the vertex set of a graph into two sets of equal sizes while minimizing the number of edges between these two sets. We consider this problem in bounded degree graphs with a given tree decomposition \((T, \mathcal{X})\) and prove an upper bound for their minimum bisection width in terms of the structure and width of \((T, \mathcal{X})\). When \((T, \mathcal{X})\) is provided as input, a bisection satisfying our bound can be computed in time proportional to the encoding length of \((T, \mathcal{X})\). Furthermore, our result can be generalized to \(k\)-section, which is known to be APX-hard even when restricted to trees with bounded degree.

Keywords: Minimum Bisection, Minimum \(k\)-Section, tree decomposition.

\textsuperscript{1} Partially supported by CNPq, FAPESP, and Project MaCLinC of NUMEC/USP.
\textsuperscript{2} Supported by the Evangelische Studienwerk Villigst e.V.
The cooperation of the three authors was supported by PROBRAL CAPES/DAAD Proc. 430/15 (February 2015 to December 2016, DAAD Projekt-ID 57143515).
\textsuperscript{3} Email: cris@ime.usp.br
\textsuperscript{4} Email: \{tina.janne.schmidt, taraz\}@tuhh.de
1 Introduction and Results

Let us first fix some basic terminology. A cut \((V_1, V_2, \ldots, V_k)\) in a graph \(G\) is a partition of its vertex set. An edge \(\{x, y\}\) of \(G\) is cut by \((V_1, V_2, \ldots, V_k)\) if \(x\) and \(y\) belong to different sets \(V_i\) and \(V_j\). The number of edges cut by \((V_1, V_2, \ldots, V_k)\) is called the width of the cut and is denoted by \(e(V_1, V_2, \ldots, V_k)\). A \(k\)-section is a cut \((V_1, V_2, \ldots, V_k)\) such that the sizes of \(V_i\) and \(V_j\) differ by at most one for all \(i, j \in [k]\), where \([k] := \{1, 2, \ldots, k\}\). The Minimum \(k\)-Section Problem asks to find a minimum \(k\)-section \((V_1, V_2, \ldots, V_k)\) in a graph \(G\), i.e., a \(k\)-section of minimum width among all \(k\)-sections in \(G\), and \(\text{MinSec}(k, G)\) is defined to be the width of \((V_1, V_2, \ldots, V_k)\). The special case \(k = 2\) is also called the Minimum Bisection Problem. In what follows, unless stated otherwise, \(n\) and \(\Delta(G)\) denote the number of vertices and the maximum degree of the considered graph \(G\), respectively.

1.1 Minimum Bisection

Finding a minimum bisection is a famous NP-hard optimization problem [6]. Jansen et al. showed that dynamic programming gives an exact algorithm with running time \(O(2^t n^3)\) when a tree decomposition of width \(t\) is provided as input [7]. Thus, the problem becomes polynomially tractable for graphs of bounded tree width. For general graphs, the best known approximation algorithm achieves an approximation ratio of \(O(\log n)\) [9]. Further, the Minimum Bisection Problem restricted to 3-regular graphs is as hard to approximate as its general version [2]. Here, we focus on upper bounds for the minimum bisection width in bounded degree graphs with a given tree decomposition of small width. Lower bounds are more difficult to derive and only few are known. One example is the spectral bound \(\text{MinSec}(2, G) \geq \frac{1}{4} \lambda_2 n\), where \(\lambda_2\) denotes the second eigenvalue of the Laplacian of \(G\) [8].

In [4], we have shown that for every tree \(T\)

\[
\text{MinSec}(2, T) \leq \frac{8\Delta(T)}{\text{diam}^*(T)},
\]

where \(\text{diam}^*(T) := (\text{diam}(T) + 1)/n\) denotes the relative diameter of the tree \(T\), i.e., the fraction of vertices of \(T\) on a longest path in \(T\). This implies that every tree with linear diameter and bounded maximum degree allows a bisection of constant width. In general, every tree with bounded degree allows a bisection of width \(O(\log_2 n)\) and the perfect ternary tree shows that this is tight up to a constant factor.
Here, we improve the bound in (1) to be polylogarithmic in $1/\text{diam}^*(T)$, more precisely $\text{MinSec}(2, T) \leq \Delta(T)((\log_2(1/d))^2 + 9 \log_2(1/d) + 8)$, where $d := \text{diam}^*(T)$. Also, we give a linear-time algorithm that computes a bisection satisfying this bound. Furthermore, we establish a similar bound for general graphs by using a tree decomposition $(T, \mathcal{X})$. Instead of using the relative diameter, we define a parameter $r(T, \mathcal{X})$ that roughly measures how close the tree decomposition $(T, \mathcal{X})$ is to a path decomposition, which is a tree decomposition $(\tilde{T}, \tilde{\mathcal{X}})$ where $\tilde{T}$ is a path. For example, every path $P$ has $\text{diam}^*(P) = 1$ and allows a bisection of width 1. When the relative diameter of a tree decreases, it looks less like a path. Similarly, consider a graph $G$ and a path decomposition $(P, \mathcal{X})$ of $G$ of width $t - 1$. It is easy to see that $G$ allows a bisection of width at most $t\Delta(G)$ by walking along the path $P$ until we have seen $n/2$ vertices of $G$ in the bags and then bisecting $G$ there. Therefore, we will define $r(T, \mathcal{X})$ in such a way that $r(T, \mathcal{X})$ is 1 for path decompositions $(T, \mathcal{X})$ and $r(T, \mathcal{X})$ decreases when $(T, \mathcal{X})$ is less path-like. Let $G = (V, E)$ be a graph and $(T, \mathcal{X})$ a tree decomposition of $G$ with $\mathcal{X} = (X_i)_{i \in V(T)}$. Define $w(T', \mathcal{X}) := |\bigcup_{i \in V(T')} X_i|$ for $T' \subseteq T$ and let $P$ be a path in $T$ for which $w(P, \mathcal{X})$ is maximum among all paths in $T$. Then, we define $r(T, \mathcal{X}) := w(P, \mathcal{X})/w(T, \mathcal{X})$ to be the relative weight of a heaviest path in $(T, \mathcal{X})$. Observe that $w(T, \mathcal{X})$ is the number of vertices of $G$ and hence we always have $\frac{1}{n} \leq r(T, \mathcal{X}) \leq 1$. Furthermore, every tree $T'$ allows a tree decomposition $(T, \mathcal{X})$ with $r(T, \mathcal{X}) \geq \text{diam}^*(T')$. To state the improved version of (1) for general graphs, define the size of a tree decomposition $(T, \mathcal{X})$ with $\mathcal{X} = (X_i)_{i \in V(T)}$ as $\| (T, \mathcal{X}) \| := |V(T)| + \sum_{i \in V(T)} |X_i|$, which measures its encoding length.

**Theorem 1.1** Every graph $G$ on $n$ vertices that allows a tree decomposition $(T, \mathcal{X})$ of width $t - 1$ satisfies

$$\text{MinSec}(2, G) \leq \frac{1}{2} t \Delta(G) \left( \left( \log_2 \frac{1}{r(T, \mathcal{X})} \right)^2 + 9 \log_2 \frac{1}{r(T, \mathcal{X})} + 8 \right).$$

If $V(G) = [n]$ and the tree decomposition $(T, \mathcal{X})$ is provided, a bisection satisfying this bound can be computed in $O(\| (T, \mathcal{X}) \|)$ time.

Note that the algorithm corresponding to Theorem 1.1 does not necessarily compute a minimum bisection, but it is much faster than the algorithm by Jansen et al. in [7], which computes a minimum bisection. Moreover, Theorem 1.1 implies that every graph $G$ that allows a path-like tree decomposition $(T, \mathcal{X})$ of width $t$, i.e., with $r(T, \mathcal{X}) = \Omega(1)$, has a bisection of width $O(t\Delta(G))$. We conclude this section with the following lemma that
relaxes the size constraint on the sets of the cut and also gives an upper bound of $O(t\Delta(G))$ on the cut width without requiring the tree decomposition to be path-like. It is a useful tool to prove Theorem 1.1 and might be of independent interest. For a real $x$ we use $\lceil x \rceil$ to denote the smallest integer $i$ with $x \leq i$.

**Lemma 1.2** Let $G$ be a graph on $n$ vertices that allows a tree decomposition $(T, X)$ of width $t - 1$. For every $m \in [n]$ and every $0 \leq c < 1$, there is a cut $(V_1, V_2)$ in $G$ such that $cm \leq |V_1| \leq m$ and $e(V_1, V_2) \leq \left\lceil \log_2 \frac{1}{1-c} \right\rceil t\Delta(G)$. If $V(G) = [n]$ and the tree decomposition $(T, X)$ is provided, then a cut satisfying these requirements can be computed in $O(\|T, X\|)$ time.

**1.2 Generalization to Minimum $k$-Section**

The algorithm of Jansen et al. in [7] for computing a minimum bisection can be modified to compute a minimum $k$-section in time polynomial in $n$ but not in $k$. Also, when $k$ is part of the input, the width of a minimum $k$-section cannot be approximated within any finite factor for general graphs [1] unless P=NP. Furthermore, the problem remains APX-hard when restricted to trees with bounded maximum degree, and it is NP-hard to approximate the width of a minimum $k$-section within a factor of $n^c$ for any $c < 1$, even when restricted to trees with bounded diameter [3]. In this section, we generalize Theorem 1.1 from bisection to $k$-section using similar ideas as in our generalization of (1) for $k$-section in trees, see [5].

**Theorem 1.3** Every graph $G$ on $n$ vertices that allows a tree decomposition $(T, X)$ of width $t - 1$ satisfies

$$\text{MinSec}(k, G) \leq (k - 1) \frac{t\Delta(G)}{2} \left( \left(\log_2 \frac{1}{r(T, X)}\right)^2 + 11 \log_2 \frac{1}{r(T, X)} + 24 \right).$$

If $V(G) = [n]$ and the tree decomposition $(T, X)$ is provided, a $k$-section with these properties can be computed in $O(k\|T, X\|)$ time.

Note that, if $k \geq n$, then any graph on $n$ vertices has only one $k$-section. Therefore, we can assume without loss of generality that $k < n$ and hence the running time in Theorem 1.3 is always polynomial in the input length. Furthermore, for connected graphs $G$ on $n$ vertices with bounded maximum degree that allow a path-like tree decomposition $(T, X)$ of bounded width $t$, the factor $t\Delta(G)((\log_2(1/r(T, X)))^2 + 11 \log_2(1/r(T, X)) + 24)$ becomes constant. As, for $k < n$, every $k$-section of a connected graph has width at least $k - 1$, the algorithm in Theorem 1.1 achieves a constant factor approximation for $\text{MinSec}(k, G)$ for this class of graphs.
Although the statements look similar, it is not straightforward to generalize Theorem 1.1 to Theorem 1.3, even for $k = 4$. For an arbitrary graph $G$, the natural approach of constructing a 4-section by first constructing a bisection $(V_1, V_2)$ and then constructing one bisection in $G[V_1]$ and one in $G[V_2]$ can give a 4-section far from optimal, even when a minimum bisection is used in each step [10]. This applies similarly to the setting that we are considering here. For instance, consider the graph $G$ obtained from a perfect ternary tree on $n/2$ vertices and a cycle on $n/2$ vertices by adding an edge between a vertex in the cycle and the root of the tree. Then, a minimum bisection $(V_1, V_2)$ in $G$ puts all vertices of the ternary tree in the set $V_1$ and all vertices of the cycle in the set $V_2$, or vice versa. Also the algorithm in Theorem 1.1 can produce the bisection $(V_1, V_2)$, when applied with the normal tree decomposition $(T, \mathcal{X})$ of $G$ of width 2. Now, in the next step of constructing a 4-section of $G$, a bisection in a perfect ternary tree is needed, which has width $\Omega(\log n)$ and therefore the recursively constructed 4-section has width $\Omega(\log n)$. However, Theorem 1.3 promises a 4-section of constant width for $G$ as $r(T, \mathcal{X}) \geq \frac{1}{2}$.

2 Proof Sketch for Theorem 1.1

The proof of Theorem 1.1 recursively builds the set $V_1$ and therefore we consider a more general version, where a graph’s vertex set is partitioned into two sets $V_1$ and $V_2$ such that $V_1$ contains a certain number of vertices. The following lemma is the heart of the proof for Theorem 1.1.

Lemma 2.1 Let $G$ be a graph on $n$ vertices, and let $(T, \mathcal{X})$ be a tree decomposition of $G$ of width $t - 1$. For every $m \in [n]$, the vertex set of $G$ can be partitioned into three pieces $V_1, V_2, Z$ such that one of the following holds:

(i) $|V_1| = m$, $Z = \emptyset$, and $e(V_1, V_2) \leq 2t\Delta(G)$, or

(ii) $|V_1| \leq m \leq |V_1| + |Z|$, $0 < |Z| \leq \frac{1}{2}n$, $e(V_1, V_2, Z) \leq \log_2 \left(\frac{16}{r(T, \mathcal{X})}\right) t\Delta(G)$, and there is a tree decomposition $(T', \mathcal{X}')$ for $G[Z]$ of width at most $t - 1$ with $r(T', \mathcal{X}') \geq 2r(T, \mathcal{X})$.

The last lemma states that we can either find a partition into two sets $V_1$ and $V_2$ of sizes exactly $m$ and $n - m$, or there is a partition with an additional set $Z$, such that $|V_1| \leq m \leq |V_1| + |Z|$ and $r(T', \mathcal{X}') \geq 2r(T, \mathcal{X})$. Hence, applying Lemma 2.1 with parameter $m' = m - |V_1|$ recursively to $G' = G[Z]$ and $(T', \mathcal{X}')$, the relative weight of a heaviest path can be doubled in each round, until it exceeds $\frac{1}{2}$ and Option (ii) in Lemma 2.1 becomes infeasible, which will then prove the existence of the cut in Theorem 1.1.
Concerning the algorithmic aspects in Theorem 1.1, a heaviest path in $(T, \mathcal{X})$ can be computed in time proportional to $||(T, \mathcal{X})||$ by dynamic programming. Furthermore, there is an algorithmic version of Lemma 2.1 where a path $P \subseteq T$ is considered and, if Option (ii) occurs, then a tree decomposition $(T', \mathcal{X}')$ for $G' = G[Z]$ and a path $P'$ with $w(P', \mathcal{X}')/|V(G')| \geq 2w(P, \mathcal{X})/n$ are computed. Therefore, we do not need to compute a heaviest path for each application of Lemma 2.1. By adjusting computed parameters and leaving $(T', \mathcal{X}')$ implicit, we can ensure the desired running time.

We conclude this section with a few words about the proof of Lemma 2.1. Let $\mathcal{X} = (X^i)_{i \in V(T)}$, consider a heaviest path $P$ in $T$, and denote by $i_0$ and $j_0$ its first and last node. Let $R$ be the union of the bags $X^i$ for all $i$ in $P$. For $i$ in $V(P)$, denote by $T_i$ the component of $T - E(P)$ that contains $i$. We label the vertices of $G$ with $1, 2, \ldots, n$ by traversing $P$ from $i_0$ to $j_0$. When traversing a node $i$ in $P$, the vertices not yet labeled in $X^i$ receive consecutive labels and the vertices not yet labeled in $X^i$ receive the largest labels among them. Let $R_i \subseteq R$ and $S_i \subseteq V(G) \setminus R$ be the sets of vertices labeled when traversing $i$. We identify the vertices of $G$ with their labels and define $f(x) = x + m$ cyclically for all $x \in V(G)$. Using properties of tree decompositions, it is easy to show the following proposition, where $E_G(i)$ denotes the set of edges of $G$ that are incident with some vertex in $X^i$ for $i$ in $T$.

**Proposition 2.2** For every $i$ in $P$, in the graph $G - E_G(i)$, every vertex in $R_i$ is isolated and there are no edges between the following three sets: the set $S_i$, the union of $R_j \cup S_j$ over all $j \neq i$ that are between $i_0$ and $i$ on $P$, and the union of $R_j \cup S_j$ over all remaining $j \neq i$ in $P$.

Using this, it is easy to see that, if there is a vertex $x \in R$ with $f(x) \in R$, then the cut $(V_1, V_2)$ with $V_1 = \{x + 1, \ldots, x + m\}$ satisfies Option (i) in Lemma 2.1. Otherwise, we can show that there is a node $i$ in $P$ such that (nearly) all vertices that are mapped into the set $S_i$ by $f$ form a set $Z$ with the property needed for Option (ii) in Lemma 2.1 when using the tree decomposition obtained from $(T, \mathcal{X})$ by deleting the vertices not in $Z$ from the bags. This set $Z$ will contain a vertex $x$ such that $f(x)$ is in $S_i$. Now, the cut $(W'_1, W'_2)$ with $W'_1 = \{x + 1, \ldots, x + m\}$ might cut too many edges, but we can find a subset $W_1 \subseteq W'_1$ such that $W_1 \cap Z = \emptyset$, $|W_1 \cup S_i| \geq m$, and $(W_1, V(G) \setminus W_1)$ cuts at most $2t\Delta(G)$ edges by Proposition 2.2, see also Figure 1a). Then, we can apply Lemma 1.2 to the subgraph of $G$ induced by $S_i$ to obtain a set $\tilde{W}_1$ such that $(\tilde{W}_1, S_i \setminus \tilde{W}_1)$ cuts only few edges in $G[S_i]$ and the set $V_1 = W_1 \cup \tilde{W}_1$ has the desired property for Option (ii) in Lemma 2.1.
3 Proof Sketch for Theorem 1.3

The main idea for the proof of Theorem 1.3 is to find a cut \((V_1, V_2)\) in \(G\) with \(|V_1| = m\) for a parameter \(m\) and the additional property that \(G[V_2]\) allows a tree decomposition \((T', X')\) of width at most \(t - 1\) with \(r(T', X') \geq r(T, X)\). Finding such a cut \(k - 1\) times then produces the desired \(k\)-section. Let us now sketch how to find such a cut \((V_1, V_2)\).

Consider a heaviest path \(P\) in \((T, X)\) and let \(r := r(T, X)\). Define the sets \(R, R_i, \) and \(S_i\) for \(i\) in \(P\) as in Section 2, and consider the same labeling of the vertices of \(G\). For \(x, y \in V(G)\), we define the \(R\)-distance as the number of vertices \(v \in R \setminus \{y\}\) that are between \(x\) and \(y\) in the cyclical labeling. Using that \(|d_R(x, y) - d_R(x + 1, y + 1)| \leq 1\) for all \(x, y\) in \(G\) and an averaging argument, we can show that there is a vertex \(v\) in \(G\) with \(d_R(v, v + m) = \lfloor rm \rfloor\). Without loss of generality we may assume that \(v \in R\) or \(v + m \in R\), because otherwise we can increase \(v\) until this is satisfied. Let \(M\) be the set of vertices \(u \neq v + m\) in \(G\) that are between \(v\) and \(v + m\) in the cyclical labeling. If \(v \in R\) and \(v + m \in R\), then the cut \((M, V(G) \setminus M)\) has the desired properties by Proposition 2.2 and because exactly \(\lfloor rm \rfloor\) vertices from \(R\) are in \(M\). Otherwise, the cut \((M, V(G) \setminus M)\) might cut too many edges. Assume that \(v \in R\) and \(v + m \notin R\); the other case is similar. Let \(i\) be the node in \(P\) with \(v + m \in S_i\). By applying Lemma 1.2 to \(G[S_i]\), we can partition the set \(S_i\) into \(W_1\) and \(W_2\) by cutting only few edges and such that \(\tilde{V} := (M \setminus S_i) \cup W_1\) satisfies \(m \leq |\tilde{V}| \leq 2m\), see also Figure 1b). Furthermore, the set \(\tilde{V}\) contains \(\lfloor rm \rfloor\) vertices from \(R\). On one hand, this will ensure...
that there is a tree decomposition $(\tilde{T}, \tilde{X})$ of $G[\tilde{V}]$ with $r(\tilde{T}, \tilde{X}) \geq r/2$ and hence, we can use Theorem 1.1 to cut off $m$ vertices from $G[\tilde{V}]$ for the set $V_1$ without cutting too many edges. On the other hand, there are at most $\lfloor rm \rfloor$ vertices from $R$ in $V_1$ and therefore the tree decomposition $(T', X')$ obtained from $(T, X)$ by deleting the vertices in $V_1$ satisfies $r(T', X') \geq r(T, X)$. The algorithmic ideas for computing the $k$-section are similar to the ones used in Section 2.

References


