Exponential-Time Approximation Schemes via Compression

Tanmay Inamdar
Indian Institute of Technology, Jodhpur, India

Madhumita Kundu
University of Bergen, Norway

Pekka Parviainen
University of Bergen, Norway

M. S. Ramanujan
University of Warwick, UK

Saket Saurabh
Institute of Mathematical Sciences, Chennai, India
University of Bergen, Norway

Abstract

In this paper, we give a framework to design exponential-time approximation schemes for basic graph partitioning problems such as k-way cut, Multiway Cut, Steiner k-cut and Multicut, where the goal is to minimize the number of *edges* going across the parts. Our motivation to focus on approximation schemes for these problems comes from the fact that while it is possible to solve them exactly in $2^n n^{\mathcal{O}(1)}$ time (note that this is already faster than brute-forcing over all partitions or edge sets), it is not known whether one can do better. Using our framework, we design the first $(1+\epsilon)$-approximation algorithms for the above problems that run in time $2^{f(\epsilon)n}$ (for $f(\epsilon) < 1$) for all these problems.

As part of our framework, we present two compression procedures. The first of these is a "lossless" procedure, which is inspired by the seminal randomized contraction algorithm for Global Min-cut of Karger [SODA '93]. Here, we reduce the graph to an equivalent instance where the total number of edges is linearly bounded in the number of edges in an optimal solution of the original instance. Following this, we show how a careful combination of greedy choices and the best exact algorithm for the respective problems can exploit this structure and lead to our approximation schemes.

Our first compression procedure bounds the number of edges linearly in the optimal solution, but this could still leave a dense graph as the solution size could be superlinear in the number of vertices. However, for several problems, it is known that they admit significantly faster algorithms on instances where solution size is linear in the number of vertices, in contrast to general instances. Hence, a natural question arises here. Could one reduce the solution size to linear in the number of vertices, at least in the case where we are willing to settle for a near-optimal solution, so that the aforementioned faster algorithms could be exploited?

In the second compression procedure, using cut sparsifiers (this time, inspired by Benczúr and Karger [STOC '96]) we introduce "solution linearization" as a methodology to give an approximation-preserving reduction to the regime where solution size is linear in the number of vertices for certain cut problems. Using this, we obtain the first *polynomial-space* approximation schemes faster than $2^n n^{\mathcal{O}(1)}$ for Minimum bisection and Edge Bipartization. Along the way, we also design the first polynomial-space *exact* algorithms for these problems that run in time faster than $2^n n^{\mathcal{O}(1)}$, in the regime where solution size is linear in the number of vertices. The use of randomized contraction and cut sparsifiers in the exponential-time setting is novel to the best of our knowledge and forms our conceptual contribution.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Approximation algorithms analysis

Keywords and phrases Exponential-Time Algorithms, Approximation Algorithms, Graph Algorithms, Cut Problems

Digital Object Identifier 10.4230/LIPIcs.ITCS.2024.64

Funding *Tanmay Inamdar*: This research was done when the author was affiliated with University of Bergen, and was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819416).
M. S. Ramanujan: Supported by Engineering and Physical Sciences Research Council (EPSRC) grants EP/V007793/1 and EP/V044621/1.
Saket Saurabh: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819416), and Swarnajayanti Fellowship (No. DST/SJF/MSA01/2017-18).

1 Introduction

In the area of exact exponential-time algorithms, the objective is to design algorithms that outperform brute-force for computationally intractable problems [3, 5, 27, 25, 13, 31]. Because the problems are intractable we do not hope for polynomial-time algorithms. Instead the aim is to allow super-polynomial time and design algorithms that are significantly faster than brute-force. For example, for NP-complete problems on graphs on n vertices and m edges whose solutions are either subsets of vertices or edges, the brute-force or trivial algorithms basically enumerate all subsets of vertices or edges. This mostly leads to algorithms of time complexity $2^n n^{\mathcal{O}(1)}$ or $2^m n^{\mathcal{O}(1)}$, based on whether we are enumerating vertex sets or edge sets. Thus our goal is typically to design an algorithm with running time $c^n n^{\mathcal{O}(1)}$ ($c^m n^{\mathcal{O}(1)}$) for $c < 2$, and we try to minimize the constant c. We refer to the textbook of Fomin and Kratsch [15] for an introduction to the field.

While there are numerous papers dedicated to designing algorithms with running time $c^n n^{\mathcal{O}(1)}$ for NP-complete problems on graphs where the solution is a set of *vertices*, the same cannot be said about edge-subset problems [13, 14, 24, 6, 9, 22, 31]. In this paper we focus on NP-complete problems on graphs where the solution is subset of the edges. These include basic graph partitioning problems such as k-way cut (delete a minimum set of edges such that the resulting graph has at least k connected components) [16, 23], Multiway Cut (remove a minimum set of edges such that every connected component contains at most one out of a given set of terminal vertices) [11], Steiner k-cut (the task is to remove a minimum set of edges whose removal results in k connected components, each of which contains at least one of the given terminal vertices) [8], and Multicut (given a graph G and pairs of terminals called requests, remove a minimum set of edges such that the vertices in each request are disconnected, where the goal is to minimize the number of edges going across the parts) [20]. For these problems it is straightforward to design either a $2^m n^{\mathcal{O}(1)}$-time algorithm based on enumerating edge subsets or enumerating all partitions of $V(G)$ into desired number of parts, resulting in $n^n n^{\mathcal{O}(1)} = 2^{\mathcal{O}(n \log n)} n^{\mathcal{O}(1)}$-time algorithm. Further, it is possible to solve these problems exactly in $2^n n^{\mathcal{O}(1)}$ time (note that this is already faster than brute-forcing over all partitions or edge sets), using subset convolution based dynamic programming. So, the natural question that arises here and which has remained open in the area, is the following.

> Do the above-mentioned basic graph partitioning problems such as k-way cut, Steiner k-cut, Multiway Cut, and Multicut admit an algorithm with running time $c^n n^{\mathcal{O}(1)}$ for $c < 2$?

In light of the lack of progress on the above question [1], the next best target would be an algorithm with running time $c^n n^{\mathcal{O}(1)}$ for $c < 2$, if we are allowed to relax the condition of finding an exact solution to one of finding an "almost-optimal" solution. That is, we ask

> Is it possible to get an algorithm for the above problems with running time $c^n n^{\mathcal{O}(1)}$ for $c < 2$, if we allow loss in the solution size. That is, is it possible to design $(1+\epsilon)$-approximation algorithms for the above problems that run in time $2^{f(\epsilon)n} n^{\mathcal{O}(1)}$ (for $f(\epsilon) < 1$)?

This idea of relaxing the question of finding an exact solution to one of approximation is motivated by similar research on Max-SAT (given a CNF-formula Φ on n variables and m clauses, find an assignment that maximizes the number of satisfied clauses). It is easily seen that we can solve this problem in time $2^n(n+m)^{\mathcal{O}(1)}$, however it is a big open question whether it is possible to design an algorithm for Max-SAT running in time $c^n(n+m)^{\mathcal{O}(1)}$ for $c < 2$. In fact the decision version of Max-SAT, i.e., SAT, forms the basis for the well-known Strong Exponential Time Hypothesis (SETH) and has been used as a starting point for numerous lower bound results for problems solvable in polynomial time as well as super-polynomial time [18, 7, 21, 26]. One the other hand, it is possible to obtain a $(1-\epsilon)$-approximation algorithm for Max-SAT running in time $2^{f(\epsilon)n}$ (for $f(\epsilon) < 1$) [17, 1, 12]. See Alman, Chan and Williams [1] for the current best deterministic and randomized $(1-\epsilon)$-approximation algorithms for Max-SAT.

In this paper, we give a framework to design exponential-time approximation schemes for graph partitioning problems such as k-way cut, Multiway Cut, Steiner k-cut and Multicut. Using our framework, we design the first $(1+\epsilon)$-approximation algorithms for the above problems that run in time $2^{f(\epsilon)n}$ (for $f(\epsilon) < 1$) for all these problems.

▶ **Theorem 1.** *For every $\epsilon > 0$,* Steiner k-cut *and* Multicut *can be $(1+\epsilon)$-approximated in time $2^{f(\epsilon)n} n^{\mathcal{O}(1)}$ for some $f(\epsilon) < 1$.*

Since Steiner k-cut generalizes both k-way cut (set the terminals to be the entire vertex set) and Multiway Cut (set k to be the number of terminals), we have the following corollary.

▶ **Corollary 2.** k-way cut *and* Multiway Cut *can be $(1+\epsilon)$-approximated in time $2^{f(\epsilon)n} n^{\mathcal{O}(1)}$ for some $f(\epsilon) < 1$.*

To the best of our knowledge, this is the first exponential-time approximation scheme explicitly designed for *minimization* problems.

The key component of our framework behind Theorem 1 is a "lossless" procedure, which is inspired by the seminal randomized contraction algorithm of Karger [19] for Global Min-cut. Here, we reduce the graph to an equivalent instance where the total number of edges is linearly bounded in the number of edges in an optimal solution. Following this, we show how a careful combination of greedy choices and the best exact algorithm for the respective problems can exploit this structure and lead to our approximation schemes. In order to give the reader more insight into our algorithm, let us take a simple example. Suppose that the number of edges in the solution is denoted by OPT and the total number

[1] One notable exception is the k-way Cut problem, which can be seen a special case of Steiner k-Cut. Very recently, Lokshtanov et al. [23] designed a $(2-c)^n n^{\mathcal{O}(1)}$ time *exact* algorithm k-Cut, where c is a very small positive constant.

of edges in the graph is 10 OPT. Then, if we pick an edge uniformly at random, it can be correctly contracted without affecting a solution with probability 9/10, reducing the number of vertices by 1. If we succeed in this contraction step for roughly αn consecutive steps, then we would have reduced the number of vertices in the graph to $(1-\alpha)n$. At this point, we can solve the problem on the residual instance in time $2^{(1-\alpha)n}n^{\mathcal{O}(1)}$ (assuming such an exact algorithm exists) and we would have succeeded with probability $(9/10)^{\alpha n}$. By repeating this sufficiently (i.e., $(10/9)^{\alpha n}$ times), we end up with a running time that is bounded by $2^{(1-\alpha)n}n^{\mathcal{O}(1)} \cdot (10/9)^{\alpha n}$ and constant success probability. Clearly, this will beat $2^n n^{\mathcal{O}(1)}$ for any choice of α. Indeed, a success probability in any single step of more than 1/2 would be sufficient to beat $2^n n^{\mathcal{O}(1)}$. The only scenario where this algorithm does not succeed is when the total number of edges in the graph is at most, say, 10 OPT. In this case, deleting all the edges is obviously already a 10-approximation. In order to do better, we select a subset of vertices of smallest degrees, take all the edges incident on them into the solution and solve the problem exactly on the residual graph. A careful balancing of these two cases gives us our algorithm. Note that this approach is oblivious to the actual problem as long as it satisfies some basic conditions such as the existence of a $2^n n^{\mathcal{O}(1)}$ algorithm and being able to contract edges disjoint from a solution "safely", i.e., the solution is not affected.

Our first compression procedure bounds the number of edges linearly in the optimal solution, but this could still leave a dense graph as the solution size could be superlinear in the number of vertices. However, for several problems, it is known that they admit significantly faster algorithms on instances where solution size is linear in the number of vertices, in contrast to general instances. Hence, a natural question arises here. Could one reduce the solution size to linear in the number of vertices, at least in the case where we are willing to settle for a near-optimal solution, so that the aforementioned faster algorithms could be exploited?

In the second compression procedure, using cut sparsifiers (this time, inspired by Benczúr and Karger [2]) we introduce "solution linearization" as a methodology to give an approximation-preserving reduction to the regime where solution size is linear in the number of vertices for certain cut problems. As before, we convey the main insights using the following scenario. Fix $\epsilon > 0$ and suppose that OPT is at least n/ϵ (we will need this lower bound to apply union bound when performing our probability computation). Suppose we aim to reduce the value of OPT to αn. Towards this, let us sample each edge with probability $p = \alpha n/\mathsf{OPT}$. Notice that in the sampled instance, for any partition (A, B) of the vertex set, the expected number of edges in this cut is p times the original size. Thus, the expected size of the "new" optimal solution is αn. However, in order to show that an optimum solution to the sampled instance leads to a $(1+\epsilon)$-approximation in the original instance, we argue that with good probability, *every* cut is within a $(1 \pm \epsilon)$ factor of its expected value. The choice of lower bound on OPT enables us to achieve this using Chernoff bounds.

Using this compression procedure, we obtain the first *polynomial-space* approximation schemes faster than $2^n n^{\mathcal{O}(1)}$ for Minimum bisection (find a partition (A, B) of the vertex set minimizing the number of edges in the cut and such that $|A|$ and $|B|$ differ by at most 1) and Edge Bipartization (delete fewest number of edges to obtain a bipartite graph).

▶ **Theorem 3.** *For every $\epsilon > 0$,* Minimum Bisection *can be $(1+\epsilon)$-approximated in time $2^{f(\epsilon)n}n^{\mathcal{O}(1)}$ for some $f(\epsilon) < 1$ and polynomial space.*

▶ **Theorem 4.** *For every $\epsilon > 0$,* Edge Bipartization *can be $(1+\epsilon)$-approximated in time $2^{f(\epsilon)n}n^{\mathcal{O}(1)}$ for some $f(\epsilon) < 1$ and polynomial space.*

In a seminal paper, Williams [28] gave $2^{\omega n/3}n^{\mathcal{O}(1)}$-time exponential-space algorithms for both these problems. It has remained open since then whether it is possible to achieve better than $2^n n^{\mathcal{O}(1)}$ time with polynomial space [29, 30]. Using the insights gained while proving the above results, we also design the first polynomial-space exact algorithms for both problems above, that run in time faster than $2^n n^{\mathcal{O}(1)}$, *in the regime* where solution size is linear in the number of edges. Here, we would like to note that while Edge Bipartization and Max Cut are the same as far as exact algorithms are concerned, they exhibit a stark difference in terms of approximation. In particular, an exponential-time approximation scheme for Max-SAT implies the same for Max Cut. However, this does not translate to approximation for Edge Bipartization.

In summary, our use of randomized contraction and cut sparsifiers in the area of exponential-time algorithms is novel to the best of our knowledge and forms the conceptual contribution of our paper.

2 Our Framework: Compression Procedure 1

In this section, we present our first compression procedure based on randomized contraction that leads to Theorem 1.

▶ **Definition 5.** The input to a *graph partitioning* problem Π is a multigraph G and a set of constraints given implicitly or explicitly. The goal is to partition the vertex set of G such that the constraints are met by the partition and the number of edges going across distinct sets of the partition (called the *solution size*) is minimized.

Notice that k-way cut, Multiway Cut, Steiner k-cut, Multicut, Edge Bipartization and Minimum Bisection are all graph partitioning problems. In the case of Steiner k-cut, the constraints are implied by the set X of terminals and the number k. Similarly, for Multicut, the terminal pairs express the constraints, and so on.

We next identify four properties that a graph partitioning problem is required to satisfy in order to be amenable to our framework.

▶ **Definition 6** (Well-behaved problems)**.** We say that a graph partitioning problem Π is *well-behaved* if it has the following properties.

1. It can be solved optimally in time $2^n n^{\mathcal{O}(1)}$ on a given n-vertex (multi)graph G. Call this algorithm `ExactAlgorithm`.
2. If $S \subseteq E(G)$ is an optimal solution for Π, then S remains an optimal solution for the graph G' obtained by contracting an edge $uv \notin S$ and deleting any self-loops that arise.
3. There is a polynomial-time algorithm, called `Contract`, which takes a multigraph G and an edge $uv \in E(G)$, and returns a graph G' such that the following holds. Either, G' is obtained by contracting the edge uv, as well as making some problem-specific modifications [2] (e.g., modifying the set of terminal pairs when Π is Multicut); or if the contraction of uv is disallowed (e.g., if the edge is between a pairs of a request for Multicut or between terminals for Multiway Cut), then $G' := \perp$ is an invalid graph. Further, there is a polynomial-time algorithm `ValidityCheck`(G') that returns `FAIL` iff G' is $\perp$.

[2] We do not attempt to give a formal definition of *problem-specific modifications*. Intuitively speaking, if $G' = G/uv$, then these are the appropriate modifications to the other parts of the input, that make the most sense depending on the edge uv and the problem.

4. Given any $X \subseteq V(G)$, one can define a "projected instance" of Π on the (multi)graph $G' = G - X$, such that $\mathsf{OPT}_\Pi(G') \leq \mathsf{OPT}_\Pi(G)$. Further, given a feasible solution $S' \subseteq E(G')$ for the projected instance and the corresponding partition $\mathcal{P}'$ of $V(G) \setminus X$ obtained by deleting S' from G', there exists a polynomial-time algorithm to extend $\mathcal{P}'$ to a partition $\mathcal{P}$ of $V(G)$ by appropriately adding the vertices of X to various parts (or creating new parts) such that, if $S \subseteq E(G)$ is the set of edges whose deletion leads to $\mathcal{P}$, then it must satisfy $|S| \leq |S'| + \sum_{v \in X} deg_G(v)$.

Property 2 is trivial to check for our problems in Theorem 1. To ensure that Property 3 holds, we do the following. For Steiner k-cut (and so, for Multiway Cut), if an edge between a terminal and a non-terminal is contracted, then the new vertex is a terminal. For Multiway Cut, it is forbidden to contract an edge between two terminals. Similarly, for Multicut, if an edge between a terminal and a non-terminal is contracted, then the new vertex is a terminal. If an edge uv is contracted where both endpoints are elements of requests, say s_i, t_i and s_j, t_j where $u = s_i$ and $v = s_j$, then the new vertex x forms a request with both t_i and t_j. An edge between two endpoints of a request cannot be contracted. Property 4 is not difficult to show for the problems in Theorem 1 and Corollary 2 as well. Let X be a vertex set as described in this property. Indeed, for Multicut, we simply make the vertices of X singleton sets of the new partition. For Steiner k-cut, we begin by making the vertices of X singleton sets of the new partition and then merging sets (subject to obeying the constraint) until we have only k sets in the partition. In the next section, we show that Property 1 holds for these problems.

For the rest of this section, we assume that Π is a well-behaved problem.

2.1 Sparse Algorithm

We first prove that at any stage of our algorithm, if we are dealing with a graph where the number of edges in the graph is at most β times the optimal solution in the original graph, then Π can be $(1+\epsilon)$-approximated in better than $2^n n^{\mathcal{O}(1)}$-time. Formally, we prove the following.

▶ **Lemma 7.** *Let $\gamma \in [0,1]$ be fixed, $\alpha > 0, \beta > 2$, and $\epsilon \in (0,1)$. There exists an algorithm* `SparseAlgorithm` *that, given $(G', \alpha, \beta, \epsilon)$ where G' is a multigraph on n' vertices satisfying:*

- $\gamma n \leq n' \leq n$,
- $|E(G')| \leq \alpha\beta n$, *and*
- $\mathsf{OPT}(G') = \alpha n$,

runs in time $2^{(1-c)n'} \cdot n^{\mathcal{O}(1)}$ for some constant $c = c(\beta, \epsilon) > 0$, and returns a solution $S \subseteq E(G')$ such that $|S| \leq (1+\epsilon)\mathsf{OPT}(G')$.

Proof. Since $|E(G')| \leq \alpha\beta n \leq \frac{\alpha\beta n'}{\gamma}$, the average degree of a vertex in G' is bounded by $\frac{2\alpha\beta}{\gamma}$. Thus, there exists a subset $X \subseteq V(G')$ such that (1) $\sum_{v \in X} d(v) \leq \epsilon\alpha n$, and $|X| \geq \frac{\gamma\epsilon n}{4\beta} \geq \frac{\gamma\epsilon n'}{4\beta}$. This can be computed greedily. Hence, we let S' be an optimal solution for $G' - X$ returned by calling `ExactAlgorithm` on the graph $G' - X$. Then, we use Property 4 to obtain a solution $S \subseteq E(G')$ with $|S| \leq |S'| + \sum_{v \in X} deg(v) \leq \mathsf{OPT}(G' - X) + \epsilon \cdot \mathsf{OPT}(G') \leq (1+\epsilon) \cdot \mathsf{OPT}(G')$. The running time is dominated by `ExactAlgorithm` that takes time $2^{n'-|X|} \cdot n^{\mathcal{O}(1)} \leq 2^{n' - \frac{\gamma\epsilon n'}{4\beta}} \cdot n^{\mathcal{O}(1)} = 2^{(1-c)n'} \cdot n^{\mathcal{O}(1)}$ for $c = \frac{\gamma\epsilon}{4\beta}$. Recall that γ is fixed. Hence, this completes the proof of the lemma. ◀

Algorithm 1 `RandomContraction`(G_i, α, ϵ).

$n = \#$ vertices in the *original* graph G and $\alpha > 0$ s.t. αn is a conjectured value of $\mathsf{OPT}(G)$
$\beta \geq 2, \gamma \in [0,1]$, and integer $q \geq 1, t \geq 1$ are parameters that can be suitably chosen

1: **for** $r = 1, 2, \ldots, t$ **do** ▷ Repeat each *block* inside t times
2: **for** $\ell = 0, 1, \ldots, q-1$ **do** ▷ Start of a *block*
3: **if** $n_{i+\ell} \leq \gamma n$ **then**
4: $S_r \leftarrow$ `ExactAlgorithm`$(G_{i+\ell})$
5: **continue** to the next iteration of the outer **for** loop (line 1)
6: **end if**
7: **if** $m_{i+\ell} = |E(G_{i+\ell})| \leq \alpha\beta n$ **then**
8: $S_r \leftarrow$ `SparseAlgorithm`$(G_{i+\ell}, \alpha, \beta, \epsilon)$ ▷ Find $(1+\epsilon)$-apx via Lem. 7
9: **continue** to the next iteration of the outer **for** loop (line 1)
10: **else** ▷ i.e., $m_{i+\ell} > \alpha\beta n$
11: Pick an edge uv of $E(G_{i+\ell})$ uniformly at random ▷ accounting for parallel edges
12: $G_{i+\ell+1} \leftarrow$ `Contract`$(G_{i+\ell}, uv)$ ▷ $G_{i+\ell}/uv$ with potentially some more modifications
13: **if** `ValidityCheck`$(G_{i+\ell+1}, \alpha, \epsilon) =$ `FAIL` **then** ▷ Problem-specific validity check
14: $S_r \leftarrow \perp$ and **continue** to the next iteration of the outer **for** loop (line 1)
15: **end if**
16: **end if**
17: **end for**
18: $S_r \leftarrow$ `RandomContraction`$(G_{i+q}, \alpha, \epsilon)$ ▷ Recursive call on G_{i+q} after a completed block.
19: **end for**
20: **return** the best solution out of $S_1, S_2, \ldots, S_t$ ▷ avoiding invalid ($\perp$) solutions

Figure 1 This algorithm uses some problem-specific subroutines, namely `ExactAlgorithm`, `SparseAlgorithm`, `Contract`, and `ValidityCheck` (denoted in blue color).

2.2 Random Contraction for Bounding the Number of Edges

In Algorithm 1, we describe a recursive algorithm called `RandomContraction` that takes as an input a (multi)graph G_i and a parameter $\alpha > 0$. When this algorithm is executed, we will be assuming that $\alpha n = \mathsf{OPT}(G)$, where G is the original graph and $n = |V(G)|$. Note that α is *not necessarily* a constant and can be guessed. Roughly speaking, our strategy is the following. As long as the "current" graph is not small enough compared to the original graph (in which case we just use the assumed exact algorithm) and the total number of edges in the current graph is not linearly bounded by the optimal solution for the original instance (in which case we use the algorithm of Lemma 7), we randomly contract an edge and recurse. We show that with good probability, this strategy succeeds.

We next formally describe `RandomContraction` (also see Algorithm 1). We run an outer **for** loop for $t = 6$ iterations (lines 1-19), which we refer to as a *block*. We compute a candidate solution S_r in each block $r \in \{1, \ldots, t\}$, and in line 20 we return the best solution found over each of the t blocks. The executions in different blocks are independent of each other; in particular, each block starts with the input graph G_i.

Now we describe a particular block $1 \leq r \leq t$. Each block consists of (up to) $q = 17$ iterations of the inner for loop (lines 2-17). At the start of iteration $1 \leq \ell \leq q$, we have a *current* graph $G_{i+\ell}$. First, in line 3 we check whether the number of vertices in the current

graph $G_{i+\ell}$ is at most γn, in which case we use `ExactAlgorithm` to find an optimal solution to the problem on $G_{i+\ell}$. Letting S_r denote the optimal solution for $G_{i+\ell}$ thus found, we continue to the next block in line 5.

Then, in line 7 we check whether the number of edges in the graph $m_{i+\ell}$ is at most $\alpha\beta n$, which forms our "base case". In this case, we use the algorithm of Lemma 7 to find a $(1+\epsilon)$-approximate solution (line 8). By Lemma 7, this algorithm runs in time $2^{(1-c)|V(G_{i+\ell})|} \cdot n^{\mathcal{O}(1)}$ time, where $c = c(\beta, \epsilon)$, and returns a solution S_r, such that $|S_r| \leq (1+\epsilon) \cdot \mathsf{OPT}(G_{i+\ell})$. Then, in line 9, we continue to the next block.

Otherwise (line 10), the number of edges in the graph is more than $\alpha\beta n$, i.e., at least β times the conjectured upper bound on the size of an optimal solution in G. In this case, in line 11, we sample an edge uv from $E(G_{i+\ell})$ uniformly at random. In line 12, we use a problem-specific subroutine `Contract` which essentially returns the graph $G_{i+\ell+1}$ obtained by contracting uv in $G_{i+\ell}$ (we keep parallel edges but remove self-loops); however this subroutine may also perform additional problem-specific steps, such as modifying terminal sets (in a problem such as Edge Multiway Cut). Then, in line 13, we perform a problem-specific validity check of the graph $G_{i+\ell+1}$, and if $G_{i+\ell+1}$ fails the validity check (this may happen if we just contracted an edge between a terminal pair, in case of Edge Multiway Cut), then we define S_r to be an invalid solution.

Otherwise, we continue to the next *iteration* in the current block; and this happens up to q iterations. If the block runs for exactly q iterations without satisfying the **if** conditions in lines 7 or 13, then we refer to it as a *completed block*. At the end of a completed block, we recursively call the algorithm on the graph G_{i+q}, and obtain a solution S_r.

We set the values of the parameters used in Algorithm 1, as follows. $\beta = 10, t = 6$ and $q = 17$. These values are chosen so as to satisfy the following properties and make subsequent probability computations go through.

1. $\left(\frac{\beta}{\beta-1}\right)^q \approx 5.996$. In other words, $\frac{1.0006}{\left(\frac{\beta-1}{\beta}\right)^q} \lessapprox 6 = t$.
2. Let β' satisfy that $6^{1/17} = 1.111152 = \frac{\beta'}{\beta'-1}$, then $\beta' \approx 9.9966$.

Now we are ready to prove that the running time of the algorithm is bounded away from 2^n – note that throughout the proof, n denotes the number of vertices in the *original graph*.

▶ **Lemma 8.** *There exists a constant $\gamma \in [0,1]$ such that for any $0 < \epsilon < 1, \alpha > 0, \beta > 2$ and any (multi)graph G with $\mathsf{OPT}(G) \leq \alpha n$, `RandomContraction`$(G, \alpha, \epsilon)$ runs in time $\mathcal{O}^*(c'^n)$, where $c' = \max\left\{1.112, 2^{1-c}\right\}$, and with probability at least 10^{-5}, returns a solution $S \subseteq E(G)$ such that $|S| \leq (1+\epsilon) \cdot \mathsf{OPT}(G)$. Here, c is the constant from Lemma 7.*

Proof. We divide the proof into two parts: (1) running time, and (2) approximation guarantee and the success probability.

Running time. We initially call `RandomContraction`(G_0, α, ϵ), where $G_0 = G$ is the original graph with $\mathsf{OPT}(G) \leq \alpha n$, where $n = |V(G)|$. Consider the execution of `RandomContraction` on a graph G_i with $n_i = \lambda n$ vertices for some $\gamma < \lambda \leq 1$, and consider a particular block $1 \leq r \leq t$.

Suppose the block does not complete. This can happen due to two reasons. (1) In line 3, the number of vertices drops to at most γn, and we use `ExactAlgorithm` to compute an optimal solution for $G_{i+\ell}$, which takes at $2^{\gamma n} n^{\mathcal{O}(1)}$ time. Otherwise, (2) in some iteration $m_{i+\ell} \leq \alpha\beta n$, in which case the **if** condition in line 7 holds. Then, the number of vertices in $G_{i+\ell}$ is $n_i - \ell$, which is at least $n_i - q$ (recall that $\ell < q$). Therefore, Lemma 7 implies that `SparseAlgorithm` on $G_{i+\ell}$ runs in time at most $2^q \cdot 2^{(1-c)\cdot(1-\lambda)n} \cdot n^{\mathcal{O}(1)} = 2^{(1-c)\cdot(1-\lambda)n} \cdot n^{\mathcal{O}(1)}$,

since 2^q is a constant. Further, recall that $c = c(\beta, \epsilon)$ is a constant. Finally, if the block completes with q iterations, then we make a recursive call to `RandomContraction` on the graph G_{i+q}.

Now we argue that that the algorithm runs in time $2^{(1-c)n}$. Note that the running time is upper bounded by

$$T(n) \leq \max\left\{\left(\frac{\beta'}{\beta'-1}\right)^{(1-\gamma)n} \cdot 2^{\gamma n} \cdot n^{\mathcal{O}(1)}, \max_{\gamma \leq \lambda \leq 1}\left(\frac{\beta'}{\beta'-1}\right)^{(1-\lambda)n} \cdot 2^{(1-c)\lambda n} \cdot n^{\mathcal{O}(1)}\right\}$$

We analyze each of the two terms separately. Note that the first term is $2^{(\lg(\beta'/(\beta'-1))(1-\gamma)+\gamma)n}$. $n^{\mathcal{O}(1)} \leq 2^{(1-c_\gamma)n} \cdot n^{\mathcal{O}(1)}$ for some constant $c_\gamma > 0$, since γ can be chosen to be bounded away from 1.

Now we bound the second term. Let $f(\lambda) \coloneqq \lg\left(\frac{\beta'}{\beta'-1}\right) \cdot (1-\lambda) + (1-c) \cdot \lambda$. Then, in order to show that the running time of the algorithm is $2^{(1-c)n}$, we need to equivalently argue that $\max_{0 \leq \delta \leq 1} f(\lambda) \leq 1 - c$, where $c = c(\beta, \epsilon) > 0$.

Consider,

$$\frac{d}{d\lambda} f(\lambda) = -\lg\left(\frac{\beta'}{\beta'-1}\right) + (1-c) \tag{1}$$

Recall that $\beta' = 9.9966 \implies \lg\left(\frac{\beta}{\beta-1}\right) \approx 0.152$. Now we consider two cases. First, if $\frac{d}{d\lambda} f(\lambda) \geq 0$, i.e., $0.152 \leq (1-c)$, then the function $f(\lambda)$ is increasing in $[\gamma, 1]$, and the maximum is achieved at $\lambda = 1$. In this case, the running time of the algorithm is bounded by $2^{(1-c)n} \cdot n^{\mathcal{O}(1)}$. Otherwise, $\frac{d}{d\lambda} f(\lambda) < 0$, i.e., $0.152 \geq (1-c)$, then the function $f(\lambda)$ is decreasing in $[\gamma, 1]$, and the maximum is achieved at $\lambda = \gamma$. In this case, the running time of the algorithm is bounded by $\left(\frac{\beta'}{\beta'-1}\right)^{(1-\gamma)n} \cdot 2^{(1-c)\gamma n} \cdot n^{\mathcal{O}(1)}$, which is bounded away from 2^n as argued for the first term.

Success Probability. We start with the following technical claim.

▷ Claim 9. Let $G_i, G_{i+1}, \ldots, G_{i+t}$ be a maximal sequence of multigraphs obtained during the execution of a particular block (Algorithms 1–1) for some $0 \leq t \leq q-1$, where each G_{i+j+1} is obtained by contracting a random edge of G_{i+j}, such that $m_i \geq m_{i+1} \geq \ldots \geq m_{i+t} \geq \alpha\beta n$. Then, with probability at least $\left(\frac{\beta-1}{\beta}\right)^t \geq \left(\frac{\beta-1}{\beta}\right)^q \approx 0.16677$, it holds that $\mathsf{OPT}(G_i) = \mathsf{OPT}(G_{i+1}) = \ldots = \mathsf{OPT}(G_{i+t})$.

Proof. Since G_{i+j+1} is a graph obtained after contraction of an edge in G_{i+j}, the set of feasible solutions in G_{i+j+1} is a subset of that in G_{i+j}. Thus, $\mathsf{OPT}(G_{i+j+1}) \geq \mathsf{OPT}(G_{i+j})$, regardless of which edge is contracted. It follows that $\mathsf{OPT}(G_{i+t}) \geq \ldots \geq \mathsf{OPT}(G_{i+1}) \geq \mathsf{OPT}(G_i)$. Now we prove the other direction.

Let $S \subseteq E(G_i)$ be the set of edges corresponding to an optimal solution for G_i. Then, in the first iteration of the for loop, the probability that the edge uv is picked in line 11 belongs to S, is at most $\frac{|S|}{m_i} \leq \frac{\alpha n}{\alpha\beta n} = \frac{1}{\beta}$. Thus, with probability at least $\frac{\beta-1}{\beta}$, the set S remains in $E(G_{i+1})$, and Property 2 implies that S is an optimal solution for G_{i+1}. [3] This lower bound on the probability holds in each of the iterations, since the number of edges in $G_i, G_{i+1}, \ldots, G_{i+t}$ is at least $\alpha\beta n$. Thus, with probability at least $\frac{\beta-1}{\beta}^t \geq \left(\frac{\beta}{\beta-1}\right)^q = \left(\frac{10}{9}\right)^{17} \approx 0.16677$, it holds that $\mathsf{OPT}(G_{i+t}) \leq \ldots \leq \mathsf{OPT}(G_{i+1}) \leq \mathsf{OPT}(G_i)$, which shows the claim. ◁

[3] Note that due to vertex contraction, some edges in S may be incident to a contracted vertex, in which case their "name" might change.

Let $\mathcal{B}(n_i)$ be the event that $\texttt{RandomContraction}(G_i, \alpha)$ does not return a $(1+\epsilon)$-approximate solution, where G_i is a multigraph on $n_i \leq n$ vertices such that $\mathsf{OPT}(G_i) \leq \alpha n$. We inductively assume that $\Pr(\mathcal{B}(n_i')) \leq \delta = 0.99999$ for all $n_i' < n_i$, and aim to show $\Pr(\mathcal{B}(n_i)) \leq \delta$. In the base case, we use `SparseAlgorithm` on the current graph $G_{i+\ell}$, and due to Lemma 7, we find a $(1+\epsilon)$-approximate solution of G_i. Now we proceed to the inductive case.

Now consider $\texttt{RandomContraction}(G_i, \alpha, \epsilon)$ with $|V(G_i)| = n_i$ and $\mathsf{OPT}(G_i) \leq \alpha n$. In the execution of `RandomContraction`, we say that a block $1 \leq r \leq t$ is *bad* if the solution S_r found is not a $(1+\epsilon)$-approximate solution for G_i; and *good* otherwise. We show the following claim.

▷ Claim 10. For any block $1 \leq r \leq t$, $\Pr(\text{Block } r \text{ is bad}) \leq \left(1 - (1-\delta) \cdot \left(\frac{\beta-1}{\beta}\right)^q\right)$.

Proof. There are two cases.

In the first case, suppose that the block is incomplete, i.e., for some $0 \leq \ell < q-1$, the line 7 does not hold, and S_r is equal to a $(1+\epsilon)$-approximation of $G_{i+\ell}$, computed using $\texttt{SparseAlgorithm}(G_{i+\ell}, \alpha, \epsilon)$. In this case, the block r is good iff $\mathsf{OPT}(G_{i+\ell}) = \mathsf{OPT}(G_i)$, since `SparseAlgorithm` is a deterministic algorithm that always returns a $(1+\epsilon)$-approximate solution of the given graph. From Claim 9, the probability that the block is good is at least $\left(\frac{\beta-1}{\beta}\right)^q$, which implies that the probability that the block is bad is at most $1 - \left(\frac{\beta-1}{\beta}\right)^q \leq 1 - (1-\delta) \cdot \left(\frac{\beta-1}{\beta}\right)^q$.

In the second case, suppose the block is complete. Then, $S_r = \texttt{RandomContraction}(G_{i+q})$. Fix an optimal solution S of G_i. Note that the block is bad only if either of the following two events happen:

1. $\mathcal{B}_1$ be the event that we contract an edge of S at any point during the block.
2. $\mathcal{B}_2$ be the event that, conditioned on $\mathcal{B}_1$ not occurring, i.e., conditioned on S surviving in G_{i+q}, $\texttt{RandomContraction}(G_{i+q})$ does not return a $(1+\epsilon)$-approximate solution. Note that, the number of vertices of G_{i+q} is smaller than G_i. Thus by induction hypothesis, the event $\mathcal{B}_2$ happens with probability at most $(1 - \Pr(\mathcal{B}_1)) \cdot \Pr(\mathcal{B}(n_{i+q}))$.

Therefore,

$$\begin{aligned}
\Pr(|S_r| > (1+\epsilon) \cdot \mathsf{OPT}(G_i)) &\leq \Pr(\mathcal{B}_1) + (1 - \Pr(\mathcal{B}_1)) \cdot \Pr(\mathcal{B}(n_{i+q})) \\
&\leq \Pr(\mathcal{B}_1) + (1 - \Pr(\mathcal{B}_1)) \cdot \delta \\
&\qquad \text{(Using inductive hypothesis on } \Pr(n_{i+q})) \\
&\leq \left(1 - (1-\delta) \cdot \left(\frac{\beta-1}{\beta}\right)^q\right) \qquad \text{(via Claim 9)}
\end{aligned}$$

Thus, the claimed upper bound holds in each of the two cases. ◁

Given Claim 10, we finish induction as follows. $\Pr(\mathcal{B}(n_i))$ is upper bounded by the probability of the event that each of the r blocks is bad. Since the events corresponding to different blocks are pairwise independent, it follows that,

$$\begin{aligned}
\Pr(\mathcal{B}(n_i)) \leq \left(1 - (1-\delta)\left(\frac{\beta-1}{\beta}\right)^q\right)^t &\leq \exp\left(-(1-\delta)t \cdot \left(\frac{\beta-1}{\beta}\right)^q\right) \\
&\leq \exp\left(-(0.00001) \cdot (1.0006)\right) \\
&\qquad \text{(Recalling that } 6 \cdot \left(\frac{\beta-1}{\beta}\right)^q \geq 1.0006) \\
&\leq 0.99998 \leq \delta.
\end{aligned}$$

This finishes the proof of the lemma. ◀

Note that the space complexity of our framework is effectively dominated by that of the exact algorithm we use. To obtain Theorem 1, we use exact algorithms that may take exponential space.

3 Applications of our Framework

In this section, we show how to apply this framework to obtain our approximation schemes for Steiner k-Cut (which generalizes both k-Way Cut and Edge Multiway Cut), and Edge Multicut.

Towards this, we need to show that the 4 properties stated in Definition 6 hold. We have already argued that the last three properties hold. The only non-trivial property is Property 1 (i.e., existence of a $2^n n^{\mathcal{O}(1)}$-time exact algorithm) and so, we focus on that for the rest of this section.

3.1 Property 1: Exact Algorithms for our Problems

Our algorithms use the *fast subset convolution* framework that we define below.

Subset Convolutions. Given two functions $f, g : 2^X \to \mathbb{Z}$, the *subset convolution* of f and g is the function $(f * g) : 2^X \to \mathbb{Z}$, defined as follows.

$$\forall Y \subseteq X : \qquad (f * g)(Y) = \sum_{\substack{A \cup B = Y \\ A \cap B = \emptyset}} f(A) \cdot g(B) \tag{2}$$

It is known that, given all the 2^n values of f and g in the input, all the 2^n values of $f * g$ can be computed in $\mathcal{O}(2^n \cdot n^3)$ arithmetic operations (see, for example, Theorem 10.15 in [10]). This is known as *fast subset convolution*. Now, let

$$(f \oplus g)(Y) = \min_{\substack{A \cup B = Y \\ A \cap B = \emptyset}} f(A) + g(B).$$

We observe that $f \oplus g$ is equal to the subset convolution $f * g$ in the integer min-sum semiring $(\mathbb{Z} \cup \{\infty\}, \min, +)$, i.e., in Equation (2), we use the mapping $+ \mapsto \min$, and $\cdot \mapsto +$. This, combined with a simple "embedding trick" enables one to compute all values of $f \oplus g : 2^X \to \{-N, \ldots, N\}$ in time $2^n n^{\mathcal{O}(1)} \cdot \mathcal{O}(N \log N \log\log N)$ using fast subset convolution – see [10, Theorem 10.17] and [4, Theorem 3]. We summarize this discussion in the following proposition.

▶ **Proposition 11.** *The subset convolution over the integer min-sum semiring can be computed in $2^n n^{\mathcal{O}(1)} \cdot \mathcal{O}(M \log M \log\log M)$ time, provided that the range of the input functions is $\{-M, -M+1, \cdots, M\}$.*

3.1.1 Edge Multicut

Recall that the input to the Multicut problem consists of an undirected graph G and pairs of terminals $\{s_1, t_1\}, \cdots, \{s_\ell, t_\ell\}$; the task is to remove a minimum set of edges such that s_i and t_i are disconnected for every $1 \leq i \leq \ell$. A simple dynamic programming algorithm running in time $3^n n^{\mathcal{O}(1)}$ follows from the following recurrence. For a vertex subset $X \subseteq V(G)$, let $f(X)$ denotes the size of a minimum set of edges required to delete from $G_X = G[X]$ in order to separate the pairs s_i and t_i for every $1 \leq i \leq \ell$, such that $\{s_1, t_1\} \subseteq X$ (of course

from the graph G_X). Let $X \subseteq V$ is called *valid* if for every $j \in [i], |X \cap \{s_j, t_j\}| \leq 1$. For a subset $X \subseteq V(G)$, let $\partial(X)$ denotes the set of edges with exactly one endpoint in X. Define $g : 2^{V(G)} \to \mathbb{Z}$, as $g(X) = |\partial(X)|$, $X \subseteq V(G)$. Clearly,

$$f(X) = \min_{\substack{X' \subsetneq X \\ X' \text{ is valid}}} f(X \setminus X') + g(X')$$

Since, each term takes $2^{|X|}n^{\mathcal{O}(1)}$ time for evaluation, we have that $\sum_{i=0}^{n} \binom{n}{i} 2^i n^{\mathcal{O}(1)} = 3^n n^{\mathcal{O}(1)}$.

Now to convert the above recurrence to fit into subset convolution framework, we define "a ranked version of f". Let $f_i(X)$ be the minimum number of edges required to delete from $G_X = G[X]$ in order to separate the pairs s_j and t_j for every $1 \leq j \leq i$ (of course from the graph G_X). Hence $f_\ell(V(G))/2$ is the solution to our problem (factor of $1/2$ comes from the fact that every cut edge is counted twice).

We evaluate the functions f_i in the increasing order of i. The base function is f_0, which is defined as $f_0[X] = 0$, for every subset $X \subseteq V(G)$. Let $g_i : 2^{V(G)} \to \mathbb{Z}$ be a function defined as follows. For $X \subseteq V(G)$, if X is valid and $|X \cap \{s_i, t_i\}| = 1$, then define $g_i(X) = |\partial(X)|$, else $g_i(X) = \infty$ (here, ∞ just denotes a large positive integer). The recurrence for $f_i(X)$ is defined as follows (and indeed it depends on $f_{i-1}(X)$).

$$f_i(X) = \min_{X' \subseteq X} \Big\{ f_{i-1}(X \setminus X') + g_i(X') \Big\}$$

For each integer $i = 1, 2, \ldots, \ell$, we assume that the function f_{i-1} is computed. Hence, using Proposition 11 and the preceding discussion we get the following.

▶ **Lemma 12.** *Given all the 2^n values of f_{i-1} and g in the input, all the 2^n values of f_i can be computed in time $2^n n^{\mathcal{O}(1)}$. In particular,* Multicut *can be solved in time $2^n n^{\mathcal{O}(1)}$.*

3.1.2 Steiner k-Cut

Recall that the input to the Steiner k-Cut problem consists of an undirected graph G, a set of terminals $T \subseteq V(G)$ of size at least k; the task is to remove a minimum set of edges whose removal results in k connected components, each of which contains at least one terminal. For a vertex subset $X \subseteq V(G)$, let $f_i(X)$ denotes the size of a minimum set of edges required to delete from $G_X = G[X]$ such that removal of these edges results in i connected components, each of which contains at least one terminal. For a subset $X \subseteq V(G)$, recall that $\partial(X)$ denotes the set of edges with exactly one endpoint in X. Let $g : 2^{V(G)} \to \mathbb{Z}$ be a function. For, $X \subseteq V(G)$, if $|X \cap T| \geq 1$, then define $g(X) = |\partial(X)|$, else define $g(X) = \infty$. Hence $f_k[V(G)]$ is the solution to our problem. We evaluate the functions f_i in the increasing order of i. The base function is f_1, which is defined as $f_1[X] = 0$, for every subset $X \subseteq V(G)$ such that $|X \cap T| \geq 1$; and ∞ otherwise (here, ∞ just denotes a large positive integer). The value of $f_i(X)$ is governed by the following recurrence.

$$f_i(X) = \min_{X' \subseteq X} \Big\{ f_{i-1}(X \setminus X') + g(X') \Big\}$$

For each integer $i = 1, 2, \ldots, k$, we assume that the function f_{i-1} is computed. Hence, using Proposition 11 and the preceding discussion we get the following.

▶ **Lemma 13.** *Given all the 2^n values of f_{i-1} and g in the input, all the 2^n values of f_i can be computed in time $2^n n^{\mathcal{O}(1)}$. In particular,* Steiner k-Cut *can be solved in time $2^n n^{\mathcal{O}(1)}$.*

4 Our Framework: Compression Procedure 2

First in Section 4.1 we define some necessary definitions and prove some basic properties. Then, in Section 4.2, we state and prove the properties of our solution linearlization procedure. This will then be used later to prove Theorem 3 and Theorem 4.

4.1 Notation and Basic Properties

Let $G = (V, E)$ be a finite undirected multigraph without self loops (henceforth, simply a *multigraph*). Let $\mathcal{P}$ be an ℓ-partition of $V(G)$ for some $\ell \geq 1$, i.e., $\mathcal{P}$ is a partition of $V(G)$ with exactly ℓ non-empty parts. Then, we define $\mathsf{cut}_G(\mathcal{P})$ as the multiset of edges with endpoints in different parts of $\mathcal{P}$, and $\kappa_G(\mathcal{P}) = |\mathsf{cut}_G(\mathcal{P})|$. We also refer to the edges in $\mathsf{cut}_G(\mathcal{P})$ as the *exterior* edges of the partition $\mathcal{P}$. We sometimes write $\mathsf{cut}_G(V_1, \ldots, V_\ell)$ (resp. $\kappa_G(V_1, \ldots, V_\ell)$) to denote $\mathsf{cut}_G(\{V_1, \ldots, V_\ell\})$ (resp. $\kappa_G(\{V_1, \ldots, V_\ell\})$), and may omit the subscript from $\mathsf{cut}_G(\cdot)$ and $\kappa_G(\cdot)$ when the graph is clear from the context. Analogously, we define $\mathsf{int}_G(\mathcal{P})$ as the multiset of edges of G whose both endpoints belong to the same part of $\mathcal{P}$, and $\iota_G(\mathcal{P}) = |\mathsf{int}_G(\mathcal{P})|$. We call the edges in $\iota_G(\mathcal{P})$ the *interior* edges of the partition $\mathcal{P}$. The same notational conventions as $\mathsf{cut}.(\cdot)$ apply to $\mathsf{int}.(\cdot)$ and $\iota.(\cdot)$. We say that a partition $\mathcal{P}'$ coarsens another partition $\mathcal{P}$ if every set in $\mathcal{P}'$ is comprised of the union of some subset of $\mathcal{P}$.

▶ **Proposition 14** (Chernoff bound). *Let $X_1, X_2, \ldots, X_n$ be independent random variables taking values in $\{0, 1\}$, and let $X = \sum_{i=1}^{n} X_i$ with $\mu = E[X]$. Then, for any $0 < \delta < 1$, the following inequality holds:*

$$\Pr\left(|X - \mu| \geq \delta\mu\right) \leq 2 \cdot e^{-\mu\delta^2/3} \tag{3}$$

▶ **Definition 15** ((α, β)-coarsening partition of $\mathcal{P}$). Let G be a multigraph and let $\mathcal{P}$ be a partition of $V(G)$. We say that $\mathcal{P}'$ is a (α, β)-*coarsening of* $\mathcal{P}$ if the following hold.

1. $\mathcal{P}'$ coarsens $\mathcal{P}$.
2. $\mathcal{P}'$ contains at most α parts.
3. $\kappa_G(\mathcal{P}') \geq \beta \cdot \kappa_G(\mathcal{P})$.

In the above definition, one can think of the partition $\mathcal{P}'$ as an approximation of the partition $\mathcal{P}$. Depending on the application, we will set α and β appropriately.

▶ **Lemma 16.** *Let G be a multigraph. For any integer $i \geq 0$, there exists a t-partition $\mathcal{P}$ of $V(G)$ such that (i) $1 \leq t \leq 2^i$, and (ii) $\kappa(\mathcal{P}) \geq (1 - 2^{-i}) \cdot |E(G)|$.*

Proof. We prove this by induction on i. Note that if $i = 1$, then the claim trivially follows by placing all the vertices into a single part. Now suppose that the claim holds for some $i > 0$ and we prove it for $i + 1$. To this end, let (V_1, V_2) be a maximum cut of $V(G)$. It is a well-known fact that $\kappa_G(V_1, V_2) \geq |E(G)|/2$. Let $G_j = G[V_j]$ and $E_j = E(G_j)$ for $j \in \{1, 2\}$. Note that $E(G) = \mathsf{cut}_G(V_1, V_2) \uplus E_1 \uplus E_2$. It follows that, $|E_1| + |E_2| \leq |E(G)|/2$.

By induction, there exists a t'-partition $\mathcal{P}_j$ of V_j such that $\kappa_{G_j}(\mathcal{P}_j) \geq (1 - 2^{-i}) \cdot |E_j|$ for $j \in \{1, 2\}$, and each $\mathcal{P}_j$ contains at most 2^i parts. We claim that $\mathcal{P} := \mathcal{P}_1 \cup \mathcal{P}_2$, which is a partition of $V(G)$, satisfies the desired properties. Note that number of parts in $\mathcal{P}$ is at most $2 \cdot 2^i = 2^{i+1}$. Now we prove the second property.

$$\begin{aligned}|\mathsf{cut}_G(\mathcal{P})| = \sum_{U_q \in \mathcal{P}} |\partial_G(U_q)| &= \sum_{U_q \in \mathcal{P}_1} |\partial_G(U_q)| + \sum_{U_q \in \mathcal{P}_2} |\partial_G(U_q)| \\ &= \mathsf{cut}_{G_1}(\mathcal{P}_1) + \mathsf{cut}_{G_2}(\mathcal{P}_2) + \kappa_G(V_1, V_2) \\ &\geq (1 - 2^{-i}) \cdot (|E_1|) + (1 - 2^{-i}) \cdot |E_2|) + \kappa_G(V_1, V_2) \\ &= |\mathsf{cut}_G(V_1, V_2)| - 2^{-i} \cdot (|E_1| + |E_2|) \\ &\geq |E(G)| - 2^{-i} \cdot |E(G)|/2 = (1 - 2^{-(i+1)}) \cdot |E(G)|. \end{aligned}$$

◀

▶ **Corollary 17.** *Let G be a multigraph on $n \geq 1$ vertices, and $\mathcal{P}$ be a partition of $V(G)$. Then, for any $0 < \epsilon \leq 1$, there exists a partition $\mathcal{P}'$ of $V(G)$ such that $\mathcal{P}'$ is a $(\lceil 1/\epsilon \rceil, 1 - \epsilon)$ coarsening of $\mathcal{P}$.*

Proof. Let H be a multigraph obtained by contracting each part of $\mathcal{P}$ into a single vertex. The claim then follows by invoking Lemma 16 on H with $i = \lceil \log(1/\epsilon) \rceil$. ◀

4.2 Properties of Solution Linearization

We next prove two lemmas for solution linearization. The first lemma (Lemma 18) "preserves" interior edges of partitions with bounded number of parts. This makes it useful for problems such as Edge Bipartization, where the solution edges are deleted from within the sets of the optimal partition. On the other hand, the second lemma (Lemma 19) achieves the same goal, but for exterior edges of partitions. This makes it useful for problems such as Minimum Bisection. Moreover, the second lemma works for arbitrarily many parts, which could find applications for other problems as well.

▶ **Lemma 18.** *For any (multi)graph G on n vertices, $0 < \epsilon < 1$, and a parameter $t \geq 1$ independent of n, if $\beta \geq \tau \geq \frac{3}{\epsilon^2} \ln(3(t+1))$, then the (multi)graph H obtained by sampling each edge of G independently with probability p, where $p = \frac{\tau}{\beta}$, satisfies the following. With probability at least $1 - (2/3)^n$, for all t'-partitions $\mathcal{P}$ of $V(G)$ with $t' \leq t$, and $\iota_G(\mathcal{P}) \geq \beta n$, it holds that*

$$\left|\iota_H(\mathcal{P}) - \iota_G(\mathcal{P}) \cdot p\right| \leq \epsilon \cdot p \cdot \iota_G(\mathcal{P}) \tag{4}$$

Proof. Fix a (multi)graph G and $0 < \epsilon < 1$ as in the statement of the lemma. Suppose $\beta \geq \tau \geq \frac{3}{\epsilon^2} \ln(3(t+1))$. Let H be a (multi)graph obtained by sampling as stated in the lemma.

Consider a t-partition $\mathcal{P}$ with $\iota_G(\mathcal{P}) \geq \beta n$, and $t \geq 1$ that is independent of n. Note that each edge in $E(G)$ is sampled independently with probability p, which implies that $\mathbb{E}[\iota_H(\mathcal{P})] = p \cdot \iota_G(\mathcal{P}) \geq \frac{\tau}{\beta} \cdot \beta n = \tau n \geq \frac{3}{\epsilon^2} \ln(3(t+1))n$. Using Chernoff Bound (Proposition 14), the following inequality holds:

$$\begin{aligned}\Pr\left(\left|\iota_H(\mathcal{P}) - p \cdot \iota_G(\mathcal{P})\right| \geq \epsilon p \cdot \iota_G(\mathcal{P})\right) &\leq \frac{1}{\exp\left(p \cdot \iota_G(\mathcal{P})\epsilon^2/3\right)} \leq \frac{2}{\exp\left(\frac{3}{\epsilon^2}\ln(3(t+1))n\epsilon^2/3\right)} \\ &= \frac{1}{(3(t+1))^n}.\end{aligned}$$

Note that we use the lower bound on $p \cdot \iota_G(\mathcal{P}) \geq \frac{3}{\epsilon^2} \ln(3(t+1))n$ in the second inequality in the above.

Finally, note that the number of t'-partitions $\mathcal{P}$ satisfying $\iota_G(\mathcal{P}) \geq \beta n$ is upper bounded by the total number of t'-partitions with $1 \leq t' \leq t$, which is $\sum_{t'=1}^{t} (t')^n \leq (t+1)^n$. Thus, by taking a union bound, we obtain that (4) holds with probability at least $1 - \frac{2 \cdot (t+1)^n}{(3(t+1))^n} \geq 1 - (2/3)^n$. ◀

In the next lemma, we show an analogous bound that shows that the sparsification procedure also preserves the *cut sizes* of partitions. However, this lemma is stronger, since we do not require that the number of parts in the partition is bounded by a parameter independent of n. By using Corollary 17, we ultimately reduce the general case to the bounded parts case as in Lemma 18.

▶ **Lemma 19.** *For any (multi)graph G on n vertices, and $0 < \epsilon < 1$, if $\beta \geq \tau \geq \frac{24}{\epsilon^2}\ln(\frac{9}{\epsilon})$, then the (multi)graph H obtained by sampling primitive (G, p), where $p = \frac{\tau}{\beta}$, satisfies the following with probability at least $1 - (2/3)^n$.*

1. *For each partition $\mathcal{P}$ of $V(G)$ with $\kappa_G(\mathcal{P}) \geq \beta n$, there exists $R_G(\mathcal{P}) \subseteq \mathsf{cut}_G(\mathcal{P})$ such that $|R_G(\mathcal{P})| \leq \epsilon/2 \cdot \kappa_G(\mathcal{P})$, and if $R_H(\mathcal{P}) \subseteq R_G(\mathcal{P})$ is the subset of $R_G(\mathcal{P})$ that is sampled in H, then the following holds.*

$$\left| |\mathsf{cut}_H(\mathcal{P}) \setminus R_H(\mathcal{P})| - \kappa_G(\mathcal{P}) \cdot p \right| \leq \epsilon \cdot p \cdot \kappa_G(\mathcal{P}) \tag{5}$$

2. *Furthermore, for any t'-partition $\mathcal{P}'$ with $1 \leq t' \leq \lceil 2/\epsilon \rceil$ and $\kappa_G(\mathcal{P}') \geq \beta n/2$, then the following stronger bound holds.*

$$\left| \kappa_H(\mathcal{P}') - \kappa_G(\mathcal{P}') \cdot p \right| \leq (\epsilon/2) \cdot p \cdot \kappa_G(\mathcal{P}') \tag{6}$$

Proof. The initial setup is similar to Lemma 18 with minor differences. Fix a (multi)graph G and $0 < \epsilon < 1$ as in the statement of the lemma. Let $t \coloneqq \lceil 2/\epsilon \rceil$, and suppose $\beta \geq \tau \geq \frac{24}{\epsilon^2}\ln(\frac{9}{\epsilon}) \geq \frac{24}{\epsilon^2}\ln(3(t+1))$. Let H be a (multi)graph obtained by sampling primitive (G, p), where $p \coloneqq \tau/\beta$ – note that $p \leq 1$.

First, we consider t'-partitions $\mathcal{P}'$ where (1) $1 \leq t' \leq t$, and $\kappa_G(\mathcal{P}') \geq \beta n/2$. Then, by using an argument similar to Lemma 18, an application of Chernoff bound (Proposition 14) implies the following.

$$\begin{aligned}
\Pr\left(\left|\kappa_H(\mathcal{P}') - p \cdot \kappa_G(\mathcal{P}')\right| \geq (\epsilon/2)p \cdot \kappa_G(\mathcal{P}')\right) &\leq \frac{2}{\exp\left(p \cdot \kappa_G(\mathcal{P}')\epsilon^2/12\right)} \\
&\leq \frac{2}{\exp\left(\frac{24}{\epsilon^2}\ln(3(t+1))n\frac{\epsilon^2}{24}\right)} \\
&= \frac{2}{(3(t+1))^n}.
\end{aligned}$$

where we use the assumption that $\mathbb{E}[\kappa_H(\mathcal{P}')] = p \cdot \kappa_G(\mathcal{P}') \geq \tau n/2 \geq \frac{24}{\epsilon^2}\ln(3(t+1)) \cdot \frac{n}{2}$. Now, by taking a union bound over all t'-partitions $\mathcal{P}'$ with $t' \leq t$ (there are at most $(t+1)^n$ of them), it follows that with probability at least $1 - \frac{2\cdot(t+1)^n}{(3(t+1))^n} \geq 1 - (2/3)^n$, if $\kappa_G(\mathcal{P}') \geq \beta n/2$, then the following bound holds:

$$\left|\kappa_H(\mathcal{P}') - p \cdot \kappa_G(\mathcal{P}')\right| \leq (\epsilon/2) \cdot p \cdot \kappa_G(\mathcal{P}') \tag{7}$$

This shows (6) holds for with probability at least $1/2$. Henceforth, we condition on this event, and prove that (5) holds for *all* partitions $\mathcal{P}$ with $\kappa_G(\mathcal{P}) \geq \beta n$. Consider one such partition $\mathcal{P}$ satisfying $\kappa_G(\mathcal{P}) \geq \beta n$. Using Corollary 17, there exists a partition $\mathcal{P}'$ that is a $(\lceil 2/\epsilon \rceil, 1 - \epsilon/2)$ coarsening of $\mathcal{P}$. In particular, it satisfies the following: (1) the number of parts in $\mathcal{P}'$ is at most $t = \lceil 2/\epsilon \rceil$, and (2) $\kappa_G(\mathcal{P}') \geq (1 - \epsilon/2) \cdot \kappa_G(\mathcal{P}) \geq \beta n/2$. Due to our assumption, (7) holds for this $\mathcal{P}'$. Let $R_G(\mathcal{P}) \coloneqq \mathsf{cut}_G(\mathcal{P}) \setminus \mathsf{cut}_G(\mathcal{P}')$ denote the (multi)set of edges of $\mathsf{cut}_G(\mathcal{P})$ that do not appear in $\mathsf{cut}_G(\mathcal{P})$, and note that $|R| \leq \epsilon/2 \cdot \kappa_G(\mathcal{P})$. Let $R_H(\mathcal{P}) \subseteq R_G(\mathcal{P})$ denote the subset that is sampled and appears in $E(H)$. For brevity, we refer to $R_G(\mathcal{P})$ and $R_H(\mathcal{P})$ as R and R', respectively. Now, consider

$$|\mathsf{cut}_H(\mathcal{P}) \setminus R_H(\mathcal{P})| = |\mathsf{cut}_H(\mathcal{P}')| = \kappa_H(\mathcal{P}') \leq (1+\epsilon/2)\cdot p \cdot \kappa_G(\mathcal{P}') \leq (1+\epsilon/2)\cdot p \cdot \kappa_G(\mathcal{P}) \quad (8)$$

$$|\mathsf{cut}_H(\mathcal{P}) \setminus R_H(\mathcal{P})| = \kappa_H(\mathcal{P}') \geq (1-\epsilon/2)\cdot p \cdot \kappa_G(\mathcal{P}') \geq (1-\epsilon/2)^2\cdot p \cdot \kappa_G(\mathcal{P}) \geq (1-\epsilon)\cdot p \cdot \kappa_G(\mathcal{P}) \quad (9)$$

By combining (8) and (9), we obtain that (5) holds with probability at least 1/2 for all partitions $\mathcal{P}$ satisfying $\kappa_G(\mathcal{P}) \geq \beta n$. ◀

5 Poly-space Approximation Schemes Faster than $2^n n^{\mathcal{O}(1)}$

In this section, we use the solution linearization procedures defined in the preceding section, to obtain polynomial-space approximation schemes for Minimum Bisection and Edge Bipartization (equivalently, Max Cut). The base case of both algorithms is the case where the optimal solution is bounded linearly in n. Here, we exploit the fact that such instances have a vertex cover of bounded size. In the following subsection, we first give polynomial-space $2^k n^{\mathcal{O}(1)}$ algorithms for both these problems parameterized by the vertex cover number k. We do this by working with a common generalization of both Minimum Bisection as well as Max-Cut.

5.1 Cut Algorithms Parameterized by Vertex Cover

In this section, we consider the following problem, which generalizes (vertex-weighted) Minimum Bisection as well as Max-Cut.

(s,c)-Cut
Input: A graph $G = (V,E)$ with costs $b : V \to \{0,1,2,\ldots,B\}$ and weights $w : E \to \{0,1,\ldots,W\}$, a vertex cover $L \subseteq V$; and two integers $s, c \geq 0$.
Task: Determine whether there exists a partition (V_1, V_2) of V, such that (1) $\sum_{v\in V_1} b(v) = s$, and (2) $\sum_{e\in\partial(V_1)} w(e) = c$.

For this problem, we design a $2^{|L|}\cdot(nBW)^{\mathcal{O}(1)}$-time algorithm that takes polynomial space.

Let (A,B) be a partition of a subset $U \subseteq V$. Then, we say that (A,B) is an (p,q)-partition of U iff the total value of the vertices in A is exactly p, and the total weight of the edges going across (A,B) is exactly q. For $U \subseteq U' \subseteq V(G)$, we say that a partition (A',B') of U' *extends* a partition (A,B) of U if $A \subseteq A'$ and $B \subseteq B'$.

Now we describe our algorithm `FastCutbyVC`(G,s,c,L) that takes as an input a graph G with non-negative integer costs on vertices and weights on edges. s is the target cost and c is the target weight for the cut. $L \subseteq V(G)$ is a vertex cover of G. We iterate over all partitions (A,B) of L. For each such partition (A,B), we run the dynamic programming algorithm, which is described in the following paragraph, to determine whether (A,B) can be extended to an (s,c)-partition of $V(G)$, and return **yes**. Otherwise, if no partition can be extended, then we say **no**.

Dynamic Programming. Let (A,B) be a given partition of a vertex cover L of $V(G)$. Let s' denote the cost of the vertices in A, and c' denote the weight of the edges with one endpoint in A and other endpoint in B. If $s' > s$ or $c' > c$ then we return no. Otherwise, we proceed as follows. Let us arbitrarily order the vertices of I as $\{u_1,u_2,\ldots,u_t\}$ and $I_j = \{u_1,u_2,\ldots,u_j\}$.

Then, we define a table T with boolean entries, where $T[j,p,q] = \texttt{true}$ iff there exists a subset $I' \subseteq I_j$ such that $(A \cup I', B \cup (I_j \setminus I'))$ is a (p,q)-partition of $L \cup I_j$. We use the following recurrence to compute the table entries.

$$T[j,p,q] = \begin{cases} \texttt{true} & \text{if } j = 0, p = s', q = c' \\ \texttt{false} & \text{if } (j = 0) \wedge (p \neq s' \vee q \neq c') \\ T[j-1, p - b(u_j), E(u_j, B)] \vee T[j-1, p, E(u_j, A)] & \text{otherwise} \end{cases}$$

The final answer is given by the entry $T[t,s,c]$. This implies the following lemma.

▶ **Lemma 20.** *For (G,s,c,L), $\texttt{FastCutbyVC}(G,s,c,L)$ correctly solves (s,c)-Cut in time $2^{|L|} \cdot (nBW)^{\mathcal{O}(1)}$ time and $(nBW)^{\mathcal{O}(1)}$ space. Furthermore, one can also find an (s,c)-partition of $V(G)$ in the same time and space.*

We obtain the following corollaries.

▶ **Corollary 21.** *Given a vertex-weighted (multi)graph $G = (V,E)$, a vertex cover $L \subseteq V$ for G, such that (1) G' has $n' \leq n$ vertices, (2) the total weight of vertices is n, and (3) the total number of edges is n^2, one can find a minimum bisection of G in time $2^{|L|} \cdot n^{\mathcal{O}(1)}$ and polynomial space.*

▶ **Corollary 22.** *Given a (multi)graph $G = (V,E)$, a vertex cover $L \subseteq V$ for G, such that (1) G' has $n' \leq n$ vertices, and (2) the total number of edges is n^2, one can find a maximum cut in G in time $2^{|L|} \cdot n^{\mathcal{O}(1)}$ and polynomial space.*

5.2 Minimum Bisection

The framework from Section 2 can be used to obtain a $2^{f(\epsilon)n} \cdot n^{\mathcal{O}(1)}$ time for some $f(\epsilon) < 1$ and *polynomial-space* algorithm for Minimum Bisection that finds a $(1+\epsilon)$-approximation. Note that although Minimum Bisection as it is usually defined does not satisfy Property 2 in Definition 6, this can be overcome by defining it as a vertex-weighted problem where all the vertex weights are integers. The goal is to find a partition with fewest edges going across subject to the condition that the total weights of the partitions differ by at most 1. Moreover, every time we contract an edge while following our framework, the new vertex gets the combined weight of both vertices. Then, the algorithm of Lemma 8 can be used for this problem.

To this end, we work with vertex-weighted graphs (hence our motivation to define (s,c)-Cut using weights). `ExactAlgorithm` simply iterates over all bisections (accounting for weights) and returns the best solution – it is easy to see that this takes polynomial space.

However, in this section, we design an approximation algorithm for Minimum Bisection with running time $2^{f(\epsilon)n} \cdot n^{\mathcal{O}(1)}$ for some $f(\epsilon) < 1$, polynomial-space, *and* the following additional property. If the size of the minimum bisection is linear in n, then the algorithm actually returns an *optimal solution*. Towards this, we need to use Procedure 1 (the random contraction algorithm) along with sparsification (Lemma 19).

Let G be a graph, then $\mathsf{bisection}(G)$ denotes the size of the optimal solution to Minimum Bisection on G. Our idea is to separate out the case where the optimal solution is bounded linearly in n from the case where this is not true. In the former case, we use Lemma 8 and in the latter case, with the knowledge of a lower bound on the size of the optimal solution, we invoke Lemma 23 below to compute an approximate solution.

5.2.1 Sparsifying to Linear Solution Size

▶ **Lemma 23.** *Let $0 < \epsilon < 1$, and $\tau = \frac{24}{\epsilon^2}\ln(\frac{9}{\epsilon})$. There exists a randomized polynomial-time algorithm that, given a graph G, and a parameter $\beta \geq \tau$, returns a subgraph H of G and a real number r. Furthermore, if $\mathsf{bisection}(G) = \beta n$, then the following properties hold with high probability.*

1. $(1-\epsilon)\cdot\mathsf{bisection}(G) \leq r\cdot\mathsf{bisection}(H) \leq (1+\epsilon)\cdot\mathsf{bisection}(G)$, *and*
2. $\mathsf{bisection}(H) \leq \tau(1+\epsilon)n$, *and if $\{V_1, V_2\}$ is an optimal bisection in H, then $\kappa_G(V_1, V_2) \leq (1+2\epsilon)\cdot\mathsf{bisection}(G)$.*

Proof. Given the input parameters satisfying the given conditions, the algorithm defines $p \coloneqq \frac{\tau}{\beta}$, and returns the graph G' obtained by sampling each edge of G independently with probability p, and $r \coloneqq 1/p$. Now, suppose β satisfies $\beta n = \mathsf{bisection}(G)$. First, observe that for *each* 2-partition $\mathcal{P} = \{V_1, V_2\}$ of $V(G)$ with $||V_1| - |V_2|| \leq 1$, it holds that $\kappa_G(\mathcal{P}) \geq \beta n$. Then, Lemma 19 (inequality (6)) implies that, with high probability, for each 2-partition $\mathcal{P}$ of $V(G)$,

$$\left|\kappa_H(\mathcal{P}) - \kappa_G(\mathcal{P})\cdot p\right| \leq \frac{\epsilon}{2}\cdot p\cdot\kappa_G(\mathcal{P}) \tag{10}$$

Henceforth, we assume that this event happens. Let $\mathcal{P}^*, \mathcal{P}'$ denote the partitions corresponding to a minimum bisection in G and H, respectively.

From (10), it follows that $\kappa_H(\mathcal{P}')/p = \mathsf{bisection}(H)/p \leq \kappa_H(\mathcal{P}^*) \leq (1+\frac{\epsilon}{2})\cdot\kappa_G(\mathcal{P}^*) = (1+\frac{\epsilon}{2})\cdot\mathsf{bisection}(G)$. In the reverse direction, $\mathsf{bisection}(H)/p = \kappa_H(\mathcal{P}')/p \geq (1-\epsilon/2)\cdot\kappa_G(\mathcal{P}') \geq (1-\epsilon/2)\cdot\mathsf{bisection}(G)$, since $\mathcal{P}'$ is a feasible solution for bisection in G. Finally, consider,

$$(1-\epsilon/2)\cdot p\kappa_G(\mathcal{P}') \leq \kappa_H(\mathcal{P}') \leq \kappa_H(\mathcal{P}^*) \leq (1+\epsilon/2)\cdot p\cdot\kappa_G(\mathcal{P}^*)$$

From combining the first and the last term in the previous inequality, we obtain that $\kappa_G(\mathcal{P}') \leq \frac{1-\epsilon/2}{1+\epsilon/2}\cdot\kappa_G(\mathcal{P}^*) \leq (1+2\epsilon)\cdot\kappa(\mathcal{P}^*)$. ◀

▶ **Theorem 24.** *There exists a randomized algorithm that, given a graph G and $0 < \epsilon < 1$, runs in time $2^{(1-\delta_\epsilon)n}$, and with high probability, returns a set $S \subseteq E(G)$ such that $|S| \leq (1+2\epsilon)\cdot\mathsf{bisection}(G)$. Furthermore, if $\mathsf{bisection}(G) \leq \frac{24}{\epsilon^2}\ln(\frac{9}{\epsilon})$, then with high probability, the solution returned by the algorithm is an optimal solution.*

Proof. In the first step, we run the algorithm in Lemma 8 with $\alpha \coloneqq \frac{24}{\epsilon^2}\ln(\frac{9}{\epsilon})$. If the algorithm returns a solution within $2^{(1-\delta_\alpha)n}$ steps, we keep it as a candidate solution; otherwise we terminate the algorithm and proceed to the next step. In the second step, by trying each value of $\ell = \lceil\alpha n\rceil, \lceil\alpha n\rceil + 1, \ldots, |E(G)|$, we use the algorithm of Lemma 23, and obtain a graph G' and a number r. Then, we run the algorithm of Lemma 8 on each such G'. Let $\alpha' = \alpha(1+\epsilon)$. If the algorithm in this iteration runs in time more than $2^{(1-\delta_{\alpha'})n}$, we terminate the algorithm and proceed with the next iteration. Otherwise, the algorithm returns a candidate solution. Finally, we return the minimum solution found in step 1, as well as over all iterations in step 2.

Now suppose $\mathsf{bisection}(G) \leq \alpha n$. Then, we use a recursive procedure that is similar to `RandomContraction` with some modifications. We keep contracting edges chosen uniformly at random until the number of edges in the graph is at most $\alpha\beta n$. Once the number of edges in the current graph G' is bounded by $\alpha\beta n$ (which is linear in n, since α and β are constants). Suppose at this point, the number of vertices in the graph G' is $(1-\lambda)n$. In this case, one can argue using average degree arguments that the size of the minimum vertex cover in G' is at most $n\cdot\frac{2\alpha\beta(1-\lambda)}{2\alpha\beta+1-\lambda}$. Thus, we use Corollary 21 to find an optimal bisection in G' in time

$2^{\mathsf{vc}(G')} \cdot n^{\mathcal{O}(1)}$ and polynomial space. By closely following the proof of Lemma 8, one can show that the running time of the algorithm is bounded by $\mathcal{O}^*(c^n)$, where $c = \max\left\{1.49, 2^{\frac{20\alpha}{1+20\alpha}}\right\}$. Furthermore, arguments similar to Lemma 8 imply that the algorithm returns an optimal bisection with at least a constant probability. This concludes the proof of the theorem. ◀

Theorem 3 follows from the above by scaling $\epsilon \leftarrow \epsilon/2$.

5.3 Edge Bipartization

For a graph G, let $\mathsf{edbip}(G)$ denote the size of the smallest cardinality edge set S such that $G - S$ is bipartite. Similar to Minimum Bisection, the idea for Edge Bipartization is to separate out the case where the optimal solution is bounded linearly in n from the case where this is not true. In the former case, we show that this bounds the size of the minimum vertex cover, in which case one can use fast fixed-parameter algorithms for Edge Bipartization parameterized by the vertex cover number of the graph. In the latter case, with the knowledge of a lower bound on the size of the optimal solution, we invoke Lemma 26 (see below) to compute an approximate solution.

We begin by using average-degree arguments to bound the minimum vertex cover size when $\mathsf{edbip}(G)$ is bounded linearly in n.

▶ **Lemma 25.** *If $\mathsf{edbip}(G) \leq \alpha n$, then minimum vertex cover is $\leq \frac{(1+8\alpha)}{2(1+4\alpha)}n$.*

▶ **Lemma 26.** *Let $\alpha > 0$ be a fixed constant and G be a graph on n vertices. Then, in time $2^{(1-\delta_\alpha)n}$ we can either return a set $S \subseteq E(G)$, such that $G-S$ is bipartite and $\mathsf{edbip}(G) = |S|$ or conclude that $\mathsf{edbip}(G) \geq \alpha n$.*

Proof. First we find a minimum vertex cover of G in $(1.22)^n$ time [31]. Let $VC(\mathrm{G})$ be a minimum vertex cover of G with size $|VC(\mathrm{G})| = vc(G)$. We consider two cases now:

- Case 1: If $vc(G) \leq \frac{(1+8\alpha)}{2(1+4\alpha)}n < n$, then we find minimum edge bipartization $S \subseteq E(G)$ by finding a maximum cut of G in $2^{vc(G)}n^{\mathcal{O}(1)} = 2^{(1-\delta_\alpha)n}$ time exactly where $\delta_\alpha = \frac{1}{2(1+4\alpha)}$. So in this case, $\mathsf{edbip}(G) = |S|$.
  Here, we are using the algorithm of Corollary 22 and the fact that computing a maximum cut in the graph is equivalent to computing a minimum solution to edge bipartization.
- Case 2: If $vc(G) > \frac{(1+8\alpha)}{2(1+4\alpha)}n$, then using Lemma 25, we can conclude that $\mathsf{edbip}(G) > \alpha n$. ◀

Due to the above lemma, the "interesting" regime for us is when $\mathsf{edbip}(G)$ is at least αn, in which case we can rely on Lemma 18.

▶ **Lemma 27.** *There exists a randomized polynomial time algorithm that given a graph $G, 0 < \epsilon < 1$, and a parameter $\frac{3\ln 9}{\epsilon^2} \leq \beta \leq \frac{m}{n}$, returns a graph H and a real number r. Furthermore, if $\mathsf{edbip}(G) = \beta n$, then the following properties hold with high probability.*

1. $(1-\epsilon)\mathsf{edbip}(G) \leq r \cdot \mathsf{edbip}(H) \leq (1+\epsilon)\mathsf{edbip}(G)$
2. $\mathsf{edbip}(H) \leq \tau(1+\epsilon)n$ *and if* $\mathcal{P}' = (V_1 \uplus V_2)$ *is an optimum edge bipartization in H, then* $\iota_G(\mathcal{P}') \leq (1+4\epsilon)\mathsf{edbip}(G)$

Lemma 26 and Lemma 27 combined, give us the following.

▶ **Theorem 28.** *Let $\epsilon > 0$ be a constant and G be a graph on n vertices. Then, in time $2^{(1-\delta_\epsilon)n}$ we can return a set $S \subseteq E(G)$, such that $G-S$ is bipartite and $|S| \leq (1+4\epsilon)\mathsf{edbip}(G)$.*

Theorem 4 follows from the above by scaling $\epsilon \leftarrow \epsilon/2$.

6 Conclusion and Discussion

In this paper, we have given a framework to design exponential-time approximation schemes for basic graph partitioning problems. Our work points to several interesting directions for follow-up work.

1. A natural set of questions in this direction is that of obtaining exact algorithms faster than $2^n n^{\mathcal{O}(1)}$ for Steiner k-cut and Multicut. For k-way cut, such an algorithm has been recently obtained [23]. In fact, the same question is open for 2-CSP, also Min 2-SAT.
2. Note that our techniques lead to randomized algorithms. Obtaining deterministic versions of our algorithms is an interesting question.
3. Could one design an appropriate "solution linearization" procedure for Multicut when the number of requests is unbounded? If the number of requests is constant, then Lemma 19 can be used to obtain such a procedure.
4. Our techniques (e.g., random contraction) break down for directed graphs. Could one build a similar framework that can handle directed problems?

References

1 Josh Alman, Timothy M. Chan, and R. Ryan Williams. Faster deterministic and las vegas algorithms for offline approximate nearest neighbors in high dimensions. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 637–649. SIAM, 2020. doi:10.1137/1.9781611975994.39.

2 András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 47–55. ACM, 1996. doi:10.1145/237814.237827.

3 Andreas Björklund. Determinant sums for undirected hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014. doi:10.1137/110839229.

4 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In *STOC 2007*, pages 67–74, New York, 2007. ACM.

5 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM J. Comput.*, 39(2):546–563, 2009. doi:10.1137/070683933.

6 Ivan Bliznets, Fedor V. Fomin, Michal Pilipczuk, and Yngve Villanger. Largest chordal and interval subgraphs faster than 2^n. *Algorithmica*, 76(2):569–594, 2016. doi:10.1007/s00453-015-0054-2.

7 Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, volume 5917 of *Lecture Notes in Computer Science*, pages 75–85. Springer, 2009. doi:10.1007/978-3-642-11269-0_6.

8 Chandra Chekuri, Sudipto Guha, and Joseph Naor. The steiner k-cut problem. *SIAM J. Discret. Math.*, 20(1):261–271, 2006. doi:10.1137/S0895480104445095.

9 Rajesh Chitnis, Fedor V. Fomin, Daniel Lokshtanov, Pranabendu Misra, M. S. Ramanujan, and Saket Saurabh. Faster exact algorithms for some terminal set problems. *J. Comput. Syst. Sci.*, 88:195–207, 2017. doi:10.1016/j.jcss.2017.04.003.

10 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

11 Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23(4):864–894, 1994. doi:10.1137/S0097539792225297.

12 Bruno Escoffier, Vangelis Th. Paschos, and Emeric Tourniaire. Approximating MAX SAT by moderately exponential and parameterized algorithms. *Theor. Comput. Sci.*, 560:147–157, 2014. doi:10.1016/j.tcs.2014.10.039.

13 Fedor V. Fomin, Serge Gaspers, Daniel Lokshtanov, and Saket Saurabh. Exact algorithms via monotone local search. *J. ACM*, 66(2):8:1–8:23, 2019. doi:10.1145/3284176.

14 Fedor V. Fomin, Serge Gaspers, Artem V. Pyatkin, and Igor Razgon. On the minimum feedback vertex set problem: Exact and enumeration algorithms. *Algorithmica*, 52(2):293–307, 2008. doi:10.1007/s00453-007-9152-0.

15 Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer, 2010. An EATCS Series: Texts in Theoretical Computer Science.

16 Olivier Goldschmidt and Dorit S. Hochbaum. A polynomial algorithm for the k-cut problem for fixed k. *Math. Oper. Res.*, 19(1):24–37, 1994. doi:10.1287/moor.19.1.24.

17 Edward A. Hirsch. Worst-case study of local search for max-k-sat. *Discret. Appl. Math.*, 130(2):173–184, 2003. doi:10.1016/S0166-218X(02)00404-3.

18 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

19 David R. Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In Vijaya Ramachandran, editor, *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas, USA*, pages 21–30. ACM/SIAM, 1993. URL: http://dl.acm.org/citation.cfm?id=313559.313605.

20 Frank Thomson Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999. doi:10.1145/331524.331526.

21 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bull. EATCS*, 105:41–72, 2011. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/92.

22 Daniel Lokshtanov, Saket Saurabh, and Ondrej Suchý. Solving multicut faster than 2 n. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms – ESA 2014 – 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, volume 8737 of *Lecture Notes in Computer Science*, pages 666–676. Springer, 2014. doi:10.1007/978-3-662-44777-2_55.

23 Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan. Breaking the all subsets barrier for min k-cut. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 90:1–90:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.90.

24 Matthias Mnich and Eva-Lotta Teutrine. Improved bounds for minimal feedback vertex sets in tournaments. *J. Graph Theory*, 88(3):482–506, 2018. doi:10.1002/jgt.22225.

25 Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for *k*-sat. *J. ACM*, 52(3):337–364, 2005. doi:10.1145/1066100.1066101.

26 Aviad Rubinstein and Virginia Vassilevska Williams. SETH vs approximation. *SIGACT News*, 50(4):57–76, 2019. doi:10.1145/3374857.3374870.

27 Uwe Schöning. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 410–414. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814612.

28 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.

29 Gerhard J. Woeginger. Exact algorithms for NP-hard problems: A survey. In *Combinatorial Optimization – Eureka, you shrink!*, volume 2570 of *LNCS*, pages 185–207. Springer-Verlag, Berlin, 2003.

30 Gerhard J. Woeginger. Space and time complexity of exact algorithms: Some open problems. In *IWPEC 2004*, volume 3162 of *LNCS*, pages 281–290. Springer-Verlag, Berlin, 2004.

31 Mingyu Xiao and Hiroshi Nagamochi. Exact algorithms for maximum independent set. *Inf. Comput.*, 255:126–146, 2017. doi:10.1016/j.ic.2017.06.001.