




MaxCut Above Guarantee

Ivan Bliznets  

Utrecht University, The Netherlands

Vladislav Epifanov 

HSE University, St. Petersburg, Russia

Abstract

In this paper, we study the computational complexity of the Maximum Cut problem parameterized above guarantee. Our main result provides a linear kernel for the Maximum Cut problem in connected graphs parameterized above the spanning tree. This kernel significantly improves the previous $O(k^5)$ kernel given by Madathil, Saurabh, and Zehavi [ToCS 2020]. We also provide subexponential running time algorithms for this problem in special classes of graphs: chordal, split, and co-bipartite. We complete the picture by lower bounds under the assumption of the ETH. Moreover, we initiate a study of the Maximum Cut problem above $\frac{2}{3}|E|$ lower bound in tripartite graphs.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Fixed parameter tractability

Keywords and phrases Tripartite, 3-colorable, chordal, maximum cut, FPT-algorithm, linear kernel

Digital Object Identifier 10.4230/LIPIcs.MFCS.2023.22

Funding Supported by the project CRACKNP that has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 853234).

Acknowledgements We want to thank anonymous reviewers for their suggestions that helped to improve the presentation of the paper.

1 Introduction

The maximum cut problem, MAX-CUT for short, is a well-studied optimization problem in computer science and graph theory. In this problem, we need to split the vertices of a given graph into two parts, such that the sum of weights of edges going between two parts is maximum. The weighted version of the decision problem was one of Karp’s 21 NP-complete problems [14]. However, it is known that even unweighted MAXIMUM CUT is NP-complete [10]. It is easy to see that the problem admits an FPT-algorithm parameterized by the solution size, since the size of solution is relatively large. Specifically, if m is the number of edges in the input graph, then using a greedy argument we can construct a cut of size $\frac{m}{2}$. That is why it is natural to study parameterization above lower bounds. Under such parameterization [13], our goal is to find a solution of size $k + l(I)$ where I is the input instance, $l(I)$ is a polynomial time computable lower bound and k is our parameter. For more details about parameterizations of this type, we refer interested readers to a very nice survey written by Gutin and Mnich [13]. Mahajan and Raman were the first who studied the MAXIMUM CUT problem parameterized above lower bounds [16]. They showed that MAXIMUM CUT parameterized above lower bound $\frac{m}{2}$ admits an FPT-algorithm. Later Crowston et al. [4, 5] presented a $2^{O(k)}$ algorithm and a kernel with $O(k^5)$ vertices for MAXIMUM CUT in the connected graph under more refined lower bound $\frac{m}{2} + \frac{n-1}{4}$. Subsequently, Crowston et al. [3] presented an $O(k^3)$ -vertex kernel, and finally, Etscheid and Mnich [8] constructed an $O(k)$ -vertex kernel. However, MAXIMUM CUT admits other lower bounds. One such lower bounds for a connected graph is $n - 1$. To construct a cut of this size, it is enough to find a spanning tree and then partition vertices such that all edges of the



© Ivan Bliznets and Vladislav Epifanov;

licensed under Creative Commons License CC-BY 4.0

48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023).

Editors: Jérôme Leroux, Sylvain Lombardy, and David Peleg; Article No. 22; pp. 22:1–22:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

tree appear in the cut. Note that lower bounds $\frac{m}{2} + \frac{n-1}{4}$, $n-1$ are incomparable. We denote MAXIMUM CUT parameterized above spanning tree by MAX-CUT-AST. Madathil et al. [15] presented an $\mathcal{O}^*(8^k)$ -time algorithm and a kernel with $O(k^5)$ vertices for MAX-CUT-AST. Note that the lower bounds $\frac{m}{2} + \frac{n-1}{4}$ and $n-1$ are incomparable, since in very sparse graphs $n-1 > \frac{m}{2} + \frac{n-1}{4}$ but in dense graphs $\frac{m}{2} + \frac{n-1}{4} > n-1$. So, the linear kernel designed by Etscheid and Mnich [9] does not provide a linear kernel for the parameterization above $n-1$.

In this paper, we continue the study of MAX-CUT-AST. Our main result establishes an $O(k)$ -vertex kernel for this problem. This result resolves an Open Problem 2 from the survey [13] and significantly improves upon the previous $O(k^5)$ -vertex kernel.

► **Theorem 1.** *MAX-CUT-AST has a kernel with $\mathcal{O}(k)$ vertices and $\mathcal{O}(k)$ edges.*

► **Theorem 2.** *There is no kernel with sublinear number of vertices for MAX-CUT-AST unless ETH fails.*

MAXIMUM CUT does not admit a $2^{o(m)}$ -time algorithm unless the ETH fails [2]. Hence, MAX-CUT-AST also does not admit $2^{o(k)}$ -time algorithm unless the ETH fails. However, we show that we can get a significant improvement if we restrict our input to special classes of graphs. Specifically, we proved the following theorems.

► **Theorem 3.** *MAX-CUT-AST can be solved in a connected co-bipartite graph G in $2^{\mathcal{O}(\sqrt{k})}$ poly time.*

► **Theorem 4.** *MAX-CUT-AST on connected chordal graphs and split graphs admits a $2^{\mathcal{O}(\sqrt{k})}$ poly-time algorithm.*

We complement these results with conditional lower bounds assuming the Exponential Time Hypothesis.

► **Theorem 5.** *MAX-CUT-AST does not admit $2^{o(\sqrt{k})}$ poly or $2^{o(\sqrt{n})}$ poly algorithms on split graphs and on chordal graphs unless the ETH is false.*

► **Theorem 6.** *MAX-CUT-AST does not admit $2^{o(\sqrt[4]{k})}$ poly algorithm on co-bipartite graphs unless the ETH is false.*

We also initiate a new parameterization of MAXIMUM CUT on tripartite graphs. First, we show that MAX-CUT-AST on tripartite graphs is essentially as hard as MAX-CUT-AST on general graphs. Note that it is easy to see that a tripartite graph admits cut of size $\frac{2}{3}m$ where m is the number of edges. However, it is not clear how to find such cut if the partition is not provided in the input. We prove that we can construct a randomized polynomial time algorithm that finds such cut. In order to design the algorithm we employ semidefinite programming and technique used to find a best known approximation of MAXIMUM CUT [12].

► **Theorem 7.** *There is a polynomial time randomized algorithm that finds an edge cut of size $\frac{2}{3}|E(G)|$ on an input tripartite graph G .*

Note that the lower bound $\frac{2}{3}m$ is essentially tight since disjoint union of k triangles is a tripartite graph and the maximum cut equals to $2k$ while the total number of edges is $3k$. Therefore it is natural to ask whether MAXIMUM CUT above $\frac{2}{3}|E(G)|$ admits an FPT algorithm in tripartite graphs. Unfortunately, we have not resolved the question and state it as an open problem. However, we show that finding cuts significantly bigger than $\frac{2}{3}|E(G)|$ is hard even in tripartite graphs that contain such cuts. Formally, we proved the following theorem.

► **Theorem 8.** *Assuming $P \neq NP$ for any $0 < \varepsilon < \frac{1}{6}$ there is no polynomial time algorithm that finds cut of size $\geq (\frac{2}{3} + \varepsilon)|E|$ in tripartite graph if such cut exists.*

2 Preliminaries

In this paper we work with undirected simple graphs. Only in very special cases we allow multigraphs and multi-edges. We explicitly mention situation when we start working with multi-edges. We use standard graph notations which can be found in [7]. As usual we denote by n or $|V(G)|$ number of vertices in a graph G . Similarly by m or $|E(G)|$ we denote number of edges in a graph G . Generally by k we denote value of parameter that is under consideration at the moment of discussion. Complement of the graph G we denote by \bar{G} . We recall that $V(\bar{G}) = V(G)$ and $e \in E(\bar{G})$ if and only if $e \notin E(G)$. Recall that in the MAXIMUM CUT problem we need to partition vertices of a given graph into two parts such the number of edges between two parts is the maximum possible. By $E_G(A, B)$ we denote the set of edges going from the set A to the set B in the graph G . If the graph G is clear from the context we omit subscript G . Partition of vertices of graph into two disjoint parts A and B we denote by (A, B) . Slightly abusing notation in some cases we refer to cut as a set of edges $E(A, B)$ and in other cases we refer to cut as a partition of vertices (A, B) . $\text{mc}(G)$ denotes size of the maximum cut of the graph G . We call a vertex an *articulation point/vertex* if its removal increase number of connected components. Recall that *block* is a maximal 2-connected component.

Chordal graphs are graphs that do not contain induced cycles on ℓ vertices for $\ell \geq 4$. *Co-bipartite* graphs are graphs whose complement graph is bipartite.

Many of our results concern the parameterized complexity of the problems, including fixed-parameter tractable algorithms, kernelization algorithms, and some hardness results for certain parameters. For a detailed survey in parameterized algorithms we refer to the book of Cygan et al [6]. Due to the space constraints, proofs of lemmas marked by (\star) are omitted.

3 Kernel

In this section we provide a linear kernel for MAX-CUT-AST. Recall that in this problem, we are given a connected graph G , integer k , and our goal is to determine if G has a maximum cut of size at least $n - 1 + k$. Before we proceed we formulate lemma by Mathadil et al. [15] that we are using in our algorithm.

► **Lemma 9** ([15]). *There exists a polynomial time algorithm that for (G, k) either concludes that graph G has a cut of size at least $|V(G)| - 1 + k$ or finds a set $S \subseteq V(G)$ such that $|S| \leq 3k$ and each block of $G - S$ is either a clique or a cycle.*

► **Theorem 1.** *MAX-CUT-AST has a kernel with $\mathcal{O}(k)$ vertices and $\mathcal{O}(k)$ edges.*

Proof. First of all we apply algorithm from Lemma 9. After this step either we conclude that input instance is Yes-instance and we can output a trivial kernel or we find a set S with properties described in Lemma 9. In the later case we proceed with an application of reduction rules. Our reduction rules maintain the following properties: (i) graph is connected; (ii) each block of $G - S$ is either a cycle or a clique, however some of these cycles might become cycles of length 2 (our graph might become a multi-graph).

We call a vertex of $G - S$ *special* if it has an edge to S or if it is an articulation point of $G - S$.

We are ready to present our first reduction rule.

Rule 1:	Let U be a block in $G \setminus S$, $v \in U$ is the only special vertex in U and $U \neq \{v\}$
Remove:	Vertices $U \setminus \{v\}$
Parameter:	Reduce k by $\text{mc}(U) - (U - 1)$

22:4 MaxCut Above Guarantee

Proof of correctness. Note that U has no other articulation points or vertices with edges to S except v . Therefore, v is an articulation point of the whole graph G . Hence, the maximum cut in G equals to the sum of the cut in U and $G - (U \setminus \{v\})$.

This reduction deletes all edges in U , so the maximum cut decreases by $\text{mc}(U)$. Besides we delete $|U| - 1$ vertices. Hence, the parameter should be decreased by $\text{mc}(U) - (|U| - 1)$. ◀

Rule 2: Let v has no neighbors in $G \setminus S$, and exactly one neighbor in S i.e. $N(v) = u \in S$
 Remove: v
 Parameter: Reduce k by multiplicity of edge $uv - 1$

Proof of correctness. Under this constrains v is either a pendant vertex in G or vertex connected with some other vertex u by multiple edges. Removal of v decreases the maximum cut by multiplicity of edge uv and number of vertices decreases by one. ◀

Since a block can consist of one vertex only in case this vertex is isolated, exhaustive application of these two rules leave every block with at least two special vertices or the whole connected component is a single vertex with at least two edges towards S and neighborhood of size at least 2.

► **Lemma 11.** *In a graph where Rules 1, 2 do not apply, every connected component of $G - S$ has at least two edges to S .*

Proof. Let C be a connected component of $G - S$ such that $|E(S, C)| = 1$. Therefore, at most one vertex from C is a special vertex but not an articulation point.

If C is composed of a single block then it has no articulation points and therefore has a block with only one special vertex. Hence, either Rule 1 or Rule 2 must be applicable.

Otherwise C has at least two leaf blocks, each of them having exactly one articulation point. At least one of these leaf blocks has no other special vertices and can be reduced by Rule 1. ◀

► **Lemma 12.** *Let f be the number of edges between S and those connected components of $G - S$ which have at least 3 edges to S . If $f \geq 24k$ then (G, k) is a YES-instance.*

Proof. Let U_1, \dots, U_m be a connected components of $G - S$. Without loss of generality we can assume that the first l of them have more than three edges to S and the other have only 2. Note that $l \leq \frac{f}{3}$.

If $f \geq 24k$ we construct a cut of G with at least $|V(G)| + k$ edges. First of all, in each connected component U_i we construct a cut of size at least $|V(U_i)| - 1$ according to a spanning tree lower bound. After that we put the whole S into one of the parts so that at least half of edges between S and $G - S$ is cut. Note that number of edges between S and $G - S$ is at least $2(m - l) + f$.

So, the constructed cut has the following size:

$$\begin{aligned} \text{mc}(G) &\geq \sum_{i=1}^m (|V(U_i)| - 1) + (m - l) + \frac{f}{2} = \sum_{i=1}^m |V(U_i)| - l + \frac{f}{2} \geq \sum_{i=1}^m |V(U_i)| - \frac{f}{3} + \frac{f}{2} \\ &= \sum_{i=1}^m |V(U_i)| + \frac{f}{6} \geq \sum_{i=1}^m |V(U_i)| + 4k \geq \sum_{i=1}^m |V(U_i)| + |S| + k \geq V(G) + k \end{aligned}$$

Constructed cut has size at least $\geq V(G) - 1 + k$ which means that (G, k) is a Yes-instance. ◀

Before we state the next reduction rule we prove the following auxiliary claim.

▷ **Claim 13.** Let M be a maximum cut in a graph G , and edge e is a bridge in G then $e \in M$.

Proof. If $e \in M$ then there is nothing to prove. Otherwise, $e \notin M$. Let $e = uv$. The maximum cut M induces a partition of vertices in G into two parts V_1, V_2 . Without loss of generality $u, v \in V_1$. The edge e also induces a partition of $V(G)$ into two parts W_1, W_2 , each part corresponds to a connected component in a graph $G - e$. Without loss of generality, we assume that $u \in W_1, v \in W_2$. Let us consider the following partitioning (A, B) where $A = (V_1 \cap W_1) \cup (V_2 \cap W_2), B = (V_1 \cap W_2) \cup (V_2 \cap W_1)$. Note that $u \in A, v \in B$. So the cut generated by partitioning (A, B) contains edge e . Moreover, if edge $e' \neq e$ then either e' connects two vertices from W_1 or two vertices from W_2 . Hence, if $e' \in M$ then e' has one endpoint in A and the other in B . So, M is not a maximum cut which leads to a contradiction. ◁

Rule 3	Let U be a block in $G - S$ which is a clique, $W \subseteq U$ is the set of special vertices such that $ W \leq \lceil U /2 \rceil$
Remove:	Vertices $V(U) \setminus W$ and a maximal subset of edges F in $G[W]$ such that $(G - (U - W)) \setminus F$ is connected
Parameter:	Reduce k by $\text{mc}(U) - V(U) + W - E(G[W] \setminus F) $

Proof of correctness. Let G' be a graph obtained from G by deletion of all edges from U and vertices $V(U) \setminus W$.

Since U is a clique, its maximum cut is achieved when its vertices are divided between partitions equally (or almost equally if $|U|$ is odd). Since $|W| \leq \lceil |U|/2 \rceil$, any cut of G' can be extended to induced maximum cut of U (simply add vertices from $U \setminus W$ to both parts such that they contain $\lceil |U|/2 \rceil$ and $\lfloor |U|/2 \rfloor$ vertices from U). Therefore, $\text{mc}(G') + \text{mc}(U) \leq \text{mc}(G)$.

Since G' and U do not have common edges we have $\text{mc}(G') + \text{mc}(U) \geq \text{mc}(G)$. Combining the two inequalities we have $\text{mc}(G') + \text{mc}(U) = \text{mc}(G)$.

Let G'' be a graph $(G - (U - W)) \setminus F$. Note that $V(G'') = V(G')$ and $E(G'') = E(G') \cup (E(G[W]) \setminus F)$. Each edge from the set $E(G[W]) \setminus F$ is a bridge in G'' , otherwise F is not maximal. In any graph maximum cut contains all bridges, by Claim 13, so we have: $\text{mc}(G'') = \text{mc}(G') + |E(G[W]) \setminus F| = \text{mc}(G) - \text{mc}(U) + |E(G[W]) \setminus F|$. And this proves correctness of the reduction.

Note that the reduction preserves the clique-cycle-forest property of $G - S$ as we delete the subset $V(U) \setminus W$ and the clique W was replaced by a spanning tree. So each edge in the spanning tree of W can be seen as a clique on two vertices. ◀

Rule 4	Let $e = bc$ be an edge from $E(G - S)$ such that $\deg_G b = \deg_G c = 2$, and a, d be the other neighbours of b and c respectively (note that a and d may be the same vertex)
Remove:	vertices b, c and add edge ad if $a \neq d$
Parameter:	No changes applied

Proof of correctness. Let G' be graph obtained after application of the reduction rule. We prove that $\text{mc}(G') = \text{mc}(G) - 2$.

First, we show that $\text{mc}(G') \geq \text{mc}(G) - 2$. Let (A, B) be partition of G that produces the maximum cut. Induced partition in the subgraph G' has cut of size $\geq \text{mc}(G) - 2$. Indeed, if all 3 edges ab, bc, cd are in cut $E(A, B)$, then a, d are in different partitions. Hence, the

induced partition cuts an edge ad in G' having a cut of size $\text{mc}(G) - 3 + 1 = \text{mc}(G) - 2$ in G' . If at most 2 edges from the set $\{ab, bc, cd\}$ belong to the cut $E(A, B)$ then obviously induced vertex partition has cut at least $|E(A, B)| - 2 = \text{mc}(G) - 2$.

To prove that $\text{mc}(G) \geq \text{mc}(G') + 2$ we construct partitioning of vertices $V(G)$ based on partitioning of vertices $V(G')$. Let (A', B') be the maximum cut in G' . We construct partition of $V(G)$ simply by adding vertices b, c to partition (A', B') . We place c be in the same part as a and b be in the other. Let us call obtained partition (A, B) . In this case edges ab, bc belong to cut $E(A, B)$, edge cd is in the cut if and only if ad was in the cut of the graph G' . Hence, $|E_G(A, B)| = |E_{G'}(A', B')| + 2$.

Since the lower bound and the maximum cut size decrease by two, the parameter should not be changed.

Note that after application of the reduction rule $G' - S$ is a clique-cycle-forest as $G - S$. However, the reduction can remove a cycle of length 3 or turn a cycle of size 4 into two multiple edges. ◀

► **Lemma 14.** *If none of the above rules is applicable to G then at least half of all vertices in every block of $G - S$ are special.*

Proof. For cliques it is true since otherwise rule 3 is applicable. Cycles can not have 2 consecutive non-special vertices since otherwise we can apply reduction rule 4. ◀

Rule 5 If there are at least k even cycles blocks in $G - S$
 Return: YES-instance.

Proof of correctness. Consider a subgraph H containing k even cycles of $G - S$. We construct a spanning forest in H , to do this we remove an arbitrary edge in each cycle. This forest can be extended to a spanning tree F of the whole G . If we add back previously deleted edges of the subgraph H we obtain a bipartite graph (all introduced cycle have even length). The bipartite subgraph has exactly $|V(G)| - 1 + k$ edges, hence, G has a required cut and we can output YES. ◀

Now we can bound the total size of connected components in $G - S$ that have at least 3 edges to S (note that here we are speaking about connected components and not blocks). We split blocks of these components into several types and bound the total number of vertices in block of each type. We consider the following types of blocks from connected components with at least 3 edges going to S :

1. Blocks with ≥ 3 articulation points;
2. Blocks with < 3 articulation points and at least one special vertex that is not an articulate vertex;
3. Even cycles with exactly two special vertices which are articulation points;
4. Cliques and odd cycles with exactly two special vertices which are articulation points.

These types of blocks cover all possibilities, since in connected components with at least three edges towards S all blocks with only one special vertex were removed by rule 1. Below for each type we bound the total size of blocks of this type.

1. For the graph $G - S$ we consider a bipartite forest of blocks and articulation points F : for every block X we introduce a vertex a_X and for every articulation point y we introduce a vertex b_y . Moreover, $a_X b_y \in E(F)$ if and only if $y \in V(X)$. Each articulation point belong to at least two blocks. Therefore leaves and isolated vertices in F correspond to blocks with one and zero articulation points respectively. We recall, that we removed all

blocks with at most one special vertex except isolated vertices with at least two edges going to S . Hence, each block corresponding to isolated vertex or leaf in the forest F has an edge going to S , as every non-isolated block left has at least two special vertices each of these blocks has its own edge to S . Hence, by Lemma 12 we have that there are at most $24k$ leaves and isolated vertices in F .

It is known that for any tree or forest T the following holds

$$\sum_{v \in T: \deg_T v > 2} (\deg(v) - 2) \leq \text{number of leaves in } T.$$

Hence, we have

$$\sum_{v \in F: \deg_F v > 2} (\deg_F(v) - 2) \leq 24k. \quad (1)$$

Moreover, $\sum_{v \in F: \deg_F v > 2} \deg_F(v) \leq 3 \sum_{v \in F: \deg_F v > 2} (\deg_F(v) - 2) \leq 3 \cdot 24k = 72k$.

Note that the total number of articulation points in blocks with at least 3 articulation points is bounded by $\sum_{v \in F: \deg_F v > 2} \deg_F(v)$. The total number of special vertices that are not articulation points in such blocks is bounded by $24k$, by Lemma 12. Therefore, the total number of special vertices in blocks with at least 3 articulation points is bounded by $72k + 24k = 96k$. Hence, by Lemma 14 these blocks have at most $2 \cdot 96k = 192k$ vertices.

2. Each block of these type has at least one special vertex v that is not an articulation point. This vertex must have an edge to S . Therefore at least one third of special vertices in this type blocks are vertices with an edge to S . By Lemma 12, number of vertices with an edge to S is at most $24k$. Hence, blocks of this type have at most $3 \cdot 24k = 72k$ special vertices. Moreover, by lemma 14 we conclude that the total number of vertices in these blocks is at most $2 \cdot 72k = 144k$.
3. There is at most k such blocks otherwise the reduction rule 5 is applicable. By Lemma 14 each of these blocks has at most 4 vertices, so their total size is at most $4k$.
4. We note, that the only blocks in this category are simple edge between two articulation points. Indeed, a clique having more vertices would be reduced by Rule 3. An odd cycle with only two special vertices must have size 3 by Lemma 14. However, a cycle on 3 vertices is also a clique, so it is also subject to reduction by rule 3.

To bound the number of vertices in these blocks we again use forest of blocks and articulation points F constructed before. Let us mark and count vertices in this forest that belong to one of the following classes:

- a. **vertices of degree ≥ 3** : there is at most $24k$ of such vertices, follows from inequality 1
- b. **other vertices corresponding to blocks of type 2 and 3**: Note that blocks of first type have degree ≥ 3 and therefore are counted in previous point. Before we showed that there are at most $24k$ blocks of the second type and at most k blocks of the third type, so there are at most $25k$ vertices of these type.
- c. **vertices corresponding to articulation points with an edge to S** : by Lemma 12 there is at most $24k$ such vertices.

Recall that we mark all vertices from bullets a, b, c . Hence, in total there are at most $24k + 25k + 24k = 73k$ marked vertices. Note that every block of type 4 corresponds to an unmarked vertex.

Unmarked vertices correspond to blocks of type 4 and articulation points that (i) does not have an edge to S , (ii) belongs to exactly two blocks. We cannot have a path of unmarked vertices of length 6 since otherwise Rule 4 is applicable.

Note that every unmarked vertex is of degree 2. We consider a forest F' obtained from F by contracting every path of unmarked vertices into an edge. F' is a forest with only marked vertices so it has at most $73k$ vertices and at most $\leq 73k - 1$ edges. No more than 6 unmarked vertices can be mapped to a single edge of F . Hence, there are at most $6(73k - 1) = 438k - 6$ unmarked vertices in F . Recall that number of blocks of type 4 is at most the number of unmarked vertices. Moreover, each such block contains at most 2 vertices. Hence, number of vertices in such blocks is at most $876k - 12$.

We just have shown that number of vertices in connected components with at least three edges to S is at most $O(k)$. Now it is left to obtain a bound for the total size of connected components that have exactly 2 edges to S (one edge is not possible due to Rules 1, 2, zero edges is impossible due to connectivity of the graph). From now on we are considering only blocks of connected components that have exactly 2 edges to S . Since each leaf block has an edge to S , these components are either a single block with 2 edges to S , or a chain of blocks with two leaf blocks having exactly one edge to S , or a single vertex having two edges going to S .

Note that every block in these components has exactly 2 special vertices or the whole component is a single vertex with two edges going to S . Hence, each such block is a cycle C_4 , or a cycle C_2 (a double edge), or a single edge K_2 , or the whole component is just one vertex.

Rule 6 If the total number of vertices in the connected components of $G - S$ with 2 edges to S is at least $20k$
 Return: YES.

Proof of correctness. Let $S' = S \cup$ (even cycle blocks of connected components of $G - S$ with 2 edges to S) and $U =$ other vertices of these components. Due to Rule 5 we have at most k even cycles in $G - S$. Moreover, we are considering blocks that have at most 4 vertices. Hence, at most $4k$ vertices can belong to an even cycles. Therefore, $|S'| \leq 3k + 4k = 7k$ and $|U| \geq 20k - 4k \geq 16k$.

Every vertex in U has a degree 2 in G . Hence, there are no edges between vertices in U , since otherwise to this edge Rule 4 is applicable. Therefore each vertex in U has 2 edges going to S' . We prove that in this case the graph contains a cut of size at least $V(G) + k$. To do this we consider the following procedure:

- take a spanning forest H in G such that each tree in the forest has exactly one vertex in S' ;
- uniformly randomly partition vertices in S' in two parts;
- place the rest of all vertices in the part so that all edges from trees in H are in cut.

Note that every two vertices from different trees are placed in parts randomly independently. Hence, each edge between trees can be in the constructed cut with probability $\frac{1}{2}$. Since $|S'| \leq 7k$ and each tree has one vertex in S' there is at least $|V(G)| - 7k$ edges in H . Every vertex in U has two edges to vertices in S' . Hence, one of the edges is an edge between two trees in H . Therefore, the expected number of edges in the cut is at least $\geq |E(H)| + \frac{16k}{2} = |V(G)| - |S'| + \frac{16k}{2} \geq |V(G)| - 7k + \frac{16k}{2} = |V(G)| + k$. Hence, we proved that a cut of size at least $|V(G)| - 1 + k$ exists.

Note that this procedure can be derandomized using method of conditional probabilities [17] which allows to find a required cut deterministically, if needed. ◀

Now we are ready to bound the overall number of vertices. Recall that there are at most $192k$ vertices in blocks of type 1, at most $144k$ vertices in blocks of type 2, at most $4k$ vertices in blocks of type 3, at most $876k - 12$ vertices in blocks of type 4. Hence,

number of vertices in connected components with at least 3 edges going to S is at most $192k + 144k + 4k + 876k - 12 = 1216k - 12$. We also have at most $20k$ vertices in other connected components of $G - S$ and at most $3k$ vertices in S . In total we have at most $1216k - 12 + 20k + 3k = \mathcal{O}(k)$ vertices.

In order to bound number of edges we employ the final reduction rule.

Rule 7 If graph contains at least $2k + 2|V(G)|$ edges
Return: YES-instance.

Proof of correctness. Every graph has a cut of size at least $\frac{|E(G)|}{2}$. Hence, in this case graph has a cut of size at least $\frac{2k+2|V(G)|}{2} = |V(G)| + k$ which means that our input is a YES-instance. ◀

Taking into account that $|V(G)| = \mathcal{O}(k)$ after application of Rule 7 we conclude that $|E(G)| = \mathcal{O}(k)$.

The algorithm either produces a graph with $\mathcal{O}(k)$ vertices and edges or concludes that the graph has a required cut. In the latter we can output a trivial YES-instance. Hence, we constructed a linear kernel.

Note that the final graph might contain multi-edges. However, in this case we can apply reduction rule 4 backwards and replace every multi-edge with a cycle of length 4. It is easy to see that exhaustive application of this operations increase kernel size only linearly. ◀

The following theorem shows that the obtained kernel has asymptotically optimal size unless ETH fails.

► **Theorem 2.** *There is no kernel with sublinear number of vertices for MAX-CUT-AST unless ETH fails.*

Proof. A trivial brute force algorithm solves MAX-CUT in $\mathcal{O}(2^{|V(G)|})$. A kernel with $o(k)$ vertices would imply that we can solve MAX-CUT in $2^{o(k)} = 2^{o(|E(G)|)}$ which is impossible unless ETH fails. ◀

4 Subexponential Algorithms for Max-Cut-AST

In this section we present subexponential algorithms for MAX-CUT-AST in chordal and co-bipartite input graphs.

4.1 Co-bipartite Graphs

► **Theorem 3.** *MAX-CUT-AST can be solved in a connected co-bipartite graph G in $2^{\mathcal{O}(\sqrt{k})}$ poly time.*

Proof. Note that co-bipartite graphs consist of two disjoint cliques and edges between them. At least one of the cliques contains at least half of the vertices. Therefore, each co-bipartite graph contains at least $\frac{n}{2}(\frac{n}{2} - 1)/2$ edges. It is well known that a maximum edge cut in any graph contains at least half of all edges. Hence, in any co-bipartite graph maximum cut size is at least $\frac{n}{2}(\frac{n}{2} - 1)/4$. Therefore, if $\frac{n}{2}(\frac{n}{2} - 1)/4 \geq n - 1 + k$ we can output Yes immediately. Otherwise, $k = \Omega(n^2)$ and the simple brute-force algorithm with working time $2^n n^{\mathcal{O}(1)}$ is actually $2^{\mathcal{O}(\sqrt{k})}$ poly algorithm. ◀

4.2 Chordal Graphs

First of all we recall some well-known results and prove auxiliary lemma.

► **Lemma 17** ([11]). *For an input chordal graph G we can find a tree decomposition of minimum width in polynomial time. Moreover, in this tree decomposition all vertices of each bag induce a clique in the original input graph G .*

► **Lemma 18** ([1]). *Given an input graph G and its tree decomposition of width k one can find the size of maximum cut in $2^{\mathcal{O}(k)}$ poly time.*

► **Lemma 19.** *Let H be an induced subgraph in a connected graph G and graph H has a cut with at least ℓ edges then the graph G has a cut of size at least $|V(G)| - |V(H)| + \ell$.*

Proof. Assume that (A, B) is the partitioning of $V(H)$ into two parts such that $|E(A, B)| \geq \ell$. We complete the partitioning (A, B) with vertices from $G \setminus H$ such that we get a cut of size at least $|V(G)| - |V(H)| + \ell$.

Let us consider a spanning forest F of the graph G such that each tree T from the forest F has only one vertex from $V(H)$ and the vertex is a root node of the tree. For each vertex $v \in V(G)$ we assign a tree T_v such that $v \in T_v$ and $T_v \in F$. Moreover, for each $v \in V(G)$ we assign vertex $r_v \in V(H)$ such that $r_v \in T_v$ and r_v is a root vertex of T_v . Now we are ready to present the partitioning. If $d_{T_v}(v, r_v)$ is even we place v in the same part as r_v , otherwise we place v in the opposite part of r_v . In the constructed partitioning each edge of forest F is in the cut, since for each vertex its ancestor belong to a different part. Since the graph G is connected we have that $|E(F)| = |V(G)| - |V(H)|$. Also the partitioning contains all edges from the initial cut of subgraph H . Therefore, we constructed a cut of size at least $|V(G)| - |V(H)| + \ell$. ◀

Now we have all ingredients to prove the main theorem of this subsection.

► **Theorem 4.** *MAX-CUT-AST on a connected chordal graphs admits a $2^{\mathcal{O}(\sqrt{k})}$ poly-time algorithm.*

Proof. Using Lemma 17 we find tree decomposition of the input graph G . In this tree decomposition each bag is a clique. We denote by $\ell - 1$ the treewidth of the input graph G . Hence, there is a bag of size ℓ . It means that G contains a subgraph H such that H is a clique on ℓ vertices. If $\frac{\ell(\ell-1)}{4} \geq k + \ell - 1$ then by Lemma 19 we conclude that the graph G contains a cut of size at least $n + k - 1$.

Otherwise, $\frac{\ell(\ell-1)}{4} < k + \ell - 1$ and we have $\ell < 5 + 2\sqrt{k}$. Hence, $\ell = \mathcal{O}(\sqrt{k})$ and we can find an exact value of the maximum cut by standard dynamic programming using tree decomposition. The running time of this algorithm is $2^{\mathcal{O}(\ell)}$ poly = $2^{\mathcal{O}(\sqrt{k})}$ poly. ◀

► **Corollary 21.** *MAX-CUT-AST on a connected split graph admits a $2^{\mathcal{O}(\sqrt{k})}$ poly algorithm.*

Proof. The statement immediately follows from the previous theorem since the class of all split graphs is a subclass of all chordal graphs. ◀

4.3 Lower Bounds

Under assumption of the Exponential Time Hypothesis we show that algorithms for chordal and split graphs are essentially tight.

► **Theorem 5.** *MAX-CUT-AST does not admit $2^{o(\sqrt{k})}$ poly or $2^{o(\sqrt{n})}$ poly algorithms on split graphs and on chordal graphs unless the ETH is false.*

Proof. Bodlaender and Jansen [1] presented a reduction of MAX-CUT on arbitrary graph G to MAX-CUT on split graph. This reduction transform graph G with a cut K into a graph G' on $|V(G)| + |E(\bar{G})|$ vertices with $|V(G)| \cdot (|V(G)| - 1)/2 + 2|E(\bar{G})|$ edges. Moreover, G has the maximum edge cut K if and only if G' has the maximum edge cut $2|E(\bar{G})| + K$. Note that $2|E(\bar{G})| + K = O(n^2)$ and $|V(G)| + |E(\bar{G})| = O(n^2)$. Hence, by solving MAX-CUT-AST in $2^{o(\sqrt{k})}$ poly time (or in $2^{o(\sqrt{|V(G')|})}$ poly) on split graphs we can solve MAX-CUT on arbitrary graphs in $2^{o(n)}$ poly time. However, it is known that under assumption of ETH MAX-CUT does not admit $2^{o(n)}$ poly algorithm [6]. So we get a desired contradiction. ◀

► **Theorem 6.** *MAX-CUT-AST does not admit $2^{o(\sqrt[4]{k})}$ poly algorithm on co-bipartite graphs unless the ETH is false.*

Proof. Bodlaender and Jansen [1] presented a reduction of MAX-CUT on split graphs to MAX-CUT on co-bipartite graph. If the initial split graph G contains n vertices then the obtained co-bipartite graph will contain $O(n)$ vertices and $O(n^2)$ edges. It means that $k = O(n^2)$. If MAX-CUT-AST admits $2^{o(\sqrt[4]{k})}$ poly = $2^{o(\sqrt[4]{n^2})}$ poly = $2^{o(\sqrt{n})}$ poly algorithm on co-bipartite graphs then we can solve MAX-CUT on split graphs in $2^{o(\sqrt[4]{k})}$ poly = $2^{o(\sqrt[4]{n^2})}$ poly = $2^{o(\sqrt{n})}$ poly which contradict the previous theorem. ◀

5 Tripartite Graphs

5.1 Parametrization Above a Spanning Tree

► **Lemma 24.** *Let us assume that MAX-CUT-AST on tripartite graphs with n vertices admits algorithm with running time $T(k, n)$ where k is the parameter. Then MAX-CUT-AST on arbitrary graph G with n' vertices and m' edges can be solved in $T(k, n' + 2m') + \text{poly}(n')$ time.*

Proof. To prove the statement we reduce MAX-CUT-AST on arbitrary graph G to MAX-CUT-AST on tripartite graph G' such that $|V(G')| = |V(G)| + |2E(G)|$. We construct G' in the following way, for each edge $e = uv \in E(G)$: (i) create two vertices e_u, e_v ; (ii) delete edge uv ; (iii) add edges $ue_u, e_u e_v, e_v v$. Essentially we subdivide each edge twice. It is easy to see that $V(G)$ is an independent set in graph G' and $G' \setminus G$ is a disjoint union of edges. Hence, G' is a tripartite graph.

Now we provide a connection between the size of maximum cut in the graph G and the size of maximum cut in G' . We claim that $\text{mc}(G') = \text{mc}(G) + 2|E(G)|$. Having partition (A', B') of vertices $V(G')$ we construct partitioning (A, B) of $V(G)$ such that $|E(A, B)| \geq |E(A', B')| - 2|E(G)|$. We take $A = A' \cap V(G), B = B' \cap V(G)$. All edges in $E(G')$ we can partition in triples of the following type $ue_u, e_u e_v, e_v v$. It is easy to see that if cut $E(A', B')$ contains all three edges $ue_u, e_u e_v, e_v v$ then cut $E(A, B)$ contains at least one edge. So it means that for each triple we lose at most 2 edges in $E(A, B)$ compared to $E(A', B')$. Hence, $|E(A, B)| \geq |E(A', B')| - 2|E(G)|$.

Now, it is enough to prove that having a cut (A, B) of graph G we can construct a cut (A', B') of G' such that $|E(A', B')| \geq |E(A, B)| + 2|E(G)|$. In order to do this for each edge $e = uv$ we place vertices e_u, e_v in the opposite part to vertices u, v correspondingly, i.e. if $u \in A, v \in B$ then we put u, e_v to A' and v, e_u to B' , similarly, if $u, v \in A$ then we put u, v to A' and e_u, e_v to B' . It is easy to see that for such cut (A', B') we have $|E(A', B')| \geq |E(A, B)| + 2|E(G)|$.

So we proved that $\text{mc}(G') = \text{mc}(G) + 2|E(G)|$. Recall that $|V(G')| = |V(G)| + |2E(G)|$. Hence, $\text{mc}(G') - (|V(G')| - 1) = \text{mc}(G) - (|V(G)| - 1)$. Therefore, answer for MAX-CUT-AST on graph G' is the same as the answer for MAX-CUT-AST on graph G . As G' is tripartite we have proved the desired result. ◀

Above lemma implies the following result.

► **Corollary 25.** *There is no $2^{o(k)}$ poly algorithm for MAX-CUT-AST on tripartite graphs under assumption of the ETH.*

Proof. Such algorithm would imply that MAX-CUT-AST can be solved on arbitrary graph in $2^{o(k)}$ poly time by lemma 24. However, existence of such algorithm contradict to the ETH [15]. ◀

5.2 Parameterization Above $\frac{2}{3}|E|$

First of all we show that any tripartite graph G has a maximum cut of size at least $\frac{2}{3}|E|$. Let assume that G has parts V_1, V_2, V_3 . It is easy to see that $|E(V_1, V_2 \cup V_3)| + |E(V_2, V_1 \cup V_3)| + |E(V_3, V_1 \cup V_2)| = 2|E(G)|$. So at least one of the cuts $(V_1, V_2 \cup V_3), (V_2, V_1 \cup V_3), (V_3, V_1 \cup V_2)$ has size $\frac{2}{3}|E|$. However, even knowing that a graph is tripartite we may not have a partition itself.

Below we show that we can find an edge cut of size $\frac{2}{3}|E|$ in tripartite graphs even if partition on three parts is not provided in the input. To get this result we employ semi-definite programming.

► **Theorem 7.** *There is a polynomial time randomized algorithm that finds an edge cut of size $\frac{2}{3}|E(G)|$ on an input tripartite graph G .*

Proof. First of all for each vertex $v_i \in G$ we assign vector $\vec{v}_i \in \mathbb{R}^n$ and formulate the following program:

$$\begin{aligned} & \min \lambda \\ & \forall i \in V(G) \langle \vec{v}_i, \vec{v}_i \rangle = 1 \\ & \forall ij \in E(G) \langle \vec{v}_i, \vec{v}_j \rangle \leq \lambda \end{aligned}$$

Essentially we are looking for n unit vectors such that minimum angle between any two vectors that correspond to adjacent vertices is maximized. We know that the input graph is tripartite, hence there is a solution such that $\lambda \leq -\frac{1}{2}$. To achieve the value $-\frac{1}{2}$ it is enough to map vertices of first part into $(1, 0, \dots, 0)$, vertices of second part into $(-\frac{1}{2}, \frac{\sqrt{3}}{2}, 0, \dots, 0)$, vertices of third part into $(-\frac{1}{2}, -\frac{\sqrt{3}}{2}, 0, \dots, 0)$. It means that in the optimum solution of the semidefinite program angle between two vectors corresponding to two adjacent vertices is at least $\frac{2\pi}{3}$. It is known [18] that for any ϵ semidefinite programming can be solved with additive error of size ϵ in polynomial time from the input and $\log \frac{1}{\epsilon}$. Therefore, we can find the optimum value of λ up to additive error ϵ . After that we generate a random hyperplane and split vertices of graph into two parts. We put vertex v_i to the first part if the vector \vec{v}_i is lying on one side of the generated hyperplane or to the second if \vec{v}_i is lying on the other side of the hyperplane. The expected value of an edge $e = v_i v_j$ being in the cut under such partitioning is equal to $\arccos(\langle \vec{v}_i, \vec{v}_j \rangle) \geq \arccos(-\frac{1}{2} + \epsilon) = \frac{2}{3} - \delta(\epsilon)$ [18, Chapter 6]. Hence, the expected value of the cut is at least $(\frac{2}{3} - \delta(\epsilon))|E|$. We choose ϵ in such way that $\delta(\epsilon)|E| \leq \frac{1}{6}$. Since derivative of $\arccos(\cdot)$ is bounded in the neighborhood of $-\frac{1}{2}$ we can take $\epsilon = \frac{1}{1000|E|^2}$. If we choose ϵ in this way we obtain a polynomial time algorithm which output a cut of expected value at least $\frac{2}{3}|E| - \frac{1}{6}$. Note that the value $\frac{2}{3}|E| - \frac{1}{6}$ is not integer and the size of a cut is an integer. So, the algorithm sometimes must output values of size at least $\lceil \frac{2}{3}|E| \rceil - \frac{1}{6} = \lceil \frac{2}{3}|E| \rceil$. Since the cut size is bounded by $|E|$, our algorithm output value of size $\lceil \frac{2}{3}|E| \rceil$ with probability at least $\Omega(\frac{1}{|E|})$. Therefore it is enough to repeat the algorithm polynomial number of times, take the largest cut among all generated. The algorithm will not output cut of size at least $\lceil \frac{2}{3}|E| \rceil$ only with exponentially small probability. ◀

► **Lemma 27** (*). *Under the assumption $P \neq NP$ there is no polynomial time algorithm that for each tripartite graph G finds a cut of size at least $\frac{5}{6}|E|$ if it exists.*

In the following theorem we show that fraction $\frac{5}{6}$ can be replaced with any fraction of the following type $\frac{2}{3} + \epsilon$ where $\epsilon \in (0, \frac{1}{6})$. Informally speaking to achieve this results it is enough to add large number of disjoint subgraphs K_3 to the construction in the previous theorem. Indeed, each partition of a triangle cuts at most 2 edges out of 3. Therefore, by adding triangles we make fraction of size of maximum cut to the total number of edges being close to $\frac{2}{3}$.

► **Theorem 8.** *Assuming $P \neq NP$ for any $0 < \epsilon < \frac{1}{6}$ there is no polynomial time algorithm that finds cut of size $\geq (\frac{2}{3} + \epsilon)|E|$ in tripartite graph if such cut exists.*

Proof. In order to refute existence of such algorithm we show that using this algorithm we can construct a polynomial time algorithm that on tripartite graphs with the number of edges divisible by 6 outputs cut of size at least $\frac{5}{6}|E|$, if such cut exists.

Let us assume that tripartite graph G contains n vertices and m edges and we need to check existence of cut of size at least $\frac{5}{6}m$. If we add k disjoint triangles to the graph G then essentially we need to check if in the new graph exists a cut of size at least $\geq \frac{5m}{6} + 2k$, since in each triangle we can cut at most two edges. However, the number of edges in the new graph is $m + 3k$. Therefore the ratio between the cut and the overall number of edges becomes $\frac{5m/6+2k}{m+3k}$. This fraction tends to $\frac{2}{3}$ as long as we increase k . Consider the maximum integer k such that the fraction is not smaller than $\frac{2}{3} + \epsilon$. Note that, for fixed ϵ we have $k = \mathcal{O}(m)$. Since we chose the maximum k with such properties we have that:

$$\frac{5m/6 + 2(k+1)}{m + 3(k+1)} < \frac{2}{3} + \epsilon \quad (2)$$

Now it is left to show that cuts with $\frac{5}{6}m - 1$ edges does not satisfy required equation, i.e. that:

$$\frac{5m/6 + 2k - 1}{m + 3k} < \frac{2}{3} + \epsilon$$

To do this it is enough to show that $\frac{5m/6+2k-1}{m+3k} < \frac{5m/6+2(k+1)}{m+3(k+1)}$. Below we provide detailed computations.

$$\begin{aligned} & \frac{5m/6 + 2k - 1}{m + 3k} < \frac{5m/6 + 2(k+1)}{m + 3(k+1)} \\ 0 & < \frac{5m/6 + 2k + 2}{m + 3k + 3} - \frac{5m/6 + 2k - 1}{m + 3k} \\ 0 & < \frac{(5m/6 + 2k + 2)(m + 3k) - (5m/6 + 2k - 1)(m + 3k + 3)}{(m + 3k + 3)(m + 3k)} \\ 0 & < \frac{(5m/6 + 2k)(m + 3k) + 2(m + 3k) - (5m/6 + 2k)(m + 3k + 3) + (m + 3k + 3)}{(m + 3k + 3)(m + 3k)} \\ 0 & < \frac{(5m/6 + 2k)(m + 3k) + 2(m + 3k) - (5m/6 + 2k)(m + 3k) - 3(5m/6 + 2k) + (m + 3k + 3)}{(m + 3k + 3)(m + 3k)} \\ 0 & < \frac{2(m + 3k) - 3(5m/6 + 2k) + (m + 3k + 3)}{(m + 3k + 3)(m + 3k)} \\ 0 & < \frac{m(2 - 5/2 + 1) + k(6 - 6 + 3) + 3}{(m + 3k + 3)(m + 3k)} \\ 0 & < \frac{m/2 + 3k + 3}{(m + 3k + 3)(m + 3k)} \end{aligned}$$

Hence, the constructed graph with disjoint triangles G' contains cut of size $\frac{5m}{6} + 2k$ if and only if it contains at least $(\frac{2}{3} + \epsilon)|E(G')|$. Moreover, it happens if and only if initial graph contained a cut with at least $\frac{5}{6}|E(G)|$ edges. ◀

6 Conclusion

In Section 5 we initiate study of MAXIMUM CUT problem on tripartite graphs parameterized above $\frac{2}{3}|E|$ lower bound. We think that existence of an FPT algorithm for this problem is an interesting open question.

References

- 1 Hans L. Bodlaender and Klaus Jansen. On the complexity of the maximum cut problem. In Patrice Enjalbert, Ernst W. Mayr, and Klaus W. Wagner, editors, *STACS 94*, pages 769–780, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.
- 2 Liming Cai and David Juedes. On the existence of subexponential parameterized algorithms. *Journal of Computer and System Sciences*, 67(4):789–807, 2003.
- 3 Robert Crowston, Gregory Gutin, Mark Jones, and Gabriele Muciaccia. Maximum balanced subgraph problem parameterized above lower bound. *Theoretical Computer Science*, 513:53–64, 2013.
- 4 Robert Crowston, Mark Jones, and Matthias Mnich. Max-cut parameterized above the edwards-erdős bound. In *Automata, Languages, and Programming: 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I 39*, pages 242–253. Springer, 2012.
- 5 Robert Crowston, Mark Jones, and Matthias Mnich. Max-cut parameterized above the edwards-erdős bound. *Algorithmica*, 72(3):734–757, 2015.
- 6 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshantov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. URL: <https://www.springer.com/gp/book/9783319212746>.
- 7 R. Diestel. *Graph Theory*. Springer, 2006.
- 8 Michael Etscheid and Matthias Mnich. Linear kernels and linear-time algorithms for finding large cuts. *Algorithmica*, 80:2574–2615, 2018.
- 9 Michael Etscheid and Matthias Mnich. Linear kernels and linear-time algorithms for finding large cuts. *Algorithmica*, 80:2574–2615, 2018.
- 10 MR Garey, DS Johnson, and L Stockmeyer. Some simplified np-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.
- 11 Fănică Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974. doi:10.1016/0095-8956(74)90094-X.
- 12 Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.
- 13 Gregory Gutin and Matthias Mnich. A survey on graph problems parameterized above and below guaranteed values. *arXiv preprint arXiv:2207.12278*, 2022.
- 14 Richard M Karp et al. Complexity of computer computations. *Reducibility among combinatorial problems*, 23(1):85–103, 1972.
- 15 Jayakrishnan Madathil, Saket Saurabh, and Meirav Zehavi. Fixed-parameter tractable algorithm and polynomial kernel for max-cut above spanning tree. *Theory of Computing Systems*, 64(1):62–100, 2020.
- 16 Meena Mahajan and Venkatesh Raman. Parameterizing above guaranteed values: Maxsat and maxcut. *Journal of Algorithms*, 31(2):335–354, 1999.
- 17 Prabhakar Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37(2):130–143, 1988.
- 18 David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.