Faster Exponential-Time Approximation Algorithms Using Approximate Monotone Local Search

Barış Can Esmer
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
Saarbrücken Graduate School of Computer Science, Saarland Informatics Campus, Germany

Ariel Kulik
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Dániel Marx
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Daniel Neuen
School of Computing Science, Simon Fraser University, Burnaby, Canada

Roohani Sharma
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Abstract

We generalize the monotone local search approach of Fomin, Gaspers, Lokshtanov and Saurabh [J.ACM 2019], by establishing a connection between parameterized approximation and exponential-time approximation algorithms for *monotone subset minimization* problems. In a *monotone subset minimization* problem the input implicitly describes a non-empty set family over a universe of size n which is closed under taking supersets. The task is to find a minimum cardinality set in this family. Broadly speaking, we use *approximate monotone local search* to show that a parameterized α-approximation algorithm that runs in $c^k \cdot n^{\mathcal{O}(1)}$ time, where k is the solution size, can be used to derive an α-approximation randomized algorithm that runs in $d^n \cdot n^{\mathcal{O}(1)}$ time, where d is the unique value in $\left(1, 1+\frac{c-1}{\alpha}\right)$ such that $\mathcal{D}\left(\frac{1}{\alpha} \middle\| \frac{d-1}{c-1}\right) = \frac{\ln c}{\alpha}$ and $\mathcal{D}(a\|b)$ is the Kullback-Leibler divergence. This running time matches that of Fomin et al. for $\alpha = 1$, and is strictly better when $\alpha > 1$, for any $c > 1$. Furthermore, we also show that this result can be derandomized at the expense of a sub-exponential multiplicative factor in the running time.

We use an approximate variant of the exhaustive search as a benchmark for our algorithm. We show that the classic $2^n \cdot n^{\mathcal{O}(1)}$ exhaustive search can be adapted to an α-approximate exhaustive search that runs in time $\left(1 + \exp\left(-\alpha \cdot \mathcal{H}\left(\frac{1}{\alpha}\right)\right)\right)^n \cdot n^{\mathcal{O}(1)}$, where $\mathcal{H}$ is the entropy function. Furthermore, we provide a lower bound stating that the running time of this α-approximate exhaustive search is the best achievable running time in an oracle model. When compared to approximate exhaustive search, and to other techniques, the running times obtained by approximate monotone local search are strictly better for any $\alpha \geq 1,\ c > 1$.

We demonstrate the potential of approximate monotone local search by deriving new and faster exponential approximation algorithms for Vertex Cover, 3-Hitting Set, Directed Feedback Vertex Set, Directed Subset Feedback Vertex Set, Directed Odd Cycle Transversal and Undirected Multicut. For instance, we get a 1.1-approximation algorithm for Vertex Cover with running time $1.114^n \cdot n^{\mathcal{O}(1)}$, improving upon the previously best known 1.1-approximation running in time $1.127^n \cdot n^{\mathcal{O}(1)}$ by Bourgeois et al. [DAM 2011].

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Mathematics of computing → Approximation algorithms; Mathematics of computing → Probabilistic algorithms

Keywords and phrases parameterized approximations, exponential approximations, monotone local search

Digital Object Identifier 10.4230/LIPIcs.ESA.2022.50

Related Version *Full Version*: `https://arxiv.org/abs/2206.13481`

Funding *Dániel Marx*: Research supported by the European Research Council (ERC) consolidator grant No. 725978 SYSTEMATICGRAPH.

1 Introduction

A lot of interesting problems are computationally hard as they do not admit polynomial-time algorithms. Still, many of them can be solved significantly faster than exhaustive search. The area of exact exponential algorithms studies the design of such techniques. Typically, for *subset problems*, where the goal is to find a subset of a given n-sized universe U that satisfies some property Π, a solution can be found by enumerating all 2^n subsets of U. Therefore, the goal is to design algorithms that beat this exhaustive search and run in time $\mathcal{O}^*(c^n)$[1] for as small $1 < c < 2$ as possible.

Exact Monotone Local Search. In a seminal work Fomin, Gaspers, Lokshtanov and Saurabh [13] showed that one can derive faster exact exponential algorithms for subset problems using a parameterized extension algorithm for the problem at hand. A parameterized extension algorithm for a subset problem additionally takes as input a parameter k and a set $X \subseteq U$, runs in time $\mathcal{O}^*(f(k))$, and outputs a set $S \subseteq U$ of size at most k such that $S \cup X$ is a solution, if such a set exists. Fomin et al. [13, Theorem 1.1] showed that if a subset problem admits a parameterized extension algorithm that runs in time $\mathcal{O}^*(c^k)$ for some absolute constant $c > 1$, then it admits a randomized exact exponential algorithm that runs in time $\mathcal{O}^*((\texttt{emls}(c))^n)$, where $\texttt{emls}(c) = 2 - \frac{1}{c}$. Their algorithm, called Exact Monotone Local Search (`Exact-MLS`), is simple and is based on monotone local search: it samples a set X of t elements at random, and then extends the set X to an optimum solution using the parameterized extension algorithm. The non-trivial part of the proof of [13, Theorem 1.1] is to analyze the value of t that optimizes the running time. This simple algorithm outperforms the exhaustive search for all subset problems that have parameterized extension algorithms running in $\mathcal{O}^*(c^k)$ time. Moreover, given the existence of a large number of problems that admit the desired parameterized extension algorithm, it yields the state-of-the-art exact exponential algorithms for several problems [13, Table 1].

Exponential Approximation. Another important algorithmic paradigm which deals with NP-hardness is the design of *approximation* algorithms, which are typically polynomial-time algorithms that compute a solution which is not necessarily optimum but has a worst-case guarantee on its *quality*. Though several NP-hard problems admit such algorithms with constant approximation ratios [30], there are many that do not, under reasonable complexity assumptions, for example Directed Feedback Vertex Set [28]. Also, there are many for which the approximation guarantees cannot be improved beyond a fixed constant. For example, Vertex Cover admits a 2-approximation but no $(2-\varepsilon)$-approximation under the Unique Games Conjecture [17].

For problems where some hardness of approximation has been established, a natural question is to determine the smallest c such that the barriers of this hardness can be broken by taking $\mathcal{O}^*(c^n)$ time. Such algorithms are called *exponential approximation* algorithms and this topic has received attention in, e.g., [1, 2, 3, 7, 25].

Consider *subset minimization* problems where the goal is to find a subset of the n-sized universe U of *minimum cardinality*, also called an *optimum* solution, that satisfies some additional property Π. For any approximation ratio $\alpha \geq 1$, we say that a subset $S \subseteq U$ satisfying the property Π is an *α-approximate solution* if $|S| \leq \alpha \cdot |\texttt{OPT}|$, where $\texttt{OPT} \subseteq U$ is an optimum solution. An *exponential α-approximation algorithm* for a subset minimization problem returns an α-approximate solution and runs in $\mathcal{O}^*(c^n)$ time for some $1 < c < 2$.

[1] The $\mathcal{O}^*$ notation hides polynomial factors in the input.

Parameterized Approximation. A *parameterized α-approximation* algorithm for a subset minimization problem additionally takes as input the parameter k, runs in time $\mathcal{O}^*(f(k))$, and outputs a solution of size at most $\alpha \cdot k$, if there exists a solution of size at most k. Analogous to parameterized algorithms, one can define the notion of extension algorithms here (see Section 2). The design of parameterized α-approximation algorithms has been an active area of research in the last few years, yielding a plethora of results for problems that exhibit some hardness either in the parameterized setting or in the approximation setting [4, 5, 8, 10, 11, 12, 14, 15, 16, 18, 21, 22, 23, 24, 26, 27, 29].

Approximate Monotone Local Search (`Approximate-MLS`). In this paper, we show that one can extend the idea of `Exact-MLS` [13] to *derive faster exponential approximation algorithms from parameterized approximation algorithms.* Let $\texttt{amls}(\alpha, c)$ be the unique value in $\left(1, 1+\frac{c-1}{\alpha}\right)$ such that $\mathcal{D}\left(\frac{1}{\alpha} \middle\| \frac{\texttt{amls}(\alpha,c)-1}{c-1}\right) = \frac{\ln c}{\alpha}$ where $\mathcal{D}\left(a \| b\right)$ is the Kullback-Leibler divergence defined as $\mathcal{D}\left(a\|b\right) = a \ln\left(\frac{a}{b}\right) + (1-a)\ln\left(\frac{1-a}{1-b}\right)$ (see, e.g., [6]). Our main result can be informally stated as follows.

> If a monotone subset minimization problem admits a parameterized extension α-approximation algorithm that runs in $\mathcal{O}^*(c^k)$, then one can derive a randomized α-approximation algorithm that runs in time $\mathcal{O}^*((\texttt{amls}(\alpha, c))^n)$ (see Theorem 2.1).

Since $\texttt{amls}(1, c) = \texttt{emls}(c) = 2 - \frac{1}{c}$ for every $c > 1$, our running time matches that of `Exact-MLS` when $\alpha = 1$.

Recall the `Exact-MLS` algorithm described earlier. The non-trivial part of the proof of [13, Theorem 1.1] is the analysis of the probability that the sampled set is *contained* in an optimum solution. To obtain our result, we use the same algorithm and show that if we allow the sampled set to also contain *some* items from outside the optimum solution, then one can speed-up the resulting exponential α-approximation algorithm. Since our analysis need to take into account the calculations for error in the sampled set, this makes analyzing the choice of t more difficult.

In order to better appreciate the running time of our algorithm, that does not seem to have a closed-form formula, we give a mathematical comparison of $\texttt{amls}(\alpha, c)$ with various benchmark exponential approximation algorithms, showing that our algorithm outperforms all of them.

Benchmark 1: Brute-Force for Exponential Approximation. Since exhaustive search is a trivial benchmark against which the running times of (exact) exponential algorithms are measured, an important question to address is: *how much time does exhaustive search take to find an α-approximate solution*?

Consider a subset minimization problem that is also monotone, that is, for every $S \subseteq T \subseteq U$, if S is a solution, then T is also a solution. We show that for monotone subset minimization problems, the classic brute-force approach can be generalized to an *α-approximation brute-force algorithm* running in time $\mathcal{O}^*(\texttt{brute}(\alpha)^n)$, where $\texttt{brute}(\alpha) = 1 + \frac{(\alpha-1)^{\alpha-1}}{\alpha^\alpha} = 1 + \exp\left(-\alpha \cdot \mathcal{H}\left(\frac{1}{\alpha}\right)\right)$ and $\mathcal{H}(\alpha) = -\alpha \ln \alpha - (1-\alpha)\ln(1-\alpha)$ denotes the entropy function[2]. The running time essentially follows by showing that a uniformly sampled set of $\alpha \cdot |\texttt{OPT}|$ items is an α-approximate solution with probability at least $\texttt{brute}(\alpha)^{-n}$ (up to polynomial factors,

[2] We adopt the convention that $0^0 = 1$.

see Theorem 5.1). We complement this result by showing that this running time is *best possible*, given only membership oracle access to the problem (Theorem 5.1). For example, for $\alpha = 2$, this brute-force algorithm runs in time $\mathcal{O}^*(1.25^n)$. For $\alpha = 1.1$, it runs in time $\mathcal{O}^*(1.716^n)$ (see also Table 1).

Benchmark 2: Naive conversion from parameterized approximation to exponential approximation. Yet another upper bound on the running time of exponential approximation algorithms can be given as follows. Suppose there is a parameterized α-approximation for a monotone subset minimization problem that runs in $\mathcal{O}^*(c^k)$ time. Run the parameterized algorithm for every value of the parameter k between 0 to $\frac{n}{\alpha}$, and return the solution of minimum cardinality among the solutions returned by the parameterized algorithm. If no solution was found by the parameterized algorithm, then return the whole universe. It can be easily verified that this indeed yields an α-approximation algorithm with running time $\mathcal{O}^*((\texttt{naive}(\alpha, c))^n)$, where $\texttt{naive}(\alpha, c) = c^{\frac{1}{\alpha}}$. Observe that for large values of c and appropriate α, $\texttt{naive}(\alpha, c)$ could be much larger than even 2. But for smaller values of c, it could sometimes beat the brute-force approximation (see Section 2.4).

Benchmark 3: `Exact-MLS` in the approximate setting. From the description of the `Exact-MLS` algorithm, it is not difficult to deduce that given a parameterized extension α-approximation for a monotone subset minimization problem that runs in time $\mathcal{O}^*(c^k)$, one can derive an exponential α-approximation for the problem that runs in time $\mathcal{O}^*((\texttt{emls}(c))^n)$. This trivial generalization of `Exact-MLS` to the approximate setting already performs better than the naive conversion in cases when c is small. For example, VERTEX COVER has a parameterized (extension) 1.1-approximation algorithm that runs in time $\mathcal{O}^*(1.1652^k)$ [3]. `Exact-MLS` gives a 1.1-approximation running in time $\mathcal{O}^*(1.1417^n)$ whereas the naive conversion gives a running time of $\mathcal{O}^*(1.1462^n)$.

Comparisons. As stated earlier, we show that (see Lemma 1.1) `Approximate-MLS` is *strictly* faster than the brute-force algorithm (Benchmark 1) or the naive-conversion approach (Benchmark 2) described earlier, for every $\alpha \geq 1$ and $c > 1$. In fact, Lemma 1.1 shows that $\texttt{amls}(\alpha, c)$ converges to $\texttt{brute}(\alpha)$ as $c \to \infty$, which would be the expected behavior, because as the parameterized algorithm becomes "less useful", the running time of our algorithm gets closer to the running time possible without the use of any problem-specific algorithm. Since $\texttt{amls}(1, c) = \texttt{emls}(c) = 2 - \frac{1}{c}$, Lemma 1.1 also shows that the running time is strictly better than that of `Exact-MLS` (Benchmark 3) when $\alpha > 1$.

▶ **Lemma 1.1** ($\star$[3]). *For every $c > 1$ the following holds:*

1. $\texttt{amls}(\alpha, c) < \min\{\texttt{brute}(\alpha), \texttt{naive}(\alpha, c)\}$ *for every $\alpha \geq 1$. In fact,* $\texttt{amls}(\alpha, c) \xrightarrow[c \to \infty]{} \texttt{brute}(\alpha)$.
2. $\texttt{amls}(\alpha, c) < \texttt{emls}(c)$ *for every $\alpha > 1$. In fact,* $\texttt{amls}(\alpha, c)$ *is a strictly decreasing function of α.*

Applications and Derandomization. We show in Section 2.4 that `Approximate-MLS` can be used to derive new and faster exponential approximation algorithms for VERTEX COVER, 3-HITTING SET, DIRECTED FEEDBACK VERTEX SET, DIRECTED SUBSET FEEDBACK VERTEX SET, DIRECTED ODD CYCLE TRANSVERSAL and UNDIRECTED MULTICUT. We also show in Section 4 that, as in [13], our algorithm can be derandomized at the expense of a multiplicative sub-exponential factor in the running time.

[3] The proofs of statements marked with $\star$ appear in the full version of the paper [9].

2 Definitions and Our Results

2.1 Formal Definitions

We now give some formal definitions that will be required to formally state and describe our main results. An *implicit set system* is a function Φ that takes as input a string $I \in \{0,1\}^\star$, called an *instance*, and returns a set system $(U_I, \mathcal{F}_I)$ where U_I is a universe and $\mathcal{F}_I$ is a collection of subsets of U_I. We use n to denote the size of the universe, that is, $n \coloneqq |U_I|$. We say that an implicit set system Φ is *polynomial-time computable* if there are two polynomial-time algorithms, the first one, given $I \in \{0,1\}^\star$, computes the set U_I, and the second one, given $S \subseteq U_I$, correctly decides if $S \in \mathcal{F}_I$. A family $\mathcal{F} \subseteq 2^U$ is called *monotone* if $U \in \mathcal{F}$ and for every $S \subseteq T \subseteq U$, if $S \in \mathcal{F}$, then $T \in \mathcal{F}$. We say that an implicit set system Φ is *monotone* if $\mathcal{F}_I$ is monotone for every input I. Throughout the remainder of this work, we only deal with implicit set systems that are polynomial-time computable and monotone. So for the sake of convenience, we refer to a polynomial-time computable and monotone implicit set system simply as an implicit set system.

For an implicit set system Φ, the problem Φ_{MIN}-SUBSET takes as input a string $I \in \{0,1\}^\star$ and asks to find $S \in \mathcal{F}_I$ such that $|S|$ is minimum. We refer to the sets in $\mathcal{F}_I$ as solutions of I and we call the sets in $\mathcal{F}_I$ of minimum cardinality as minimum solutions or optimal solutions of I. For $\alpha \geq 1$, we say that an algorithm is a (randomized) α-approximation algorithm for Φ_{MIN}-SUBSET if on input I, it returns a set $S \in \mathcal{F}_I$ such that $|S| \leq \alpha \cdot |\texttt{OPT}|$ (with a constant probability), where $\texttt{OPT}$ is an optimum solution of Φ_{MIN}-SUBSET.

One can observe that many fundamental graph problems, such as VERTEX COVER, FEEDBACK VERTEX SET, DIRECTED FEEDBACK VERTEX SET, etc., can be cast as a Φ_{MIN}-SUBSET problem. Consider for example the VERTEX COVER problem. Given a graph $G = (V, E)$ we say a subset $S \subseteq V$ is a *vertex cover* if for every $(u, v) \in E$ it holds that $u \in S$ or $v \in S$. The input for the VERTEX COVER problem is a graph G and the objective is to find a vertex cover S of G such that $|S|$ is minimum. We can cast VERTEX COVER as a Φ_{MIN}-SUBSET problem for the implicit set system Φ_{VC} defined as follows. The instance of the problem is interpreted as a graph $G = (V, E)$. We define the universe U_G as the set of vertices V and the set of solutions $\mathcal{F}_G = \{S \subseteq V \mid S \text{ is a vertex cover of } G\}$ is the set of all vertex covers of G. Finally, we define $\Phi_{\text{VC}}(G) = (U_G, \mathcal{F}_G)$. It can be easily verified that Φ_{VC} is an implicit set system (i.e., it is polynomial-time computable and monotone).

We say an algorithm is a *parameterized (randomized) α-approximate Φ-extension* if, given an instance $I \in \{0,1\}^\star$, $X \subseteq U_I$ and a parameter $k \in \mathbb{N}$, it returns a set $Y \subseteq U_I$ which satisfies the following property (with a constant probability): if there exists a set $S \subseteq U_I$ such that $S \cup X \in \mathcal{F}_I$ and $|S| \leq k$, then it holds that $Y \cup X \in \mathcal{F}_I$ and $|Y| \leq \alpha \cdot k$. We use the shorthand *(randomized) (α, Φ)-extension algorithm* to refer to a *parameterized (randomized) α-approximate Φ-extension algorithm*. Observe that a parameterized α-approximation algorithm for VERTEX COVER can be turned into an $(\alpha, \Phi_{\text{VC}})$-extension algorithm with the same running time, by taking the instance (G, X, k) of the $(\alpha, \Phi_{\text{VC}})$-extension algorithm, and running the parameterized α-approximation algorithm on the instance $(G - X, k)$. Observe that this way of converting parameterized α-approximation algorithms to (α, Φ)-extension algorithms holds for various implicit set systems Φ, for example, when Φ corresponds to a vertex deletion problem to a hereditary graph class.

2.2 Our results

Given an (α, Φ)-extension algorithm with running time $\mathcal{O}^*(c^k)$ we design an α-approximation algorithm for Φ_{MIN}-SUBSET with running time $\mathcal{O}^*(\texttt{amls}(\alpha, c)^n)$ where $\texttt{amls}$ is defined as the unique value $\gamma \in \left(1, 1 + \frac{c-1}{\alpha}\right)$ such that $\mathcal{D}\left(\frac{1}{\alpha} \middle\| \frac{\gamma-1}{c-1}\right) = \frac{\ln c}{\alpha}$. Note that $\texttt{amls}(\alpha, c)$ is indeed well-defined because for every $\alpha \geq 1$, the function $f(\delta) := \mathcal{D}\left(\frac{1}{\alpha} \| \delta\right)$ is monotonically decreasing in the interval $\delta \in \left(0, \frac{1}{\alpha}\right)$ as well as $f(\delta) \xrightarrow[\delta \to 0]{} \infty$ and $f(\delta) \xrightarrow[\delta \to \frac{1}{\alpha}]{} 0$.

▶ **Theorem 2.1** (Approximate Monotone Local Search). *Let Φ be an implicit set system and $\alpha \geq 1$. If there is a randomized (α, Φ)-extension algorithm that runs in time $\mathcal{O}^*(c^k)$, then there is a randomized α-approximation algorithm for Φ_{MIN}-SUBSET that runs in time $\mathcal{O}^*((\texttt{amls}(\alpha, c))^n)$.*

The formula for $\texttt{amls}(\alpha, c)$ (which describes the running time of Theorem 2.1) is not a closed-form formula, and we do not expect a closed-form formula for general α, c to exist. However, it represents a *tight* analysis of our algorithm. Despite being represented as an implicit formula, its basic properties can be deduced (see Lemma 1.1). Also, $\texttt{amls}$ can be easily evaluated for every $\alpha, c > 1$. Indeed, for every $\alpha \geq 1$, the function $f(\gamma) = \mathcal{D}\left(\frac{1}{\alpha} \middle\| \frac{\gamma-1}{c-1}\right)$ is monotonically decreasing in the interval $\left(1, 1 + \frac{c-1}{\alpha}\right)$. This means that $\texttt{amls}(\alpha, c)$ can be evaluated to an arbitrary precision, for every $\alpha \geq 1$ and $c > 1$, using binary search. In particular, the running time implied by Theorem 2.1 can be evaluated.

Theorem 2.1 can be used to obtain faster (than the state-of-art) exponential approximation algorithms for some Φ_{MIN}-SUBSET problems. For example, the brute-force 1.1-approximation algorithm runs in time $\mathcal{O}^*(\texttt{brute}(1.1)^n) = \mathcal{O}^*(1.716^n)$. The best parameterized 1.1-approximation for VERTEX COVER runs in time $\mathcal{O}^*(1.1652^k)$ [18], where k is the parameter. Using the naive conversion, we obtain a 1.1-approximation that runs in time $\mathcal{O}^*(\texttt{naive}(1.1, 1.1652)^n) = \mathcal{O}^*(1.149^n)$. The previously known fastest 1.1-approximation algorithm for VERTEX COVER runs in time $\mathcal{O}^*(1.127^n)$ [3]. Using Theorem 2.1 in conjunction with the $\mathcal{O}^*(1.1652^k)$ algorithm of [18], we get a 1.1-approximation algorithm for VERTEX COVER with running time $\mathcal{O}^*(1.114^n)$, improving over all of the above. We provide additional applications in Section 2.4.

In Section 4, we show that the algorithm of Theorem 2.1 can be derandomized, at the cost of a sub-exponential factor in the running time, by generalizing the construction of set inclusion families from [13].

▶ **Theorem 2.2** (Derandomization Approximate Monotone Local Search). *Let Φ be an implicit set system and $\alpha \geq 1$. If there is an (α, Φ)-extension algorithm that runs in time $\mathcal{O}^*(c^k)$, then there is an α-approximation algorithm for Φ_{MIN}-SUBSET that runs in time $\mathcal{O}^*\left((\texttt{amls}(\alpha, c))^{n+o(n)}\right)$.*

2.3 Approximate Monotone Local Search

We now present the algorithm underlying Theorem 2.1 and give a sketch for its analysis. Recall Φ is the implicit set family and $\alpha \geq 1$ is the approximation ratio. Let $\mathcal{A}_{\text{ext}}$ denote the (α, Φ)-extension algorithm that runs in time $\mathcal{O}^*(c^k)$ where k is the parameter. Given an instance $I \in \{0, 1\}^\star$, let $\Phi(I) = (U_I, \mathcal{F}_I)$. The algorithm for Theorem 2.1 is described in Algorithm 2, and it is denoted by `Approximate-MLS`. It uses the subroutine `Sample` (Algorithm 1) which samples a random set from U_I, which is subsequently extended, using $\mathcal{A}_{\text{ext}}$, to yield the solution. Algorithm 2 coincides with the algorithm of [13, Theorem 1.1] when $\alpha = 1$.

Algorithm 1 `Sample`(I, k, t).

```
Input: $I \in \{0,1\}^\star, k \in \mathbb{N}, t \in \mathbb{N}$
1: Sample a set $X$ of size $t$ from $U_I$ uniformly at random.
2: $Y \leftarrow \mathcal{A}_{\text{ext}}\left(I, X, k - \lceil \frac{t}{\alpha} \rceil\right)$.
3: $Z \leftarrow X \cup Y$.
4: If $Z \in \mathcal{F}_I$ and $|Z| \leq \alpha \cdot k$, then return $Z$, otherwise return $U_I$.
```

Let `OPT` be an optimum solution of the instance I of Φ_{MIN}-Subset. Consider the execution of `Sample` on the instance (I, k, t) where $k = |\texttt{OPT}|$. In Step 1 of `Sample` if $|X \cap \texttt{OPT}| \geq \frac{t}{\alpha}$ then $|\texttt{OPT} \setminus X| \leq k - \frac{t}{\alpha}$. Therefore, in Step 2 $\mathcal{A}_{\text{ext}}$ must return a set Y such that $X \cup Y \in \mathcal{F}_I$ and $|Y| \leq \alpha \cdot (k - \frac{t}{\alpha}) = \alpha k - t$. Thus, the set $Z = X \cup Y$ computed in Step 3 is an α-approximate solution of I. Let $\texttt{hyper}(n, k, t, x)$ be the probability that a uniformly random set X of t items out of $[n] := \{1, \ldots, n\}$ satisfies $|X \cap [k]| \geq x$. The distribution of $|X \cap [k]|$ is commonly referred as *hyper-geometric*. Since $\Pr\left(|X \cap \texttt{OPT}| \geq \frac{t}{\alpha}\right) = \texttt{hyper}(n, k, t, \frac{t}{\alpha})$, `Sample` returns an α-approximate solution of I with probability $\texttt{hyper}(n, k, t, \frac{t}{\alpha})$. Observe that the running time of the `Sample` subroutine is proportional (up to polynomial factors) to the running time of the call to $\mathcal{A}_{\text{ext}}$ in Step 2. Thus, the `Sample` subroutine runs in time $\mathcal{O}^*(c^{k - \frac{t}{\alpha}})$.

Algorithm 2 `Approximate-MLS`(I).

```
Input: $I \in \{0,1\}^\star$
1: Define $n = |U_I|$ and $\mathcal{S} \leftarrow \emptyset$.
2: for $k$ from 0 to $\frac{n}{\alpha}$ do
3:     $t \leftarrow \operatorname*{argmin}_{t \in [0, \alpha k] \cap \mathbb{N}} \left( \frac{c^{k - \frac{t}{\alpha}}}{\texttt{hyper}\left(n, k, t, \frac{t}{\alpha}\right)} \right)$.
4:     Run $\mathcal{S} = \mathcal{S} \cup \{\texttt{Sample}(I, k, t)\}$ for $\left(\texttt{hyper}\left(n, k, t, \frac{t}{\alpha}\right)\right)^{-1}$ times.
5: Return a minimum sized set in $\mathcal{S}$.
```

Now, let us consider the execution of `Approximate-MLS` on input I. The analysis of `Approximate-MLS` focuses on the iteration of Step 2 when $k = |\texttt{OPT}|$. In this iteration, each call to $\texttt{Sample}(I, k, t)$ returns an α-approximate solution with probability $\texttt{hyper}(n, k, t, \frac{t}{\alpha})$ (as argued above). Since in Step 4, `Sample` is invoked $\left(\texttt{hyper}(n, k, t, \frac{t}{\alpha})\right)^{-1}$ times, at the end of the execution of Step 4, the set $\mathcal{S}$ contains an α-approximate solution of I with a constant probability.

The running time of a *fixed iteration* in Step 2 of `Approximate-MLS` is $(\texttt{hyper}(n, k, t, \frac{t}{\alpha}))^{-1}$ times the running time of `Sample`, that is, $\frac{c^{k - \frac{t}{\alpha}}}{\texttt{hyper}(n,k,t,\frac{t}{\alpha})}$. Let us denote $\texttt{iteration}_{n,k,c}(t) = \frac{c^{k - \frac{t}{\alpha}}}{\texttt{hyper}(n,k,t,\frac{t}{\alpha})}$. Observe that the value of t selected in Step 3 minimizes $\texttt{iteration}_{n,k,c}(t)$. From the algorithmic perspective the selection of the optimal value of t is straightforward as $\texttt{iteration}_{n,k,c}(t)$ can be computed in polynomial time for each value of t (given n and k). However, the asymptotic analysis of $\texttt{iteration}_{n,k,c}(t)$, and hence the overall running time, requires an in-depth understanding of the random process and serves as the main technical contribution of this paper.

As described earlier, when $\alpha = 1$, Algorithm 2 coincides with the algorithm of [13, Theorem 1.1]. The analysis of [13, Theorem 1.1] lower bounds the probability that the set X sampled on Step 1 of `Sample` satisfies $X \subseteq \texttt{OPT}$. The analysis of our algorithm lower bounds the probability that $\frac{|X \cap \texttt{OPT}|}{|X|} \geq \frac{1}{\alpha}$. In particular, the sampling step may select items which are not in OPT, though the number of such items is restricted. This allows for an improved running time in comparison to that of [13] (see Lemma 1.1), but renders the analysis of the running time to be more involved.

For the analytical estimation of t, which is selected in Step 3 of `Approximate-MLS`, the question that one needs to understand is *how many items should the algorithm sample before it decides to use $\mathcal{A}_{\text{ext}}$ to extend the sampled set.* Assume that the algorithm already sampled a set X of t items such that $|X \cap \texttt{OPT}| \approx \frac{t}{\alpha}$. Let $\varepsilon > 0$ be some small number. Observe that $U_I \setminus X$ contains $\approx k - \frac{t}{\alpha}$ items from OPT, and thus $\frac{|(U_I \setminus X) \cap \texttt{OPT}|}{|U_I \setminus X|} \approx \frac{k - \frac{t}{\alpha}}{n-t}$. The algorithm now has two options: it can either further sample a set A of additional $\varepsilon \cdot n$ items or, use $\mathcal{A}_{\text{ext}}$ with the parameter $k - \frac{t}{\alpha}$ to extend X to a final solution. In the first case, the time taken to sample a set A of $\varepsilon \cdot n$ items such that $|A \cap \texttt{OPT}| \geq \frac{|A|}{\alpha} = \frac{\varepsilon \cdot n}{\alpha}$ holds with constant probability, is $\left(\Pr\left(|A \cap \texttt{OPT}| \geq \frac{\varepsilon \cdot n}{\alpha}\right)\right)^{-1}$. In the second case, the algorithm spends an additional factor of $c^{\frac{\varepsilon \cdot n}{\alpha}}$ time to extend the set X, instead of $X \cup A$, to the final solution. Thus, if $\Pr\left(|A \cap \texttt{OPT}| \geq \frac{\varepsilon \cdot n}{\alpha}\right) > c^{-\frac{\varepsilon \cdot n}{\alpha}}$, it is better to continue sampling, and otherwise, it is better to run $\mathcal{A}_{\text{ext}}$ on the instance $(I, X, k - \frac{t}{\alpha})$. Therefore, to understand the analytics of the chosen t, one needs to upper bound $\Pr\left(|A \cap \texttt{OPT}| \geq \frac{\varepsilon \cdot n}{\alpha}\right)$.

We view the sampling of A as an iterative process in which the items are sampled one after the other. When sampling the ℓ-th item, the ratio between the remaining items in OPT and the available items is $\approx \frac{k - \frac{t}{\alpha} - \Delta}{n - t - \ell}$, where Δ is the number of items from OPT sampled in previous iterations and $0 \leq \Delta \leq \ell \leq \varepsilon \cdot n$. As $\varepsilon \cdot n$ is small, we estimate $\frac{k - \frac{t}{\alpha} - \Delta}{n - t - \ell} \approx \frac{k - \frac{t}{\alpha}}{n - t}$. Therefore, the probability that the ℓ-th sampled item is in OPT is roughly $\frac{k - \frac{t}{\alpha}}{n - t}$. Thus, $|A \cap \texttt{OPT}|$ can be estimated as the sum of $\varepsilon \cdot n$ Bernoulli random variables $\mathsf{x}_1, \ldots, \mathsf{x}_{\varepsilon \cdot n}$ with probability $\frac{k - \frac{t}{\alpha}}{n - t}$. Thus,

$$\Pr\left(|A \cap \texttt{OPT}| \geq \frac{\varepsilon \cdot n}{\alpha}\right) \approx \Pr\left(\sum_{i=1}^{\varepsilon \cdot n} \mathsf{x}_i \geq \frac{\varepsilon \cdot n}{\alpha}\right) \approx \exp\left(-\varepsilon \cdot n \cdot \mathcal{D}\left(\frac{1}{\alpha} \middle\| \frac{k - \frac{t}{\alpha}}{n - t}\right)\right),$$

where the last estimation follows from a large deviation property of binomial distributions [6, Theorem 11.1.4] and assumes $|\texttt{OPT}| \leq \frac{n}{\alpha}$.

Therefore, the (optimal) selection of t which minimizes $\texttt{iteration}_{n,k,c}(t)$ is the largest t which satisfies $\exp\left(-\varepsilon \cdot n \cdot \mathcal{D}\left(\frac{1}{\alpha} \middle\| \frac{k - \frac{t}{\alpha}}{n - t}\right)\right) > c^{-\frac{\varepsilon \cdot n}{\alpha}}$, or equivalently, $\mathcal{D}\left(\frac{1}{\alpha} \middle\| \frac{k - \frac{t}{\alpha}}{n - t}\right) \approx \frac{\ln c}{\alpha}$. We use this value of t to bound the running time of an iteration of Step 2, that is, to upper bound $\min_{t \in [0, \alpha k] \cap \mathbb{N}} \texttt{iteration}_{n,k,c}(t)$. This analytical estimation of t forms the crux in analyzing the overall running time of Algorithm 2.

2.4 Applications of `Approximate-MLS`

In this section we use Theorem 2.1 to get faster randomized exponential approximation algorithms for Vertex Cover, 3-Hitting Set, Directed Feedback Vertex Set (DFVS), Directed Subset Feedback Vertex Set (Subset DFVS), Directed Odd Cycle Transversal (DOCT) and Undirected Multicut. One can observe that all these problems can be described as some Φ_{MIN}-Subset problem. Since all these problems can be interpreted as vertex deletion problems to some hereditary graph class, any parameterized α-approximation algorithm for these problems can be used as an (α, Φ)-extension algorithm, for the respective Φ.

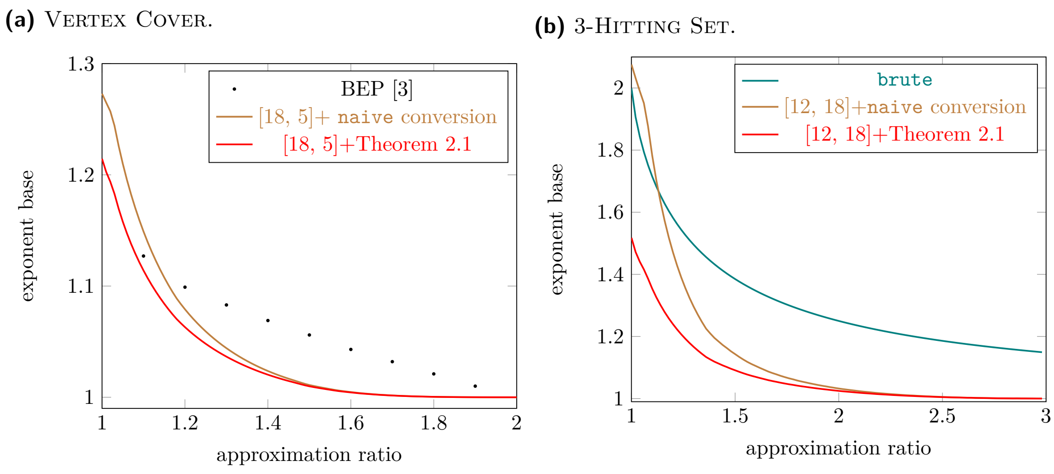

Figure 1 Results for Vertex Cover and 3-Hitting Set. A dot at (α, d) means that the respective algorithm outputs an α-approximation in time $\mathcal{O}^*(d^n)$.

Vertex Cover (VC). In [3] Bourgeois, Escoffier and Paschos designed several exponential approximation algorithms for VC for approximation ratios in the range $(1, 2)$. For any $\alpha \in (1, 2)$ the best known running time of a parameterized randomized α-approximation algorithm for VC is attained in [18] if $\alpha \gtrsim 1.03$, and in [5] if $\alpha \lesssim 1.03$. We use these algorithms in conjunction with Theorem 2.1 to obtain faster randomized exponential α-approximation algorithms for VC for values of α in the range $(1, 2)$. We compare our running times to the running times obtained by the naive conversion (Benchmark 2) and to the running times in [3].[4] We present the running time for selected approximation ratios in Table 1 and give a graphical comparison in Figure 1a.

Table 1 Results for Vertex Cover and 3-Hitting Set. A value d at the column of an approximation ratio α means that the respective algorithm outputs an α-approximation in time $\mathcal{O}^*(d^n)$.

Vertex Cover

ratio	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9
`brute`(α)	1.716	1.583	1.496	1.433	1.385	1.347	1.317	1.291	1.269
BEP [3]	1.127	1.099	1.083	1.069	1.056	1.043	1.032	1.021	1.01
[18]+Naive Conv.	1.149	1.079	1.044	1.0236	1.0110	1.00469	1.00162	1.000406	1.0000432
[18]+Theorem 2.1	1.114	1.064	1.036	1.0203	1.0099	1.00435	1.00156	1.000397	1.0000428

3-Hitting Set

ratio	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8
`brute`(α)	1.583	1.433	1.347	1.291	1.251	1.220	1.196	1.177	1.162
[18]+Naive Conv.	1.471	1.196	1.105	1.0582	1.0326	1.0173	1.00831	1.00324	1.000903
[18]+Theorem 2.1	1.240	1.119	1.0698	1.0417	1.0248	1.0140	1.00711	1.00292	1.000853

[4] The result of [3] provides an α-approximation algorithm for every $\alpha \in (1, 2)$. As the evaluation of these running times is not trivial, we only provide the running times which were explicitly given in [3] for selected approximation ratios.

3-Hitting Set (3-HS). The problem admits a simple polynomial-time 3-approximation algorithm which cannot be improved assuming UGC [17]. For any $\alpha \in (1,3)$ the best known running time of a parameterized α-approximation algorithm for 3-HS is attained by either [12] if $\alpha \lesssim 1.08$, or [18] if $\alpha \gtrsim 1.08$. Using these algorithms as parameterized extension algorithms, we calculate the running times of α-approximation algorithms for 3-HS attained using the naive conversion (Benchmark 2) and Theorem 2.1, for values of $\alpha \in (1,3)$. We provide the running times for selected approximation ratios in Table 1 and a graphical comparison in Figure 1b.

DFVS, Subset DFVS, DOCT, Undirected Multicut. For all these problems [22] gave parameterized 2-approximation algorithms that run in time $\mathcal{O}^*(c^k)$, for some constant $c > 1$. One can easily observe from the description of the DFVS algorithm in [22] that it runs in time $\mathcal{O}^*(1024^k)$. Using Theorem 2.1 we get that DFVS admits an exponential 2-approximation algorithm that runs in time $\mathcal{O}^*(1.2498^n)$. This running time is significantly better than the running time derived using the naive conversion (Benchmark 2) of the algorithm of [22], which does not give anything meaningful for this problem. It is also significantly better than using `Exact-MLS` with the algorithm of [22], which gives $\mathcal{O}^*((\texttt{emls}(1024))^n) = \mathcal{O}^*(1.9991^n)$. It is also qualitatively better than the brute-force 2-approximation algorithm (Benchmark 1), which runs in time $\mathcal{O}^*(1.25^n)$.

Using Lemma 1.1, we can show that we get faster 2-approximation algorithms for all mentioned problems compared to the brute-force 2-approximation algorithm, or the naive conversion of the parameterized algorithms in [22] or the application of `Exact-MLS` with the algorithms of [22]. Note that even though the algorithms derived from Theorem 2.1 are only *qualitatively* better than brute-force approximation, we emphasize that `Approximate-MLS` is always strictly better than brute-force approximation (and the other benchmarks described earlier). Also, it reflects Part 1 of Lemma 1.1, that as c increases (that is, as the parameterized extension algorithm becomes slower), our algorithm converges to the brute-force approximation.

3 Analysis of Approximate Monotone Local Search

This section is dedicated to the proof of Theorem 2.1. As explained earlier, the algorithm promised in Theorem 2.1 is `Approximate-MLS` (Algorithm 2). In Lemma 3.1 we prove the correctness of Algorithm 2 and in Lemma 3.2 we provide a formula for its running time. Finally, in Lemma 3.4 we upper bound the running time of the formula obtained in Lemma 3.2 with $\mathcal{O}^*(\texttt{amls}(\alpha,c)^n)$, thereby proving Theorem 2.1.

▶ **Lemma 3.1** (Correctness ⋆). `Approximate-MLS` *(Algorithm 2) is a randomized α-approximation algorithm for Φ_{MIN}-Subset.*

▶ **Lemma 3.2** (Running time). `Approximate-MLS` *(Algorithm 2) runs in time $f_{\alpha,c}(n) \cdot n^{\mathcal{O}(1)}$ where*

$$f_{\alpha,c}(n) := \sum_{k=0}^{\lfloor \frac{n}{\alpha} \rfloor} \min_{t \in [0,\alpha k] \cap \mathbb{N}} \frac{c^{k-\frac{t}{\alpha}}}{\texttt{hyper}\left(n,k,t,\frac{t}{\alpha}\right)}. \tag{1}$$

Proof. For each choice of k, Algorithm 2 chooses in Step 3 a number t that minimizes $\frac{c^{k-\frac{t}{\alpha}}}{\texttt{hyper}\left(n,k,t,\frac{t}{\alpha}\right)}$. Clearly, this step can be performed in time $n^{\mathcal{O}(1)}$. Afterwards, the algorithm calls Algorithm 1 $\texttt{hyper}\left(n,k,t,\frac{t}{\alpha}\right)^{-1}$ times which takes $\frac{c^{k-\frac{t}{\alpha}}}{\texttt{hyper}(n,k,t,\frac{t}{\alpha})} \cdot n^{\mathcal{O}(1)}$ time in total. So overall, the running time is upper bounded by $f_{\alpha,c}(n) \cdot n^{\mathcal{O}(1)}$. ◀

We now proceed with the main part of the analysis which is to bound $f_{\alpha,c}(n)$ by $\texttt{amls}(\alpha, c)^n$ up to some polynomial factors. We remark at this point (without giving a proof) that our analysis is in fact tight, that is, it can be shown that $f_{\alpha,c}(n)$ is equal to $\texttt{amls}(\alpha, c)^n$ up to some polynomial factors in n.

Recall that $\mathcal{H}(p) = -p \ln p - (1-p)\ln(1-p)$ denotes the entropy function. We will use the following bound on binomial coefficients (see, e.g., [6, Example 11.1.3]):

$$\frac{1}{n+1} \cdot \exp\left(n \cdot \mathcal{H}\left(\frac{k}{n}\right)\right) \le \binom{n}{k} \le \exp\left(n \cdot \mathcal{H}\left(\frac{k}{n}\right)\right) \tag{2}$$

for all $n, k \in \mathbb{N}$ such that $0 \le k \le n$.

Moreover, we also need the following technical lemma. Intuitively speaking, it says that small perturbations to the values of a and b do not change the value of $a \cdot \mathcal{H}\left(\frac{b}{a}\right)$ by a large amount.

▶ **Lemma 3.3** (⋆). *For $0 \le b \le a \le n$ and $\varepsilon, \delta \in [-1, 1]$ such that $a + \varepsilon \ge 0$ and $0 \le b + \delta \le a + \varepsilon$, we have $\left|a \cdot \mathcal{H}\left(\frac{b}{a}\right) - (a+\varepsilon) \cdot \mathcal{H}\left(\frac{b+\delta}{a+\varepsilon}\right)\right| = \mathcal{O}(\log(n))$.*

Finally, recall that $\mathcal{D}\left(a \| b\right) = a \ln \frac{a}{b} + (1-a) \ln \frac{1-a}{1-b}$ denotes the Kullback-Leibler divergence (see, e.g., [6]).

▶ **Lemma 3.4.** *It holds that*

$$f_{\alpha,c}(n) = \mathcal{O}^*(\texttt{amls}(\alpha, c)^n).$$

Proof. Recall that $\texttt{hyper}\left(n, k, t, \frac{t}{\alpha}\right)$ denotes the probability that a uniformly random set X of t elements out of $[n]$ satisfies that $|X \cap [k]| \ge \frac{t}{\alpha}$. Thus, we have that

$$\texttt{hyper}\left(n, k, t, \frac{t}{\alpha}\right) = \sum_{y \ge \lceil \frac{t}{\alpha} \rceil} \frac{\binom{k}{y}\binom{n-k}{t-y}}{\binom{n}{t}} \ge \frac{\binom{k}{\lceil \frac{t}{\alpha} \rceil}\binom{n-k}{t - \lceil \frac{t}{\alpha} \rceil}}{\binom{n}{t}} = \frac{\binom{t}{\lceil \frac{t}{\alpha} \rceil}\binom{n-t}{k - \lceil \frac{t}{\alpha} \rceil}}{\binom{n}{k}}, \tag{3}$$

where the last equality holds since the distribution of $|X \cap [k]|$, where $X \subseteq [n]$ is a uniformly random set of cardinality t, is identical to the distribution of $|Y \cap [t]|$ where $Y \subseteq [n]$ is a uniformly random set of cardinality k.

Using (3) we have

$$\begin{aligned} f_{\alpha,c}(n) &= \sum_{k=0}^{\lfloor \frac{n}{\alpha} \rfloor} \min_{t \in [0, \alpha k] \cap \mathbb{N}} \frac{c^{k - \frac{t}{\alpha}}}{\texttt{hyper}\left(n, k, t, \frac{t}{\alpha}\right)} \le \sum_{k=0}^{\lfloor \frac{n}{\alpha} \rfloor} \min_{t \in [0, \alpha k] \cap \mathbb{N}} \frac{c^{k - \frac{t}{\alpha}} \cdot \binom{n}{k}}{\binom{t}{\lceil \frac{t}{\alpha} \rceil}\binom{n-t}{k - \lceil \frac{t}{\alpha} \rceil}} \\ &\le n^{\mathcal{O}(1)} \cdot \sum_{k=0}^{\lfloor \frac{n}{\alpha} \rfloor} \binom{n}{k} \exp\left(\min_{t \in [0, \alpha k] \cap \mathbb{N}} \left(\left(k - \frac{t}{\alpha}\right) \ln(c) - t \cdot \mathcal{H}\left(\frac{\lceil \frac{t}{\alpha} \rceil}{t}\right) - (n-t) \cdot \mathcal{H}\left(\frac{k - \lceil \frac{t}{\alpha} \rceil}{n-t}\right)\right)\right) \quad (4) \\ &\le n^{\mathcal{O}(1)} \cdot \sum_{k=0}^{\lfloor \frac{n}{\alpha} \rfloor} \binom{n}{k} \exp\left(\min_{t \in [0, \alpha k] \cap \mathbb{N}} \left(\left(k - \frac{t}{\alpha}\right) \ln(c) - t \cdot \mathcal{H}\left(\frac{1}{\alpha}\right) - (n-t) \cdot \mathcal{H}\left(\frac{k - \frac{t}{\alpha}}{n-t}\right)\right)\right) \end{aligned}$$

where the second inequality follows from (2) and the third inequality follows from Lemma 3.3.

Define

$$g_{n,k}(t) := \left(k - \frac{t}{\alpha}\right) \ln(c) - t \cdot \mathcal{H}\left(\frac{1}{\alpha}\right) - (n-t) \cdot \mathcal{H}\left(\frac{k - \frac{t}{\alpha}}{n-t}\right).$$

By Lemma 3.3 it holds that $|g_{n,k}(t) - g_{n,k}(t-\varepsilon)| = \mathcal{O}(\log n)$ for any $t \in [0, \alpha k]$ and $0 \leq \varepsilon \leq \min\{1, t\}$. Using this observation and (4) we get

$$\begin{aligned} f_{\alpha,c}(n) &\leq n^{\mathcal{O}(1)} \cdot \sum_{k=0}^{\lfloor \frac{n}{\alpha} \rfloor} \binom{n}{k} \exp\left(\min_{t\in[0,\alpha k]\cap\mathbb{N}} g_{n,k}(t) \right) \\ &\leq n^{\mathcal{O}(1)} \cdot \sum_{k=0}^{\lfloor \frac{n}{\alpha} \rfloor} \binom{n}{k} \exp\left(\min_{t\in[0,\alpha k]} g_{n,k}(t) \right). \end{aligned} \tag{5}$$

Observe that in the last term the range of t is not restricted to integral values.

Let $\delta^* = \frac{\texttt{amls}(\alpha,c)-1}{c-1}$. By the definition of `amls` it holds that $\delta^* \in (0, \frac{1}{\alpha})$ and $\mathcal{D}\left(\frac{1}{\alpha} \middle\| \delta^*\right) = \frac{\ln(c)}{\alpha}$. Define $t^*(n,k) := \frac{k - n\delta^*}{\frac{1}{\alpha} - \delta^*}$, thus $\frac{k - \frac{t^*(n,k)}{\alpha}}{n - t^*(n,k)} = \delta^*$ and $\mathcal{D}\left(\frac{1}{\alpha} \middle\| \frac{k - \frac{t^*(n,k)}{\alpha}}{n - t^*(n,k)}\right) = \frac{\ln c}{\alpha}$ for every $n \in \mathbb{N}$ and $k \in \mathbb{N}$. It can be verified that $g_{n,k}(t)$ is convex and has a global minimum at $t^*(n,k)$, though this observation is not directly used by our proof.

For any $n \in \mathbb{N}$ and $k \in \left[0, \frac{n}{\alpha}\right] \cap \mathbb{N}$ it holds that $t^*(n,k) = \frac{\alpha k(\frac{1}{\alpha} - \delta^*) + \alpha k \delta^* - n\delta^*}{\frac{1}{\alpha} - \delta^*} \leq \alpha k$ since $\alpha k \leq n$. Furthermore, $t^*(n,k) \geq 0$ if and only if $k \geq n\delta^*$. Following this observation we partition the summation in (5) into two parts. Define

$$A(n) = \sum_{k=0}^{\lfloor n\cdot\delta^* \rfloor} \binom{n}{k} \exp\left(\min_{t\in[0,\alpha k]} g_{n,k}(t) \right) \text{ and } B(n) = \sum_{k=\lfloor n\cdot\delta^*\rfloor+1}^{\lfloor \frac{n}{\alpha} \rfloor} \binom{n}{k} \exp\left(\min_{t\in[0,\alpha k]} g_{n,k}(t) \right).$$

Thus, $f_{\alpha,c}(n) \leq n^{\mathcal{O}(1)} \cdot (A(n) + B(n))$. We bound each of the sums $A(n)$ and $B(n)$ separately.

In order to bound $B(n)$ we use the following algebraic identity.

▷ Claim 3.5 (⋆). It holds that

$$g_{n,k}(t) = \left(k - \frac{t}{\alpha}\right)\ln(c) + t \cdot \mathcal{D}\left(\frac{1}{\alpha} \middle\| \frac{k - \frac{t}{\alpha}}{n-t}\right) + k \cdot \ln\left(\frac{k - \frac{t}{\alpha}}{n-t}\right) + (n-k)\cdot\ln\left(1 - \frac{k - \frac{t}{\alpha}}{n-t}\right).$$

For any $n \in \mathbb{N}$ and $k \in \left[n\delta^*, \frac{n}{\alpha}\right] \cap \mathbb{N}$ it holds that $0 \leq t^*(n,k) \leq \alpha k$. Thus,

$$\begin{aligned} \min_{t\in[0,\alpha k]} g_{n,k}(t) &\leq g_{n,k}\left(t^*(n,k)\right) \\ &= \left(k - \frac{t^*(n,k)}{\alpha}\right)\ln(c) + t^*(n,k)\cdot\mathcal{D}\left(\frac{1}{\alpha} \middle\| \delta^*\right) + k\cdot\ln(\delta^*) + (n-k)\cdot\ln(1-\delta^*) \\ &= k\cdot\ln(c) + k\cdot\ln(\delta^*) + (n-k)\cdot\ln(1-\delta^*) \\ &= k\cdot\ln\left(\frac{c\cdot\delta^*}{1-\delta^*}\right) + n\cdot\ln(1-\delta^*), \end{aligned}$$

where the first equality uses Claim 3.5 and the second equality follows from $\mathcal{D}\left(\frac{1}{\alpha} \middle\| \delta^*\right) = \frac{\ln(c)}{\alpha}$. Therefore,

$$\begin{aligned} B(n) &= \sum_{k=\lceil n\delta^*\rceil+1}^{\lfloor \frac{n}{\alpha} \rfloor} \binom{n}{k} \cdot \exp\left(\min_{t\in[0,\alpha k]} g_{n,k}(t) \right) \leq \sum_{k=0}^{n} \binom{n}{k} \left(\frac{c\cdot\delta^*}{1-\delta^*}\right)^k (1-\delta^*)^n \\ &= (1-\delta^*)^n \left(\frac{c\cdot\delta^*}{1-\delta^*} + 1\right)^n = ((c-1)\delta^* + 1)^n \end{aligned} \tag{6}$$

using the Binomial Theorem.

We now proceed to bound $A(n)$. For every $n \in \mathbb{N}$ and $0 \leq k \leq \delta^* n$ it holds that

$$\min_{t\in[0,\alpha k]} g_{n,k}(t) \leq g_{n,k}(0) = k \cdot \ln(c) - n \cdot \mathcal{H}\left(\frac{k}{n}\right) \leq k \cdot \ln c - \ln \binom{n}{k},$$

where the last inequality follows from (2). Therefore,

$$A(n) = \sum_{k=0}^{\lfloor n\delta^* \rfloor} \binom{n}{k} \cdot \exp\left(\min_{t\in[0,\alpha k]} g_{n,k}(t)\right) \leq \sum_{k=0}^{\lfloor n\delta^* \rfloor} c^k \leq n \cdot (c^{\delta^*})^n. \tag{7}$$

Finally, by using (6) and (7), we get

$$\begin{aligned} f_{\alpha,c}(n) &\leq n^{\mathcal{O}(1)} \cdot (A(n) + B(n)) \leq n^{\mathcal{O}(1)} \cdot \left((c^{\delta^*})^n + ((c-1)\delta^* + 1)^n\right) \\ &\leq n^{\mathcal{O}(1)} \cdot ((c-1)\delta^* + 1)^n, \end{aligned}$$

where the third inequality uses $c^{\delta^*} \leq (c-1)\delta^* + 1$ which holds because $f(x) := c^x - (c-1)x - 1$ is convex and has two roots at 0 and 1. By the definition of δ^* it holds that $(c-1)\delta^* + 1 = \texttt{amls}(\alpha, c)$ and thus, $f_{\alpha,c}(n) \leq n^{\mathcal{O}(1)} \cdot \texttt{amls}(\alpha, c)^n$. ◀

Finally, Theorem 2.1 is implied by Lemmas 3.1, 3.2 and 3.4.

4 Derandomization

In this section, we show how to derandomize Algorithm 2. In particular, we prove Theorem 2.2. As usual, let $(U_I, \mathcal{F}_I)$ be a set system and let $k, t, n \in \mathbb{N}$ be the variables from Algorithm 2 and let $\alpha \geq 1$. In order to derandomize the algorithm, it is sufficient to find a collection $\mathcal{C}$ of subsets of U_I of size t such that, for every possible solution set $S \subseteq U_I$ of size k, there is some set $X \in \mathcal{C}$ such that $|X \cap S| \geq \frac{t}{\alpha}$. We refer to such a family $\mathcal{C}$ as an $(n, k, t, \frac{t}{\alpha})$-*set-intersection-family* which is formally defined below.

▶ **Definition 4.1.** *Let U be a universe of size n and let $p, q, r \geq 1$ such that $n \geq p \geq r$ and $n - p + r \geq q \geq r$. A family $\mathcal{C} \subseteq \binom{U}{q}$ is a (n, p, q, r)-*set-intersection-family *if for every $T \in \binom{U}{p}$ there is some $X \in \mathcal{C}$ such that $|T \cap X| \geq r$.*

Given a $(n, k, t, \frac{t}{\alpha})$-*set-intersection-family* $\mathcal{C}$ we can derandomize Algorithm 2 by iterating over all choices $X \in \mathcal{C}$ instead of repeatedly sampling a set X uniformly at random. Observe that the derandomized algorithm (for a fixed k, t) runs in time $\mathcal{O}^*(|\mathcal{C}| \cdot c^{k-\frac{t}{\alpha}})$. Now, we define

$$\kappa(n, p, q, r) := \frac{\binom{n}{q}}{\binom{p}{r} \cdot \binom{n-p}{q-r}}.$$

The following theorem computes the desired set-intersection-family of small size.

▶ **Theorem 4.2** ($\star$)**.** *There is an algorithm that, given a set U of size n and $p, q, r \geq 1$ such that $n \geq p \geq r$ and $n - p + r \geq q \geq r$, computes an (n, p, q, r)-set-intersection-family of size $\kappa(n, p, q, r) \cdot 2^{o(n)}$ in time $\kappa(n, p, q, r) \cdot 2^{o(n)}$.*

For the proof of Theorem 4.2 we extend the arguments from [13] which provide such a result for the special case when $q = r$ (which corresponds to the case $\alpha = 1$).

Proof of Theorem 2.2. We proceed analogously to the proof of Theorem 2.1 with the following changes. In Algorithm 2, Step 3 we define

$$t := \operatorname*{argmin}_{t \in [0, \alpha k] \cap \mathbb{N}} \kappa\left(n, k, t, \left\lceil \frac{t}{\alpha} \right\rceil\right) c^{k - \frac{t}{\alpha}}$$

and then compute an $(n, k, t, \frac{t}{\alpha})$-set-intersection-family $\mathcal{C}$ of size $\kappa(n, p, q, r) \cdot 2^{o(n)}$ using Theorem 4.2. Afterwards, we repeatedly execute Algorithm 1 for every $X \in \mathcal{C}$ (instead of sampling X uniformly at random). Repeating the analysis of Lemma 3.2 the running time is bounded by $\mathcal{O}^*(f_{\alpha,c}(n))$ where,

$$f_{\alpha,c}(n) = \sum_{k=0}^{\lfloor \frac{n}{\alpha} \rfloor} \min_{t \in [0, \alpha k] \cap \mathbb{N}} \kappa\left(n, k, t, \left\lceil \frac{t}{\alpha} \right\rceil\right) c^{k - \frac{t}{\alpha}} \cdot 2^{o(n)}.$$

As we already proved in Lemma 3.4 it holds that

$$\begin{aligned} \sum_{k=0}^{\lfloor \frac{n}{\alpha} \rfloor} \min_{t \in [0, \alpha k] \cap \mathbb{N}} \kappa\left(n, k, t, \left\lceil \frac{t}{\alpha} \right\rceil\right) c^{k - \frac{t}{\alpha}} &= \sum_{k=0}^{\lfloor \frac{n}{\alpha} \rfloor} \min_{t \in [0, \alpha k] \cap \mathbb{N}} \frac{c^{k - \lceil \frac{t}{\alpha} \rceil} \cdot \binom{n}{t}}{\binom{k}{\lceil \frac{t}{\alpha} \rceil} \binom{n-k}{t - \lceil \frac{t}{\alpha} \rceil}} \\ &= \sum_{k=0}^{\lfloor \frac{n}{\alpha} \rfloor} \min_{t \in [0, \alpha k] \cap \mathbb{N}} \frac{c^{k - \lceil \frac{t}{\alpha} \rceil} \cdot \binom{n}{k}}{\binom{t}{\lceil \frac{t}{\alpha} \rceil} \binom{n-t}{k - \lceil \frac{t}{\alpha} \rceil}} \leq \mathtt{amls}(\alpha, c)^n \cdot n^{\mathcal{O}(1)} \end{aligned}$$

which results in the running time stated in the theorem. ◀

5 The Brute-Force Approximation Algorithm

In this section we describe an α-approximate variant of exhaustive search that runs in time $\mathcal{O}^*\left((\mathtt{brute}(\alpha))^n\right)$, where $\mathtt{brute}(\alpha) = 1 + \exp(-\alpha \cdot \mathcal{H}\left(\frac{1}{\alpha}\right))$. We complement this result by showing that $\mathcal{O}^*\left((\mathtt{brute}(\alpha))^n\right)$ is the best possible running time of an α-approximation algorithm for a subset minimization problem in the *oracle model* defined below.

A (randomized) *oracle α-approximation minimum subset* algorithm takes as input a universe U and receives a membership oracle to a *monotone* family $\mathcal{F} \subseteq 2^U$. The algorithm returns a set $S \in \mathcal{F}$ such that $|S| \leq \alpha \cdot \min\left\{|T| \;\middle|\; T \in \mathcal{F}\right\}$ (with constant probability).

▶ **Theorem 5.1.** *For any $\alpha \geq 1$, there is a deterministic oracle α-approximation minimum subset algorithm which runs in time $\mathcal{O}^*((\mathtt{brute}(\alpha))^n)$. Moreover, there is no randomized oracle α-approximation minimum subset algorithm which uses $\mathcal{O}^*(c^n)$ oracle queries for any $c < \mathtt{brute}(\alpha)$.*

The proof of Theorem 5.1 utilizes the technical bound proved in Lemma 5.2. The proof of Lemma 5.2 uses arguments that similar to the ones used in the proof of Lemma 3.4.

▶ **Lemma 5.2** ($\star$). *For any $n \in \mathbb{N}$ and $\alpha \geq 1$ it holds that*

$$n^{-\mathcal{O}(1)} \cdot (\mathtt{brute}(\alpha))^n \leq \max_{k \in \left[0, \frac{n}{\alpha}\right) \cap \mathbb{N}} \frac{\binom{n}{k}}{\binom{\lfloor \alpha k \rfloor}{k}} \leq n^{\mathcal{O}(1)} \cdot (\mathtt{brute}(\alpha))^n$$

To obtain the claimed algorithm of Theorem 5.1 the basic idea is to sample $\mathcal{O}^*((\mathtt{brute}(\alpha))^n)$ random sets (of some size k) and show that the desired approximate solution is found with constant probability. This algorithm can be derandomized by using

Theorem 4.2 for the special case when $p = r$. However, this introduces another sublinear term in the exponent of the running time. Instead, in this special case, we can rely on existing results on covering families [19].

Let $k < t < n$ be natural numbers and recall $[n] := \{1, 2, \dots, n\}$. An (n, t, k)-*covering* is a family $\mathcal{C} \subseteq \{X \mid X \subseteq [n], |X| = t\}$ such that, for every $S \subseteq [n]$ of size $|S| = k$, there is some $X \in \mathcal{C}$ such that $S \subseteq X$. To construct the algorithm for Theorem 5.1, we exploit known constructions of (n, t, k)-coverings of almost optimal size.

▶ **Theorem 5.3** (Kuzjurin [19]). *There is an algorithm that, given $k < t < n$, computes an (n, t, k)-covering $\mathcal{C}$ of size $(1 + o(1)) \cdot \binom{n}{k}/\binom{t}{k}$ in time $|\mathcal{C}| \cdot n^{\mathcal{O}(1)}$.*

Proof of Theorem 5.1. We first show the algorithmic part. By renaming elements, we may assume $U = [n]$. For every $k \leq n/\alpha$ we computes a $(n, \lfloor \alpha k \rfloor, k)$-covering $\mathcal{C}_k$ using Theorem 5.3 and check for every set $X \in \mathcal{C}_k$ whether it is contained in $\mathcal{F}$ using the oracle. We return the smallest set that is contained in $\mathcal{F}$. If no such set is found, we return the entire universe.

It is easy to see that this algorithm is an α-approximation algorithm. Indeed, let $\mathtt{OPT} \subseteq U$ be a solution set of minimum cardinality and let $k := |\mathtt{OPT}|$. If $k \geq n/\alpha$, then the algorithm is clearly correct since even returning the entire universe gives the desired approximation ratio. So suppose that $k \leq n/\alpha$. By definition of an $(n, \lfloor \alpha k \rfloor, k)$-covering there is some $X \in \mathcal{C}_k$ such that $\mathtt{OPT} \subseteq X$ and $|X| = \lfloor \alpha k \rfloor \leq \alpha k$. Since $\mathcal{F}$ is monotone we get that $X \in \mathcal{F}$ and the algorithm returns a solution set of size at most $|X| \leq \alpha k$.

By Theorem 5.3, the algorithm runs in time $\max_{k \in [0, \frac{n}{\alpha}) \cap \mathbb{N}} \binom{n}{k}/\binom{\lfloor \alpha k \rfloor}{k} \cdot n^{\mathcal{O}(1)}$. From Lemma 5.2 we get that the running time is bounded by $(\mathtt{brute}(\alpha))^n \cdot n^{\mathcal{O}(1)}$.

Next, we prove the lower bound. For $n \in \mathbb{N}$ define $\kappa(n) = \operatorname{argmax}_{k \in [0, \frac{n}{\alpha}) \cap \mathbb{N}} \binom{n}{k}/\binom{\lfloor \alpha k \rfloor}{k}$. For every $n \in \mathbb{N}$ we define $\mathcal{F}_{\mathtt{adv}}(n) = \{S \subseteq [n] \mid |S| > \alpha\kappa(n)\}$. Also, for every $n \in \mathbb{N}$ and $X \subseteq [n]$ we define $\mathcal{F}(n, X) = \{S \subseteq [n] \mid X \subseteq S\} \cup \mathcal{F}_{\mathtt{adv}}(n)$. It can be easily observed that $\mathcal{F}(n, X)$ and $\mathcal{F}_{\mathtt{adv}}(n)$ are monotone set families. Our bound is based on the difficulty that algorithms have to distinguish between $\mathcal{F}_{\mathtt{adv}}(n)$ and $\mathcal{F}(n, X)$.

Let $\mathcal{A}$ be a randomized oracle α-approximation minimum subset algorithm. Without loss of generality we assume $\mathcal{A}$ only returns a set S if it queried the oracle with that set (and got back a positive answer). Let $q(n)$ be the maximal number of oracle queries the algorithm uses given a universe of size n. As the algorithm is randomized, we use R to denote the random sequence of bits used by the algorithm.

Fix $n \in \mathbb{N}$. For any $j \in [q(n)]$ the j-th query to the oracle is a function of the previous answers the algorithm received from the oracle and the sequence of random bits the algorithm uses. Thus, there is a function $S_j(R)$ which returns the j-th query the algorithm sends to the oracle, given that the algorithm gets an oracle to $\mathcal{F}_{\mathtt{adv}}(n)$. If the algorithm does not issue the j-th query given R we arbitrarily define $S_j(R) = \emptyset$.

Let $X \subseteq [n]$ be a random set of size $\kappa(n)$ which is sampled uniformly (and independently of R). Consider the execution of $\mathcal{A}$ with the universe $[n]$ and an oracle for $\mathcal{F}(n, X)$. Define

$$\mathcal{C}(R) = \bigcup_{j=1}^{q(n)} \begin{cases} \{T \subseteq [n] \mid |T| = \kappa(n),\ T \subseteq S_j(R)\} & \text{if } |S_j(R)| \leq \alpha \cdot \kappa(n) \\ \emptyset & \text{otherwise} \end{cases}. \tag{8}$$

If $X \notin \mathcal{C}(R)$ then the answers the algorithm receives to its queries are identical to the answers it would have received if it was given an oracle $\mathcal{F}_{\mathtt{adv}}(n)$. It therefore asks the same queries, and must return a set $S \in \mathcal{F}_{\mathtt{adv}}(n)$. As $\mathcal{A}$ is a randomized α-approximation algorithm, there is $\gamma \in (0, 1]$ such that $\mathcal{A}$ returns a set $S \in \mathcal{F}(n, X)$ which satisfies $|S| \leq \alpha \cdot |X| = \alpha \cdot \kappa(n)$ with probability at least γ. As all the sets in $\mathcal{F}_{\mathtt{adv}}$ have cardinality greater than $\alpha \cdot \kappa(n)$ it follows that $\Pr(X \notin \mathcal{C}(R)) \leq 1 - \gamma$, or equivalently, $\Pr(X \in \mathcal{C}(R)) \geq \gamma$.

By the definition of $\mathcal{C}(R)$ (8), each query $S_j(R)$ adds at most $\binom{\lfloor \alpha\cdot\kappa(n)\rfloor}{\kappa(n)}$ sets to $\mathcal{C}(R)$. Thus $\mathcal{C}(R) \leq q(n)\cdot\binom{\lfloor \alpha\cdot\kappa(n)\rfloor}{\kappa(n)}$. Since X is independent of R we have,

$$\gamma \leq \Pr(X \in \mathcal{C}(R)) \leq \frac{q(n)\cdot\binom{\lfloor \alpha\cdot\kappa(n)\rfloor}{\kappa(n)}}{\binom{n}{\kappa(n)}},$$

and hence

$$q(n) \geq \gamma \cdot \frac{\binom{n}{\kappa(n)}}{\binom{\lfloor \alpha\cdot\kappa(n)\rfloor}{\kappa(n)}} = \gamma \cdot \max_{k\in\left[0,\frac{n}{\alpha}\right)\cap\mathbb{N}} \frac{\binom{n}{k}}{\binom{\lfloor \alpha k\rfloor}{k}} \geq n^{-\mathcal{O}(1)}\cdot\left(\mathtt{brute}(\alpha)\right)^n,$$

where the equality follows from the definition of $\kappa(n)$ and the last inequality follows from Lemma 5.2. In particular, $\mathcal{A}$ does not use $\mathcal{O}^*(c^n)$ oracle queries for any $c < \mathtt{brute}(\alpha)$. ◀

6 Concluding Remarks

We introduced and analyzed approximate monotone local search `Approximate-MLS` which can be used to obtain faster exponential approximation algorithms from parameterized (extension) approximation algorithms for monotone subset minimization problems. In particular, we obtain faster exponential approximation algorithms for Vertex Cover, 3-Hitting Set, DFVS, Subset DFVS, DOCT and Undirected Multicut (for some approximation ratios).

Following the submission of this work we became aware of a similar application of monotone local search in the PhD thesis of Lee [20]. However, Lee mainly provides experimental evaluations of the running time of approximate monotone local search for specific problems such as Vertex Cover and Feedback Vertex Set, and does not provide a rigorous analysis of the general running time which is the main focus of this work.

The significance of `Exact-MLS` stems from the abundance of existing parameterized (extension) algorithms which can be used to obtain the state-of-art exponential algorithms for multiple problems. `Approximate-MLS` has a similar potential in the context of exponential approximation algorithms. Thus, our result further emphasizes the importance of the already-growing field of parameterized approximation, by exhibiting its strong connections with exponential-time approximations.

Some interesting follow-up questions of our work are the following.

▶ **Problem 6.1.** *Can an α-approximate algorithm running in time $\mathcal{O}^*((\mathtt{brute}(\alpha)-\varepsilon)^n)$ be derived from a parameterized extension β-approximation algorithm for any $\beta > \alpha$?*

For example, for Directed Feedback Vertex Set only a parameterized 2-approximation algorithm running in time $\mathcal{O}^*(c^k)$ [22] is currently available. The question is whether this algorithm can also be used to obtain an exponential 1.1-approximation algorithm that runs in time $\mathcal{O}^*((\mathtt{brute}(1.1)-\varepsilon)^n)$, for some $\varepsilon > 0$?

We also described and showed that the exhaustive search analog in the α-approximate setting achieves the best possible running time of $\mathcal{O}^*((\mathtt{brute}(\alpha))^n)$ when one only has access to a membership oracle for the problem. Observe that for $\alpha = 1$, SETH asserts that $(\mathtt{brute}(1))^n = 2^n$ is tight.

▶ **Problem 6.2.** *Does there exist a monotone subset minimization problem for which there is no α-approximation algorithm that runs in time $\mathcal{O}^*\left((\mathtt{brute}(\alpha)-\varepsilon)^n\right)$, assuming* SETH*?*

Recall that the `Approximate-MLS` algorithm only uses random sampling and the given parameterized α-approximation extension algorithm. Another interesting lower bound question is the following.

▶ **Problem 6.3.** *Can one show that the running time of* `Approximate-MLS` *is tight (up to polynomial factors) when one is only given access to a membership oracle and a parameterized α-approximation extension algorithm as a black-box?*

References

1 Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. *J. ACM*, 62(5):42:1–42:25, 2015. `doi:10.1145/2775105`.

2 Nikhil Bansal, Parinya Chalermsook, Bundit Laekhanukit, Danupon Nanongkai, and Jesper Nederlof. New tools and connections for exponential-time approximation. *Algorithmica*, 81(10):3993–4009, 2019. `doi:10.1007/s00453-018-0512-8`.

3 Nicolas Bourgeois, Bruno Escoffier, and Vangelis Th. Paschos. Approximation of max independent set, min vertex cover and related problems by moderately exponential algorithms. *Discret. Appl. Math.*, 159(17):1954–1970, 2011. `doi:10.1016/j.dam.2011.07.009`.

4 Ljiljana Brankovic and Henning Fernau. Parameterized approximation algorithms for hitting set. In Roberto Solis-Oba and Giuseppe Persiano, editors, *Approximation and Online Algorithms – 9th International Workshop, WAOA 2011, Saarbrücken, Germany, September 8-9, 2011, Revised Selected Papers*, volume 7164 of *Lecture Notes in Computer Science*, pages 63–76. Springer, 2011. `doi:10.1007/978-3-642-29116-6_6`.

5 Ljiljana Brankovic and Henning Fernau. A novel parameterised approximation algorithm for minimum vertex cover. *Theor. Comput. Sci.*, 511:85–108, 2013. `doi:10.1016/j.tcs.2012.12.003`.

6 Thomas M. Cover and Joy A. Thomas. *Elements of information theory.* Wiley-Interscience, 2nd edition, 2006. `doi:10.1002/047174882X`.

7 Marek Cygan, Lukasz Kowalik, and Mateusz Wykurz. Exponential-time approximation of weighted set cover. *Inf. Process. Lett.*, 109(16):957–961, 2009. `doi:10.1016/j.ipl.2009.05.003`.

8 Pavel Dvorák, Andreas Emil Feldmann, Dusan Knop, Tomás Masarík, Tomás Toufar, and Pavel Veselý. Parameterized approximation schemes for Steiner trees with small number of Steiner vertices. *SIAM J. Discret. Math.*, 35(1):546–574, 2021. `doi:10.1137/18M1209489`.

9 Barış Can Esmer, Ariel Kulik, Dániel Marx, Daniel Neuen, and Roohani Sharma. Faster exponential-time approximation algorithms using approximate monotone local search. *CoRR*, abs/2206.13481, 2022. `doi:10.48550/arXiv.2206.13481`.

10 Uriel Feige and Mohammad Mahdian. Finding small balanced separators. In Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 375–384. ACM, 2006. `doi:10.1145/1132516.1132573`.

11 Andreas Emil Feldmann, Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6):146, 2020. `doi:10.3390/a13060146`.

12 Michael R. Fellows, Ariel Kulik, Frances A. Rosamond, and Hadas Shachnai. Parameterized approximation via fidelity preserving transformations. *J. Comput. Syst. Sci.*, 93:30–40, 2018. `doi:10.1016/j.jcss.2017.11.001`.

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 Anupam Gupta, Euiwoong Lee, and Jason Li. Faster exact and approximate algorithms for k-cut. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 113–123. IEEE Computer Society, 2018. `doi:10.1109/FOCS.2018.00020`.

15 Anupam Gupta, Euiwoong Lee, and Jason Li. An FPT algorithm beating 2-approximation for k-cut. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2821–2837. SIAM, 2018. doi:10.1137/1.9781611975031.179.

16 Ken-ichi Kawarabayashi and Bingkai Lin. A nearly 5/3-approximation FPT algorithm for min-k-cut. 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 990–999. SIAM, 2020. doi:10.1137/1.9781611975994.59.

17 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-ε. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008. doi:10.1016/j.jcss.2007.06.019.

18 Ariel Kulik and Hadas Shachnai. Analysis of two-variable recurrence relations with application to parameterized approximations. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 762–773. IEEE, 2020. doi:10.1109/FOCS46700.2020.00076.

19 Nikolai N. Kuzjurin. Explicit constructions of Rödl's asymptotically good packings and coverings. *Comb. Probab. Comput.*, 9(3):265–276, 2000. doi:10.1017/S0963548300004235.

20 Edward Lee. *Exponential time algorithms via separators and random subsets.* PhD thesis, University of New South Wales, 2021. doi:10.26190/unsworks/22740.

21 Euiwoong Lee. Partitioning a graph into small pieces with applications to path transversal. *Math. Program.*, 177(1-2):1–19, 2019. doi:10.1007/s10107-018-1255-7.

22 Daniel Lokshtanov, Pranabendu Misra, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. FPT-approximation for FPT problems. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10–13, 2021*, pages 199–218. SIAM, 2021. doi:10.1137/1.9781611976465.14.

23 Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan. A parameterized approximation scheme for min k-cut. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 798–809. IEEE, 2020. doi:10.1109/FOCS46700.2020.00079.

24 Pasin Manurangsi. A note on max k-vertex cover: Faster fpt-as, smaller approximate kernel and improved approximation. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San Diego, CA, USA*, volume 69 of *OASIcs*, pages 15:1–15:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/OASIcs.SOSA.2019.15.

25 Pasin Manurangsi and Luca Trevisan. Mildly exponential time approximation algorithms for vertex cover, balanced separator and uniform sparsest cut. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 – Princeton, NJ, USA*, volume 116 of *LIPIcs*, pages 20:1–20:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.20.

26 Dániel Marx. Parameterized complexity and approximation algorithms. *Comput. J.*, 51(1):60–78, 2008. doi:10.1093/comjnl/bxm048.

27 Dániel Marx and Igor Razgon. Constant ratio fixed-parameter approximation of the edge multicut problem. *Inf. Process. Lett.*, 109(20):1161–1166, 2009. doi:10.1016/j.ipl.2009.07.016.

28 Igor Razgon. Computing minimum directed feedback vertex set in O*(1.9977^n). In Giuseppe F. Italiano, Eugenio Moggi, and Luigi Laura, editors, *Theoretical Computer Science, 10th Italian Conference, ICTCS 2007, Rome, Italy, October 3-5, 2007, Proceedings*, pages 70–81. World Scientific, 2007. doi:10.1142/9789812770998_0010.

29 Piotr Skowron and Piotr Faliszewski. Chamberlin-Courant rule with approval ballots: approximating the MaxCover problem with bounded frequencies in FPT time. *J. Artificial Intelligence Res.*, 60:687–716, 2017. doi:10.1613/jair.5628.

30 Vijay V. Vazirani. *Approximation algorithms*. Springer, 2001. doi:10.1007/978-3-662-04565-7.