Journal of Discrete Algorithms 14 (2012) 136–149

Contents lists available at SciVerse ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

The complexity of finding uniform sparsest cuts in various graph classes [☆]

Paul Bonsma [a,*,1], Hajo Broersma [b,2], Viresh Patel [b,2], Artem Pyatkin [b,2,3]

[a] *Humboldt Universität zu Berlin, Computer Science Department, Unter den Linden 6, 10099 Berlin, Germany*
[b] *School of Engineering and Computing Sciences, Durham University, Science Laboratories, South Road, Durham DH1 3LE, UK*

A R T I C L E   I N F O

Article history:
Available online 7 December 2011

Keywords:
Sparsest cut
Parameterized complexity
Treewidth
Clique-width
Unit interval graph

A B S T R A C T

Given an undirected graph $G = (V, E)$, the (uniform, unweighted) sparsest cut problem is to find a vertex subset $S \subset V$ minimizing $|E(S, \bar{S})|/(|S||\bar{S}|)$. We show that this problem is NP-complete, and give polynomial time algorithms for various graph classes. In particular, we show that the sparsest cut problem can be solved in linear time for unit interval graphs, and in cubic time for graphs of bounded treewidth. For cactus graphs and outerplanar graphs this can be improved to linear time and quadratic time, respectively. For graphs of clique-width k for which a short decomposition is given, we show that the problem can be solved in time $O(n^{2k+1})$, where n is the number of vertices in the input graph. We also establish that a running time of the form $n^{O(k)}$ is optimal in this case, assuming that the Exponential Time Hypothesis holds.

1. Introduction

The sparsest cut of a graph is a measure of its expansion and is closely related to other graph expansion measures including edge expansion, vertex expansion, and conductance [4]; graph expansion is well studied and plays an important role in discrete mathematics and theoretical computer science. The algorithmic solution of problems such as the sparsest cut problem is useful in the design of divide-and-conquer algorithms for many other problems [23], including cluster analysis, VLSI circuit layout, and so on.

We now define the different variants of the sparsest cut problem that we consider. Given a graph $G = (V, E)$, for disjoint $S, T \subseteq V$, we write $E_G(S, T)$ for the set of all edges of G having one end in S and the other in T. We write $\bar{S} = V \setminus S$. Any set of edges of the form $E_G(S, \bar{S})$ with $S \neq \emptyset$, $S \neq V$ is called a *cut* of G. Define the *density* of the cut $E_G(S, \bar{S})$ to be

$$d_G(S, \bar{S}) = \frac{|E_G(S, \bar{S})|}{|S||\bar{S}|}.$$

We omit subscripts when the graph is clear from the context. The *sparsest cut problem for unweighted graphs* is the following: given a graph $G = (V, E)$ and a positive rational D, determine whether there exists a subset $S \subset V$ such that $d(S, \bar{S}) \leqslant D$.

The problem can be generalized to weighted graphs as follows. Given a positive edge-weighting $w_G : E \to \mathbb{Z}^+$ of G, we define the weight of $E_G(S, T)$ to be

[☆] This paper includes results that have previously been presented in the conference papers (Bonsma, 2007) [9] and (Bonsma et al., 2011) [10].
[*] Corresponding author. Tel.: +49 30 2093 3086; fax: +49 30 2093 3081.
E-mail addresses: bonsma@informatik.hu-berlin.de (P. Bonsma), hajo.broersma@dur.ac.uk (H. Broersma), viresh.patel@dur.ac.uk (V. Patel), artem.pyatkin@dur.ac.uk (A. Pyatkin).
[1] The author is supported by DFG Grant BO 3391/1-1.
[2] The authors are supported by EPSRC Grant EP/F064551/1.
[3] The author is also supported by RFBR (projects 08-01-00516, 09-01-00032, and 08-01-00370).

1570-8667/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.jda.2011.12.008

$$w_G(S,T) = \sum_{e \in E_G(S,T)} w_G(e),$$

and the *density* of the cut $E_G(S,\bar{S})$ to be $d_G(S,\bar{S}) = w_G(S,\bar{S})/(|S||\bar{S}|)$. This definition of density defines the *sparsest cut problem for weighted graphs*.

Note that $E_G(S,\bar{S})$ is a densest cut of G if and only if $E_{\bar{G}}(S,\bar{S})$ is a sparsest cut of the complementary graph $\bar{G}$. (The complement of an edge weighted graph or *weighted complement* is obtained by first introducing edges with zero weight between all non-adjacent vertex pairs, and subsequently changing every edge weight $w(e)$ to $M - w(e)$, where M is the maximum edge weight.) So, the problems of finding sparsest and densest cuts are equivalent, for both weighted and unweighted graphs.

We remark that in the literature this problem is also called the *uniform* sparsest cut problem (see e.g. [23]), to distinguish it from the more general problem where, in addition, an edge weighted *demand graph* H with $V(H) = V(G)$ is given, and the objective is to minimize $w_G(S,\bar{S})/w_H(S,\bar{S})$. We will call this more general problem the *non-uniform sparsest cut problem*. (The sparsest cut problem corresponds to the case where H is an unweighted complete graph.) In this paper, we shall primarily be concerned with the (weighted and unweighted) uniform sparsest cut problem, which we usually refer to simply as the sparsest cut problem.

The unweighted uniform sparsest cut problem has been assumed to be NP-complete by various authors, but no proof is known to us. One of the reasons for this assumption is that these authors either directly or indirectly refer to a paper by Matula and Shahrokhi [24] in which the NP-completeness of the weighted uniform version of the problem has been established. We give an NP-completeness proof for the unweighted case, thereby providing a solid basis for all the papers that build on the assumption that the unweighted version is also NP-complete. As an example, the *Minimum Sum of Squares Clustering* (*MSSC*) *problem* has been assumed to be NP-complete by more than 20 authors (see [1] for further details). The first supposed proof of the NP-completeness of the MSSC problem appeared in [17], but it was shown in [1] that this proof contained an error. Two years later Aloise et al. [2] provided a correct NP-completeness reduction for the MSSC problem; however, they gave a reduction from the unweighted uniform densest cut problem (which is equivalent to the unweighted uniform sparsest cut problem). Here we bridge this small gap in the literature by giving a reduction similar to that in [24] for the unweighted uniform sparsest cut problem.

Approximation algorithms for the sparsest cut problem and other similar problems have received a lot of interest, see e.g. [5,4,23]. The best approximation algorithm known approximates the sparsest cut of an n-vertex graph to within a factor $O(\sqrt{\log n})$ in polynomial time [5,4]. Hardness of approximation results are also known: Ambühl et al. [3] proved that the unweighted sparsest cut problem admits no polynomial-time approximation scheme unless NP-hard problems can be solved in randomized subexponential time. Since this is a stronger assumption than $P \neq NP$, this does not imply NP-completeness however.

In light of the hardness and inapproximability of the sparsest cut problem, it is natural to search for efficient algorithms on restricted graph classes. Surprisingly, not much work has been done in this direction. Matula and Shahrokhi [24] showed that the uniform and non-uniform sparsest cut problem can be solved in linear and quadratic time on (weighted) trees, respectively. They also show that the non-uniform sparsest cut problem can be solved in polynomial time on (weighted) 3-connected planar graphs G when the demand graph H only contains edges between vertices that lie on the outer face of G. In [8] it is shown that sparsest cuts of Cartesian product graphs $G \times H$ can be obtained from sparsest cuts of G and H, which gives polynomial time algorithms for various graph classes. The sparsest cut is also shown to be polynomially solvable for graphs of bounded genus [26]. In this paper, we collect some further results about sparsest cuts on restricted graph classes.

In Section 2 we prove that the sparsest cut problem can be solved in cubic time for graphs of bounded treewidth. A tree decomposition (X,T) of a graph G consists of a tree T, the vertices of which are called *nodes* (of the tree decomposition), and a mapping X of nodes $v \in V(T)$ to subsets $X_v \subseteq V(G)$, satisfying certain properties (see Section 2 for details). The *width* of (X,T) is $\max_{v \in V(T)} |X_v| - 1$.

Theorem 1. *Let G be a weighted graph on n vertices for which a tree decomposition of width k on $O(kn)$ nodes is given. In time $O^*(n^3 2^k)$, a sparsest cut of G can be found.*[4]

For graphs of treewidth bounded by a constant k, a tree decomposition of width at most k and with at most $O(kn)$ nodes can be found in linear time [6]. Thus, Theorem 1 shows that the sparsest cut problem can be solved in cubic time for such graphs. Examples of graph classes with bounded treewidth include cactus graphs, series-parallel graphs, outerplanar graphs and Halin graphs, which have treewidth at most 2, 2, 2 and 3, respectively. In Section 4 we improve the running time of Theorem 1 for cactus graphs and outerplanar graphs: we show how to find sparsest cuts for unweighted cactus graphs and weighted outerplanar graphs in linear and quadratic time, respectively.

Note that many graph problems can be shown to be linear-time solvable on bounded-treewidth graphs by expressing the problem in monadic second order logic [14]. However, it seems that the sparsest cut problem cannot be expressed in

[4] The O^* notation omits polynomial factors, provided that exponential factors in the same variable are present.

this way. Theorem 1 uses a fairly standard dynamic programming approach [7], for computing for every $i \in \{1, \dots, n-1\}$, the minimum number of edges of a cut $E_G(S, \overline{S})$ with $|S| = i$. This large number of cuts that need to be considered leads to the cubic complexity bound. After presenting the conference version of this result [10], we have become aware that a dynamic programming algorithm very similar to ours has first been described by Jansen et al. [22], for the purpose of finding minimum α-balanced cuts or minimum bisections (see Section 3 for details).

In Section 3, we study graphs of bounded clique-width. These are graphs that can be constructed with the following four operations, using vertex labels in $\{1, \dots, k\}$.

- Create a new graph on a single vertex, with vertex label $i \in \{1, \dots, k\}$.
- Take the disjoint union of two vertex disjoint graphs in the class.
- Add all edges between vertices with label i and vertices with label j.
- Change all vertex labels i to j.

A recipe for constructing a graph G using these operations is called a *k-expression for G* and the number of such operations used is called the *length* of the k-expression. For detailed definitions related to clique-width and k expressions, see Section 3. Examples of graph classes with bounded clique-width are co-graphs and distance-hereditary graphs, which have clique-width at most 2 and 3, respectively [21]. We remark that graphs of bounded treewidth have bounded clique-width as well [13]. In Section 3 we prove the following theorem.

Theorem 2. *Given an unweighted n-vertex graph G of clique-width k (with $1 \leqslant k \leqslant n$), together with a k-expression of length l for constructing G, a sparsest cut of G can be computed in time $O(n^{2k}l)$.*

In time $k^{O(1)}l$, a k-expression of length l can be transformed into an equivalent k-expression of length $k^{O(1)}n$ [16], so for fixed k we may assume that $l \in O(n)$ at a negligible cost. In [25], an $O(n^9 \log n)$-time algorithm is given that, for fixed k, either returns a $(2^{3k+2} - 1)$-expression for a graph G, or decides correctly that G has clique-width at least $k+1$. Combined with Theorem 2, this shows that sparsest cuts can be found in polynomial time for any graph class of bounded clique-width. We remark that for graphs of clique-width at most 3, a much faster recognition algorithm is known; these can be recognized in time $O(n^2 m)$, with $m = |E(G)|$ [13].

Theorem 2 (like Theorem 1) is based on a dynamic programming algorithm for computing, for every $i \in \{1, \dots, n-1\}$, the minimum number of edges of a cut $E_G(S, \overline{S})$ with $|S| = i$. Because of this, using similar ideas to those in [22], we are able to solve other cut problems, such as computing minimum α-balanced cuts or minimum bisections (see Section 3 for details).

Comparing Theorem 1 with Theorem 2, we see that in both cases, the problem can be solved in polynomial time for graphs where k is bounded by a constant. However, the running time from Theorem 1 is much more practical since the parameter k does not appear in the exponent of n in the running-time bound. An algorithm with running time of the form $f(k)O(n^c)$, for a constant c and arbitrary computable function $f(k)$, is called a *Fixed Parameter Tractable* (*FPT*) algorithm for parameter k. A natural question is whether the unweighted sparsest cut problem admits an FPT algorithm when choosing the clique-width as parameter k. We show that this is unlikely. More precisely, the running time of Theorem 2 is essentially optimal up to a constant factor in the exponent, assuming that the *Exponential Time Hypothesis* (*ETH*) holds. See [18] for more background on FPT algorithms and ETH.

Theorem 3. *The unweighted sparsest cut problem cannot be solved in time $f(k)n^{o(k)}$ unless ETH fails. Here k is the clique-width of the input graph.*

Similar results for other problems are given in [19].

Finally, in Section 4 we study *unit interval graphs*. A graph is an *interval graph* if its vertices can be mapped to intervals of the real line (not necessarily all of the same length) such that two vertices are adjacent if and only if the intervals intersect. A graph is a *unit interval graph* if it can be represented using unit length intervals in this way. The main result of Section 4 is the following theorem.

Theorem 4. *In linear time, a sparsest cut of an unweighted unit interval graph can be computed.*

The proof demonstrates the novel use of average density arguments to prove results on uniform sparsest cuts, which may be useful for proving other results as well. Sections 2, 3 and 4 can be read independently.

2. Graphs of bounded treewidth

A tuple (X, T) is a *tree decomposition* of a graph $G = (V, E)$ if T is a tree, and $X = \{X_v\colon v \in V(T)\}$ is a family of subsets of V such that:

- $\bigcup_{v \in V(T)} X_v = V$,
- for all $xy \in E$, there exists a $v \in V(T)$ with $x, y \in X_v$, and
- for every $x \in V$, the subgraph $T[\{v \in V(T)\colon\ x \in X_v\}]$ is connected.

The *width* of a tree decomposition (X, T) is $\max_{v \in V(T)} |X_v| - 1$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G. To distinguish between vertices of G and vertices of T, the latter will be called *nodes*. If T is a rooted tree, (X, T) is called a *rooted* tree decomposition. A rooted tree decomposition (X, T) of G is *nice* [7] if every node of T is of one of the following types:

- *leaf nodes* u are leaves of T and have $|X_u| = 1$;
- *introduce nodes* u have one child v with $X_u = X_v \cup \{x\}$ for some $x \in V(G)$;
- *forget nodes* u have one child v with $X_u = X_v \backslash \{x\}$ for some $x \in X_v$;
- *join nodes* u have two children v and z, with $X_u = X_v = X_z$.

For fixed k, it can be decided in linear time if a given graph has treewidth at most k, and in that case, a tree decomposition of width at most k can be found [6]. In fact, it can be checked that in linear time this can be made into a *nice* tree decomposition (X, T) of width at most k, with $|V(T)| \in O(kn)$, where $|V(G)| = n$. For two nodes u and v of a rooted tree T, we write $v \succcurlyeq u$ if u is a predecessor of v, or $u = v$. For a rooted tree decomposition (X, T) of G and a node $v \in V(T)$, we define the subgraph $G(v) = G[\bigcup_{z \succcurlyeq v} X_z]$.

Let (X, T) be a rooted tree decomposition of a graph G on n vertices with edge weights w. Denote the root node by r. For $u \in V(T)$, $S' \subseteq X_u$, $i \leqslant n$, we define $w(u, S', i)$ to be the minimum cut weight over all cuts $E_{G(u)}(S, \bar{S})$ of $G(u)$ that satisfy $S \cap X_u = S'$ and $|S| = i$, if such a cut exists. If $i = 0$ then $w(u, S', i) = 0$, provided that $S' = \emptyset$. If $i = |V(G(u))|$ then $w(u, S', i) = 0$, provided that $S' = X_u$. In all other cases, we define $w(u, S', i) = \infty$. Since $G(r) = G$, the following proposition follows immediately from the above definition.

Proposition 5. *Let (X, T) be a tree decomposition of G with root r. The density of a sparsest cut of G equals the minimum of $\frac{w(r, S', i)}{i(n-i)}$ taken over all $S' \subseteq X_r$ and $1 \leqslant i \leqslant n - 1$.*

So to compute the density of a sparsest cut, we only need to compute the values of $w(r, S', i)$ for all $S' \subseteq X_r$ (possibly empty) and $i \in \{0, \dots, n\}$. The way we compute these values is very similar to the algorithm given in [22], but we include the details for completeness.

Lemma 6. *Let (X, T) be a nice tree decomposition of width k, of a graph G on n vertices. In time $O^*(n^3 2^k)$, the values $w(u, S', i)$ can be computed for all combinations of $u \in V(T)$, $S' \subseteq X_u$ and $i \in \{0, \dots, n\}$.*

Proof. We show how $w(u, S', i)$ can be computed, when all values $w(v, S'', j)$ are known for all children v of u.

First suppose u is a leaf node of T (so $|X_u| = 1$). Then $w(u, S', i) = 0$ if $|S'| = i$, and $w(u, S', i) = \infty$ otherwise. Next, suppose u is an introduce node with a child v, and let $X_u \backslash X_v = \{x\}$. For all $S' \subseteq X_u$:

$$w(u, S', i) = w(v, S' \setminus \{x\}, i - 1) + w_G(\{x\}, X_u \backslash S') \qquad \text{if } x \in S',$$
$$w(u, S', i) = w(v, S', i) + w_G(\{x\}, S') \qquad \text{if } x \notin S'.$$

Suppose u is a forget node with child v, and let $X_v \backslash X_u = \{x\}$. Then

$$w(u, S', i) = \min\{w(v, S', i), w(v, S' \cup \{x\}, i)\}.$$

Finally, suppose u is a join node with children v and z. By the third property in the definition of tree decomposition, we know that $V(G(v)) \cap V(G(z)) = X_u$, so for the cut $E_{G(u)}(S, \bar{S})$ of $G(u)$ that determines $w(u, S', i)$ the set S contains j vertices of $G(v)$ and $i - j + |S'|$ vertices of $G(z)$, for some $j \in \{|S'|, \dots, i\}$. Therefore,

$$w(u, S', i) = \min_{j\colon |S'| \leqslant j \leqslant i} w(v, S', j) + w(z, S', i + |S'| - j) - w_G(S', X_u \backslash S').$$

So all values $w(u, S', i)$ can be computed using the above expressions, if the nodes of T are treated in the proper order. Now we consider the time complexity. As a first step, we build an adjacency matrix for G in time $O(n^2)$, which also contains the weights of the edges. This allows us to determine the existence and weight of a possible edge between two vertices in constant time. For every node, at most $n2^{k+1}$ values need to be computed. In the case of leaf, introduce or forget nodes, computing a value using the above expressions takes time $k^{O(1)}$. (Note that for introduce nodes, the weight of some cut in $G[X_u]$ needs to be computed. To bound the time this takes by a function of k, we have to use the adjacency matrix.) In the case of join nodes, the computation requires time $nk^{O(1)}$. So for every node, the complexity is bounded by $n^2 O^*(2^k)$. Since we can ensure that $|V(T)| \in O(kn)$, computing all values for all nodes of T then requires time $O^*(n^3 2^k)$. □

Lemma 6 and Proposition 5 together prove Theorem 1:

Proof of Theorem 1. First, in linear time, we transform the given tree decomposition into a nice tree decomposition on $O(kn)$ nodes. By Lemma 6, we can then determine all values $w(r, S', i)$ in time $O^*(n^3 2^k)$, where r is the root node in our tree-decomposition. By Proposition 5, the minimum value of $w(r, S', i)/i(n-i)$ over all relevant S' and i gives the density of a sparsest cut of G. Computing this minimum takes time $O(n2^k)$. □

Note that we can not only compute the density of a sparsest cut, but also construct one with the same time complexity.

3. Graphs of bounded clique-width

In this section we consider graphs of clique-width at most k, as defined in [16]. We first give precise definitions. A *k-labeled graph* is a tuple $G_L = (V_1, \ldots, V_k, E)$ of sets such that the sets $V_1, \ldots, V_k$ are pairwise disjoint, and $G = (V_1 \cup \cdots \cup V_k, E)$ is a graph. G_L is called a *k-labeling* of G.

- For $i \in \{1, \ldots, k\}$, $i(v)$ denotes $(V_1, \ldots, V_k, \emptyset)$ with $V_i = \{v\}$ and $V_j = \emptyset$ for all $j \neq i$.
- If $G = (V_1, \ldots, V_k, E)$ and $G' = (V'_1, \ldots, V'_k, E')$ are k-labeled graphs with $(V_1 \cup \cdots \cup V_k) \cap (V'_1 \cup \cdots \cup V'_k) = \emptyset$, then $G \oplus G'$ is defined as $(V_1 \cup V'_1, \ldots, V_k \cup V'_k, E \cup E')$.
- If $G = (V_1, \ldots, V_k, E)$ is a k-labeled graph, then $\eta_{i,j}(G)$ is defined as $(V_1, \ldots, V_k, E \cup E_{ij})$, with $E_{ij} = \{uv\colon\ u \in V_i, v \in V_j\}$.
- If $G = (V_1, \ldots, V_k, E)$ is a k-labeled graph, then $\rho_{i\to j}(G)$ is defined as $(V'_1, \ldots, V'_k, E)$, with $V'_i = \emptyset$, $V'_j = V_i \cup V_j$ and $V'_l = V_l$ for all $l \notin \{i, j\}$.

Observe that $G \oplus G'$, $\eta_{i,j}(G)$ and $\rho_{i\to j}(G)$ are all k-labeled graphs again. A *k-expression* is an algebraic expression using the above operations, where only labels in $\{1, \ldots, k\}$ are used. For instance, the following expression ϕ is a 3-expression, which constructs a path on four vertices u, v, w and x:

$$\phi = \eta_{1,3}\big(\rho_{1\to 2}\big(\eta_{2,3}\big(\eta_{1,2}\big(1(u) \oplus 2(v)\big) \oplus 3(w)\big)\big) \oplus 1(x)\big).$$

We write $G_L := \phi$ to denote that evaluating the k-expression ϕ yields the k-labeled graph G_L. The *clique-width* of a graph G is the minimum k for which a k-expression ϕ exists such that $G_L := \phi$ is a k-labeling of G. The above 3-expression shows that the path P_4 has clique-width at most 3. It is well known and can easily be checked that its clique-width is exactly 3. The *length* of a k-expression is the total number of operations in the expression. The above expression uses the operation $i(v)$ four times, $\oplus$ three times, $\eta_{i,j}$ three times, and $\rho_{i\to j}$ once, so its length is 11.

3.1. The algorithm

We now set about giving the various short lemmas necessary for our dynamic programming algorithm. A k-expression ϕ is called *η-minimal* if removing any $\eta_{i,j}$-operation in ϕ results in a different k-labeled graph.

Proposition 7. *Let ϕ be a k-expression, and suppose $\eta_{x,y}(\phi)$ is an η-minimal k-expression. If $G_L := \phi$ then $E_{G_L}(V_x, V_y) = \emptyset$.*

Proof. Note that the operation $\eta_{i,j}$ is the only operation that adds edges. Let $M = E_{ij}$ be the set of edges added by an occurrence of operation $\eta_{i,j}$ in the k-expression ϕ. Observe that all four operations preserve the following property, where $G = (V_1, \ldots, V_k, E)$ is the resulting graph:

- either there are distinct $a, b \in \{1, \ldots, k\}$ such that $M \subseteq E_G(V_a, V_b)$, or
- there is an $a \in \{1, \ldots, k\}$ such that $M \subseteq G[V_a]$.

In particular, this property holds for $G_L := \phi$. Suppose now that $M' = E_{G_L}(V_x, V_y) \neq \emptyset$. Then by the above property, some operation $\eta_{i,j}$ occurs in ϕ that introduces an edge set $E_{ij} \subseteq M'$. Since $\eta_{x,y}$ adds all edges between V_x and V_y anyway, this operation $\eta_{i,j}$ can be removed from ϕ without affecting the resulting graph, contradicting that $\eta_{x,y}(\phi)$ is η-minimal. □

Proposition 8. *Let ϕ be a k-expression of length l. In time $O(k^2 l)$, an equivalent η-minimal k-expression can be constructed.*

Proof. (*Sketch.*) A k-expression can be transformed in linear time to a rooted binary tree which can be parsed in the usual (post order) way. For all distinct pairs $x, y \in \{1, \ldots, k\}$, we maintain a list $L_{x,y}$ of pointers to the $\eta_{i,j}$-operations (tree nodes) that have introduced edges that are currently between the sets V_x and V_y. Observe that for every operation, these lists can be updated in time $O(k^2)$. The previous proof shows that whenever an $\eta_{i,j}$-operation is applied and the list $L_{i,j}$ is non-empty, all nodes in $L_{i,j}$ are redundant and may be suppressed. Computing the lists $L_{i,j}$ for all nodes takes time $O(k^2 l)$, and suppressing all redundant tree nodes takes time $O(l)$. The resulting tree can be made into an η-minimal k-expression in time $O(l)$. □

Let $G_L = G = (V_1, \ldots, V_k, E)$ be a k-labeling of $G = (V, E)$. A tuple $(\sigma_1, \ldots, \sigma_k)$ of natural numbers is called *relevant* (for G_L) if $\sigma_i \leqslant |V_i|$ for all i. For all relevant tuples, we define $m_{G_L}(\sigma_1, \ldots, \sigma_k)$ to be the minimum number of edges $|E_G(S, \bar{S})|$ over all $S \subseteq V$ with $|S \cap V_i| = \sigma_i$ for all i. The following proposition follows easily from the definitions.

Proposition 9. *Let G_L be a k-labeling of an unweighted graph $G = (V, E)$. The density of a sparsest cut of G equals the minimum of*

$$\frac{m_{G_L}(\sigma_1, \ldots, \sigma_k)}{(\sum_{i=1}^k \sigma_i) \cdot (|V| - \sum_{i=1}^k \sigma_i)},$$

taken over all relevant tuples $(\sigma_1, \ldots, \sigma_k)$ with $1 \leqslant \sum_{i=1}^k \sigma_i \leqslant |V| - 1$.

Lemma 10. *Let ϕ be an η-minimal k-expression of length l and let $H := \phi$ be a graph on n vertices. In time $O(n^{2k}l)$, the values $m_H(\sigma_1, \ldots, \sigma_k)$ can be computed for all relevant tuples $(\sigma_1, \ldots, \sigma_k)$.*

Proof. First suppose $H := i(v)$. Then clearly $m_H(\sigma_1, \ldots, \sigma_k) = 0$ if $\sigma_i \in \{0, 1\}$ and $\sigma_j = 0$ for all $j \neq i$, and these are the only two relevant tuples. Hence in constant time, all necessary values $m_H(\sigma_1, \ldots, \sigma_k)$ can be computed.

Now suppose $H := G \oplus G'$. Then observe that

$$m_H(\sigma_1, \ldots, \sigma_k) = \min_{\forall i:\ 0 \leqslant \tau_i \leqslant \sigma_i} \bigl(m_G(\tau_1, \ldots, \tau_k) + m_{G'}(\sigma_1 - \tau_1, \ldots, \sigma_k - \tau_k)\bigr),$$

where the minimum is taken over all combinations $(\tau_1, \ldots, \tau_k)$ with $0 \leqslant \tau_i \leqslant \sigma_i$ for all i that yield relevant combinations for both G and G'. There are less than n^k such combinations, so computing $m_H(\sigma_1, \ldots, \sigma_k)$ can be done in time $O(n^k)$. Less than n^k relevant tuples exist for H, so computing these values for all relevant tuples takes time at most $O(n^{2k})$, assuming that the values for G and G' are known.

Next, suppose $H := \eta_{i,j}(G)$ and $G = (V_1, \ldots, V_k, E)$. Since we only consider *minimal* k-expressions, we may assume that $E_G(V_i, V_j) = \emptyset$ (Proposition 7). In that case, observe that

$$m_H(\sigma_1, \ldots, \sigma_k) = m_G(\sigma_1, \ldots, \sigma_k) + \sigma_i\bigl(|V_j| - \sigma_j\bigr) + \sigma_j\bigl(|V_i| - \sigma_i\bigr).$$

This shows that $m_H(\sigma_1, \ldots, \sigma_k)$ can be computed in constant time (assuming that $|V_i|$ and $|V_j|$ can be evaluated in constant time). Computing the values for all relevant combinations can therefore be done in time $O(n^k)$.

Finally, suppose $H := \rho_{i \to j}(G)$. Then, relevant combinations $(\sigma_1, \ldots, \sigma_k)$ have $\sigma_i = 0$. For these, note that

$$m_H(\sigma_1, \ldots, \sigma_k) = \min_{\forall l \neq i,j:\ \tau_l = \sigma_l,\ \tau_i + \tau_j = \sigma_j} m_G(\tau_1, \ldots, \tau_k),$$

where the minimum is taken over all relevant combinations $(\tau_1, \ldots, \tau_k)$ for G with $\tau_i + \tau_j = \sigma_j$, and $\tau_l = \sigma_l$ for $l \notin \{i, j\}$. There are at most n such combinations, so computing these values for all $O(n^{k-1})$ relevant combinations can be done in time $O(n^k)$.

Summarizing, we have shown for all four operations that the desired values can be computed in time $O(n^{2k})$, assuming that all values are known for the operand(s), and that all cardinalities $|V_i|$ can be evaluated in constant time. A simple structural induction proof then gives the stated complexity bound of $O(n^{2k}l)$. □

We can now prove Theorem 2.

Proof of Theorem 2. Let G be an unweighted graph on n vertices, and let ϕ be a k-expression of length l such that $G_L := \phi$ is a k-labeling of G, with $1 \leqslant k \leqslant n$. First, we construct an equivalent η-minimal k-expression in time $O(k^2 l) \subseteq O(n^{2k}l)$ (Proposition 8). By Lemma 10, we can then determine all values $m_{G_L}(\sigma_1, \ldots, \sigma_k)$ for all relevant tuples $(\sigma_1, \ldots, \sigma_k)$ in time $O(n^{2k}l)$. By Proposition 9, the density of a sparsest cut of G is obtained by finding the minimum of

$$\frac{m_{G_L}(\sigma_1, \ldots, \sigma_k)}{(\sum_{i=1}^k \sigma_i) \cdot (|V| - \sum_{i=1}^k \sigma_i)},$$

taken over all relevant tuples $(\sigma_1, \ldots, \sigma_k)$; this incurs a time cost of $O(n^k)$. Thus the total running time is $O(n^{2k}l)$. □

By inspecting the values $m_{G_L}(\sigma_1, \ldots, \sigma_k)$ we can solve other important cut problems as well. A cut $E_G(S, \bar{S})$ is an *α-balanced cut* if $\min\{|S|, |\bar{S}|\} \geqslant \alpha|V(G)|$, for $0 \leqslant \alpha \leqslant \frac{1}{2}$. In particular, if $\alpha = \frac{1}{2}$, it is a *bisection*.

Theorem 11. *Let G be an unweighted n-vertex graph of clique-width k, for which a k-expression ϕ of length l is given. For any $\alpha \leqslant \frac{1}{2}$, in time $O(n^{2k}l)$, a minimum α-balanced cut of G can be found.*

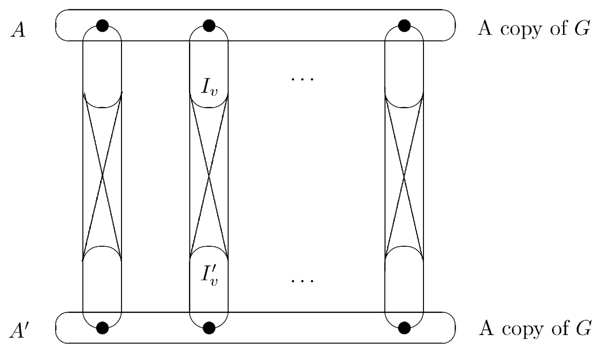

Fig. 1. Graph G'.

3.2. The hardness result

In this section, we show that the dynamic programming algorithm from the last subsection for determining the sparsest cut of n-vertex graphs of clique-width at most k, which has a running time of $n^{O(k)}$, is optimal up to a linear factor in the exponent of n, subject to the Exponential Time Hypothesis. The Exponential Time Hypothesis (ETH) is the conjecture that the 3-satisfiability problem cannot be solved in time $2^{o(n)}$, where n is the number of variables. See [18] for more information.

In [19], Theorem 3.1, it is shown that solving the max cut problem for graphs of clique-width k cannot be done in time $f(k)n^{o(k)}$ (where f is a function of k only) unless ETH fails. We state the theorem formally so we can refer to it later.

Theorem 12. *(See [19].) The max cut problem cannot be solved in time $f(k)n^{o(k)}$ unless ETH fails. Here k is the clique-width of the input graph.*

Recall that for the max cut problem, we are given an instance of a graph G and a positive integer r and wish to determine whether there exists a cut $E(S,\bar{S})$ of G satisfying $|E(S,\bar{S})| \geqslant r$.

We shall give a polynomial-time reduction from an instance (G,r) of the max cut problem to an instance (G^*,r') of the sparsest cut problem, but with the added constraint that the clique-width of G^* is linear in the clique-width of G. Thus if we can solve the sparsest cut problem for n-vertex graphs of clique-width at most k in time $f(k)n^{o(k)}$, then we can also solve the max cut problem for n-vertex graphs of clique-width at most k in time $g(k)n^{o(k)}$, contradicting Theorem 12 (subject to ETH).

We start by describing the reduction. We shall then prove its correctness and show that the reduction increases clique-width only linearly. The reduction we give is similar to that given in [24] for weighted graphs.

Reduction. Let the graph $G=(V,E)$ and the positive integer r be an instance of max cut. Let $V=\{v_1,\dots,v_n\}$ and let m be the number of edges in G. Construct the graph G' in the following way.

For each $v \in V$ we have two sets I_v and I'_v of vertices in G', each of size $M=n^2$. Thus, G' has $2nM$ vertices. For each $v \in V$, connect each vertex in I_v to each vertex in I'_v. Pick one distinguished vertex from each I_v to form a set A of n vertices, and pick one distinguished vertex from each I'_v to form a set A' of n vertices. Insert edges in A and A' to create two copies of G. The resulting graph is G' (see Fig. 1). Note that the degree of every vertex in G' is equal to M plus possibly the degree of the corresponding vertex in G.

Let G^* be the complementary graph of G', and set

$$r' = 1 - \frac{nM^2 + 2r}{(Mn)^2}.$$

We consider the instance (G^*,r') of the sparsest cut problem. Clearly we can construct (G^*,r') from (G,r) in polynomial time. Lemma 14 establishes the correctness of this reduction, but first we need the following easy fact.

Proposition 13. *The maximum cut of the complete bipartite graph $K_{n,n}$ with parts A and B is $E(A,B)$ with cardinality n^2. All other cuts have cardinality at most n^2-n.*

Proof. The first part is trivial. For the second part, let $S \subset A \cup B$ such that $S \neq A$ and $S \neq B$. Let $a=|S \cap A|$ and $b=|S \cap B|$. Then

$$\left|E(S,\bar{S})\right| = a(n-b) + b(n-a) = (a+b)n - 2ab.$$

If $|S| < |\bar{S}|$, then $(a+b) \leqslant n-1$ and $|E(S,\bar{S})| \leqslant n^2 - n$. If $|S| = |\bar{S}|$, then $a+b=n$, but $a \neq n$ and $b \neq n$, so that $ab \geqslant n-1$. Then we have $|E(S,\bar{S})| \leqslant n^2 - 2(n-1) \leqslant n^2 - n$ for $n \geqslant 2$. The case $n = 1$ is trivial. □

Lemma 14. *A graph G has a cut of cardinality at least r if and only if the graph G^* has a cut of density at most r'.*

Proof. We observe that if a cut of a graph has density d then the corresponding cut in the complementary graph has density $1-d$. Therefore, it is sufficient for us to prove that G has a cut of cardinality r if and only if G' has a cut of density at least $1 - r = (nM^2 + 2r)/(Mn)^2$.

First of all we show that if $E_{G'}(S,\bar{S})$ is a densest cut of G' then $|S| = Mn$. Indeed, assume without loss of generality that $y = |S| < |\bar{S}|$, so $y < nM = n^3$. We define

$$X = \bigcup_{v \in V} I_v \quad \text{and} \quad X' = \bigcup_{v \in V} I'_v.$$

We bound $d_{G'}(S,\bar{S})$:

$$\frac{My + n(n-1)}{y(2nM - y)} \geqslant d(S,\bar{S}) \geqslant d\big(X, X'\big) = nM^2/(nM)^2 = 1/n.$$

Rearranging, we obtain $n^2(n-1) \geqslant nMy - y^2 = y(n^3 - y)$, which implies $n^3 - 1 \geqslant y(n^3 - y)$, a contradiction (since $1 \leqslant y \leqslant n^3 - 1$).

Now assume that for some vertex v of G both $S \cap I_v$ and $S \cap I'_v$ are nonempty. Then the cut has at most $n(n-1)$ edges within X and X', at most $M^2(n-1)$ edges between $X \setminus I_v$ and $X' \setminus I'_v$ and at most $M^2 - M$ edges between I_v and I'_v (by Proposition 13). Since $M = n^2$, we have

$$d_{G'}(S,\bar{S}) \leqslant \frac{n(n-1) + M^2(n-1) + M^2 - M}{(Mn)^2} = \frac{M^2 n - n}{(Mn)^2} < \frac{1}{n},$$

a contradiction since $d(X,X') = 1/n$. So, without loss of generality for every vertex v of G, either $S \cap (I_v \cup I'_v) = I_v$ or $S \cap (I_v \cup I'_v) = I'_v$. Let $T = \{v \in V \mid I_v \subseteq S\}$. Then clearly

$$d_{G'}(S,\bar{S}) = \frac{M^2 n + 2|E_G(T,\bar{T})|}{(Mn)^2}$$

and so $d_{G'}(S,\bar{S}) \geqslant (nM^2 + 2r)/(Mn)^2$ if and only if $|E_G(T,\bar{T})| \geqslant r$. Thus Lemma 14 is proved. □

We note that Lemma 14 gives an NP-completeness proof for the unweighted sparsest cut problem, since the max cut problem is NP-complete [20].

Theorem 15. *The (uniform) unweighted sparsest cut problem is NP-complete.*

Finally, we prove that clique-width increases only linearly in our reduction.

Lemma 16. *If G has clique-width k then G^* has clique-width at most $4k + 10$.*

Proof. It is shown in [15] that if a graph H has clique-width k, then its complement has clique-width at most $2k$. Therefore, it is sufficient for us to show that if G has clique-width k, then G' has clique-width at most $2k + 5$ (since G^* is the complement of G').

Let ϕ be a k-expression for $G = (V,E)$ that uses vertex labels $1,\dots,k$. Using vertex labels $1,\dots,k,1',\dots,k',a,a',b,b',c$, we can construct a $(2k+5)$-expression ϕ' for G' as follows.

For each $v \in V$ and $i \in \{1,\dots,k\}$, if $i(v)$ occurs in ϕ, it is replaced in ϕ' by

$$\rho_{b\to c}\rho_{b'\to c}\rho_{a\to i}\rho_{a'\to i'}\big(\eta_{aa'}\eta_{bb'}\eta_{ab'}\eta_{a'b}\big(a(v) \oplus b(v_1) \oplus \cdots \oplus b(v_{M-1}) \oplus a'\big(v'\big) \oplus b'\big(v'_1\big) \oplus \cdots \oplus b'\big(v'_{M-1}\big)\big)\big);$$

each occurrence of η_{ij} in ϕ is replaced in ϕ' by $\eta_{i'j'}\eta_{ij}$; and each occurrence of $\rho_{i\to j}$ in ϕ is replaced by $\rho_{i'\to j'}\rho_{i\to j}$ in ϕ'. One can easily check that ϕ' gives a labeled copy of G', but we give a brief description below of how the operations of ϕ' give G'.

In words, whenever a vertex labeled i is introduced in ϕ, we introduce sets of vertices $I_v = \{v, v_1, \dots, v_{M-1}\}$ and $I'_v = \{v', v'_1, \dots, v'_{M-1}\}$ in ϕ' with all edges present between I_v and I'_v, and where v is labeled i, v' is labeled i', and all other vertices of I_v and I'_v are labeled c. Thus we have a distinguished vertex v of I_v labeled i and a distinguished vertex v' of I'_v labeled i'. The two copies of G are obtained by applying each operation of ϕ twice, once to the unprimed labels, and once to the primed labels. □

Theorem 3 now follows.

Proof of Theorem 3. Theorem 3 follows from Lemmas 14 and 16, together with Theorem 12. □

4. Sparsest cuts in unit interval graphs, cactus graphs and outerplanar graphs

In this section we characterize sparsest cuts of unweighted unit interval graphs, and show that they can be found in linear time (Section 4.3). We also give a linear-time algorithm for finding sparsest cuts in unweighted cactus graphs and a quadratic-time algorithm for finding sparsest cuts in weighted outerplanar graphs (Section 4.2). First, we introduce some simple tools that will be used in the subsequent proofs (Section 4.1).

4.1. Average densities

We start by generalizing the definition of density to any pair of disjoint vertex sets: let $S, T \subseteq V(G)$ with $S \cap T = \emptyset$, $S \neq \emptyset$, $T \neq \emptyset$. The *density* between S and T is $d(S,T) = w_G(S,T)/|S||T|$. The following notation is introduced to improve the readability of our proofs. Suppose $G = (V, E)$ is a graph with edge weighting $w : E \to \mathbb{Z}^+$ and which has a sparsest cut $E(S, \bar{S})$, where $d = d(S, \bar{S}) = w(S, \bar{S})/|S||\bar{S}|$. For non-empty $A, B \subseteq V$ with $A \cap B = \emptyset$, the *normalized density* between A and B is defined to be $e(A, B) := d(A, B) - d$. Note that if $A \cup B \neq V(G)$, then $e(A, B)$ may be negative. Densities and normalized densities can be expressed as weighted averages, as we show in the following useful proposition.

Proposition 17. *Suppose $G = (V, E)$ is a graph with edge weighting $w : E \to \mathbb{Z}^+$. If A, B and C are disjoint non-empty subsets of V, then $d(A, B \cup C) = \frac{d(A,B)|B|+d(A,C)|C|}{|B|+|C|}$, and $e(A, B \cup C) = \frac{e(A,B)|B|+e(A,C)|C|}{|B|+|C|}$.*

Proof. First, we observe that

$$d(A, B \cup C) = \frac{w(A, B \cup C)}{|A|(|B|+|C|)} = \frac{w(A,B)+w(A,C)}{|A|(|B|+|C|)} = \frac{\frac{w(A,B)}{|A||B|}|B| + \frac{w(A,C)}{|A||C|}|C|}{|B|+|C|} = \frac{d(A,B)|B|+d(A,C)|C|}{|B|+|C|}.$$

Let d be the sparsest cut density of G. Then we have

$$\frac{e(A,B)|B|+e(A,C)|C|}{|B|+|C|} = \frac{d(A,B)|B|+d(A,C)|C|}{|B|+|C|} - d = e(A, B \cup C). \qquad \square$$

The following simple lemma is the key to our approach for unit interval graphs.

Lemma 18. *Let G be a graph with edge weighting $w : E \to \mathbb{Z}^+$. If $E(A \cup B, C)$ is a sparsest cut of G, with A and B disjoint and non-empty, then $e(A, B) \geqslant 0$. If $e(A, B) = 0$, then $E(A, B \cup C)$ or $E(B, A \cup C)$ is also a sparsest cut of G.*

Proof. Since $E(A \cup B, C)$ is a sparsest cut, we have $e(A \cup B, C) = 0$. Since this is a weighted average of $e(A, C)$ and $e(B, C)$ (Proposition 17), one of $e(A, C)$ and $e(B, C)$ is at most 0 and the other is at least 0. Assume first $e(A, C) \leqslant 0$ and $e(B, C) \geqslant 0$. If $e(A, B) < 0$, then

$$e(A, B \cup C) = \frac{e(A,B)|B|+e(A,C)|C|}{|B|+|C|} < 0,$$

contradicting the fact that $E(A \cup B, C)$ is a sparsest cut. This shows that $e(A, B) \geqslant 0$. If $e(A, B) = 0$, then similarly we get $e(A, B \cup C) \leqslant 0$. So $0 = e(A \cup B, C) \leqslant e(A, B \cup C) \leqslant 0$, and therefore $E(A, B \cup C)$ is also a sparsest cut.

If instead we assume $e(B, C) \leqslant 0$ and $e(A, B) = 0$, then we find in the same way that $E(B, A \cup C)$ is also a sparsest cut. □

Corollary 19. *If $E(S, T)$ is a sparsest cut in a connected graph G, then $G[S]$ and $G[T]$ are connected.*

Proof. Since G is connected and edge weights are strictly positive, the sparsest cut density d of G is strictly positive. Suppose w.l.o.g. that $G[S]$ is not connected, so S can be partitioned into two non-empty sets A and B such that $w_G(A, B) = 0$. Then $e(A, B) < 0$, contradicting Lemma 18. □

Alternatively, the above corollary states that every sparsest cut is an (inclusionwise) minimal edge cut.

4.2. Cactus graphs and outerplanar graphs

Before we present our main result for unit interval graphs, we illustrate applications of Corollary 19 by giving a linear-time algorithm for finding sparsest cuts in unweighted cactus graphs, and a quadratic-time algorithm for finding sparsest cuts in weighted outerplanar graphs. A graph is a *cactus* if it is connected and every edge is part of at most one cycle. In other words, the blocks of a cactus graph are either cycles or K_2 graphs; recall that a *block* of a graph is a maximal subgraph without cut vertices. Cactus graphs have treewidth at most 2, but Theorem 1 only guarantees a cubic-time algorithm.

Theorem 20. *A sparsest cut of an unweighted n-vertex cactus graph $G = (V, E)$ can be computed in time $O(n)$.*

Proof. (*Sketch.*) Note that $|E| \in O(n)$. The blocks of G can be found in linear time [27]. For every block B and every vertex $v \in V(B)$, let $n(B, v)$ be the number of vertices in the component of $G - E(B)$ that contains v. These values can be computed in linear time as follows: first assign $n'(v) := 1$ for every vertex v. Now repeatedly consider a block B that contains at most one cut vertex u of G (such a block always exists). For all $v \in V(B)$ with $v \neq u$, set $n(B, v) = n'(v)$. The correct value of $n(B, u)$ can now be computed from the fact that $\sum_{v \in V(B)} n(B, v) = n$. This yields all values $n(B, v)$ for B. Next, contract B into the single vertex u (the cut vertex), and set $n'(u) := \sum_{v \in V(B)} n'(v)$. Continue this procedure with the next block that contains at most one cut vertex (in the resulting graph), until no blocks remain. This correctly computes the values $n(B, v)$ for all B and all $v \in B$, in time $O(n)$.

Every 2-connected block B of G is a cycle. Label its vertices $v_1, \ldots, v_k$ along the cycle. The edge cut $E(S, \bar{S})$ of smallest density containing only edges of B can be computed as follows in time $O(k)$. By Corollary 19, $S \cap V(B) = \{v_x, v_{x+1}, \ldots, v_y\}$ for some $1 \leqslant x \leqslant y \leqslant k$. We also know that $|E(S, \bar{S})| = 2$, and so

$$d(S, \bar{S}) = \frac{2}{(\sum_{i=x}^{y} n(B, v_i)) \cdot (n - (\sum_{i=x}^{y} n(B, v_i))}. \tag{1}$$

The values of x and y that minimize (1) can be found in at most $2k$ steps as follows. Start with $x = 1$ and $y = 1$; whenever $\sum_{i=x}^{y} n(B, v_i) \leqslant n/2$, increase y by 1, and otherwise increase x by 1, computing the value of $d(S, \bar{S})$ at each stage. Continue until either $y > k$ or $x > y$. It can be verified that the values of (x, y) considered include one that minimizes (1).

All blocks B of G that are not 2-connected correspond to a single edge uv, and the unique cut that only contains uv has density $\frac{1}{n(B,u)n(B,v)}$. Corollary 19 shows that if we consider the densities of cuts of the above types, a sparsest cut is considered (a minimal cut is always contained within a single block). The algorithm takes time $O(n)$. □

With a similar proof to the one above, Theorem 1 can be improved for outerplanar graphs, which have treewidth 2 and generalize cactus graphs. *Outerplanar graphs* are graphs that admit a planar embedding such that every vertex lies on the boundary of the outer (unbounded) face.

Theorem 21. *A sparsest cut of an n-vertex weighted outerplanar graph $G = (V, E)$ can be computed in time $O(n^2)$.*

Proof. (*Sketch.*) For outerplanar graphs, since every vertex lies on the boundary of the outer face, every 2-connected block B consists of a cycle containing all vertices of B, and chords of this cycle (these are exactly the edges that are not incident with the outer face). As in the case of cactus graphs, Corollary 19 implies that there exists a sparsest cut of G that is a minimal edge cut, and hence can only contain edges from a single block. Suppose $E(S, \bar{S})$ is a sparsest cut of G that contains only edges of a 2-connected block B, whose vertices, labeled around the cycle, are $v_1, \ldots, v_k$. Then, again by Corollary 19, $S \cap V(B) = \{v_x, v_{x+1}, \ldots, v_y\}$ for some $1 \leqslant x \leqslant y \leqslant k$. To find a cut of this form of least density, we need to consider $O(k^2)$ combinations of x and y (in contrast to cactus graphs where we needed to consider only $O(k)$ combinations). Given a 2-connected block B, we show how to compute a cut of smallest density contained entirely within B in time $O(k^2)$; the result then follows analogously to the proof of Theorem 20.

First we note that in linear time, an outerplanar embedding of G can be constructed, and the blocks can be identified. This implies that the aforementioned cycle $v_1, \ldots, v_k$ in B can be found in linear time. In addition, using an $O(n^2)$-time preprocessing step, we can compute a weighted adjacency matrix for every block of G. Thus we may assume that for the block B, we can check the existence of an edge $v_i v_j$, and evaluate its edge weight $w(v_i v_j)$ if it exists, in constant time for all i, j.

The values $n(B, v)$ are defined and computed as in the proof of Theorem 20 (we may assume G is connected). For $1 \leqslant x \leqslant y \leqslant k$, define $S_{x,y} = \{v_i\colon x \leqslant i \leqslant y\}$, and $w_B(x, y) = w(S_{x,y}, V(B) \backslash S_{x,y})$. The remaining task is to evaluate, for every $1 \leqslant x \leqslant y \leqslant k$, the density of the cut of G that contains only edges of B, and that separates $S_{x,y}$ from $V(B) \backslash S_{x,y}$. The density of such a cut is

$$\frac{w_B(x, y)}{\left(\sum_{i=x}^{y} n(B, v_i)\right) \cdot \left(n - \left(\sum_{i=x}^{y} n(B, v_i)\right)\right)}. \tag{2}$$

For $i, j \in \{1, \ldots, k\}$, define $p_{i,j} = w(\{v_i\}, S_{1,j})$. (In order to have less cluttered notation, we abuse the definitions a little by allowing that $v_i \in S_{1,j}$, but note that this is not a problem.) Clearly, in time $O(k^2)$ the values $p_{i,j}$ can be computed for

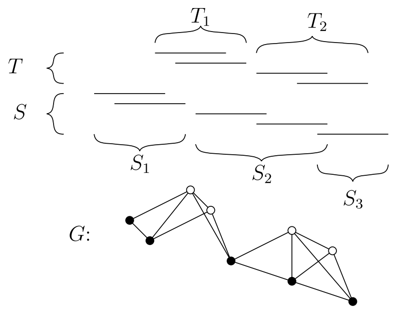

Fig. 2. An I-partition for S and T.

every i and j dynamically, since $p_{i,j} = p_{i,j-1}$ if $v_iv_j \notin E(B)$, and $p_{i,j} = p_{i,j-1} + w(v_iv_j)$ otherwise. Recall that this can be evaluated in constant time. Now observe that for $y > x$,

$$\begin{aligned} w_B(x, y) &= w_B(x, y-1) - w\big(\{v_y\}, S_{x,y-1}\big) + w\big(\{v_y\}, S_{1,x-1}\big) + w\big(\{v_y\}, S_{y+1,k}\big) \\ &= w_B(x, y-1) - \big(w\big(\{v_y\}, S_{1,y-1}\big) - w\big(\{v_y\}, S_{1,x-1}\big)\big) \\ &\quad + w\big(\{v_y\}, S_{1,x-1}\big) + \big(w\big(\{v_y\}, S_{1,k}\big) - w\big(\{v_y\}, S_{1,y}\big)\big) \\ &= w_B(x, y-1) - p_{y,y-1} - p_{y,y} + 2p_{y,x-1} + p_{y,k}. \end{aligned}$$

In addition, $w_B(x,x) = p_{x,k}$. This shows that, when considering the different combinations of x and y in the proper order, the density (2) can be evaluated in constant time.

Thus, for each 2-connected block B, we can evaluate the cut of least density that is contained entirely in B. We can do the same for the other blocks, which are single edges, in the same way as for cactus graphs. At least one of these cuts is a sparsest cut of G, and it is found in $O(n^2)$ time. $\square$

4.3. Unit interval graphs

A graph G is a *unit interval graph* if a function $I : V(G) \to \mathbb{R}$ exists such that $uv \in E(G)$ if and only if $I(u) - 1 \leqslant I(v) \leqslant I(u) + 1$. I is called a *unit interval representation* of G.

In order to give a short expression for the form of a sparsest cut, we introduce the following definition. Let $I : V(G) \to \mathbb{R}$ be a unit interval representation for the graph G. If A and B are disjoint non-empty subsets of $V(G)$, we write $A \prec B$ if for all $u \in A$ and $v \in B$, $I(u) \leqslant I(v)$ holds.

Let G be a unit interval graph with unit interval representation I. We show that G has a sparsest cut $E(S, \bar{S})$ such that $S \prec \bar{S}$. This is done by considering an arbitrary cut $E(S, T)$, and partitioning S and T into $S_1, \ldots, S_k$ resp. $T_1, \ldots, T_l$ with $k - 1 \leqslant l \leqslant k$ such that $S_1 \prec T_1 \prec S_2 \prec T_2 \prec \cdots$. For a given cut $E(S, T)$ and unit interval representation I of G, a partition of S and T into non-empty subsets with this property is called an *I-partition* of S and T. Observe that without loss of generality we can always find an I-partition. See Fig. 2 for an example.

We show that if $k > 1$, then we can reassign these subsets into disjoint non-empty subsets S' and T' with $S' \cup T' = V(G)$, such that $d(S', T') \leqslant d(S, T)$. The property that is stated in the next lemma is the only property of unit interval graphs we will use in the proof of our main result (Theorem 23).

Lemma 22. *Let $I : V(G) \to \mathbb{R}$ be a unit interval representation for the graph G. If $A, B, C \subseteq V(G)$ with $A \prec B \prec C$, then $d(B, A \cup C) \geqslant d(A, C)$.*

Proof. Let $A_1 \subseteq A$ be the vertices in A that have at least one neighbor in C. Similarly, $C_1 \subseteq C$ are the vertices in C that have at least one neighbor in A. Let $\alpha = |A_1|/|A|$, and $\gamma = |C_1|/|C|$.

$$d(A, C) = \frac{|E(A, C)|}{|A||C|} \leqslant \frac{|A_1||C_1|}{|A||C|} = \alpha\gamma \leqslant \min\{\alpha, \gamma\}.$$

Now we will show that $\min\{\alpha, \gamma\}$ is a lower bound for $d(B, A \cup C)$. Suppose $a \in A$ and $c \in C$ are adjacent, so $I(a) + 1 \geqslant I(c)$. Consider $b \in B$. $I(a) \leqslant I(b)$ (since $A \prec B$), and $I(b) \leqslant I(c)$ (since $B \prec C$). It follows that $I(a) + 1 \geqslant I(b)$, so a and b are adjacent, and $I(b) + 1 \geqslant I(c)$, so b and c are adjacent. This proves that if a vertex $a \in A$ has a neighbor in C, then every vertex in B is adjacent to a, and a similar statement holds for vertices in C. Now we write

$$d(B, A \cup C) = \frac{|E(B, A \cup C)|}{|B|(|A| + |C|)} \geqslant \frac{|B|(|A_1| + |C_1|)}{|B|(|A| + |C|)} = \frac{\alpha|A| + \gamma|C|}{|A| + |C|},$$

which is a weighted average of α and γ, so

$$d(B, A \cup C) \geqslant \min\{\alpha, \gamma\}. \qquad \square$$

Theorem 23. *Let $I : V(G) \to \mathbb{R}$ be a unit interval representation for the unweighted graph G. G has a sparsest cut $E(S,T)$ such that $S \prec T$.*

Proof. Consider a sparsest cut $E(S,T)$ and I-partition $\{S_1,\dots,S_k\}$ and $\{T_1,\dots,T_l\}$ of S and T, that has $k+l$ minimum, among all such sparsest cuts and I-partitions. We use the following shorthand notation: $S_{i\dots j} = S_i \cup \dots \cup S_j$, and $T_{i\dots j} = T_i \cup \dots \cup T_j$.

If $k = 1$, then $S = S_1 \prec T_1 = T$, and we have found the desired sparsest cut. Otherwise, we distinguish two cases: $l = k-1$, and $l = k \geqslant 2$.

Case 1. $l = k-1$.

For all $1 \leqslant t \leqslant k-1$, $e(S_{1\dots t}, S_{t+1\dots k}) \geqslant 0$ (Lemma 18). If this is an equality, then $E(S_{1\dots t}, S_{t+1\dots k} \cup T_{1\dots k-1})$ or $E(S_{t+1\dots k}, S_{1\dots t} \cup T_{1\dots k-1})$ is also a sparsest cut (Lemma 18). The first cut has an I-partition with $2t < 2k-1 = k+l$ classes, and the second cut has an I-partition with $2(k-t) < 2k-1 = k+l$ classes, both contradictions with our choice of $E(S,T)$. We conclude that $e(S_{1\dots t}, S_{t+1\dots k}) > 0$ for every $1 \leqslant t \leqslant k-1$.

Since $S_{1\dots t} \prec T_t \prec S_{t+1\dots k}$ and $S_{1\dots t} \cup S_{t+1\dots k} = S$, it follows from Lemma 22 that $e(T_t, S) \geqslant e(S_{1\dots t}, S_{t+1\dots k}) > 0$, for all $1 \leqslant t \leqslant k-1$. Since

$$e(S,T) = \frac{e(T_1,S)|T_1| + \dots + e(T_{k-1},S)|T_{k-1}|}{|T_1| + \dots + |T_{k-1}|},$$

we have $e(S,T) > 0$, a contradiction with the fact that $E(S,T)$ is a sparsest cut. This concludes the case $l = k-1$.

Case 2. $l = k \geqslant 2$.

We again have $e(T_t, S) > 0$ for all $1 \leqslant t \leqslant k-1$ (see the previous case), but it is possible that $e(T_k, S) < 0$, so we cannot immediately obtain a contradiction this way.

First we show that for every $2 \leqslant t \leqslant k$,

$$e(S_t, T) > e(T_{1\dots t-1}, S)\frac{|S|}{|T|}, \tag{3}$$

and for every $1 \leqslant t \leqslant k-1$,

$$e(T_t, S) > e(S_{t+1\dots k}, T)\frac{|T|}{|S|}. \tag{4}$$

For a fixed t with $2 \leqslant t \leqslant k$, we denote $T_L = T_{1\dots t-1}$ and $T_H = T_{t\dots k}$. We showed that $e(T_i, S) > 0$ for all $i < k$, so we have $e(T_L, S) = \alpha > 0$ ($e(T_L,S)$ is a weighted average of $e(T_i,S)$ for $i = 1,\dots,t-1$). Since

$$0 = e(T_L \cup T_H, S) = \frac{\alpha|T_L| + e(T_H,S)|T_H|}{|T_L| + |T_H|},$$

we have $e(T_H, S) = -\alpha\frac{|T_L|}{|T_H|}$. Now we consider the cut $E(T_H, V(G)\backslash T_H)$. Since

$$0 \leqslant e\bigl(T_H, V(G)\backslash T_H\bigr) = \frac{e(T_H,S)|S| + e(T_H,T_L)|T_L|}{|S| + |T_L|},$$

we have $e(T_H, T_L) \geqslant -e(T_H, S)\frac{|S|}{|T_L|} = \alpha\frac{|S|}{|T_H|}$. Finally, using Lemma 22 and $T_L \prec S_t \prec T_H$ we obtain $e(S_t, T) \geqslant e(T_L, T_H) \geqslant \alpha\frac{|S|}{|T_H|} > e(T_L, S)\frac{|S|}{|T|}$. By symmetry (since $l = k$), (4) can be proved the same way.

Using (3) and (4), we now prove by induction on i that for all $1 \leqslant i \leqslant k-1$,

$$e(T_{1\dots i}, S) > e(S_{i+1\dots k}, T)\frac{|T|}{|S|}. \tag{5}$$

If $i = 1$, then (5) is equal to (4) for $t = 1$.

If $i > 1$ then our induction hypothesis is that $e(T_{1\dots i-1}, S) > e(S_{i\dots k}, T)\frac{|T|}{|S|}$. When we combine this with (3) we get

$$e(S_i, T) > e(T_{1\dots i-1}, S)\frac{|S|}{|T|} > e(S_{i\dots k}, T).$$

Since $e(S_{i\dots k}, T)$ is a weighted average of $e(S_i, T)$ and $e(S_{i+1\dots k}, T)$, it follows that

$$e(S_{i\dots k}, T) > e(S_{i+1\dots k}, T).$$

We combine this with the induction hypothesis:

$$e(T_{1\dots i-1}, S) > e(S_{i\dots k}, T)\frac{|T|}{|S|} > e(S_{i+1\dots k}, T)\frac{|T|}{|S|}.$$

From (4) we see that $e(S_{i+1\dots k}, T)\frac{|T|}{|S|}$ is also a lower bound for $e(T_i, S)$. Since $e(T_{1\dots i}, S)$ is a weighted average of $e(T_{1\dots i-1}, S)$ and $e(T_i, S)$, it follows that

$$e(T_{1\dots i}, S) > e(S_{i+1\dots k}, T)\frac{|T|}{|S|},$$

which concludes the induction proof.

Using (5) resp. (3), we obtain a contradiction for the case $l = k$:

$$e(T_{1\dots k-1}, S) > e(S_k, T)\frac{|T|}{|S|} > e(T_{1\dots k-1}, S).$$

We showed that both cases with $k > 1$ lead to a contradiction, so with our choice of S and T, k must be 1, and thus $S \prec T$. □

Unit interval graphs can be recognized in linear time, and a unit interval representation can be found in linear time [12]. It follows that for unit interval graphs, sparsest cuts can be found in linear time:

Proof of Theorem 4. In linear time, compute a unit interval representation I for G, and number the vertices $v_1, \dots, v_n$, according to the linear order given by I. Now we only have to compute and compare the densities of the cuts $E(\{v_1, \dots, v_i\}, \{v_{i+1}, \dots, v_n\})$, for $i = 1, \dots, n-1$ (Theorem 23). Note that the number of edges in the i-th cut can be deduced in time $O(d(v_i))$ from the number of edges in the $(i-1)$-th cut. Therefore the algorithm has complexity $O(|V| + |E|)$. □

Since Lemma 22 is the only property of unit interval graphs that we used to prove Theorem 23, our result actually holds for a larger graph class. Namely the class of graphs which are ‘path-like’ in the following sense: a complete order on $V(G)$ exists, and corresponding partial order $\prec$ subsets of $V(G)$ (deduced from the vertex order the same way as $\prec$ is deduced from I), such that $A \prec B \prec C$ implies $d(B, A \cup C) \geqslant d(A, C)$. Clearly this property does not hold for all graphs (for instance, it can be shown that it does not hold for sufficiently large stars), but there exist graphs that are not unit interval graphs that satisfy this property. Using the appropriate order on the vertices, it holds for instance for graphs obtained from a path by first doubling every edge, and then subdividing every edge with a single degree 2 vertex. (The result is a concatenation of 4-cycles.)

5. Conclusions and discussion

We gave an NP-completeness proof for the unit capacity densest (sparsest) cut problem. We also showed that the weighted sparsest cut problem can be solved in polynomial time for graphs of bounded treewidth. One may ask how far this can be generalized to the non-uniform sparsest cut problem. The algorithm from Section 2 can easily be generalized to give a pseudopolynomial time algorithm in the case where demands are determined by vertex weights $x(v)$ in the following way: the weight of edge uv in the demand graph equals $x(u)x(v)$. However, the algorithm can most likely not be fully generalized; very recently it has been shown that the non-uniform sparsest cut problem is NP-complete for unweighted graphs of treewidth 2, and even for the more restricted class of graphs with pathwidth 2 [11]. In the same paper, the authors give a constant-factor approximation algorithm for the non-uniform sparsest cut problem for graphs of bounded treewidth, using linear programming relaxation techniques. The complexity status of the following problems related to graphs of bounded treewidth remains open:

(1) the case where the input graph G has bounded treewidth, and both G and the demand graph H are unweighted, and
(2) the case where both G and H have bounded treewidth (but possibly both are weighted).

We gave a linear time algorithm for finding sparsest cuts in unweighted unit interval graphs. An obvious question is whether this can be generalized to *all* interval graphs G. The graph in Fig. 3 indicates that Theorem 23 does not generalize in a straightforward way to all interval graphs: The graph shown has in essence a unique interval representation, and it has a unique sparsest cut that is obtained by separating the single vertex ‘in the middle’ from the rest of the vertices. The graph in Fig. 3 has $2n + 2m - 2$ vertices, and consists of two K_n blocks, two K_m blocks and one K_2 block ($m \geqslant 2$, $n \geqslant 2$). Consider

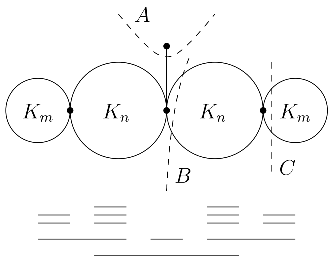

Fig. 3. An interval graph with an unexpected sparsest cut.

the three cuts illustrated in the figure. The first cut has $|E(A,\overline{A})| = 1$, and $|A| = 1$. The second has $|E(B,\overline{B})| = n-1$ and $|B| = m+n-2$. The third has $|E(C,\overline{C})| = m-1$ and $|C| = m-1$. Observe that regardless of the choice of n and m, one of these is a sparsest cut. These cuts have densities resp. $d(A,\overline{A}) = \frac{1}{2n+2m-3} \approx \frac{1}{2(n+m)}$, $d(B,\overline{B}) = \frac{n-1}{(m+n)(m+n-2)} \approx \frac{n}{(n+m)^2}$ and $d(C,\overline{C}) = \frac{m-1}{(m-1)(2n+m-1)} = \frac{1}{2n+m-1} \approx \frac{1}{2n+m}$. When $m \geqslant 3$ and $n \geqslant m+2$, $E(A,\overline{A})$ is the unique sparsest cut, even though this cut does not satisfy any partial order corresponding to a representation.

Acknowledgement

We thank Petr Golovach for his very helpful suggestions.

References

[1] D. Aloise, P. Hansen, On the complexity of minimum sum-of-squares clustering, Cahiers du GERAD, G, 2007-50, 2007, available online at http://www.gerad.ca.
[2] D. Aloise, A. Deshpande, P. Hansen, P. Popat, NP-hardness of Euclidean sum-of-squares clustering, Mach. Learn. 75 (2009) 245–248.
[3] C. Ambühl, M. Mastrolilli, O. Svensson, Inapproximability results for sparsest cut, optimal linear arrangement, and precedence constrained scheduling, in: FOCS 2007, IEEE, 2007, pp. 329–337.
[4] S. Arora, S. Rao, U.V. Vazirani, Expander flows, geometric embeddings and graph partitioning, J. ACM 56 (2) (2009) 1–37.
[5] S. Arora, E. Hazan, S. Kale, $O(\sqrt{\log n})$ approximation to SPARSEST CUT in $\tilde{O}(n^2)$ time, SIAM J. Comput. 39 (5) (2010) 1748–1771.
[6] H.L. Bodlaender, A linear-time algorithm for finding tree-decompositions of small treewidth, SIAM J. Comput. 25 (1996) 1305–1317.
[7] H.L. Bodlaender, Treewidth: algorithmic techniques and results, in: MFCS 1997, in: LNCS, vol. 1295, Springer, Berlin, 1997, pp. 19–36.
[8] P. Bonsma, Sparsest cuts and concurrent flows in product graphs, Discrete Appl. Math. 136 (2–3) (2004) 173–182.
[9] P. Bonsma, Linear time algorithms for finding sparsest cuts in various graph classes, in: CS 2006, Prague, 2006, Electronic Notes in Discrete Mathematics 28 (2007) 265–272.
[10] P. Bonsma, H.J. Broersma, V. Patel, A.V. Pyatkin, The complexity status of problems related to sparsest cuts, in: IWOCA 2010, in: LNCS, vol. 6460, Springer, Berlin, 2011, pp. 125–135.
[11] E. Chlamtac, R. Krauthgamer, P. Raghavendra, Approximating sparsest cuts in graphs of bounded treewidth, in: APPROX and RANDOM 2010, in: LNCS, vol. 6302, Springer, Berlin, 2010, pp. 124–137.
[12] D.G. Corneil, A simple 3-sweep LBFS algorithm for the recognition of unit interval graphs, Discrete Appl. Math. 138 (3) (2004) 371–379.
[13] D.G. Corneil, M. Habib, J. Lanlignel, B. Reed, U. Rotics, Polynomial time recognition of clique-width $\leqslant 3$ graphs (extended abstract), in: LATIN 2000, in: LNCS, vol. 1776, Springer, Berlin, 2000, pp. 126–134.
[14] B. Courcelle, Graph rewriting: An algebraic and logic approach, in: Handbook of Theoretical Computer Science, vol. B, Elsevier, Amsterdam, 1990, pp. 193–242.
[15] B. Courcelle, S. Olariu, Upper bounds to the clique width of graphs, Discrete Appl. Math. 101 (2000) 77–114.
[16] B. Courcelle, J.A. Makowsky, U. Rotics, Linear time solvable optimization problems on graphs of bounded clique-width, Theory Comput. Syst. 33 (2) (2000) 125–150.
[17] P. Drineas, A. Frieze, R. Kannan, S. Vempala, V. Vinay, Clustering large graphs via the singular value decomposition, Mach. Learn. 56 (1–3) (2004) 9–33.
[18] J. Flum, M. Grohe, Parameterized Complexity Theory, Springer-Verlag, Berlin, 2006.
[19] F.V. Fomin, P.A. Golovach, D. Lokshtanov, S. Saurabh, Algorithmic lower bounds for problems parameterized with clique-width, in: SODA 2010, SIAM, Philadelphia, 2010, pp. 493–502.
[20] M.R. Garey, D.S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, New York, 1979.
[21] M.C. Golumbic, U. Rotics, On the clique-width of some perfect graph classes, Internat. J. Found. Comput. Sci. 11 (3) (2000) 423–443.
[22] K. Jansen, M. Karpinski, A. Lingas, E. Seidel, Polynomial time approximation schemes for max-bisection in planar and geometric graphs, in: STACS 2001, in: LNCS, vol. 2010, Springer, Berlin, 2001, pp. 365–375.
[23] F.T. Leighton, S. Rao, Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms, J. ACM 46 (6) (1999) 787–832.
[24] D.W. Matula, F. Shahrokhi, Sparsest cuts and bottlenecks in graphs, Discrete Appl. Math. 27 (1990) 113–123.
[25] S. Oum, P. Seymour, Approximating clique-width and branch-width, J. Combin. Theory Ser. B 96 (4) (2006) 514–528.
[26] V. Patel, Determining edge expansion and other connectivity measures of graphs of bounded genus, in: Algorithms – ESA 2010, in: LNCS, vol. 6346, Springer, Berlin, 2010, pp. 561–572.
[27] R. Tarjan, Depth-first search and linear graph algorithms, SIAM J. Comput. 1 (2) (1972) 146–160.