Approximate Monotone Local Search for Weighted Problems

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
University of Bremen, Germany

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

Abstract

In a recent work, Esmer et al. describe a simple method – Approximate Monotone Local Search – to obtain exponential approximation algorithms from existing parameterized exact algorithms, polynomial-time approximation algorithms and, more generally, parameterized approximation algorithms. In this work, we generalize those results to the weighted setting.

More formally, we consider monotone subset minimization problems over a weighted universe of size n (e.g., Vertex Cover, d-Hitting Set and Feedback Vertex Set). We consider a model where the algorithm is only given access to a subroutine that finds a solution of weight at most $\alpha \cdot W$ (and of arbitrary cardinality) in time $c^k \cdot n^{\mathcal{O}(1)}$ where W is the minimum weight of a solution of cardinality at most k. In the unweighted setting, Esmer et al. determine the smallest value d for which a β-approximation algorithm running in time $d^n \cdot n^{\mathcal{O}(1)}$ can be obtained in this model. We show that the same dependencies also hold in a weighted setting in this model: for every fixed $\varepsilon > 0$ we obtain a β-approximation algorithm running in time $\mathcal{O}\left((d+\varepsilon)^n\right)$, for the same d as in the unweighted setting.

Similarly, we also extend a β-approximate brute-force search (in a model which only provides access to a membership oracle) to the weighted setting. Using existing approximation algorithms and exact parameterized algorithms for weighted problems, we obtain the first exponential-time β-approximation algorithms that are better than brute force for a variety of problems including Weighted Vertex Cover, Weighted d-Hitting Set, Weighted Feedback Vertex Set and Weighted Multicut.

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

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

Digital Object Identifier 10.4230/LIPIcs.IPEC.2023.17

Related Version *Full Version*: `https://arxiv.org/abs/2308.15306`

Funding *Ariel Kulik*: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 852780-ERC (SUBMODULAR).
Dániel Marx: Research supported by the European Research Council (ERC) consolidator grant No. 725978 SYSTEMATICGRAPH.

1 Introduction

In this work, we are interested in *subset problems*, where the goal is to find a subset of a given n-sized universe U that satisfies some property Π (e.g., Vertex Cover, Hitting Set, Feedback Vertex Set, Mulitcut). Such problems can trivially be solved in time $\mathcal{O}^*(2^n)$[1], and in the past decades there has been great interest in designing algorithms that beat this exhaustive search and run in time $\mathcal{O}^*(d^n)$ for as small $1 < d < 2$ as possible (see, e.g., [16]). On the other hand, many of the considered problems admit polynomial-time α-approximation algorithms for some constant $\alpha > 1$ (e.g., Vertex Cover admits a polynomial-time 2-approximation [5]). To bridge the gap between exact exponential-time algorithms and polynomial-time α-approximation algorithms for some possibly large constant α, there has been recent interest in *exponential-time approximation algorithms* [2,4,6,9–12,19] to obtain approximation ratios that are better than what is considered possible in polynomial time.

In a recent work, Esmer et al. [12] describe a simple method – Approximate Monotone Local Search – to obtain exponential-time approximation algorithms for certain subset problems from existing parameterized exact algorithms, polynomial-time approximation algorithms and, more generally, parameterized approximation algorithms. More precisely, the focus in [12] lies on *subset minimization problems* where, given a universe U with n elements, we are aiming to find a set $S \subseteq U$ of minimum cardinality satisfying some property Π. To allow for approximation algorithms, we restrict to *monotone* problems, i.e., the family $\mathcal{F}$ of solution sets is closed under taking supersets. In this setting, a *β-approximation algorithm* is asked to return a solution set $S \in \mathcal{F}$ such that $|S| \leq \beta \cdot |\texttt{OPT}|$ where OPT denotes a solution of minimum size. Given access to a parameterized α-approximation algorithm running in time $\mathcal{O}^*(c^k)$ (where the parameter k denotes the size of the desired optimal solution), Esmer et al. [12] determine the best possible value $d = \texttt{amls}(\alpha, c, \beta)$ such that a β-approximation algorithm running in time $\mathcal{O}^*(d^n)$ can be obtained. Using existing parameterized approximation algorithms, which in particular include polynomial-time approximation and exact parameterized algorithms, this leads to the fastest exponential-time approximation algorithms for a variety of problems including Vertex Cover, d-Hitting Set, Feedback Vertex Set and Odd Cycle Transversal.

In this work, we are interested in *weighted* monotone subset minimization problems. Here, the universe U is additionally equipped with a weight function $\mathbf{w}\colon U \to \mathbb{N}$ and we are asking for a solution set $S \in \mathcal{F}$ of minimum weight $\mathbf{w}(S) \coloneqq \sum_{u\in S} \mathbf{w}(u)$. Accordingly, in a β-approximation algorithm, we are seeking a solution set $S \in \mathcal{F}$ such that $\mathbf{w}(S) \leq \beta \cdot \mathbf{w}(\texttt{OPT})$ where OPT denotes a solution of minimum weight. Looking at [12], it can be observed that the obtained algorithms only extend to the weighted setting for the special case $\alpha = \beta = 1$. Indeed, in this particular case Fomin, Gaspers, Lokshtanov and Saurabh [15] already show in an earlier work that an exact parameterized algorithm running in time $\mathcal{O}^*(c^k)$ can be turned into an exact exponential-time algorithm running time $\mathcal{O}^*((2 - \frac{1}{c})^n)$. As already pointed out in [15], the result also holds in a weighted setting and implies exact exponential-time algorithms for, e.g., d-Hitting Set and Feedback Vertex Set. On the other hand, for $\beta > 1$, the algorithm presented in [12] does not work in a weighted setting. In a nutshell, the main idea in [12] is to randomly sample a set of vertices X and to bound the probability that it contains a sufficiently large portion of an optimum solution OPT. However, in a weighted setting, such an approach is destined to fail since even adding a single element of large weight to an optimum solution may lead to an unbounded approximation factor.

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

The main contribution of this work is to adapt the tools from [12] to the weighted setting. Given a parameterized α-approximation algorithm running in time $\mathcal{O}^*(c^k)$ for a weighted subset minimization problem (where the parameter k still denotes the *size* of the desired optimal solution, we give additional details in the next paragraph), for every fixed $\varepsilon > 0$ we obtain a β-approximation algorithm running in time $\mathcal{O}\left((\texttt{amls}(\alpha, c, \beta) + \varepsilon)^n\right)$. Note that this matches the corresponding bound in the unweighted setting up to the additive ε in the base of the exponent in the running time.

To state our main result more precisely, let us formalize the requirements for the parameterized α-approximation algorithm. Similar to [12], we require an *α-approximate extension algorithm.* Such an algorithm receives as input a set $X \subseteq U$ and a number $k \geq 0$. If there is an extension $S \subseteq U$ to a solution set (i.e., $X \cup S \in \mathcal{F}$) of size at most k, then the algorithm outputs a set $T \subseteq U$ such that $X \cup T \in \mathcal{F}$ and $\mathbf{w}(T) \leq \alpha \cdot \mathbf{w}(S^*)$ where S^* is a minimum-weight extension of X of size at most k. Note that the size of T does not need to be bounded in k; the parameter k only restricts the size of an "optimum solution" which we compare against. For example, the polynomial-time 2-approximation algorithm for Weighted Vertex Cover [5] immediately results in a 2-approximate extension algorithm: given a graph G, $X \subseteq V(G)$ and $k \geq 0$, we apply the 2-approximation algorithm to $G - X$ and return the output $T \subseteq V(G)$ (this algorithm behaves independently of k). With this, our main result can informally be stated as follows.

> Suppose a weighted monotone subset minimization problem admits an α-approximate extension algorithm running in time $\mathcal{O}^*(c^k)$. Then there is a β-approximation algorithm running in time $\mathcal{O}^*((\texttt{amls}(\alpha, c, \beta) + \varepsilon)^n)$ for every $\varepsilon > 0$.

The basic idea to achieve this result is to partition the universe U into subsets U_i of elements of roughly the same weight. We apply the results from [12] to each of the sets U_i separately which results in a "query set" for U_i, i.e., a set of queries made by the algorithm from [12] to the α-approximate extension algorithm. The crucial observation is that these queries are made in a non-adaptive way, i.e., the "query set" only depends on the set U_i. We then combine the "query sets" for the blocks U_i into a query set for the whole weighted set U. In particular, the results from [12] are only used in a black-box manner.

For many problems such as Vertex Cover [5], d-Hitting Set [5] and Feedback Vertex Set [3], polynomial-time approximation algorithms directly extend to the weighted setting, and provide α-approximate extension algorithms as discussed above. On the other hand, while many parameterized exact algorithms do not directly extend to the weighted setting, several problems have been studied in the weighted setting. For example, for Weighted Vertex Cover [21] provides a $\mathcal{O}^*(1.363^k)$ algorithm that, given a vertex-weighted graph and a number $k \geq 0$, returns a vertex cover of weight at most W where W is the minimum weight of a vertex cover of size at most k (if there is no vertex cover of size at most k, the algorithm reports failure). As before, to obtain a 1-approximate extension subroutine, we apply the algorithm to the graph $G - X$ (where X is the input set for the extension subroutine). Similar results are also available for Weighted d-Hitting Set [14, 21] and Weighted Feedback Vertex Set [1]. Also, some simple branching algorithms immediately extend to the weighted setting (e.g., deletion to $\mathcal{H}$-free graphs where $\mathcal{H}$ is a finite set of forbidden induced subgraphs). Finally, we can also rely on parameterized approximation algorithms. For example, [18] provides such algorithms for several problems including Weighted Directed FVS and Weighted Multicut.

We remark that there are also FPT algorithms for other parameterizations in the weighted setting. For example, Niedermeier and Rossmanith [20] show that Weighted Vertex Cover is fixed-parameter tractable parameterized by the weight W of a minimum-weight vertex cover. However, such subroutines are not useful in our setting since we aim to bound the running time of our approximation algorithms with respect to the number of vertices.

Similar to [12], we compare our algorithms to a brute-force search. Here, we consider a setting where the weighted monotone subset minimization problem can only be accessed via a membership oracle, i.e., given a set $X \subseteq U$, we can test (in polynomial time) whether X is a solution set. For every $\beta \geq 1$ we define $\texttt{brute}(\beta) := 1 + \exp\left(-\beta \cdot \mathcal{H}\left(\frac{1}{\beta}\right)\right)$ where $\mathcal{H}(\beta) := -\beta \ln \beta - (1-\beta)\ln(1-\beta)$ denotes the entropy function. In [11] it has been shown that, in the unweighted setting, there is a β-approximation algorithm running in time $\mathcal{O}^*((\texttt{brute}(\beta))^n)$ that only exploits a membership test. We also extend this result to the weighted setting, i.e., we show that any weighted monotone subset minimization problem can be solved in time $(\texttt{brute}(\beta))^{n+o(n)}$ given only a membership oracle.

Esmer et al. [12] show that $\texttt{amls}(\alpha, c, \beta) < \texttt{brute}(\beta)$ for all $\alpha, c \geq 1$ and $\beta > 1$. Since the same bounds are achieved in the weighted setting, all algorithms obtained above are strictly faster than the brute-force β-approximation algorithm. In particular, we obtain exponential-time approximation algorithms that are faster than the approximate brute-force search for the weighted versions of Vertex Cover, d-Hitting Set, Feedback Vertex Set, Tournament FVS, Subset FVS, Cluster Graph Vertex Deletion, Cograph Vertex Deletion, Split Vertex Deletion, Partial Vertex Cover, Directed Feedback Vertex Set, Directed Subset FVS, Directed Odd Cycle Transversal and Multicut (all problems are defined in Appendix B).

Organization

The paper is organized as follows. In Section 2 we state the problems we want to address, provide the necessary definitions and notation, and formally state our main results. In Section 3, we demonstrate how our methods can be applied to specific problems to obtain exponential-time approximation algorithms. Section 4 contains the proof of Theorem 3, our result on exponential-time approximation algorithms for the WeightedSM problem in the membership model. Similarly, Section 5 contains the proof of Theorem 8, our result on exponential-time approximation algorithms for the WeightedSM problem in the extension model. Finally, in Section 6 we conclude the paper by summarizing our main contributions and key findings.

2 Our Results

To formally state our results we use an abstract notion of a problem and oracle-based computational models. Let U be a finite set of elements. We use n to denote the cardinality of U. A set system $\mathcal{F}$ of U is a family $\mathcal{F} \subseteq 2^U$ of subsets of U. We say the set system $\mathcal{F}$ is *monotone* if (i) $U \in \mathcal{F}$ and (ii) for all $T \subseteq S \subseteq U$, if $T \in \mathcal{F}$ then $S \in \mathcal{F}$ as well.

An instance of the *Weighted Monotone Subset Minimization problem* (WeightedSM) is a triplet $(U, \mathbf{w}, \mathcal{F})$ where U is a finite set, $\mathbf{w}\colon U \to \mathbb{N}$ is a weight function over the elements of U, and $\mathcal{F}$ is a monotone set system of U. The set of solutions is $\mathcal{F}$ and the objective is to find $S \in \mathcal{F}$ with minimum total weight $\mathbf{w}(S) := \sum_{u \in S} \mathbf{w}(u)$. We use $\texttt{opt}(U, \mathbf{w}, \mathcal{F}) := \min\{\mathbf{w}(S) \mid S \in \mathcal{F}\}$ to denote the optimum value of a solution to the WeightedSM instance $(U, \mathbf{w}, \mathcal{F})$. We refer to the special case in which $\mathbf{w}(u) = 1$ for all $u \in U$ as the *Unweighted Monotone Subset Minimization problem* (UnweightedSM).

The Weighted Monotone Subset Minimization problem is a meta-problem which captures multiple well studied problems as special cases, e.g., Weighted Vertex Cover, Weighted Feedback Vertex Set and Weighted Multicut. We study the problem using two computational models. In both models the set U and the weight function $\mathbf{w}$ are given as part of the input. The set $\mathcal{F}$ can only be accessed using an oracle, and the models differ in the type of supported oracle queries.

2.1 Membership Oracles and Weighted Approximate Brute Force

In the *membership model* the input to the algorithm is a universe U and a weight function $\mathbf{w}\colon U \to \mathbb{N}$. Additionally, the algorithm has access to a membership oracle for a monotone set system $\mathcal{F}$ of U, that is, the algorithm can check if a subset $S \subseteq U$ satisfies $S \in \mathcal{F}$ in a single step. For every $\alpha \geq 1$, we say an algorithm is an α-approximation for WeightedSM (UnweightedSM) in the membership model if for every WeightedSM (UnweightedSM) instance $(U, \mathbf{w}, \mathcal{F})$ the algorithm returns a set $S \in \mathcal{F}$ such that $\mathbf{w}(S) \leq \alpha \cdot \texttt{opt}(U, \mathbf{w}, \mathcal{F})$.

One can easily attain a 1-approximation algorithm for WeightedSM in the membership model by iterating over all subsets of U and querying the oracle for each. This leads to an algorithm with running time $\mathcal{O}^*(2^n)$. Moreover, it can be easily shown there is no 1-approximation algorithm for WeightedSM (or for UnweightedSM) in the membership model which runs in time $\mathcal{O}\left((2-\varepsilon)^n\right)$. We refer to this algorithm as the (exact) brute force.

Intuitively, for every $\alpha > 1$, it should be possible to design an α-approximation algorithm for WeightedSM and UnweightedSM in the membership model which runs in time $\mathcal{O}(c^n)$ for some $c < 2$. However, the value of the optimal c in this setting is not obvious. In [11] the authors studied UnweightedSM in the membership model and pinpointed the right value of c. For every $\alpha \geq 1$ we define

$$\texttt{brute}(\alpha) = 1 + \exp\left(-\alpha \cdot \mathcal{H}\left(\frac{1}{\alpha}\right)\right), \tag{1}$$

where $\mathcal{H}(x) = -x\ln(x) - (1-x)\ln(1-x)$ is the entropy function.

▶ **Lemma 1** ([11, Theorem 5.1]). *For every $\alpha \geq 1$ the following holds.*

1. *There is a deterministic α-approximation algorithm for* UnweightedSM *in the membership model which runs in time $(\texttt{brute}(\alpha))^n \cdot n^{\mathcal{O}(1)}$.*
2. *Let $\varepsilon > 0$. There is no α-approximation algorithm for* UnweightedSM *in the membership model which runs in time $(\texttt{brute}(\alpha) - \varepsilon)^n \cdot n^{\mathcal{O}(1)}$.*

As the algorithmic result in Lemma 1 can be viewed as an approximate analogue of the brute-force algorithm, it is commonly referred as *α-approximate brute force.* The lower bound given in [11] also holds if the algorithm is allowed to use randomization. As WeightedSM is a generalization of UnweightedSM, the following corollary is an immediate consequence of Lemma 1.

▶ **Corollary 2.** *For every $\alpha \geq 1$ and $\varepsilon > 0$ there is no α-approximation algorithm for* WeightedSM *in the membership model which runs in time $(\texttt{brute}(\alpha) - \varepsilon)^n \cdot n^{\mathcal{O}(1)}$.*

As the bound in Lemma 1 also holds if randomization is allowed, the same holds true for the bound in Corollary 2. Our first result is a generalization of the approximate brute-force algorithm of [11] for the weighted setting. That is, we provide an algorithm which matches the lower bound in Corollary 2 up to a sub-exponential factor.

▶ **Theorem 3** (Weighted Approximate Brute Force). *For every $\alpha > 1$ there is an α-approximation algorithm for* WeightedSM *in the membership model which runs in time* $(\texttt{brute}(\alpha))^{n+o(n)}$.

The proof of Theorem 3 is based on a rounding of the weight function $\mathbf{w}$ and utilizes a construction from [11] which is applied to each of the rounded weight classes.

2.2 Extension Oracles and Weighted Approximate Monotone Local Search

Our second computational model deals with *extension oracles.* The input for these oracles is a set $S \subseteq U$ and a number $\ell \in \mathbb{N}_{\geq 0}$ and the output is an *extension* of S, that is, a set $X \subseteq U$ such that $S \cup X \in \mathcal{F}$. Furthermore, the returned set X is guaranteed to have a small weight in comparison to the minimum-weight extension of S which contains at most ℓ elements. For multiple problems, such as Vertex Cover and Feedback Vertex Set, these oracles can be implemented using existing parameterized algorithms which have a running time of the form $c^\ell \cdot n^{\mathcal{O}(1)}$. We therefore associate a running time of c^ℓ with the query (S, ℓ). The formal definition of extension oracles is as follows.

▶ **Definition 4** (Extension Oracle). *Let $(U, \mathbf{w}, \mathcal{F})$ be a* WeightedSM *instance and let $\alpha \geq 1$. An α-extension oracle for $(U, \mathbf{w}, \mathcal{F})$ is a function* $\texttt{Ext} : 2^U \times \mathbb{N}_{\geq 0} \to 2^U$ *such that for every $S \subseteq U$ and $\ell \in \mathbb{N}_{\geq 0}$ the following holds:*
1. $\texttt{Ext}(S, \ell) \cup S \in \mathcal{F}$.
2. $\mathbf{w}(\texttt{Ext}(S, \ell)) \leq \alpha \cdot \min\{\mathbf{w}(X) \mid X \subseteq U,\ |X| \leq \ell,\ X \cup S \in \mathcal{F}\}$ *(we set $\min \emptyset = \infty$).*

In the (α, c)-*extension model* the input for the algorithm is a finite set U and a weight function $\mathbf{w} \colon U \to \mathbb{N}$. Furthermore, the algorithm is given oracle access to an α-extension oracle $\texttt{Ext}$ of $(U, \mathbf{w}, \mathcal{F})$ for some monotone set system $\mathcal{F}$ of U. For every $\beta \geq 1$, we say an algorithm is a β-approximation algorithm for WeightedSM (UnweightedSM) in the (α, c)-extension model if for every WeightedSM (UnweightedSM) instance $(U, \mathbf{w}, \mathcal{F})$ and α-extension oracle $\texttt{Ext}$ of the instance, the algorithm returns $T \in \mathcal{F}$ such that $\mathbf{w}(T) \leq \beta \cdot \texttt{opt}(U, \mathbf{w}, \mathcal{F})$. The running time of an algorithm in this model is the number of computations steps plus c^ℓ for every query (S, ℓ) issued to the oracle during the execution. Following the standard worst-case analysis convention, we say an algorithm runs in time $f(n)$ if for every WeightedSM instance $(U, \mathbf{w}, \mathcal{F})$ and α-extension oracle $\texttt{Ext}$ of the instance the algorithm runs in time at most $f(|U|)$.

The (α, c)-extension model is studied in [12] for the special case of UnweightedSM. The authors of [12] provide a deterministic β-approximation algorithm in the (α, c)-extension model, known as *deterministic approximate monotone local search*, which runs in time $(\texttt{amls}(\alpha, c, \beta))^{n+o(n)}$, where $\texttt{amls}(\alpha, c, \beta)$ is defined as the optimal value of a continuous optimization problem. Throughout the paper we use $\texttt{amls}$ to denote this function. We provide the formal definition of $\texttt{amls}$ in the full version of the paper for completeness. We note that this formal definition is not required for the understanding of the results in this paper. It is also shown in [12] that the value of $\texttt{amls}(\alpha, c, \beta)$ can be computed up to precision of ε in time polynomial in the encoding length of α, c, β and ε.

This algorithmic result is complemented in [12] with a matching lower bound.

▶ **Lemma 5** ([12]). *For every $\alpha, \beta, c \geq 1$ and $\varepsilon > 0$ there is no deterministic β-approximation for* UnweightedSM *in the (α, c)-extension model which runs in time* $(\texttt{amls}(\alpha, c, \beta) - \varepsilon)^n \cdot n^{\mathcal{O}(1)}$.

Furthermore, it is shown that the running time of the deterministic approximate monotone local search is better than brute force for every $\beta > 1$.

▶ **Lemma 6** ([12]). *For all $\alpha, c \geq 1$ and $\beta > 1$ it holds that $\texttt{amls}(\alpha, c, \beta) < \texttt{brute}(\beta)$.*

The results of [12] also include a randomized algorithm which omits the subexponential factor in the running time and the lower bound also holds for randomized algorithms. In [12] the authors use the deterministic approximate monotone local search algorithm to obtain exponential-time approximation algorithms for multiple *unweighted* problems such as Vertex Cover and Feedback Vertex Set. We generalize the algorithmic results of [12] to the weighted setting and similarly use it to obtain exponential-time approximation algorithms for *weighted* variants of the mentioned problems (see Section 3).

Since UnweightedSM is a special case of WeightedSM, Lemma 5 immediately implies the following

▶ **Corollary 7.** *For every $\alpha, c, \beta \geq 1$ and $\varepsilon > 0$ there is no deterministic β-approximation for* WeightedSM *in the (α, c)-extension model which runs in time $(\texttt{amls}(\alpha, c, \beta) - \varepsilon)^n \cdot n^{\mathcal{O}(1)}$.*

Our second result is an algorithm which matches the running time in Corollary 7 up to an additive term of ε in the base of the running time.

▶ **Theorem 8** (Weighted Approximate Monotone Local Search). *For every $\alpha, c \geq 1$, $\beta > 1$ and $\varepsilon > 0$ there is a deterministic β-approximation for* WeightedSM *in the (α, c)-extension model which runs in time $\mathcal{O}\left((\texttt{amls}(\alpha, c, \beta) + \varepsilon)^n\right)$.*

The proof of Theorem 8 is similar to the one of Theorem 3. It applies rounding to the weights on the elements, and then utilizes a construction from [12] for each of the weight classes.

The result of Theorem 8 only applies for $\beta > 1$. If $\alpha = \beta = 1$ the result of [15] can be used to obtain an exact algorithm with running time $n^{\mathcal{O}(1)} \cdot \left(2 - \frac{1}{c}\right)^n$. For $\alpha > \beta = 1$ Corollary 7 implies that the best possible running time is $\mathcal{O}^*(2^n)$ which can be attained by brute force.

3 Applications

Our results provide exponential-time approximation algorithms for a variety of weighted vertex-deletions problems. For illustration purposes, let us first focus on the Weighted Vertex Cover problem where we are given a graph G with vertex weights $\mathbf{w}\colon V(G) \to \mathbb{N}$, and we ask for a vertex cover $S \subseteq V(G)$ of minimum weight $\mathbf{w}(S) = \sum_{v \in S} \mathbf{w}(v)$. It is well-known that Weighted Vertex Cover admits a polynomial-time 2-approximation algorithm [5]. Also, the problem can be solved exactly in time $\mathcal{O}^*(1.238^n)$ [22].

For the unweighted version Unweighted Vertex Cover, Bourgeois, Escoffier and Paschos [6] designed several exponential-time approximation algorithms for approximation ratios in the range $(1, 2)$. For example, they obtain a 1.1-approximation algorithm running in time $\mathcal{O}^*(1.127^n)$ where n denotes the number of vertices of the input graph. These running times are further improved in [11, 12] using the framework of Approximate Monotone Local Search. Indeed, the fastest known 1.1-approximation algorithm for Unweighted Vertex Cover runs in time $\mathcal{O}^*(1.113^n)$ [12].

For the weighted version, no such results have been obtained so far. We use Theorem 8 to design the first exponential β-approximation algorithms for Weighted Vertex Cover for all $\beta \in (1, 2)$. For the extension oracle, we can rely on the well-known polynomial-time 2-approximation algorithm. Given a set $S \subseteq V(G)$, we delete all vertices in S and apply the

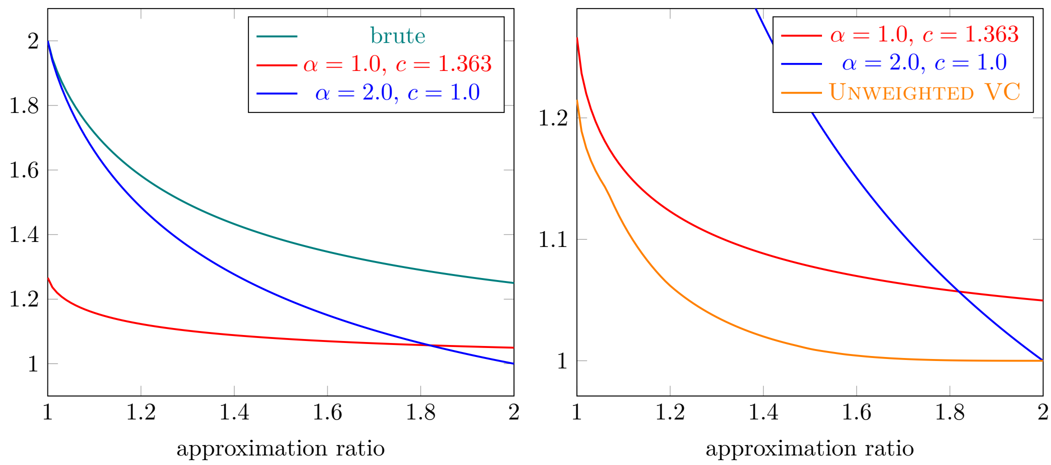


Figure 1 The left figure shows running times for WEIGHTED VERTEX COVER and the right side provides a comparison to UNWEIGHTED VERTEX COVER. A dot at (β, d) means that the respective algorithm outputs an β-approximation in time $\mathcal{O}^*(d^n)$.

Table 1 The table shows the running times for WEIGHTED VERTEX COVER (second and third row) and UNWEIGHTED VERTEX COVER (last row). An entry d in column β means that the respective algorithm outputs a β-approximation in time $\mathcal{O}^*(d^n)$.

	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9
`brute`	1.716	1.583	1.496	1.433	1.385	1.347	1.317	1.291	1.269
$(\alpha = 1, c = 1.363)$	1.158	1.123	1.103	1.089	1.078	1.07	1.064	1.058	1.054
$(\alpha = 2, c = 1)$	1.659	1.485	1.366	1.277	1.208	1.151	1.104	1.064	1.03
UNWEIGHTED VC	1.113	1.062	1.036	1.02	1.01	1.005	1.002	1.0004	1.00005

2-approximation algorithm to the graph $G - S$ which outputs a vertex cover X such that $\mathbf{w}(X) \leq 2 \cdot \mathbf{w}(\texttt{OPT})$ where $\texttt{OPT}$ denotes a minimum vertex cover of $G - S$. As a result, we can implement a 2-extension oracle in polynomial time which corresponds (up to polynomial factors) to cost $c = 1$. So Theorem 8 results in a β-approximation algorithm for WEIGHTED VERTEX COVER which runs in time $\mathcal{O}^*((\texttt{amls}(\alpha, c, \beta) + \varepsilon)^n)$ for every $\beta > 1$, where $\alpha = 2$ and $c = 1$. A visualization is given in Figure 1.

Instead of using a polynomial-time 2-approximation algorithm, we can also rely on existing FPT algorithms for WEIGHTED VERTEX COVER to simulate the extension oracle. Let us point out that different parameterizations have been considered for WEIGHTED VERTEX COVER in the literature. For example, Niedermeier and Rossmanith [20] give FPT algorithms for WEIGHTED VERTEX COVER parameterized by the weight of the optimal solution. However, in light of the computational model introduced above, we require FPT algorithms parameterized by the *size* of the solution. Given a graph G with vertex weights $\mathbf{w}\colon V(G) \to \mathbb{N}$ and an integer $k \geq 0$, we ask for a vertex cover of weight at most W where W is the minimum weight of a vertex cover of size at most k. The best known FPT algorithm (parameterized by k) for this problem runs in time $1.363^k \cdot n^{\mathcal{O}(1)}$ [21]. This algorithm provides an extension algorithm with parameters $\alpha = 1$ and $c = 1.363$. Using Theorem 8, we obtain a β-approximation algorithm for WEIGHTED VERTEX COVER running in time $\mathcal{O}^*((\texttt{amls}(\alpha, c, \beta) + \varepsilon)^n)$. For $\beta = 1.1$, we obtain a running time of $\mathcal{O}^*(1.158^n)$.

Table 2 List of weighted deletion problems admitting a single-exponential parameterized algorithm running in time $O^*(c_1^k)$ and/or a polynomial-time α_2-approximation algorithm. The problems Tournament FVS, Cluster Graph Vertex Deletion, Cograph Vertex Deletion and Split Vertex Deletion can be easily reduced to d-Hitting Set for appropriate values of d by exploiting known characterizations in terms of forbidden induced subgraphs. Additionally, for Tournament FVS we can rely on iterative compression to obtain a $\mathcal{O}^*(2^k)$ algorithm (see, e.g., [8]).

Problem	c_1		det.	α_2		det.
Vertex Cover	1.363	[21]	✓	2	[5]	✓
FVS	3.618	[1]	✓	2	[3]	✓
Tournament FVS	2.0		✓	3		✓
Subset FVS	-		✓	8	[13]	✓
3-Hitting Set	2.168	[21]	✓	3	[5]	✓
d-Hitting Set ($d \geq 4$)	$d - 0.832$	[14]	✓	d	[5]	✓
Cluster Graph Vertex Deletion	2.168		✓	3		✓
Cograph Vertex Deletion	3.168		✓	4		✓
Split Vertex Deletion	4.168		✓	5		✓
Partial Vertex Cover	-		✓	2	[7]	✓

We provide running times for selected approximation ratios for both algorithms in Table 1 and a graphical comparison in Figure 1. We also compare the running times to the approximate brute-force search and the best algorithms in the unweighted setting [12]. It can be observed that the second algorithm (using the FPT algorithm as an extension subroutine) is faster for $\beta \lesssim 1.82$. Also, there is still a noticeable gap to the unweighted setting. This can be mainly explained by the fact that the approximation algorithm for the unweighted setting [12] relies on parameterized approximation algorithms for Unweighted Vertex Cover [17] which are currently unavailable in the weighted setting.

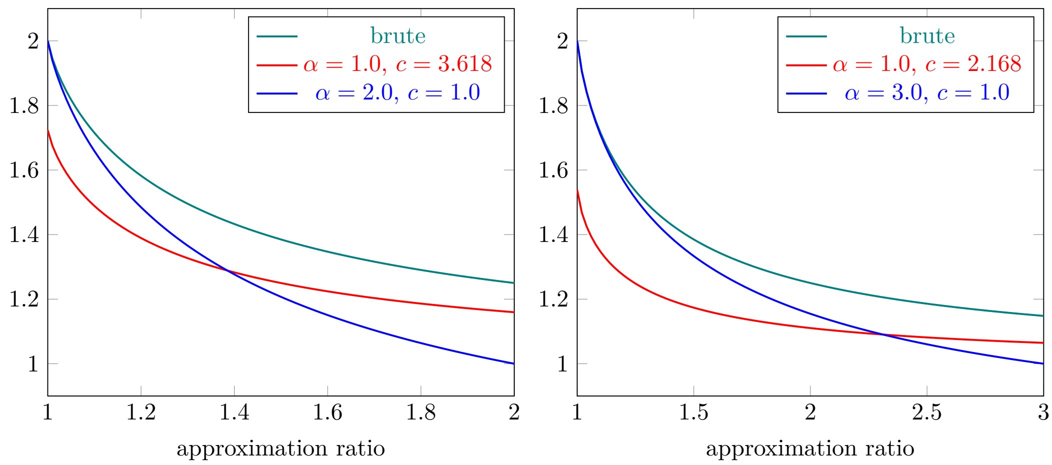
We stress that our results are not limited to Weighted Vertex Cover, but they are applicable to various vertex-deletion problems including Weighted Feedback Vertex Set and Weighted d-Hitting Set (see Figure 2). Table 2 gives an overview on problems for which we obtain exponential approximation algorithms by simulating the extension oracle by an FPT algorithm (parameterized by solution size) running in time $c_1^k \cdot n^{\mathcal{O}(1)}$ or a polynomial-time α_2-approximation algorithm. For all the problems listed in Table 2, we obtain the first exponential β-approximation algorithms for all $\beta \in (1, \alpha_2)$. Observe that these algorithms always outperform the approximate brute-force search by Lemma 6. We provide data on the running times of these algorithms in the full version of the paper.

Finally, let us point out that our methods are also applicable if there is a parameterized approximation algorithm for a weighted vertex deletion problem, i.e., an α-approximation algorithm running time $c^k \cdot n^{\mathcal{O}(1)}$. In [18], such algorithms have been obtained for Weighted Directed Feedback Vertex Set, Weighted Directed Subset FVS, Weighted Directed Odd Cycle Transversal and Weighted Multicut. As a result, we also obtain exponential β-approximation algorithms for these problems that outperform the approximate brute-force search.

4 Weighted Approximate Brute Force

In this section we prove Theorem 3. The algorithm is based on the notion of *covering families*.

▶ **Definition 9** (Covering Family). *Let U be a finite set, $\mathbf{w}\colon U \to \mathbb{N}$ be a weight function and $\alpha \geq 1$. We say $\mathcal{C} \subseteq 2^U$ is an α-*covering family *of U and $\mathbf{w}$ if for every $S \subseteq U$ there exists $T \in \mathcal{C}$ such that $S \subseteq T$ and $\mathbf{w}(T) \leq \alpha \cdot \mathbf{w}(S)$.*



(a) Feedback Vertex Set. **(b)** 3-Hitting Set.

Figure 2 The figure shows running times for Feedback Vertex Set and 3-Hitting Set. A dot at (β, d) means that the respective algorithm outputs an β-approximation in time $O^*(d^n)$.

An α-covering family $\mathcal{C}$ can be easily used to attain an α-approximation algorithm in the membership model as follows. The algorithm constructs the covering family $\mathcal{C}$ and uses the membership oracle to compute $\mathcal{C} \cap \mathcal{F}$ using $|\mathcal{C}|$ queries. The algorithm then returns the set $Q \in \mathcal{C} \cap \mathcal{F}$ of minimum weight. To show correctness, consider a set $S \in \mathcal{F}$ such that $\mathbf{w}(S) = \texttt{opt}(U, \mathbf{w}, \mathcal{F})$. Since $\mathcal{C}$ is an α-covering family there is $T \in \mathcal{C}$ such that $S \subseteq T$ and $\mathbf{w}(T) \leq \alpha \cdot \mathbf{w}(S) = \alpha \cdot \texttt{opt}(U, \mathbf{w}, \mathcal{F})$. Since $\mathcal{F}$ is monotone it holds that $T \in \mathcal{F}$, thus $T \in \mathcal{C} \cap \mathcal{F}$, and we conclude that the algorithm returns a set of weight at most $\alpha \cdot \texttt{opt}(U, \mathbf{w}, \mathcal{F})$. The running time of the algorithm, up to polynomial factors, is the construction time of $\mathcal{C}$ plus $|\mathcal{C}|$.

By the above argument, the proof of Theorem 3 boils down to the construction of α-covering families. To this end, we show the next result. Recall that n denotes the size of the universe U.

▶ **Lemma 10.** *There exists an algorithm which given a finite set U, a weight function $\mathbf{w} : U \to \mathbb{N}$ and $\alpha > 1$, constructs an α-covering family $\mathcal{C}$ of U and $\mathbf{w}$ such that $|\mathcal{C}| \leq (\texttt{brute}(\alpha))^{n+o(n)}$. Furthermore, the running time of the algorithm is $(\texttt{brute}(\alpha))^{n+o(n)}$.*

Using the argument above, Theorem 3 is an immediate consequence of Lemma 10. The construction of the covering family in Lemma 10 is based on a rounding of the weight function and a reduction to covering families in the unweighted case. The construction of such families is implicitly given in [11].

▶ **Lemma 11** ([11]). *There exists an algorithm which given a finite set U and $\alpha > 1$ returns an α-covering family $\mathcal{C}$ of U and the uniform weight function $\mathbf{w} : U \to \{1\}$ such that $|\mathcal{C}| \leq (\texttt{brute}(\alpha))^n \cdot n^{\mathcal{O}(1)}$. Furthermore, the algorithm runs in time $(\texttt{brute}(\alpha))^n \cdot n^{\mathcal{O}(1)}$.*

The proof of Lemma 10 is based on the following seemingly weaker version of Lemma 10.

▶ **Lemma 12.** *There exists an algorithm $\mathscr{A}$ which given a finite set U, a weight function $\mathbf{w} : U \to \mathbb{N}$, $\beta > 1$ and $0 < \delta < 1$, constructs a $(1+\delta) \cdot \beta$-covering family $\mathcal{C}$ of U and $\mathbf{w}$ such that $|\mathcal{C}| \leq (\texttt{brute}(\beta))^n \cdot n^{\mathcal{O}\left(\frac{1}{\delta}\log\left(\frac{n}{\delta}\right)\right)}$. Furthermore, the running time of the algorithm is $(\texttt{brute}(\beta))^n \cdot n^{\mathcal{O}\left(\frac{1}{\delta}\log\left(\frac{n}{\delta}\right)\right)}$.*

Proof. The algorithm $\mathscr{A}$ works as follows:

- Define $\gamma := 1 + \frac{\delta}{2} > 1$. For $i \geq 0$ let
$$U_i := \{u \in U \mid \gamma^i \leq \mathbf{w}(u) < \gamma^{i+1}\} \tag{2}$$
and $n_i := |U_i|$. Let $I := \{i \in \mathbb{Z}_{\geq 0} \mid U_i \neq \emptyset\}$ denote the set of indices $i \geq 0$ for which U_i is non-empty. Note that $|I| \leq n$.
- For each $i \in I$ construct a β-covering family $\mathcal{C}_i$ of the universe U_i and the uniform weight function using Lemma 11.
- Define $d := \lceil (2/\delta) \cdot \log(2n/\delta) \rceil$ and for each $k \in I$, let $I_k := \{i \in I \mid k - d \leq i \leq k\}$ denote the indices in I between $k-d$ and k.
- For every $k \in I$, let $r_k := |I_k|$ and define
$$W_k := \bigcup_{i \in I\colon 1 \leq i < k-d} U_i \quad \text{and}$$
$$\mathcal{Q}_k := \left\{ W_k \cup E_1 \cup \ldots \cup E_{r_k} \;\middle|\; (E_1, \ldots, E_{r_k}) \in \prod_{i \in I_k} \mathcal{C}_i \right\}.$$
- Return the set $\mathcal{C} := \bigcup_{k \in I} \mathcal{Q}_k$.

▷ Claim 13. The algorithm $\mathscr{A}$ returns a $(1+\delta) \cdot \beta$-covering family of U and $\mathbf{w}$.

Proof. Let us pick a set $S \subseteq U$. We show that there exists $T \in \mathcal{C}$ such that $S \subseteq T$ and $\mathbf{w}(T) \leq (1+\delta) \cdot \beta \cdot \mathbf{w}(S)$.

By the definition of γ and d, it holds that
$$\gamma^d \geq \left(1 + \frac{\delta}{2}\right)^{(2/\delta) \cdot \log(2n/\delta)} \geq 2^{\log(2n/\delta)} = \frac{2n}{\delta} \tag{3}$$
using that $(1 + \frac{1}{x})^x \geq 2$ for all $x \geq 1$. Let $k \in I$ be the largest index such that $S \cap U_k \neq \emptyset$. It holds that
$$\mathbf{w}(W_k) < n \cdot \gamma^{k-d} \leq \frac{\delta}{2} \cdot \gamma^k \leq \frac{\delta}{2} \cdot \mathbf{w}(S) \tag{4}$$
where the first inequality follows from the fact that $|W_k| \leq n$ and each $u \in W_k$ belongs to a set U_i where $i \leq k - d - 1$ and therefore $\mathbf{w}(u) < \gamma^{k-d-1+1} = \gamma^{k-d}$ by (2). The second inequality follows from (3) and finally the last inequality holds because by definition of k, there exists $u \in S \cap U_k$ such that $\mathbf{w}(u) \geq \gamma^k$ by (2).

For every $i \in I_k$ define $S_i := S \cap U_i$. Since $\mathcal{C}_i$ is a β-covering family of U_i and the *uniform weight function*, for each $i \in I_k$ there exists $T_i \in \mathcal{C}_i$ such that $S_i \subseteq T_i$ and $|T_i| \leq \beta \cdot |S_i|$. Hence, for all $i \in I_k$ it holds that
$$\mathbf{w}(T_i) \leq \gamma^{i+1} \cdot |T_i| \leq \gamma^{i+1} \cdot \beta \cdot |S_i| \leq \gamma \cdot \beta \cdot \mathbf{w}(S_i) \tag{5}$$
where the first inequality follows from the fact that $T_i \subseteq U_i$ and (2), the second inequality follows from the definition of T_i and finally the last one again follows from the fact that $S_i \subseteq U_i$ and (2).

Let $\overline{T} := \bigcup_{i \in I_k} T_i$ and define
$$T := \left(W_k \cup \overline{T}\right) \in \mathcal{Q}_k \subseteq \mathcal{C}.$$

Then we have

$$S = S \cap U = S \cap \left(\bigcup_{i \in I} U_i \right) = \left(S \cap \left(\cup_{i \in I : i < k-d} U_i \right) \right) \cup \left(S \cap \left(\cup_{i \in I_k} U_i \right) \right)$$
$$= (S \cap W_k) \cup \left(\bigcup_{i \in I_k} (S \cap U_i) \right)$$
$$\subseteq W_k \cup \left(\bigcup_{i \in I_k} T_i \right)$$
$$= T.$$

Finally, it also holds that

$$\begin{aligned} \mathbf{w}(T) &= \mathbf{w}(W_k) + \sum_{i \in I_k} \mathbf{w}(T_i) \\ &\leq \frac{\delta}{2} \cdot \mathbf{w}(S) + \sum_{i \in I_k} \gamma \cdot \beta \cdot \mathbf{w}(S_i) && \text{by (4) and (5)} \\ &\leq \frac{\delta}{2} \cdot \beta \cdot \mathbf{w}(S) + \gamma \cdot \beta \cdot \mathbf{w}(S) \\ &\leq (1+\delta) \cdot \beta \cdot \mathbf{w}(S). \end{aligned}$$

This shows that $\mathcal{C}$ is a $(1+\delta) \cdot \beta$-covering family of U and $\mathbf{w}$. ◁

▷ Claim 14.

$$|\mathcal{C}| \leq (\texttt{brute}(\beta))^n \cdot n^{\mathcal{O}\left(\frac{1}{\delta} \cdot \log\left(\frac{n}{\delta}\right)\right)}.$$

Proof. By Lemma 11, for each $i \in I$ it holds that

$$|\mathcal{C}_i| \leq (\texttt{brute}(\beta))^{n_i} \cdot n_i^c \tag{6}$$

for some $c > 0$. Thus, for every $k \in I$,

$$\begin{aligned} |\mathcal{Q}_k| \leq \left| \prod_{i \in I_k} \mathcal{C}_i \right| = \prod_{i \in I_k} |\mathcal{C}_i| \leq \prod_{i \in I_k} (\texttt{brute}(\beta))^{n_i} \cdot n_i^c \\ = (\texttt{brute}(\beta))^{\sum_{i \in I_k} n_i} \cdot n_i^{c \cdot |I_k|} \\ \leq (\texttt{brute}(\beta))^n \cdot n^{c \cdot (d+1)} \\ = (\texttt{brute}(\beta))^n \cdot n^{\mathcal{O}\left(\frac{1}{\delta} \cdot \log\left(\frac{n}{\delta}\right)\right)}. \end{aligned}$$

Finally, we have that

$$|\mathcal{C}| = \left| \bigcup_{k \in I} \mathcal{Q}_k \right| \leq \sum_{k \in I} |\mathcal{Q}_k| = (\texttt{brute}(\beta))^n \cdot n^{\mathcal{O}\left(\frac{1}{\delta} \cdot \log\left(\frac{n}{\delta}\right)\right)}$$

since $|I| \leq n$. ◁

▷ Claim 15. The running time of $\mathscr{A}$ is $(\texttt{brute}(\beta))^n \cdot n^{\mathcal{O}\left(\frac{1}{\delta} \cdot \log\left(\frac{n}{\delta}\right)\right)}$.

Proof. The construction of $\{\mathcal{C}_i\}_{i\in I}$ takes

$$\sum_{i\in I}(\texttt{brute}(\beta))^{n_i}\cdot n_i^{\mathcal{O}(1)}\leq(\texttt{brute}(\beta))^{n}\cdot n^{\mathcal{O}(1)} \tag{7}$$

which follows from Lemma 11. Finally, the construction of $\mathcal{C}$ takes time proportional to the size of $\mathcal{C}$ where we have

$$|\mathcal{C}|=(\texttt{brute}(\beta))^{n}\cdot n^{\mathcal{O}\left(\frac{1}{\delta}\cdot\log\left(\frac{n}{\delta}\right)\right)}$$

by Claim 14. All in all, the running time of $\mathscr{A}$ is upper bounded by $(\texttt{brute}(\beta))^{n}\cdot n^{\mathcal{O}\left(\frac{1}{\delta}\cdot\log\left(\frac{n}{\delta}\right)\right)}$. ◁

The lemma follows from Claims 13–15. ◀

To prove Lemma 10, we combine Lemma 12 with the following technical lemma.

▶ **Lemma 16.** *Let $f\colon I\to\mathbb{R}$ be a continuous function on an open interval $I\subseteq\mathbb{R}$ and let $\alpha>1$ such that $\alpha\in I$. Define $\beta(n):=\alpha-\frac{1}{\log(n)}$ and $\delta(n):=\frac{\alpha}{\beta(n)}-1$ for all $n\in\mathbb{N}$. Then it holds that*

$$f\big(\beta(n)\big)^{n}\cdot n^{\mathcal{O}\left(\frac{1}{\delta(n)}\cdot\log\left(\frac{n}{\delta(n)}\right)\right)}=f(\alpha)^{n+o(n)}.$$

The proof of Lemma 16 is given in the full version of the paper.

Proof of Lemma 10. We claim that the algorithm $\mathscr{A}$ from Lemma 12 with $\beta:=\alpha-\frac{1}{\log(n)}$ and $\delta:=\frac{\alpha}{\beta}-1$ satisfies the properties listed in Lemma 10. Note that β and δ are functions of n, but we write β and δ instead of $\beta(n)$ and $\delta(n)$ for the sake of readability.

Observe that we have $(1+\delta)\cdot\beta=\frac{\alpha}{\beta}\cdot\beta=\alpha$. Hence, by Lemma 12, the set returned by $\mathscr{A}$ is an α-covering family $\mathcal{C}$ of U and $\mathbf{w}$ such that $|\mathcal{C}|\leq(\texttt{brute}(\beta))^{n}\cdot n^{\mathcal{O}\left(\frac{1}{\delta}\log\left(\frac{n}{\delta}\right)\right)}$. The running time of the algorithm is also bounded by $(\texttt{brute}(\beta))^{n}\cdot n^{\mathcal{O}\left(\frac{1}{\delta}\cdot\log\left(\frac{n}{\delta}\right)\right)}$.

By (1), $\texttt{brute}(x)$ is a continuous function of x because the entropy function is continuous. Therefore by Lemma 16 it holds that $(\texttt{brute}(\beta))^{n}\cdot n^{\mathcal{O}\left(\frac{1}{\delta}\log\left(\frac{n}{\delta}\right)\right)}=(\texttt{brute}(\alpha))^{n+o(n)}$ which proves the lemma. ◀

5 Weighted Monotone Local Search

In this section we prove Theorem 8. The proof presented here follows the outline of the proof of Theorem 3 in Section 4, using the concept of an (α,β)-*extension family*, an adaptation of the term α-covering family presented in Section 4 to the setting of extension oracles. While the items in a covering family are queries to a membership oracle, and hence are subsets of U, the items in an extension family represent queries to the extension oracle, and thus are pairs (T,ℓ) of a subset T of U and a non-negative integer ℓ. A reduction to a construction from [12] is used to build the extension family, and the extension family itself can be trivially used to attain a β-approximation algorithm for WeightedSM.

▶ **Definition 17** (Extension Family)**.** *Let U be a finite set and $\mathbf{w}\colon U\to\mathbb{N}$ be a weight function. Furthermore, let $\alpha,\beta\geq 1$. We say $\mathcal{E}\subseteq 2^{U}\times\mathbb{N}$ is an (α,β)-*extension family *of U and $\mathbf{w}$ if for every $S\subseteq U$ there exists $(T,\ell)\in\mathcal{E}$ which satisfies the following:*

$$|S \setminus T| \leq \ell,$$
$$\mathbf{w}(T) + \alpha \cdot \mathbf{w}(S \setminus T) \leq \beta \cdot \mathbf{w}(S). \tag{8}$$

Let $c \geq 1$. The *c-cost* of an (α, β)-extension family $\mathcal{E}$ of U and $\mathbf{w}$ is defined as $\texttt{cost}_c(\mathcal{E}) \coloneqq \sum_{(T,\ell)\in\mathcal{E}} c^\ell$. The proof of Theorem 8 relies on the following lemma.

▶ **Lemma 18.** *Let $\alpha, c \geq 1$, $\beta > 1$ and $\varepsilon > 0$. Then there is an algorithm which given a finite set U and a weight function $\mathbf{w}\colon U \to \mathbb{N}$ returns an (α, β)-extension family $\mathcal{E}$ of U and $\mathbf{w}$ such that $\texttt{cost}_c(\mathcal{E}) = \mathcal{O}\left(\left(\texttt{amls}(\alpha, c, \beta) + \varepsilon\right)^n\right)$. Furthermore, the running time of the algorithm is $\mathcal{O}\left(\left(\texttt{amls}(\alpha, c, \beta) + \varepsilon\right)^n\right)$.*

Before heading to the proof of Lemma 18, we show how the lemma can be used to prove Theorem 8.

Proof of Theorem 8. Let $\alpha, c \geq 1$, $\beta > 1$ and $\varepsilon > 0$. Consider the following algorithm $\mathscr{A}$:

- Given the input $(U, \mathbf{w}, \mathcal{F})$, use Lemma 18 with α, β, c and ε to construct an (α, β)-extension family $\mathcal{E}$ of U and $\mathbf{w}$.
- Let $\texttt{Ext}$ be the α-extension oracle. For each $(T_i, \ell_i) \in \mathcal{E}$, use $\texttt{Ext}$ to compute $X_i \coloneqq \texttt{Ext}(T_i, \ell_i)$.
- Define $\mathcal{T} \coloneqq \{T_i \cup X_i \mid (T_i, \ell_i) \in \mathcal{E}\}$ and return a set in $\mathcal{T}$ with the minimum weight, i.e., a set $T \in \mathcal{T}$ such that $\mathbf{w}(T) = \min\{\mathbf{w}(Y) \mid Y \in \mathcal{T}\}$.

The algorithm $\mathscr{A}$ first creates an (α, β)-extension family $\mathcal{E}$. Then it goes over all elements $(T_i, \ell_i) \in \mathcal{E}$ and queries the oracle with (T_i, ℓ_i). So the running time of this algorithm in the (α, c)-extension model is equal to the running time of the algorithm from Lemma 18 plus c^{ℓ_i} for every query (T_i, ℓ_i), i.e., the cost of the extension family $\mathcal{E}$. By Lemma 18 this value is at most

$$\begin{aligned}(\texttt{amls}(\alpha, c, \beta) + \varepsilon)^n + \texttt{cost}_c(\mathcal{E}) &= \mathcal{O}\Big((\texttt{amls}(\alpha, c, \beta) + \varepsilon)^n\Big) + \mathcal{O}\Big((\texttt{amls}(\alpha, c, \beta) + \varepsilon)^n\Big) \\ &= \mathcal{O}\Big((\texttt{amls}(\alpha, c, \beta) + \varepsilon)^n\Big)\end{aligned}$$

▷ Claim 19. The algorithm $\mathscr{A}$ is a deterministic β-approximation for WeightedSM in the (α, c)-extension model.

Proof. Let $R \in \mathcal{T}$ be the set returned by the algorithm. Note that by definition of $\mathcal{T}$, $R = T_j \cup X_j$ for some $(T_j, \ell_j) \in \mathcal{E}$ and $X_j = \texttt{Ext}(T_j, \ell_j)$. In particular, $R = T_j \cup X_j = T_j \cup \texttt{Ext}(T_j, \ell_j) \in \mathcal{F}$ (see Definition 4).

Let $S \in \mathcal{F}$ be a set with minimum weight in $\mathcal{F}$, i.e., $\mathbf{w}(S) = \min\{\mathbf{w}(Y) \mid Y \in \mathcal{F}\} = \texttt{opt}(U, \mathbf{w}, \mathcal{F})$. Since $\mathcal{E}$ is an (α, β)-extension family, there exists $(T_i, \ell_i) \in \mathcal{E}$ such that S, T_i and ℓ_i satisfy (8) for $T = T_i$ and $\ell = \ell_i$. Moreover, observe that

$$S \setminus T_i \in \{X \subseteq U \mid |X| \leq \ell_i, X \cup T_i \in \mathcal{F}\}$$

because $S \subseteq (S \setminus T_i) \cup T_i \in \mathcal{F}$. Hence, by the definition of $\texttt{Ext}$ and (T_i, ℓ_i), it follows that

$$\begin{aligned}\mathbf{w}(X_i) = \mathbf{w}\left(\texttt{Ext}(T_i, \ell_i)\right) &\leq \alpha \cdot \min\left\{\mathbf{w}(X) \mid X \subseteq U,\ |X| \leq \ell_i,\ X \cup T_i \in \mathcal{F}\right\} \\ &\leq \alpha \cdot \mathbf{w}\left(S \setminus T_i\right) \\ &\leq \beta \cdot \mathbf{w}(S) - \mathbf{w}(T_i).\end{aligned}$$

Finally, we have

$$\begin{aligned}\mathbf{w}(R) &= \min\{\mathbf{w}(Y) \mid Y \in \mathcal{T}\}\\ &\le \mathbf{w}(T_i \cup X_i)\\ &\le \mathbf{w}(T_i) + \mathbf{w}(X_i)\\ &\le \beta \cdot \mathbf{w}(S)\\ &= \beta \cdot \texttt{opt}(U, \mathbf{w}, \mathcal{F})\end{aligned}$$

which proves the claim. ◁

This shows that the algorithm $\mathscr{A}$ is a deterministic β-approximation for WeightedSM in the (α, c)-extension model with the running time given in Theorem 8. ◀

The proof of Lemma 18 relies on the following construction for extension families in the unweighted case which is implicitly given in [12].

▶ **Lemma 20** ([12]). *Let $\alpha, c \ge 1$ and $\beta > 1$. Then there is a deterministic algorithm which given a finite set U, returns an (α, β)-extension family $\mathcal{E}$ of U and the uniform weight function $\mathbf{w}\colon U \to \{1\}$ such that $\texttt{cost}_c(\mathcal{E}) \le \Big(\texttt{amls}(\alpha, c, \beta)\Big)^{n+o(n)}$. Furthermore, the running time of the algorithm is $\Big(\texttt{amls}(\alpha, c, \beta)\Big)^{n+o(n)}$.*

In a manner analogous to Section 4, we begin by introducing Lemma 21, which presents a slightly weaker form of Lemma 18. Within Lemma 21, ζ takes the place of the previously mentioned β from Lemma 18. Subsequently, in the proof of Lemma 18, we set the value of ζ as a function of β.

▶ **Lemma 21.** *Let $\alpha, c \ge 1$, $\zeta > 1$ and $0 < \delta < 1$. Then there is an algorithm which given a finite set U and a weight function $\mathbf{w}\colon U \to \mathbb{N}$ returns an $(\alpha, (1+\delta)\cdot\zeta)$-extension family $\mathcal{E}$ of U and $\mathbf{w}$ such that $\texttt{cost}_c(\mathcal{E}) \le \Big(\texttt{amls}(\alpha, c, \zeta)\Big)^{n+o(n)}$. Furthermore, the running time of the algorithm is $\Big(\texttt{amls}(\alpha, c, \zeta)\Big)^{n+o(n)}$.*

As already indicated above, the proof of Lemma 21 is similar to the proof of Lemma 12. The algorithm used to compute the desired extension family is given in Algorithm 1. The full proof of Lemma 21 can be found in Appendix A. Lastly, we use the following property of the function amls.

▶ **Lemma 22.** *For every fixed $\alpha > 1$ and $c > 1$, $\texttt{amls}(\alpha, c, x)$ is a continuous function of x on the interval $(1, \infty)$.*

The proof of Lemma 22 is given in the full version of the paper.

Proof of Lemma 18. Let $\alpha, c \ge 1$, $\beta > 1$ and $\varepsilon > 0$. Since $\texttt{amls}(\alpha, c, \zeta)$ is continuous in ζ by Lemma 22, there exists a $\zeta' \in (1, \beta)$ such that $\texttt{amls}(\alpha, c, \zeta') < \texttt{amls}(\alpha, c, \beta) + \frac{\varepsilon}{2}$. To prove the lemma, we use the algorithm from Lemma 21 (i.e., Algorithm 1) with $\zeta := \zeta'$ and $\delta := \beta/\zeta - 1$.

Note that we have $(1+\delta)\cdot\zeta = (\beta/\zeta)\cdot\zeta = \beta$. Hence, by Lemma 21, the set $\mathcal{E}$ returned by Algorithm 1 is an $\big(\alpha, \beta\big)$-extension family of U and $\mathbf{w}$ such that

$$\texttt{cost}_c(\mathcal{E}) \le \Big(\texttt{amls}(\alpha, c, \zeta')\Big)^{n+o(n)} \le \Big(\texttt{amls}(\alpha, c, \beta) + \frac{\varepsilon}{2}\Big)^{n+o(n)} = \mathcal{O}\Big(\big(\texttt{amls}(\alpha, c, \beta) + \varepsilon\big)^{n}\Big)$$

Algorithm 1 Extension Family for Arbitrary Weight Functions.

Configuration: $\alpha \geq 1$, $c \geq 1$, $\zeta > 1$ and $0 < \delta < 1$
Input: A universe U, weight function $\mathbf{w} \colon U \to \mathbb{N}$

1: Define $\gamma \coloneqq 1 + \frac{\delta}{2} > 1$. For $i \geq 0$ let

$$U_i \coloneqq \{u \in U \mid \gamma^i \leq \mathbf{w}(u) < \gamma^{i+1}\} \tag{9}$$

and $n_i \coloneqq |U_i|$. Let $I \coloneqq \{i \in \mathbb{Z}_{\geq 0} \mid U_i \neq \emptyset\}$ denote the set of indices $i \geq 0$ for which U_i is non-empty. Note that $|I| \leq n$.

2: For each $i \in I$ construct an $\left(\alpha, \zeta\right)$-extension family $\mathcal{E}_i$ of the universe U_i and the uniform weight function using Lemma 20 with respect to α, c and ζ.

3: Define $d \coloneqq \lceil (2/\delta) \cdot \log(2n/\delta) \rceil$ and for each $k \in I$, let $I_k \coloneqq \{i \in I \mid k - d \leq i \leq k\}$ denote the indices in I between $k - d$ and k.

4: For every $k \in I$, let $r_k \coloneqq |I_k|$ and define

$$W_k \coloneqq \bigcup_{i \in I \colon 1 \leq i < k-d} U_i \quad \text{and}$$

$$\mathcal{Q}_k \coloneqq \left\{ \left(W_k \cup E_1 \cup \ldots \cup E_{r_k}, \ell_1 + \ldots + \ell_{r_k}\right) \;\middle|\; \left((E_1, \ell_1), \ldots, (E_{r_k}, \ell_{r_k})\right) \in \prod_{i \in I_k} \mathcal{E}_i \right\}.$$

5: Return the set $\mathcal{E} \coloneqq \bigcup_{k \in I} \mathcal{Q}_k$.

Finally, the running time of the algorithm is also bounded by

$$\left(\mathtt{amls}(\alpha, c, \zeta')\right)^{n+o(n)} = \mathcal{O}\left(\left(\mathtt{amls}(\alpha, c, \beta) + \varepsilon\right)^n\right)$$

which proves the lemma. ◀

6 Discussion

In this paper, we study weighted monotone subset minimization problems, where given a universe U with n elements and a weight function $\mathbf{w} \colon U \to \mathbb{N}$, the goal is to find a subset $S \subseteq U$ which satisfies a certain fixed property and has a minimum weight. For such problems, we show that the Approximate Monotone Local Search framework of Esmer et al. [12] can be extended to the weighted setting. In particular, given a parameterized α-approximate extension algorithm, that runs in time $c^k \cdot n^{\mathcal{O}(1)}$ and outputs a solution whose weight is at most $\beta \cdot \mathbf{w}(\mathtt{OPT})$ where $\mathtt{OPT}$ is a solution of size at most k and minimum weight, one can design an exponential β-approximation algorithm that runs faster than the proposed (natural) brute-force algorithm.

Note that for most of our applications, the parameterized approximation algorithms that are available in the literature [1,14,18,21] provide bi-objective guarantees which are stronger than the requirements from the α-approximate extension algorithm. In particular, these algorithms run in time $c^k \cdot n^{\mathcal{O}(1)}$ and output a solution of size at most $\gamma \cdot k$ and weight at most $\beta \cdot W$, if a solution of size at most k and weight at most W exists. That is, they (approximately) optimize the size and weight of the output solution *simultaneously*.

This leads to the following natural question. Consider more restrictive weighted monotone subset minimization problems where given a universe U on n vertices, a weight function $\mathbf{w}$ on the elements of the universe, the goal is to find a subset of the universe of size at most k

that minimizes the weight and satisfies a certain fixed property. What is the analogue of brute-force in this setting? Can bi-objective parameterized approximation algorithms for weighted monotone subset minimization problems be used to design faster (than brute force) exponential approximation algorithms in this restrictive setting? What happens if we extend this setting to a bi-criteria optimization setting?

References

1 Akanksha Agrawal, Sudeshna Kolay, Daniel Lokshtanov, and Saket Saurabh. A faster FPT algorithm and a smaller kernel for block graph vertex deletion. In Evangelos Kranakis, Gonzalo Navarro, and Edgar Chávez, editors, *LATIN 2016: Theoretical Informatics - 12th Latin American Symposium, Ensenada, Mexico, April 11-15, 2016, Proceedings*, volume 9644 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2016. doi:10.1007/978-3-662-49529-2_1.

2 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.

3 Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM J. Discret. Math.*, 12(3):289–297, 1999. doi:10.1137/S0895480196305124.

4 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.

5 Reuven Bar-Yehuda and Shimon Even. A linear-time approximation algorithm for the weighted vertex cover problem. *J. Algorithms*, 2(2):198–203, 1981. doi:10.1016/0196-6774(81)90020-1.

6 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.

7 Nader H. Bshouty and Lynn Burroughs. Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem. In Michel Morvan, Christoph Meinel, and Daniel Krob, editors, *STACS 98, 15th Annual Symposium on Theoretical Aspects of Computer Science, Paris, France, February 25-27, 1998, Proceedings*, volume 1373 of *Lecture Notes in Computer Science*, pages 298–308. Springer, 1998. doi:10.1007/BFb0028569.

8 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

9 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.

10 Bruno Escoffier, Vangelis Th. Paschos, and Emeric Tourniaire. Super-polynomial approximation branching algorithms. *RAIRO Oper. Res.*, 50(4-5):979–994, 2016. doi:10.1051/ro/2015060.

11 Barış Can Esmer, Ariel Kulik, Dániel Marx, Daniel Neuen, and Roohani Sharma. Faster exponential-time approximation algorithms using approximate monotone local search. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 50:1–50:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.50.

12 Barış Can Esmer, Ariel Kulik, Dániel Marx, Daniel Neuen, and Roohani Sharma. Optimally repurposing existing algorithms to obtain exponential-time approximations. *CoRR*, abs/2306.15331, 2023. To be published at SODA 2024. arXiv:2306.15331, doi:10.48550/arXiv.2306.15331.

13 Guy Even, Joseph Naor, and Leonid Zosin. An 8-approximation algorithm for the subset feedback vertex set problem. *SIAM J. Comput.*, 30(4):1231–1252, 2000. doi:10.1137/S0097539798340047.

14 Fedor V. Fomin, Serge Gaspers, Dieter Kratsch, Mathieu Liedloff, and Saket Saurabh. Iterative compression and exact algorithms. *Theor. Comput. Sci.*, 411(7-9):1045–1053, 2010. doi:10.1016/j.tcs.2009.11.012.

15 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.

16 Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010. doi:10.1007/978-3-642-16533-7.

17 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.

18 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.

19 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.

20 Rolf Niedermeier and Peter Rossmanith. On efficient fixed-parameter algorithms for weighted vertex cover. *J. Algorithms*, 47(2):63–77, 2003. doi:10.1016/S0196-6774(03)00005-1.

21 Hadas Shachnai and Meirav Zehavi. A multivariate framework for weighted FPT algorithms. *J. Comput. Syst. Sci.*, 89:157–189, 2017. doi:10.1016/j.jcss.2017.05.003.

22 Magnus Wahlström. A tighter bound for counting max-weight solutions to 2sat instances. In Martin Grohe and Rolf Niedermeier, editors, *Parameterized and Exact Computation, Third International Workshop, IWPEC 2008, Victoria, Canada, May 14-16, 2008. Proceedings*, volume 5018 of *Lecture Notes in Computer Science*, pages 202–213. Springer, 2008. doi:10.1007/978-3-540-79723-4_19.

A Missing Proofs from Section 5

The proof of Lemma 21 contains repeated arguments from the proof of Lemma 12. To enhance the overall readability and reduce the notational burden, we keep the common arguments in both proofs. Finally, we need the following technical lemma whose proof is given in the full version of the paper.

▶ **Lemma 23.** *Let $g, d\colon \mathbb{N} \to \mathbb{N}$ be two functions such that $g \in n + o(n)$ and $d \in o(n)$. We define $f\colon \mathbb{N} \to \mathbb{N}$ via*

$$f(n) = \max_{n=n_1+\cdots+n_{d(n)}} \sum_{i=1}^{d(n)} g(n_i).$$

Then $f \in n + o(n)$.

Proof of Lemma 21. We claim that Algorithm 1 satisfies the conditions stated in the lemma.

▷ Claim 24. The set $\mathcal{E}$ returned by Algorithm 1 is an $\big(\alpha, (1+\delta)\cdot\zeta\big)$-extension family of U and $\mathbf{w}$.

Proof. Let $S \subseteq U$ be a set. We argue that there exists $(T, \ell) \in \mathcal{E}$ such that S, T and ℓ satisfy (8). By the definition of γ and d, it holds that

$$\gamma^d \geq \left(1 + \frac{\delta}{2}\right)^{(2/\delta)\cdot\log(2n/\delta)} \geq 2^{\log(2n/\delta)} = \frac{2n}{\delta} \tag{10}$$

using that $(1 + \frac{1}{x})^x \geq 2$ for all $x \geq 1$. Let $k \in I$ be the largest index such that $S \cap U_k \neq \emptyset$. It holds that

$$\mathbf{w}(W_k) < n \cdot \gamma^{k-d} \leq \frac{\delta}{2} \cdot \gamma^k \leq \frac{\delta}{2} \cdot \mathbf{w}(S) \tag{11}$$

where the first inequality follows from the fact that $|W_k| \leq n$ and each $u \in W_k$ belongs to a set U_i where $i \leq k - d - 1$ and therefore $\mathbf{w}(u) < \gamma^{k-d-1+1} = \gamma^{k-d}$ by (9). The second inequality follows from (10) and finally the last inequality holds because by the definition of k, there exists $u \in S \cap U_k$ such that $\mathbf{w}(S) \geq \mathbf{w}(u) \geq \gamma^k$ by (9).

For $i \in I_k$ let $S_i := S \cap U_i$. Since $\mathcal{E}_i$ is an (α, ζ)-extension family of U_i and the *uniform weight function*, for each $i \in I_k$ there exists $(T_i, \ell_i) \in \mathcal{E}_i$ such that

$$\begin{aligned} |S_i \setminus T_i| &\leq \ell_i \\ |T_i| + \alpha \cdot |S_i \setminus T_i| &\leq \zeta \cdot |S_i|. \end{aligned} \tag{12}$$

Let $\overline{T} := \bigcup_{i \in I_k} T_i$ and define

$$\begin{aligned} T &:= W_k \cup \overline{T} \\ \ell &:= \sum_{i \in I_k} \ell_i. \end{aligned}$$

By definition of $\mathcal{Q}_k$ in Algorithm 1 it holds that $(T, \ell) \in \mathcal{Q}_k \subseteq \mathcal{E}$. Observe that

$$\begin{aligned} S \setminus T &= (S \setminus T) \cap U \\ &= (S \setminus T) \cap \left(W_k \cup \left(\bigcup_{i \in I_k} U_i\right)\right) \\ &= \Big((S \setminus T) \cap W_k\Big) \cup \left((S \setminus T) \cap \left(\bigcup_{i \in I_k} U_i\right)\right) \\ &= (S \setminus T) \cap \left(\bigcup_{i \in I_k} U_i\right) \\ &= \bigcup_{i \in I_k} (U_i \cap (S \setminus T)) \\ &= \bigcup_{i \in I_k} S_i \setminus T_i \end{aligned} \tag{13}$$

where the second equality follows from $S \subseteq \bigcup_{1 \leq i \leq k} U_i = W_k \cup \left(\bigcup_{k-d \leq i \leq k} U_i\right)$ and the fourth inequality holds because $W_k \subseteq T$ which further implies $(S \setminus T) \cap W_k = \emptyset$. Therefore it holds that

$$|S \setminus T| = \left|\bigcup_{i \in I_k} S_i \setminus T_i\right| \leq \sum_{i \in I_k} |S_i \setminus T_i| \leq \sum_{i \in I_k} \ell_i = \ell.$$

where the second inequality follows from (12).

For all $i \in I_k$ we also have that

$$\begin{aligned}\mathbf{w}(T_i) + \alpha \cdot \mathbf{w}(S_i \setminus T_i) &\leq |T_i| \cdot \gamma^{i+1} + \alpha \cdot \gamma^{i+1} \cdot |S_i \setminus T_i| \\ &\leq \gamma^{i+1} \cdot (|T_i| + \alpha \cdot |S_i \setminus T_i|) \\ &\leq \gamma^{i+1} \cdot \zeta \cdot |S_i| \\ &\leq \zeta \cdot \gamma \cdot \mathbf{w}(S_i) \end{aligned} \tag{14}$$

where the third inequality follows from (12). Therefore we have that

$$\begin{aligned}\mathbf{w}(T) + \alpha \cdot \mathbf{w}(S \setminus T) &= \mathbf{w}(W_k) + \mathbf{w}\left(\bigcup_{i \in I_k} T_i\right) + \alpha \cdot \mathbf{w}\left(S \setminus T\right) \\ &\leq \mathbf{w}(W_k) + \sum_{i \in I_k}\Bigg(\mathbf{w}(T_i) + \alpha \cdot \mathbf{w}(S_i \setminus T_i)\Bigg) && \text{by (13)} \\ &< \frac{\delta}{2} \cdot \mathbf{w}(S) + \sum_{i \in I_k}\Bigg(\mathbf{w}(T_i) + \alpha \cdot \mathbf{w}(S_i \setminus T_i)\Bigg) && \text{by (11)} \\ &\leq \frac{\delta}{2} \cdot \mathbf{w}(S) + \sum_{i \in I_k} \zeta \cdot \gamma \cdot \mathbf{w}(S_i) && \text{by (14)} \\ &< \zeta \cdot \frac{\delta}{2} \cdot \mathbf{w}(S) + \sum_{i \in I_k} \zeta \cdot \gamma \cdot \mathbf{w}(S_i) && \text{since } \zeta > 1 \\ &\leq \zeta \cdot \frac{\delta}{2} \cdot \mathbf{w}(S) + \zeta \cdot \gamma \cdot \mathbf{w}(S) \\ &\leq (1+\delta) \cdot \zeta \cdot \mathbf{w}(S)\end{aligned}$$

which proves the claim. ◁

▷ Claim 25.

$$\mathtt{cost}_c(\mathcal{E}) \leq \Big(\mathtt{amls}(\alpha, c, \zeta)\Big)^{n+o(n)}.$$

Proof. By Lemma 20, for each $i \in I$ it holds that

$$\mathtt{cost}_c(\mathcal{E}_i) \leq \Big(\mathtt{amls}(\alpha, c, \zeta)\Big)^{n_i+o(n_i)}. \tag{15}$$

For $i \in \mathbb{Z}_{\geq 0} \setminus I$ we define $\mathcal{E}_i = \{(\emptyset, 0)\}$. By doing so, we can make the assumption, without loss of generality, that $\mathcal{E}_i$ is nonempty for all $i \in \mathbb{Z}_{\geq 0}$. Note that this assumption does not have any effect on the value of $\mathtt{cost}_c$ and it simplifies the following analysis. Also let $\mathbb{E}$ denote the Cartesian product of $\mathcal{E}_{k-d}$ up to $\mathcal{E}_k$, i.e., $\mathbb{E} \coloneqq \prod_{i \in \{k-d,\dots,k\}} \mathcal{E}_i$.

With this assumption, for all $k \in I$, we have

$$\begin{aligned}\mathtt{cost}_c(\mathcal{Q}_k) &= \sum_{\big((E_{k-d},\ell_{k-d}),\dots,(E_{k-1},\ell_{k-1}),(E_k,\ell_k)\big)\in\mathbb{E}} c^{\ell_{k-d}+\dots+\ell_{k-1}+\ell_k} \\ &= \sum_{\big((E_{k-d},\ell_{k-d}),\dots,(E_{k-1},\ell_{k-1}),(E_k,\ell_k)\big)\in\mathbb{E}} c^{\ell_{k-d}} \cdot \ldots \cdot c^{\ell_{k-1}} \cdot c^{\ell_k} \\ &= \prod_{j=k-d}^{k} \sum_{(E,\ell)\in\mathcal{E}_j} c^{\ell}\end{aligned}$$

$$= \prod_{j=k-d}^{k} \mathtt{cost}_c(\mathcal{E}_j)$$
$$= \prod_{j \in I_k} \mathtt{cost}_c(\mathcal{E}_j)$$

By (15) we obtain

$$\begin{aligned} \mathtt{cost}_c(\mathcal{Q}_k) &= \prod_{i \in I_k} \mathtt{cost}_c(\mathcal{E}_i) \\ &\leq \prod_{i \in I_k} \Big(\mathtt{amls}(\alpha, c, \zeta)\Big)^{n_i + o(n_i)} \\ &= \Big(\mathtt{amls}(\alpha, c, \zeta)\Big)^{n+o(n)} \end{aligned}$$

where the last step follows from Lemma 23.

Finally, it holds that

$$\mathtt{cost}_c(\mathcal{E}) = \mathtt{cost}_c\left(\bigcup_{k \in I} \mathcal{Q}_k\right) \leq \sum_{k \in I} \mathtt{cost}_c(\mathcal{Q}_k) = \Big(\mathtt{amls}(\alpha, c, \zeta)\Big)^{n+o(n)}$$

since $|I| \leq n$. ◁

▷ Claim 26. The running time of Algorithm 1 is $\Big(\mathtt{amls}(\alpha, c, \zeta)\Big)^{n+o(n)}$.

Proof. The construction of $\{\mathcal{E}_i\}_{i \in I}$ takes time

$$\sum_{i \in I} \Big(\mathtt{amls}(\alpha, c, \zeta)\Big)^{n_i + o(n_i)} \leq \Big(\mathtt{amls}(\alpha, c, \zeta)\Big)^{n+o(n)}$$

by Lemma 20.

Finally, the construction of each $\mathcal{Q}_k$ takes time proportional to its size, which is upper bounded by $\mathtt{cost}_c(\mathcal{Q}_k)$. Then, the construction of $\mathcal{E}$ takes at most time $\mathcal{O}(\mathtt{cost}_c(\mathcal{E}))$, where we have

$$\mathtt{cost}_c(\mathcal{E}) \leq \Big(\mathtt{amls}(\alpha, c, \zeta)\Big)^{n+o(n)}$$

by Claim 25. All in all, the running time of Algorithm 1 is upper bounded by

$$\Big(\mathtt{amls}(\alpha, c, \zeta)\Big)^{n+o(n)}.$$ ◁

The lemma follows from Claims 24–26. ◀

B Problem Definitions

In this section, we give the problem definitions of all the problems discussed in the paper. For simplicity, we define the problems in their unweighted version. In the weighted version, the vertices are equipped with weights and we are looking for a solution S of minimum weight.

VERTEX COVER (VC)
Input: An undirected graph G.
Question: Find a minimum set S of vertices of G such that $G - S$ has no edges.

Partial Vertex Cover (PVC)
Input: An undirected graph G and an integer $t \geq 0$.
Question: Find a minimum set S of vertices of G such that $G - S$ has at most $|E(G)| - t$ many edges.

d-Hitting Set (d-HS)
Input: A universe U and set family $\mathcal{F} \subseteq \binom{U}{\leq d}$.
Question: Find a minimum set $S \subseteq U$ such that for each $F \in \mathcal{F}$, $S \cap F \neq \emptyset$.

Feedback Vertex Set (FVS)
Input: An undirected graph G.
Question: Find a minimum set S of vertices of G such that $G - S$ is an acyclic graph.

Subset Feedback Vertex Set (Subset FVS)
Input: An undirected graph G and a set $T \subseteq V(G)$.
Question: Find a minimum set S of vertices of G such that $G - S$ has no cycle that contains at least one vertex of T.

Tournament Feedback Vertex Set (TFVS)
Input: A tournament graph G.
Question: Find a minimum set S of vertices of G such that $G - S$ is an acyclic tournament.

Directed Feedback Vertex Set (DFVS)
Input: A directed graph G.
Question: Find a minimum set S of vertices of G such that $G - S$ is a directed acyclic graph.

Directed Subset Feedback Vertex Set (Subset DFVS)
Input: A directed graph G and a set $T \subseteq V(G)$.
Question: Find a minimum set S of vertices of G such that $G - S$ has no directed cycle that contains at least one vertex of T.

Directed Odd Cycle Transversal (DOCT)
Input: A directed graph G.
Question: Find a minimum set S of vertices of G such that $G - S$ has no directed cycle of odd length.

Multicut
Input: An undirected graph G and a set $\mathcal{P} \subseteq V(G) \times V(G)$.
Question: Find a minimum set S of vertices of G such that $G - S$ has no path from u to v for any $(u, v) \in \mathcal{P}$

For the next problems, we require some additional definitions. A graph G is *cluster graph* if every connected component of G is a complete graph. A *cograph* is a graph G which does not contain P_4 (a path on 4 vertices) is an induced subgraph. Finally, a graph G is a *split graph* if the vertex set can be partitioned into two sets $V(G) = I \uplus C$ such that I is an independent set and C is a clique in G.

Cluster Graph Vertex Deletion
Input: An undirected graph G.
Question: Find a minimum set S of vertices of G such that $G - S$ is a cluster graph.

Cograph Vertex Deletion
Input: An undirected graph G.
Question: Find a minimum set S of vertices of G such that $G - S$ is a cograph.

Split Vertex Deletion
Input: An undirected graph G.
Question: Find a minimum set S of vertices of G such that $G - S$ is a split graph.