# Exact Algorithms via Monotone Local Search

FEDOR V. FOMIN, University of Bergen, Norway
SERGE GASPERS, UNSW Sydney & Data61, CSIRO, Australia
DANIEL LOKSHTANOV, University of Bergen, Norway
SAKET SAURABH, University of Bergen, Norway & Institute of Mathematical Sciences, Chennai, India

We give a new general approach for designing exact exponential-time algorithms for *subset problems*. In a subset problem the input implicitly describes a family of sets over a universe of size $n$ and the task is to determine whether the family contains at least one set. A typical example of a subset problem is Weighted $d$-SAT. Here, the input is a CNF-formula with clauses of size at most $d$, and an integer $W$. The universe is the set of variables and the variables have integer weights. The family contains all the subsets $S$ of variables such that the total weight of the variables in $S$ does not exceed $W$ and setting the variables in $S$ to 1 and the remaining variables to 0 satisfies the formula. Our approach is based on "monotone local search," where the goal is to extend a partial solution to a solution by adding as few elements as possible. More formally, in the extension problem, we are also given as input a subset $X$ of the universe and an integer $k$. The task is to determine whether one can add at most $k$ elements to $X$ to obtain a set in the (implicitly defined) family. Our main result is that a $c^k n^{O(1)}$ time algorithm for the extension problem immediately yields a randomized algorithm for finding a solution of any size with running time $O((2 - \frac{1}{c})^n)$.

In many cases, the extension problem can be reduced to simply finding a solution of size at most $k$. Furthermore, efficient algorithms for finding small solutions have been extensively studied in the field of parameterized algorithms. Directly applying these algorithms, our theorem yields in one stroke significant improvements over the best known exponential-time algorithms for several well-studied problems, including $d$-Hitting Set, Feedback Vertex Set, Node Unique Label Cover, and Weighted $d$-SAT. Our results demonstrate an interesting and very concrete connection between parameterized algorithms and exact exponential-time algorithms.

We also show how to derandomize our algorithms at the cost of a subexponential multiplicative factor in the running time. Our derandomization is based on an efficient construction of a new pseudo-random object that might be of independent interest. Finally, we extend our methods to establish new combinatorial upper bounds and develop enumeration algorithms.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

A preliminary version of this work appeared in proceedings of STOC 2016 (Fomin et al. 2016).
The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreements No. 267959/No. 306992 and Research Council of Norway via MULTIVAL. Serge Gaspers is the recipient of an Australian Research Council (ARC) Future Fellowship (Project No. FT140100048) and acknowledges support under the ARC's Discovery Projects funding scheme (Project No. DP150101134). Daniel Lokshtanov is supported by the Beating Hardness by Pre-processing grant under the recruitment programme of the of Bergen Research Foundation.
Authors' addresses: F. V. Fomin and D. Lokshtanov, Department of Informatics, University of Bergen, N-5020 Bergen, Norway; emails: {fomin, daniello}@ii.uib.no; S. Gaspers, UNSW Australia and Data61, CSIRO, Australia; email: sergeg@cse.unsw.edu.au; S. Saurabh, The Institute of Mathematical Sciences, Chennai-600113, India; email: saket@imsc.res.in.

0004-5411/2019/03-ART8 $15.00
https://doi.org/10.1145/3284176

General Terms: Algorithms, Design, Theory

Additional Key Words and Phrases: Exact exponential algorithm, parameterized algorithm, local search, satisfiability

**ACM Reference format:**
Fedor V. Fomin, Serge Gaspers, Daniel Lokshtanov, and Saket Saurabh. 2019. Exact Algorithms via Monotone Local Search. *J. ACM* 66, 2, Article 8 (March 2019), 23 pages.
https://doi.org/10.1145/3284176

## 1 INTRODUCTION

In the area of exact exponential-time algorithms, the objective is to design algorithms that outperform brute-force for computationally intractable problems. Because the problems are intractable, we do not hope for polynomial time algorithms. Instead, the aim is to allow super-polynomial time and design algorithms that are significantly faster than brute-force. For *subset problems* in NP, where the goal is to find a subset with some specific properties in a universe of $n$ elements, the brute-force algorithm that tries all possible solutions has running time $2^n n^{O(1)}$. Thus, our goal is typically to design an algorithm with running time $c^n n^{O(1)}$ for $c < 2$, and we try to minimize the constant $c$. We refer to the textbook of Fomin and Kratsch (2010) for an introduction to the field.

In the area of parameterized algorithms (see Cygan et al. (2015)), the goal is to design efficient algorithms for the "easy" instances of computationally intractable problems. Here, the running time is measured not only in terms of the input size $n$ but also in terms of a parameter $k$, which is expected to be small for "easy" instances. For subset problems, the parameter $k$ is often chosen to be the size of the solution sought for, and many subset problems have parameterized algorithms that find a solution of size $k$ (if there is one) in time $c^k n^{O(1)}$ for a constant $c$, which is often much larger than 2.

In this article, we address the following question: Can an *efficient* algorithm for the *easy* instances of a hard problem yield a *non-trivial* algorithm for *all* instances of that problem? More concretely, can parameterized algorithms for a problem be used to speed up exact exponential-time algorithms for the same problem? Our main result is an affirmative answer to this question: We show that, for a large class of problems, an algorithm with running time $c^k n^{O(1)}$ for any $c > 1$ immediately implies an exact algorithm with running time $O((2 - \frac{1}{c})^{n+o(n)})$ for the problem. Our main result, coupled with the fastest known parameterized algorithms, gives in one stroke the first non-trivial exact algorithm for a number of problems and simultaneously significantly improves over the best-known exact algorithms for several well-studied problems; see Table 1 for a non-exhaustive list of corollaries. Our approach is also useful to prove upper bounds on the number of interesting combinatorial objects and to design efficient algorithms that enumerate these objects; see Table 2.

At this point, it is worth noting that a simple connection between algorithms running in time $c^k n^{O(1)}$ for $c < 4$ and exact exponential-time algorithms beating $O(2^n)$ has been known for a long time. For subset problems, where we are looking for a specific subset of size $k$ in a universe of size $n$, to beat $O(2^n)$, one only needs to outperform brute-force for values of $k$ that are very close to $n/2$. Indeed, for $k$ sufficiently far away from $n/2$, trying all subsets of size $k$ takes time $\binom{n}{k} n^{O(1)}$, which is significantly faster than $O(2^n)$. Thus, if there is an algorithm deciding whether there is a solution of size at most $k$ in time $c^k n^{O(1)}$ for some $c < 4$, we can deduce that the problem can be solved in time $O((2 - \epsilon)^n)$ for an $\epsilon > 0$ that depends only on $c$. However, it is easy to see that this trade-off between $c^k$ and $\binom{n}{k}$ does not yield any improvement over $2^n$ when $c \geq 4$. Our main result significantly outperforms the algorithms obtained by this trade-off for every value of $c > 1$ and further yields better than $O(2^n)$ time algorithms even for $c \geq 4$.

As a concrete example, consider the Interval Vertex Deletion problem. Here the input is a graph $G$ and an integer $k$ and the task is to determine whether $G$ can be turned into an interval graph by deleting $k$ vertices. The fastest parameterized algorithm for the problem is due to Cao (2016) and runs in time $8^k n^{O(1)}$. Combining this algorithm with the simple trade-off scheme described above does not outperform brute-force, since $8 \geq 4$. The fastest previously known exponential-time algorithm for the problem is due to Bliznets et al. (2013) and runs in time $\mathcal{O}((2-\epsilon)^n)$ for $\epsilon < 10^{-20}$. However, combining the parameterized algorithm, as a black box, with our main result immediately yields a $1.875^{n+o(n)}$ time algorithm for Interval Vertex Deletion. While we focus mostly on decision problems, by a standard self-reduction, one can also find a smallest vertex set whose deletion makes the input graph an interval graph in time $1.875^{n+o(n)}$.

**Our Results.** We need some definitions to state our results precisely. We define an *implicit set system* as a function $\Phi$ that takes as input a string $I \in \{0,1\}^*$ and outputs a set system $(U_I, \mathcal{F}_I)$, where $U_I$ is a universe and $\mathcal{F}_I$ is a collection of subsets of $U_I$. The string $I$ is referred to as an *instance*, and we denote by $|U_I| = n$ the size of the universe and by $|I| = N$ the size of the instance. We assume that $N \geq n$. The implicit set system $\Phi$ is said to be *polynomial time computable* if (a) there exists a polynomial time algorithm that given $I$ produces $U_I$, and (b) there exists a polynomial time algorithm that given $I$, $U_I$ and a subset $S$ of $U_I$ determines whether $S \in \mathcal{F}_I$. All implicit set systems discussed in this article are polynomial time computable, except for the minimal satisfying assignments of $d$-CNF formulas, which are not polynomial time computable unless P=NP (Yato and Seta 2003).

An implicit set system $\Phi$ naturally leads to some problems about the set system $(U_I, \mathcal{F}_I)$. Find a set in $\mathcal{F}_I$. Is $\mathcal{F}_I$ non-empty? What is the cardinality of $\mathcal{F}_I$? In this article, we will primarily focus on the first and last problems. Examples of implicit set systems include the set of all feedback vertex sets of a graph of size at most $k$, the set of all satisfying assignments of a CNF formula of weight at most $W$, and the set of all minimal hitting sets of a set system. Next, we formally define subset problems.

$\Phi$-Subset
**Input:** An instance $I$
**Output:** A set $S \in \mathcal{F}_I$ if one exists.

An example of a subset problem is Min-Ones $d$-Sat. Here for an integer $k$ and a propositional formula $F$ in conjunctive normal form (CNF) where each clause has at most $d$ literals, the task is to determine whether $F$ has a satisfying assignment with Hamming weight at most $k$, i.e., setting at most $k$ variables to 1. In our setting, the instance $I$ consists of the input formula $F$ and the integer $k$, encoded as a string over 0s and 1s. The implicit set system $\Phi$ is a function from $I$ to $(U_I, \mathcal{F}_I)$, where $U_I$ is the set of variables of $F$, and $\mathcal{F}_I$ is the set of all satisfying assignments of Hamming weight at most $k$.

$\Phi$-Extension
**Input:** An instance $I$, a set $X \subseteq U_I$, and an integer $k$.
**Question:** Does there exists a subset $S \subseteq (U_I \setminus X)$ such that $S \cup X \in \mathcal{F}_I$ and $|S| \leq k$?

Our first main result gives exponential-time randomized algorithms for $\Phi$-Subset based on single-exponential, *deterministic as well as randomized*, parameterized algorithms for $\Phi$-Extension with parameter $k$. Our randomized algorithms are Monte Carlo algorithms with one-sided error. On no-instances they always return no, and on yes-instances they return yes (or output a certificate) with probability $> \frac{1}{2}$. Our algorithms are deterministic algorithms unless stated otherwise.

Theorem 1.1. *If there exists an algorithm for* $\Phi$-Extension *with running time* $c^k N^{O(1)}$*, then there exists a randomized algorithm for* $\Phi$-Subset *with running time* $(2 - \frac{1}{c})^n N^{O(1)}$.

Our second main result is that the algorithm of Theorem 1.1 can be derandomized at the cost of a subexponential factor in $n$ in the running time.

Theorem 1.2. *If there exists an algorithm for* $\Phi$-Extension *with running time* $c^k N^{O(1)}$*, then there exists an algorithm for* $\Phi$-Subset *with running time* $(2 - \frac{1}{c})^{n+o(n)} N^{O(1)}$.

To exemplify the power of these theorems, we give a few examples of applications. We have already seen the first example, the $1.875^{n+o(n)}$ time algorithm for Interval Vertex Deletion. Let us now consider the Min-Ones $d$-SAT problem described above.

A simple branching algorithm solves the extension problem for Min-Ones $d$-SAT as follows. Suppose we have already selected a set $X$ of variables to set to 1, remove all clauses containing a positive literal on $X$, and remove negative literals on $X$ from the remaining clauses. Start from the all-0 assignment on the remaining variables, with a budget for flipping $k$ variables from 0 to 1. As long as there is an unsatisfied clause, guess which one of the at most $d$ variables in this clause should be flipped from 0 to 1, and for each proceed recursively with the budget decreased by one. The recursion tree of this algorithm has depth at most $k$, and each node of the recursion tree has at most $d$ children, thus this algorithm terminates in time $d^k \cdot n^{O(1)}$.

Hence, by Theorem 1.2, Min-Ones $d$-SAT can be solved in time $(2 - \frac{1}{d})^{n+o(n)}$. For $d = 3$ there is a faster parameterized algorithm with running time $2.562^k n^{O(1)}$ due to Kutzkov and Scheder (2010). Thus, Min-Ones 3-SAT can be solved in time $O(1.6097^n)$. Since $d$-Hitting Set is a special case of Min-Ones $d$-SAT, the same bounds hold for this problem as well, and the same approach works for weighted variants of these problems. However, due to faster known parameterized algorithms for $d$-Hitting Set (Fomin et al. 2010), our theorem implies faster exact algorithms for $d$-Hitting Set with running time $(2 - \frac{1}{(d-0.9255)})^n$. That is, for $d$-Hitting Set, $d \geq 3$, there are parameterized algorithms running in time $(d - 0.9245)^k$ (Fomin et al. 2010), and thus combining this with our theorem implies exact algorithms for $d$-Hitting Set with running time $(2 - \frac{1}{(d-0.9255)})^n$.

Another interesting example is the Feedback Vertex Set problem. Here, the task is to decide, for a graph $G$ and an integer $k$, whether $G$ can be made acyclic by removing $k$ vertices. While this problem is trivially solvable in time $2^n n^{O(1)}$ for $n$-vertex graphs, breaking the $2^n$-barrier for the problem was an open problem in the area for some time. The first algorithm breaking the barrier is due to Razgon (2006). The running time $O(1.8899^n)$ of the algorithm from Razgon (2006) was improved in Fomin et al. (2008) to $O(1.7548^n)$. Then Xiao and Nagamochi (2013) gave an algorithm with running time $O(1.7356^n)$. Finally, an algorithm of running time $O(1.7347^n)$ was obtained in Fomin et al. (2015). For the parameterized version of the problem there was also a chain of improvements (Cao et al. 2015; Chen et al. 2008; Dehne et al. 2007; Guo et al. 2006), resulting in a $3^k n^{O(1)}$ time randomized algorithm (Cygan et al. 2011) and a $3.591^k n^{O(1)}$ time deterministic algorithm (Kociumaka and Pilipczuk 2014). This, coupled with our main theorem, immediately gives us randomized and deterministic algorithms of running times $O(1.6667^n)$ and $O(1.7216^n)$, respectively.

More generally, let $\Pi$ be a hereditary family of graphs. That is, if $G \in \Pi$ then so are all its induced subgraphs. Examples of hereditary families include the edgeless graphs, forests, bipartite graphs, chordal graphs, interval graphs, split graphs, and perfect graphs. Of course, this list is not exhaustive. For every hereditary graph family $\Pi$ there is a natural vertex deletion problem, which we define here.

Π-Vertex Deletion
**Input:** An undirected (or directed) graph $G$ and an integer $k$.
**Question:** Is there a set $S \subseteq V(G)$ with $|S| \leq k$ such that $G - S \in \Pi$?

We can cast Π-Vertex Deletion as a Φ-Subset problem as follows. The instance $I$ describes the graph $G$, so $U_I = V(G)$ and $\mathcal{F}_I$ is the family of subsets $S$ of $V(G)$ of size at most $k$ such that $G - S \in \Pi$. Notice that a polynomial time algorithm to determine whether a graph $G$ is in $\Pi$ yields a polynomial time algorithm to determine whether a set $S$ is in $\mathcal{F}_I$, implying that $\Phi$ is polynomial time computable. Moreover, a $c^k N^{O(1)}$ time algorithm for Π-Vertex Deletion trivially gives the same running time for its extension variant, since vertices in $X$ can simply be deleted. Also, if $\Pi$ is characterized by a set of forbidden induced subgraphs, which all have at most $d$ vertices, such as cographs ($d = 4$) (Corneil et al. 1981) and split graphs ($d = 5$) (Hammer and Földes 1977), we can reduce the Π-Vertex Deletion problem to $d$-Hitting Set where the number of elements is the number of vertices of the input graph.

In Table 1, we list more applications of Theorem 1.2. We also provide the running times of the fastest known parameterized and exact algorithms. The problem definitions are given in the appendix. For most of these problems, the results are obtained by simply using the fastest known parameterized algorithm in combination with Theorem 1.2. The results for Weighted $d$-Hitting Set follow from a variant that relies on algorithms for a permissive version of Φ-Extension; see Section 2.3.

We also extend the technique to enumeration problems and to prove combinatorial upper bounds. For example, a minimal satisfying assignment of a $d$-CNF formula is a satisfying assignment $a$ such that no other satisfying assignment sets every variable to 0 that $a$ sets to 0. In other words, a satisfying assignment $a$ is minimal, if setting any variable assigned 1 to 0 leads to an unsatisfying assignment. It is interesting to investigate the number of minimal satisfying assignments of $d$-CNF formulas, algorithms to enumerate these assignments, and upper bounds and enumeration algorithms for other combinatorial objects.

Formally, let $\Phi$ be an implicit set system and $c \geq 1$ be a real-valued constant. We say that $\Phi$ is *$c$-uniform* if, for every instance $I$, set $X \subseteq U_I$, and integer $k \leq n - |X|$, the cardinality of the collection

$$\mathcal{F}_{I,X}^k = \{S \subseteq U_I \setminus X : |S| = k \text{ and } S \cup X \in \mathcal{F}_I\}$$

is at most $c^k n^{O(1)}$. The next theorem will provide new combinatorial upper bounds for collections generated by $c$-uniform implicit set systems.

Theorem 1.3. *Let $c > 1$ and $\Phi$ be an implicit set system. If $\Phi$ is $c$-uniform, then $|\mathcal{F}_I| \leq (2 - \frac{1}{c})^n n^{O(1)}$ for every instance $I$.*

We say that an implicit set system $\Phi$ is *efficiently $c$-uniform* if there exists an algorithm that given $I$, $X$ and $k$ enumerates all elements of $\mathcal{F}_{I,X}^k$ in time $c^k N^{O(1)}$. In this case, we can enumerate $\mathcal{F}_I$ in time $(2 - \frac{1}{c})^{n+o(n)} N^{O(1)}$.

Theorem 1.4. *Let $c > 1$ and $\Phi$ be an implicit set system. If $\Phi$ is efficiently $c$-uniform, then there is an algorithm that given as input $I$ enumerates $\mathcal{F}_I$ in time $(2 - \frac{1}{c})^{n+o(n)} N^{O(1)}$.*

For minimal satisfying assignments of $d$-CNF formulas, we observe that the afore-mentioned branching algorithm for the extension version of Min-Ones $d$-Sat, which explores the Hamming ball of radius $k$ around the all-0 assignment of the reduced instance, encounters all minimal satisfying assignments extending $X$ by at most $k$ variables. Thus, minimal satisfying assignments for

Table 1. Summary of Known and New Results for Different Optimization Problems

| Problem Name | Parameterized | | New bound | Previous Bound | |
|---|---|---|---|---|---|
| FEEDBACK VERTEX SET | $3^k$ (r) | (h) | $1.6667^n$ (r) | | |
| FEEDBACK VERTEX SET | $3.592^k$ | (q) | $1.7217^n$ | $1.7347^n$ | (m) |
| SUBSET FEEDBACK VERTEX SET | $4^k$ | (v) | $1.7500^n$ | $1.8638^n$ | (l) |
| FEEDBACK VERTEX SET IN TOURNAMENTS | $1.6181^k$ | (r) | $1.3820^n$ | $1.4656^n$ | (r) |
| GROUP FEEDBACK VERTEX SET | $4^k$ | (v) | $1.7500^n$ | NPR | |
| NODE UNIQUE LABEL COVER | $\|\Sigma\|^{2k}$ | (v) | $(2 - \frac{1}{\|\Sigma\|^2})^n$ | NPR | |
| VERTEX $(r, \ell)$-PARTIZATION $(r, \ell \leq 2)$ | $3.3146^k$ | (b) | $1.6984^n$ | NPR | |
| INTERVAL VERTEX DELETION | $8^k$ | (e) | $1.8750^n$ | $(2 - \varepsilon)^n$ for $\varepsilon < 10^{-20}$ | (c) |
| PROPER INTERVAL VERTEX DELETION | $6^k$ | (n) | $1.8334^n$ | $(2 - \varepsilon)^n$ for $\varepsilon < 10^{-20}$ | (c) |
| BLOCK GRAPH VERTEX DELETION | $4^k$ | (a) | $1.7500^n$ | $(2 - \varepsilon)^n$ for $\varepsilon < 10^{-20}$ | (c) |
| CLUSTER VERTEX DELETION | $1.9102^k$ | (d) | $1.4765^n$ | $1.6181^n$ | (i) |
| THREAD GRAPH VERTEX DELETION | $8^k$ | (p) | $1.8750^n$ | NPR | |
| MULTICUT ON TREES | $1.5538^k$ | (o) | $1.3565^n$ | NPR | |
| 3-HITTING SET | $2.0755^k$ | (u) | $1.5182^n$ | $1.6278^n$ | (u) |
| 4-HITTING SET | $3.0755^k$ | (i) | $1.6750^n$ | $1.8704^n$ | (i) |
| $d$-HITTING SET $(d \geq 3)$ | $(d - 0.9245)^k$ | (i) | $(2 - \frac{1}{(d-0.9245)})^n$ | | (g) |
| MIN-ONES 3-SAT | $2.562^k$ | (s) | $1.6097^n$ | NPR | |
| MIN-ONES $d$-SAT $(d \geq 4)$ | $d^k$ | | $(2 - \frac{1}{d})^n$ | NPR | |
| WEIGHTED $d$-SAT $(d \geq 3)$ | $d^k$ | | $(2 - \frac{1}{d})^n$ | NPR | |
| WEIGHTED FEEDBACK VERTEX SET | $3.6181^k$ | (a) | $1.7237^n$ | $1.8638^n$ | (k) |
| WEIGHTED 3-HITTING SET | $2.168^k$ | (t) | $1.5388^n$ | $1.6755^n$ | (f) |
| WEIGHTED $d$-HITTING SET $(d \geq 4)$ | $(d - 0.832)^k$ | (j) | $(2 - \frac{1}{d-0.832})^n$ | | (f) |

NPR means that we are not aware of any previous algorithms faster than brute-force. All bounds suppress factors polynomial in the input size $N$. The algorithms in the first row are randomized (r).

(a) Agrawal et al. (2016)
(b) Baste et al. (2015), Kolay and Panolan (2015)
(c) Bliznets et al. (2013)
(d) Boral et al. (2014)
(e) Cao (2016)
(f) Cochefert et al. (2016)
(g) Cochefert et al. (2016), Fomin et al. (2010)
(h) Cygan et al. (2011)
(i) Fomin et al. (2010)
(j) Fomin et al. (2010), Shachnai and Zehavi (2017)
(k) Fomin et al. (2008)
(l) Fomin et al. (2014)
(m) Fomin et al. (2015)
(n) Cao (2015), van 't Hof and Villanger (2013)
(o) Kanj et al. (2014)
(p) Kanté et al. (2015)
(q) Kociumaka and Pilipczuk (2014)
(r) Kumar and Lokshtanov (2016)
(s) Kutzkov and Scheder (2010)
(t) Shachnai and Zehavi (2017)
(u) Wahlström (2007)
(v) Wahlström (2014)

$d$-CNF formulas are $d$-uniform. It follows immediately that minimal $d$-hitting sets are $d$-uniform and they are also efficiently $d$-uniform.

A tournament is a directed graph (digraph) in which every pair of distinct vertices is connected by a single directed edge. That is, it is an orientation of a complete graph. An induced subgraph of a tournament is called a subtournament. Furthermore, a (sub)tournament is called transitive if it does not contain any directed cycle. By a classical theorem of Moon from 1971 (Moon 1971) the number of maximal transitive subtournaments in an $n$-vertex tournament does not exceed $1.7170^n$. Gaspers and Mnich (2013) improved this bound to $1.6740^n$. It is well known that a (sub)tournament is transitive if and only if it does not contain a directed cycle of length 3 (directed 3-cycle). Using this characterization our approach yields immediately a better bound of $O(1.6667^n)$, since every directed 3-cycle needs to be hit. Similarly, in chordal graphs, a set is a feedback vertex set (FVS)

Table 2. Summary of Known and New Results for Different Combinatorial Bounds

| Problem Name | $c$-uniform | New bound | Previous Bound | |
|---|---|---|---|---|
| Minimal FVSs in Tournaments | 3 | $1.6667^n$ | $1.6740^n$ | [(a)] |
| Minimal 3-Hitting Sets | 3 | $1.6667^n$ | $1.6755^n$ | [(b)] |
| Minimal 4-Hitting Sets | 4 | $1.7500^n$ | $1.8863^n$ | [(b)] |
| Minimal 5-Hitting Sets | 5 | $1.8000^n$ | $1.9538^n$ | [(b)] |
| Minimal $d$-Hitting Sets | $d$ | $(2-\frac{1}{d})^n$ | $(2-\epsilon_d)^n$ with $\epsilon_d < 1/d$ | [(b)] |
| Minimal $d$-Sat ($d \geq 2$) | $d$ | $(2-\frac{1}{d})^n$ | NPR | |
| Minimal FVSs in chordal graphs | 3 | $1.6667^n$ | $1.6708^n$ | [(c)] |
| Minimal Subset FVSs in chordal graphs | 3 | $1.6667^n$ | NPR | |
| Maximal $r$-colorable induced subgraphs of perfect graphs | $r+1$ | $(2-\frac{1}{r+1})^n$ | NPR | |

NPR means that we are not aware of any previous results better than $2^n$. All bounds suppress factors polynomial in the universe size $n$.
[(a)]Gaspers and Mnich (2013)
[(b)]Cochefert et al. (2016)
[(c)]Golovach et al. (2013).

if it hits every 3-cycle. For maximal $r$-colorable induced subgraphs of perfect graphs it suffices to hit every clique of size $r + 1$. Some consequences of our results for enumeration algorithms and combinatorial bounds are given in Table 2.

**Local Search versus Monotone Local Search.** One of the successful approaches for obtaining exact exponential-time algorithms for $d$-SAT is based on sampling and local search. In his breakthrough article, Schöning (1999) introduced the following simple and elegant approach: Sample a random assignment and then do a local search in a Hamming ball of small radius around this assignment. With the right choice of the parameter for the local search algorithm (the Hamming distance, in this case), it is possible to prove that with a reasonable amount of samples this algorithm decides the satisfiability of a given formula with good probability. The running time of Schöning's algorithm on formulas with $n$ variables is $O((2-2/d)^n)$ and it was shown by Moser and Scheder (2011) how to derandomize it in almost the same running time, see also Dantsin et al. (2002).

While this method has been very successful for satisfiability, it is not clear how to apply this approach to other NP-complete problems, in particular, to optimization problems, like finding a satisfying assignment of Hamming weight at most $k$ or finding a hitting set of size at most $k$. The reason why Schöning's approach cannot be directly applied to optimization problems is that it is very difficult to get efficient local search algorithms for these problems.

Consider, for example, Min-Ones $d$-Sat. If we select some assignment $a$ as a center of Hamming ball $B_r$ of radius $r$, then there is a dramatic difference between searching for any satisfying assignment in $B_r$, and a satisfying assignment of Hamming weight at most $k$ in $B_r$. In the first case the local search problem can be solved in time $d^r \cdot n^{O(1)}$. In the second case, we do not know any better alternative to a brute-force search. Indeed, an algorithm with running time on the form $f(r) \cdot n^{O(1)}$ for any function $f$ would imply that FPT = W[1]. This issue is not specific to Min-Ones $d$-Sat: it is known that the problem of searching a Hamming ball $B_r$ of radius $r$ is W[1]-hard parameterized by $r$ for most natural optimization problems (Fellows et al. 2012).

Despite this obstacle, our approach is based on sampling an initial solution, and then performing a local search from that solution. The way we get around the hardness of local search is to make the local search problem easier, at the cost of reducing the success probability of the sampling step. Specifically, we only consider *monotone* local search, where we are not allowed to remove any elements from the solution, and only allowed to add at most $k$ new elements. Instead of searching a Hamming *ball* around the initial solution, we look for a solution in a Hamming *cone*. Monotone local search is equivalent to the extension problem, and it turns out that the extension problem can very often be reduced to the problem of finding a solution of size at most $k$. This allows us to use for our monotone local search the powerful toolbox developed for parameterized algorithms.

**Our Approach.** Our algorithm is based on random sampling. Suppose we are looking for a solution $S$ of size $k$ in a universe $U$ of size $n$, and we have already found some set $X$ of size $t$, which we know is a subset of $S$. At this point, one option we have is to run the extension algorithm—this would take time $c^{k-t} \cdot n^{O(1)}$. Another option is to pick a *random* element $x \in U \setminus X$, add $x$ to $X$, and then proceed. We succeed if $x$ is in $S \setminus X$, so the probability of success is at least $(k-t)/(n-t)$. If we succeed in picking $x$ from $S \setminus X$, then $k-t$ drops by 1, so running the extension algorithm on $X \cup \{x\}$ is a factor $c$ faster than running it on $X$. Therefore, as long as $(k-t)/(n-t) \geq 1/c$ it is better to keep sampling vertices and adding them to $X$. When $(k-t)/(n-t) < 1/c$ it is better to run the algorithm for the extension problem. This is the entire algorithm!

While the description of the algorithm used to obtain Theorem 1.1 is simple, the analysis is a bit more involved. In particular, getting the right upper bound on the running time of the algorithm requires more work. At a first glance it is not at all obvious that a $100^k \cdot n^{O(1)}$ time algorithm for the extension problem gives any advantage over trying all subsets of size $k$ in $\binom{n}{k}$ time. To see why our approach outperforms $\binom{n}{k}$, it is helpful to think of the brute-force algorithm as a randomized algorithm that picks a random subset of size $k$ by picking one vertex at a time and inserting it into the solution. The success probability of such an algorithm is at least

$$\frac{k}{n} \cdot \frac{k-1}{n-1} \cdot \frac{k-2}{n-2} \cdot \ldots \cdot \frac{2}{n-(k-2)} \cdot \frac{1}{n-(k-1)} = \frac{1}{\binom{n}{k}}.$$

Notice that in the beginning of the random process the success probability of each step is high, but that it gets progressively worse, and that in the very end it is close to $1/n$. At some point, we have picked $t$ vertices and $(k-t)/(n-t)$ drops below $1/c$. Here, we run the extension algorithm, spending time $c^{k-t}$, instead of continuing with brute-force, which would take time

$$\binom{n-t}{k-t} = \frac{n-t}{k-t} \cdot \frac{n-t-1}{k-t-1} \cdot \ldots \cdot \frac{n-k+2}{2} \cdot \frac{n-k+1}{1},$$

which is a product of $k-t$ larger and larger terms, with even the first and smallest term being greater than $c$. Thus, we can conclude that any $c^k$ algorithm will give some improvement over $2^n$.

Notice that if the algorithm is looking for a set of size $k$ in a universe of size $n$, the number $t$ of vertices to sample before the algorithm should switch from picking more random vertices to running the extension algorithm can directly be deduced from $n, k$, and $c$. The algorithm picks a random set $X$ of size $t$, and runs the extension algorithm on $X$. We succeed if $X$ is a subset of a solution, hence the success probability is $p = \binom{k}{t}/\binom{n}{t}$. To get constant success probability, we run the algorithm $1/p$ times, taking time $c^{k-t} \cdot n^{O(1)}$ for each run.

To derandomize the algorithm, we show that it is possible to construct in time $(1/p) \cdot 2^{o(n)}$ a family $\mathcal{F}$ of sets of size $t$, such that $|\mathcal{F}| \leq (1/p) \cdot 2^{o(n)}$, and every set of size $k$ has a subset of size $t$ in $\mathcal{F}$. Thus, it suffices to construct $\mathcal{F}$ and run the extension algorithm on each set $X$ in $\mathcal{F}$. The construction of the family $\mathcal{F}$ borrows ideas from Naor et al. (1995); however, their methods are not directly applicable to our setting. We also refer to the work of Alon and Gutner (2010) in this line of research. Our construction appears in Theorem 1.2 and constitutes one of the main technical contributions of the article.

The main conceptual contribution of this article is a non-trivial generalization of local-search based satisfiability algorithms to a wide class of optimization problems. Instead of covering the search space by Hamming balls, we cover it by Hamming cones and use a parameterized algorithm to search for a solution in each of the cones.

## 2 COMBINING RANDOM SAMPLING WITH FPT ALGORITHMS

In this section, we prove our main results, Theorems 1.1–1.4, which will give new algorithms to find a set in $\mathcal{F}_I$ and to enumerate the sets in $\mathcal{F}_I$. For many potential applications, the objective is to find a minimum-size set with certain properties, for example, that the removal of this set of vertices yields an acyclic graph. This can easily be done using the algorithms resulting from Theorems 1.1 and 1.2 with only a polynomial overhead by using binary search over $k$, the size of the targeted set $S$, and specifying that $\mathcal{F}_I$ contains only sets of size at most $k$.

### 2.1 Picking Random Subsets of the Solution

This subsection is devoted to the proof of Theorem 1.1 that we recall here. Note that the algorithm mentioned in Theorem 1.1 is a Monte Carlo algorithm with one-sided error. That is, on no-instances it always returns no, and on yes-instances it returns yes (or output a certificate) with probability $> \frac{1}{2}$. Recall that Φ-Extension is the following problem: Given an instance $I$, a subset $X \subseteq U_I$, and an integer $k$, find a set $S \subseteq U_I \setminus X$ with $|S| \leq k$ such that $X \cup S \in \mathcal{F}_I$.

Theorem 1.1. *If there exists an algorithm for* Φ-Extension *with running time* $c^k N^{O(1)}$*, then there exists a randomized algorithm for* Φ-Subset *with running time* $(2 - \frac{1}{c})^n N^{O(1)}$.

The theorem will follow from the following lemma, which gives a new randomized algorithm for Φ-Extension. The next lemma formalizes the intuition given in our approach paragraph of the the introduction.

Lemma 2.1. *If there is a constant* $c > 1$ *and an algorithm for* Φ-Extension *with running time* $c^k N^{O(1)}$*, then there is a randomized algorithm for* Φ-Extension *with running time* $(2 - \frac{1}{c})^{n-|X|} N^{O(1)}$.

Proof. Let $\mathcal{B}$ be an algorithm for Φ-Extension with running time $c^k N^{O(1)}$. We now give another algorithm, $\mathcal{A}$, for the same problem. $\mathcal{A}$ is a randomized algorithm and consists of the following two steps for an input instance $(I, X, k')$ with $k' \leq k$.

(1) Choose an integer $t \leq k'$ depending on $c$, $n$, $k'$ and $|X|$, and then select a random subset $Y$ of $U_I \setminus X$ of size $t$. The choice of $t$ will be discussed towards the end of the proof.
(2) Run Algorithm $\mathcal{B}$ on the instance $(I, X \cup Y, k' - t)$ and return the answer.

This completes the description of Algorithm $\mathcal{A}$. Since the running time of step (1) is polynomial in $N$, the running time of the whole algorithm is upper bounded by $c^{k'-t} N^{O(1)}$.

If $\mathcal{A}$ returns yes for $(I, X, k')$, then this is because $\mathcal{B}$ returned yes for $(I, X \cup Y, k' - t)$. In this case, there exists a set $S \subseteq U_I \setminus (X \cup Y)$ of size at most $k' - t \leq k - t$ such that $S \cup X \cup Y \in \mathcal{F}_I$. Thus, $Y \cup S$ witnesses that $(I, X, k)$ is indeed a yes-instance.

Next, we lower bound the probability that $\mathcal{A}$ returns yes in case there exists a set $S \subseteq U_I \setminus X$ of size exactly $k'$ such that $X \cup S \in \mathcal{F}_I$. The algorithm $\mathcal{A}$ picks a set $Y$ of size $t$ at random from $U_I \setminus X$. There are $\binom{n-|X|}{t}$ possible choices for $Y$. If $\mathcal{A}$ picks one of the $\binom{k'}{t}$ subsets of $S$ as $Y$, then $\mathcal{A}$ returns yes. Thus, given that there exists a set $S \subseteq U_I \setminus X$ of size $k'$ such that $X \cup S \in \mathcal{F}_I$, we have that

$$\Pr\left[\mathcal{A} \text{ returns yes}\right] \geq \Pr[Y \subseteq S] = \binom{k'}{t} \Big/ \binom{n-|X|}{t}.$$

Let $p(k') = \binom{k'}{t}/\binom{n-|X|}{t}$. For each $k' \in \{0, \ldots, k\}$, our main algorithm runs $\mathcal{A}$ independently $1/p(k')$ times with parameter $k'$. The algorithm returns yes if any of the runs of $\mathcal{A}$ return yes. If $(I, X, k)$ is a yes-instance, then the main algorithm returns yes with probability at least $\min_{k' \le k}\{1 - (1 - p(k'))^{1/p(k')}\} \ge 1 - \frac{1}{e} > \frac{1}{2}$. Next, we upper bound the running time of the main algorithm, which is

$$\begin{aligned}\sum_{k' \le k} \frac{1}{p(k')} \cdot c^{k'-t} N^{O(1)} &\le \max_{k' \le k} \frac{\binom{n-|X|}{t}}{\binom{k'}{t}} \cdot c^{k'-t} N^{O(1)} \\ &\le \max_{k \le n-|X|} \frac{\binom{n-|X|}{t}}{\binom{k}{t}} \cdot c^{k-t} N^{O(1)}.\end{aligned}$$

We are now ready to discuss the choice of $t$ in the algorithm $\mathcal{A}$. The algorithm $\mathcal{A}$ chooses the value for $t$ that gives the minimum value of $\frac{\binom{n-|X|}{t}}{\binom{k'}{t}} \cdot c^{k'-t}$. Thus, for fixed $n$ and $|X|$ the running time of the algorithm is upper bounded by

$$\max_{0 \le k \le n-|X|} \left\{ \min_{0 \le t \le k} \left\{ \frac{\binom{n-|X|}{t}}{\binom{k}{t}} c^{k-t} N^{O(1)} \right\} \right\}. \tag{1}$$

We upper bound the expression in (1) by $(2 - \frac{1}{c})^{n-|X|} N^{O(1)}$ in Lemma 2.3 below. The running time of the algorithm is thus upper bounded by $(2 - \frac{1}{c})^{n-|X|} N^{O(1)}$, completing the proof. □

*Remark 2.2.* The proof of Lemma 2.1 goes through just as well when $\mathcal{B}$ is a randomized algorithm. If $\mathcal{B}$ is deterministic or has one-sided error (possibly saying no whereas it should say yes), then the algorithm of Lemma 2.1 also has one sided error. If $\mathcal{B}$ has two sided error, then the algorithm of Lemma 2.1 has two sided error as well.

Now, we give the technical lemma that was used to upper bound the running time of the algorithm described in Lemma 2.1.

Lemma 2.3. *Let $c > 1$ be a fixed constant, and let $n$ be a non-negative integer. Then,*

$$\max_{0 \le k \le n} \left\{ \min_{0 \le t \le k} \left\{ \frac{\binom{n}{t}}{\binom{k}{t}} c^{k-t} \right\} \right\} \le \left(2 - \frac{1}{c}\right)^n n^{O(1)}.$$

Proof. For $k \le n/c$, we can choose $t = 0$ and obtain that the expression is at most $c^{n/c} \le (2 - 1/c)^n$.

Now, assume $k > n/c$. We have that

$$\begin{aligned}\frac{\binom{n}{t}}{\binom{k}{t}} c^{k-t} &= \frac{n!}{t!(n-t)!} \cdot \frac{t!(k-t)!}{k!} \cdot \frac{(n-k)!}{(n-k)!} \cdot c^{k-t} \\ &= \frac{\binom{n}{k}}{\binom{n-t}{k-t} \left(\frac{1}{c}\right)^{k-t}}.\end{aligned} \tag{2}$$

Let us lower bound the denominator. Using ordinary generating functions, one can show that, for any $x \in [0, 1)$ and any integer $m \ge 0$,

$$\sum_{i \ge 0} \binom{m+i}{i} x^i = \sum_{i \ge 0} \binom{m+i}{m} x^i = \frac{1}{(1-x)^{m+1}},$$

and this identity is well-known. Setting $m = n - k$ and $x = 1/c$, the summand at $i = k - t$ equals the denominator of Equation (2). Remember that we aim to choose a value for $t$, and therefore also for $i$ that maximizes the denominator. Let us first show that the maximum term of the sum occurs when $i \leq k$. When $i > k$, we can rewrite a term of the sum as

$$\begin{aligned}&\frac{(m+i)!}{i! \cdot m!} \cdot x^i \\ =&\frac{(m+i)\cdot(m+i-1)\cdot \ldots \cdot (m+k+1)\cdot(m+k)\cdot\ldots\cdot(m+1)}{i(i-1)\cdot\ldots\cdot(k+1)\cdot(k)\cdot\ldots\cdot 1} x^{i-k}\cdot x^k \\ =&\frac{m+i}{i}\cdot x \cdot \frac{m+i-1}{i-1}\cdot x\cdot\ldots\cdot\frac{m+k+1}{k+1}\cdot x\cdot\frac{(m+k)\cdot(m+k-1)\cdot\ldots\cdot(m+1)}{(k)\cdot(k-1)\cdot\ldots\cdot 1}\cdot x^k.\end{aligned}$$

Let $\Gamma_k = \frac{(m+k)\cdot(m+k-1)\cdot\ldots\cdot(m+1)}{(k)\cdot(k-1)\cdot\ldots\cdot 1}\cdot x^k$. The quantity $\Gamma_k$ corresponds to the $(k+1)$th term of the sum. Furthermore, since $\frac{n}{k} < c$, we have that $\frac{m+j}{j}\cdot x < 1$ whenever $j \geq k$. This implies that any term after $(k+1)$th term of the sum is less than $\Gamma_k$. From here, we conclude that the maximum term of the sum occurs for $i \leq k$. Then expanding the sum for $i > k$ and extracting $\Gamma_k$ from each term, we get

$$\frac{m+k+1}{k+1}\cdot x + \frac{m+k+2}{k+2}\cdot x\cdot\frac{m+k+1}{k+1}\cdot x + \cdots \frac{m+k+3}{k+3}\cdot x\cdot\frac{m+k+2}{k+2}\cdot x\cdot\frac{m+k+1}{k+1}\cdot x+\cdots$$

This sum is upper bounded by the following sum:

$$\sum_{j\geq 1}\left(\frac{m+k+1}{k+1}\right)^j \cdot x^j. \tag{3}$$

This is a geometric series with ratio $\frac{m+k+1}{k+1}x < 1$. Now, since the sum of a geometric series converges for coefficients in $[0;1)$, we also have that the series in Equation (3) can be upper bounded by a constant. Therefore, the sum of the terms for $i > k$ can be upper bounded by a constant times the largest summand. We conclude that the value of the largest summand is $\Omega((\frac{1}{1-x})^m)$ up to a lower order factor of $\mathcal{O}(k)$. So, Equation (2) is at most

$$\binom{n}{k}(1-x)^{n-k}n^{O(1)} = \left(2-\frac{1}{c}\right)^n n^{O(1)}$$

by the binomial theorem. □

By running the algorithm from Lemma 2.1 with $X = \emptyset$ and for each value of $k \in \{0,\ldots,n\}$, we obtain an algorithm for Φ-Subset, and this proves Theorem 1.1.

## 2.2 Derandomization

In this section, we prove Theorem 1.2 by derandomizing the algorithm of Theorem 1.1, at the cost of a subexponential factor in the running time. Recall Theorem 1.2.

Theorem 1.2. *If there exists an algorithm for* Φ-Extension *with running time* $c^k N^{O(1)}$, *then there exists an algorithm for* Φ-Subset *with running time* $(2-\frac{1}{c})^{n+o(n)}N^{O(1)}$.

The key tool in our derandomization is a new pseudo-random object, which we call set-inclusion-families, as well as an almost optimal (up to subexponential factors) construction of such objects.

*Definition 2.4.* Let $U$ be a universe of size $n$ and let $0 \leq q \leq p \leq n$. A family $\mathcal{C} \subseteq \binom{U}{q}$ is an $(n,p,q)$-*set-inclusion-family*, if for every set $S \in \binom{U}{p}$, there exists a set $Y \in \mathcal{C}$ such that $Y \subseteq S$.

Let $\kappa(n, p, q) = \binom{n}{q}/\binom{p}{q}$. Observe that that $\kappa^{-1}(n, p, q)$ is exactly the probability of a fixed $p$ sized set being covered by a randomly selected $q$-sized set. Let $t = \kappa(n, p, q) \cdot n^{O(1)}$. Construct the family $\mathcal{C} = \{C_1, \ldots, C_t\}$ by selecting each set $C_i$ independently and uniformly at random from $\binom{U}{q}$. Then, using an application of probabilistic method one can show that $\mathcal{C}$ is an $(n, p, q)$-set-inclusion-family with a positive probability. We give details of this construction in Lemma 3.1. However, to obtain a deterministic algorithm, we need an efficient and deterministic construction. In Section 3 (Theorem 3.3), we give a deterministic construction of an $(n, p, q)$-set-inclusion-family, $\mathcal{C}$, of size at most $\kappa(n, p, q) \cdot 2^{o(n)}$. The running time of the algorithm constructing $\mathcal{C}$ is also upper bounded by $\kappa(n, p, q) \cdot 2^{o(n)}$. This construction crucially uses a constructive version of Lemma 3.1.

The proof of Theorem 1.2 is now almost identical to the proof of Theorem 1.1. However, in Lemma 2.1, we replace the sampling step where the algorithm $\mathcal{A}$ picks a set $Y \subseteq U_I \setminus X$ of size $t$ at random, with a construction of an $(n - |X|, k', t)$-set-inclusion-family $\mathcal{C}$ using Theorem 3.3. Instead of $\kappa(n - |X|, k', t) \cdot n^{O(1)}$ independent repetitions of the algorithm $\mathcal{A}$, the new algorithm loops over all $Y \in \mathcal{C}$. The correctness follows from the definition of set-inclusion-families, while the running-time analysis is identical to the analysis of Lemma 2.1.

## 2.3 Extension to Permissive FPT Subroutines

For some of our applications, specifically the ones for weighted hitting set problems, our results rely on algorithms for permissive variants of the Φ-Extension problem. Permissive problems were introduced in the context of local search algorithms (Marx and Schlotter 2011) and it has been shown that permissive variants can be fixed-parameter tractable even if the strict version is W[1]-hard and the optimization problem is NP-hard (Gaspers et al. 2012).

Permissive Φ-Extension
**Input:** An instance $I$, a set $X \subseteq U_I$, and an integer $k$.
**Question:** If there is a subset $S \subseteq (U_I \setminus X)$ such that $S \cup X \in \mathcal{F}_I$ and $|S| \leq k$, then answer yes; else if $|\mathcal{F}_I| > 0$, then answer yes or no;
else answer no.

We observe that any algorithm solving Φ-Extension also solves Permissive Φ-Extension. However, using an algorithm for Permissive Φ-Extension will only allow us to solve a decision variant of the Φ-Subset problem, unless it also returns a certificate in case it answers yes.

Decision Φ-Subset
**Input:** An instance $I$
**Question:** Is $|\mathcal{F}_I| > 0$?

The proof of Lemma 2.1 can easily be adapted to the Permissive Φ-Extension problem.

Lemma 2.5. *If there exists a constant $c > 1$ and an algorithm for* Permissive Φ-Extension *with running time $c^k N^{O(1)}$, then there exists a randomized algorithm for* Permissive Φ-Extension *with running time $(2 - \frac{1}{c})^{n-|X|} N^{O(1)}$.*

Now, any algorithm for Permissive Φ-Extension also solves Decision Φ-Subset. If the algorithm for Permissive Φ-Extension also returns a certificate whenever it answers yes, then this also leads to an algorithm for Φ-Subset. Again, these algorithms can be derandomized at the cost of a factor $2^{o(n)}$ in the running time.

Theorem 2.6. *If there is an algorithm for* Permissive Φ-Extension *with running time $c^k N^{O(1)}$, then there is an algorithm for* Decision Φ-Subset *with running time $(2 - \frac{1}{c})^{n+o(n)} N^{O(1)}$. Moreover,* *if the algorithm for* Permissive $\Phi$-Extension *computes a certificate whenever it answers yes, then there is an algorithm for* $\Phi$-Subset *with running time* $(2-\frac{1}{c})^{n+o(n)}N^{O(1)}$.

## 2.4 Enumeration and Combinatorial Upper Bounds

In this subsection, we prove Theorems 1.3 and 1.4 on combinatorial upper bounds and enumeration algorithms. Let us first restate Theorem 1.3.

Theorem 1.3. *Let $c > 1$ and $\Phi$ be an implicit set system. If $\Phi$ is $c$-uniform, then $|\mathcal{F}_I| \leq (2-\frac{1}{c})^n n^{O(1)}$ for every instance $I$.*

For the intuition behind Theorem 1.3, consider the following random process:

(1) Choose an integer $t$ based on $c$, $n$, and $k$, then randomly sample a subset $X$ of size $t$ from $U_I$.
(2) Uniformly at random pick a set $S$ from $\mathcal{F}_{I,X}^{k-t}$, and output $W = X \cup S$. In the special case where $\mathcal{F}_{I,X}^{k-t}$ is empty return the empty set.

An analysis similar to the one in Lemma 2.1 shows that each set in the family $\mathcal{F}_I$ is selected with probability at least $(2-\frac{1}{c})^{-n} \cdot n^{-O(1)}$. This implies that there are at most $(2-\frac{1}{c})^n n^{O(1)}$ such sets. Next, we formalize this intuition.

Proof of Theorem 1.3. Let $I$ be an instance and $k \leq n$. We prove that the number of sets in $\mathcal{F}_I$ of size exactly $k$ is upper bounded by $(2-\frac{1}{c})^n n^{O(1)}$. Since $k$ is chosen arbitrarily the bound on $|\mathcal{F}_I|$ will follow. We describe below a random process that picks a set $W$ of size $k$ from $\mathcal{F}_I$ as follows.

(1) Choose an integer $t$ based on $c$, $n$, and $k$, then randomly sample a subset $X$ of size $t$ from $U_I$.
(2) Uniformly at random, pick a set $S$ from $\mathcal{F}_{I,X}^{k-t}$, and output $W = X \cup S$. In the corner case where $\mathcal{F}_{I,X}^{k-t}$ is empty, return the empty set.

This completes the description of the process.

For each set $Z \in \mathcal{F}_I$ of size exactly $k$, let $E_Z$ denote the event that the set $W$ output by the random process above is equal to $Z$. Now, we lower bound the probability of the event $E_Z$. We have the following lower bound.

$$\begin{aligned}\Pr[E_Z] &= \Pr[X \subseteq Z \wedge S = Z \setminus X]\\ &= \Pr[X \subseteq Z] \times \Pr[S = Z \setminus X \mid X \subseteq Z] \\ &= \frac{\binom{k}{t}}{\binom{n}{t}} \times \frac{1}{|\mathcal{F}_{I,X}^{k-t}|}.\end{aligned} \tag{4}$$

Since $\Phi$ is $c$-uniform, we have that $|\mathcal{F}_{I,X}^{k-t}| \leq c^{k-t} n^{O(1)}$, hence

$$\Pr[E_Z] \geq \frac{\binom{k}{t}}{\binom{n}{t}} c^{-(k-t)} n^{-O(1)}.$$

We are now ready to discuss the choice of $t$ in the random process. The integer $t$ is chosen such that the preceding expression for $\Pr[E_Z]$ is maximized (or, in other words, its reciprocal is minimized). By Lemma 2.3, we have that for every $k \leq n$ there exists a $t \leq k$ such that

$$\frac{\binom{k}{t}}{\binom{n}{t}} c^{-(k-t)} \geq \left(2-\frac{1}{c}\right)^{-n} \cdot n^{-O(1)}.$$

Hence, $\Pr[E_Z] \geq (2 - \frac{1}{c})^{-n} \cdot n^{-O(1)}$ for every $Z \in \mathcal{F}_I$ of size $k$. Since the events $E_Z$ are disjoint for all the different sets $Z \in \mathcal{F}_I$, we have that

$$\sum_{\substack{Z \in \mathcal{F}_I \\ |Z|=k}} \Pr[E_Z] \leq 1.$$

This, together with the lower bound on $\Pr[E_Z]$ implies that the number of sets in $\mathcal{F}_I$ of size exactly $k$ is upper bounded by $(2 - \frac{1}{c})^n n^{O(1)}$, completing the proof. □

If the implicit set system $\Phi$ is efficiently $c$-uniform, then the proof of Theorem 1.3 can be made constructive by replacing the sampling step by a construction of an $(n, k, t)$-set-inclusion-family $C$ using Theorem 3.3. For each $X \in C$ the algorithm uses the fact that $\Phi$ is efficiently $c$-uniform to loop over all sets $S \in \mathcal{F}_{I,X}^{k-t}$ and output $X \cup S$ for each such $S$. Looping over $C$ instead of sampling $X$ incurs a $2^{o(n)}$ overhead in the running time of the algorithm. To avoid enumerating duplicates, we also store each set that we output in a trie and for each set that we output, we check first in linear time whether we have already output that set.

Theorem 1.4. *Let $c > 1$ and $\Phi$ be an implicit set system. If $\Phi$ is efficiently $c$-uniform, then there is an algorithm that given as input $I$ enumerates $\mathcal{F}_I$ in time $(2 - \frac{1}{c})^{n+o(n)} N^{O(1)}$.*

## 3 EFFICIENT CONSTRUCTION OF SET-INCLUSION-FAMILIES

In this section, we give the promised construction of set-inclusion-families. The $(n, p, q)$-set-inclusion-family is actually related to $(n, k, l)$ covering design (Kuzjurin 2000; Jukna 2011), and it is very plausible that a modification of these known constructions can be used for our purposes. It is also related to the concepts of splitters and universal sets introduced by Naor et al. (1995). Here, we present a simple and self-contained construction. We start by giving a construction of set-inclusion-families with good bounds on the size, but with a poor bound on the construction time. Recall that $\kappa(n, p, q) = \binom{n}{q} / \binom{p}{q}$.

Before proceeding with the technical proof of Theorem 3.3, let us provide a high-level overview. The proof is based on the following ideas.

(a) **Existential Proof (Lemma 3.1)**. This lemma shows that there exists an $(n, p, q)$-set-inclusion-family $C$ of size at most $\kappa(n, p, q) \cdot n^{O(1)}$. Essentially, it shows that if we construct the family $C = \{C_1, \ldots, C_t\}$ by selecting each set $C_i$ independently and uniformly at random from $\binom{U}{q}$, then with positive probability $C$ is indeed an $(n, p, q)$-set-inclusion-family. This also leads to not "so fast algorithm" to design the family $C$ using an approximation algorithm for the Set Cover problem.

(b) **Universe Reduction.** The construction obtained in Lemma 3.1 has only one drawback—the time is much larger what we can afford. To overcome this lacuna, we do not apply the construction in Lemma 3.1 directly. We first prove a result that helps us in reducing the universe size to something smaller. This is done using the known construction of pairwise independent hash families of size $O(n^2)$. This makes the universe small enough that we can apply the construction given in Lemma 3.1.

Lemma 3.1. *There is an algorithm that given $n$, $p$ and $q$ outputs an $(n, p, q)$-set-inclusion-family $C$ of size at most $\kappa(n, p, q) \cdot n^{O(1)}$ in time $O(8^n)$.*

Proof. We start by giving a randomized algorithm that with positive probability constructs an $(n, p, q)$-set-inclusion-family $C$ with the claimed size. We will then discuss how to deterministically compute such a $C$ within the required time bound. Set $t = \kappa(n, p, q) \cdot (p + 1) \log n$ and construct the family $C = \{C_1, \ldots, C_t\}$ by selecting each set $C_i$ independently and uniformly at random from $\binom{U}{q}$.

By construction, the size of $C$ is within the required bounds. We now argue that with positive probability $C$ is indeed an $(n, p, q)$-set-inclusion-family. For a fixed set $A \in \binom{U}{p}$, and integer $i \leq t$, we consider the probability that $C_i \subseteq A$. This probability is $1/\kappa(n, p, q)$. Since each $C_i$ is chosen independently from the other sets in $C$, the probability that *no* $C_i$ satisfies $C_i \subseteq A$ is

$$\left(1 - \frac{1}{\kappa(n, p, q)}\right)^t \leq e^{-(p+1)\log n} \leq \frac{1}{n^{p+1}}.$$

There are $\binom{n}{p}$ choices for $A \in \binom{U}{p}$, therefore the union bound yields that the probability that there exists an $A \in \binom{U}{p}$ such that no set $C_i \in C$ satisfies $C_i \subseteq A$ is upper bounded by $\frac{1}{n^{p+1}} \cdot n^p = \frac{1}{n}$.

To construct $C$ within the stated running time, proceed as follows. We construct an instance of Set Cover, and then, using a known approximation algorithm for Set Cover, we construct the desired family. An instance of Set Cover consists of a universe $\mathcal{U}$ and a family $\mathscr{S}$ of subsets of $\mathcal{U}$. The objective is to find a minimum sized sub-collection $\mathscr{S}' \subseteq \mathscr{S}$ such that the union of elements of the sets in $\mathscr{S}'$ is $\mathcal{U}$. It is known (Johnson 1974) that Set Cover admits a polynomial time approximation algorithm with factor $O(\log |\mathcal{U}|)$. For our problem, the elements of the universe $\mathcal{U}$ are $u_A$ for every $A \in \binom{U}{p}$. For every set $B \in \binom{U}{q}$, let $F_B$ consist of all the elements $u_A \in \mathcal{U}$ such that $B \subseteq A$. The set family $\mathscr{S}$ contains $F_B$ for each choice of $B \in \binom{U}{q}$. Given a sub-collection $\mathscr{S}' \subseteq \mathscr{S}$, we construct the family $C(\mathscr{S}')$ by taking the sets $B \in \binom{U}{q}$ such that $F_B \in \mathscr{S}'$. Clearly, any $C(\mathscr{S}')$ corresponding to a sub-collection $\mathscr{S}' \subseteq \mathscr{S}$ covering $\mathcal{U}$ is a $(n, p, q)$-set-inclusion-family, and vice versa.

Let OPT denote the size of a minimum sized sub-collection $\mathscr{S}' \subseteq \mathscr{S}$ covering $\mathcal{U}$. We run the known $O(\log |\mathcal{U}|)$-factor approximation algorithm on our instance and obtain a sub-collection $\mathscr{S}' \subseteq \mathscr{S}$ covering $\mathcal{U}$. Let $C = C(\mathscr{S}')$. By the preceding discussions, we know that $C$ is an $(n, p, q)$-set-inclusion-family. Clearly, the size of $C$ is upper bounded by

$$\begin{aligned} |C| &\leq \mathrm{OPT} \cdot O(\log |\mathcal{U}|) \leq t \cdot O(\log |\mathcal{U}|) \\ &\leq O(t(\log n^p)) \leq \kappa(n, p, q) \cdot n^{O(1)}. \end{aligned}$$

It is well known that one can implement the approximation algorithm for Set Cover to run in time $O(|\mathcal{U}| \cdot \sum_{S \in \mathscr{S}} |S|) = O(\binom{n}{p} \cdot \binom{n}{q}\binom{n-q}{p-q}) = O(2^n \cdot 2^n \cdot 2^n) = O(8^n)$ (Korte and Vygen 2012). This concludes the proof. □

Next, we will reduce the problem of finding an $(n, p, q)$-set-inclusion-family to the same problem, but with a much smaller value of $n$. To that end, we will use a well-known construction of *pair-wise independent* families of functions. Let $U$ be a universe of size $n$ and $b$ be a positive integer. Let $\mathcal{X}$ be a collection of functions from $U$ to $[b]$. That is, each function $f$ in $\mathcal{X}$ takes as input an element of $U$ and returns an integer from 1 to $b$. The collection $\mathcal{X}$ is said to be *pair-wise independent* if, for every $i, j \in [b]$ and every $u, v \in U$ such that $u \neq v$ we have that

$$\Pr_{f \in \mathcal{X}}[f(u) = i \wedge f(v) = j] = \frac{1}{b^2}.$$

Observe that this implies that any pairwise independent family of functions from $U$ to $[b]$ with $|U| \geq 2$ also satisfies that for every $i \in [b]$ and $u \in U$ we have $\Pr_{f \in \mathcal{X}}[f(u) = i] = \frac{1}{b}$. We will make use of the following known construction of pair-wise independent families.

Proposition 3.2 (Alon et al. (1986)). *There is a polynomial time algorithm that given a universe $U$ and integer $b$ constructs a pair-wise independent family $\mathcal{X}$ of functions from $U$ to $[b]$. The size of $\mathcal{X}$ is $O(n^2)$.*

Using Proposition 3.2, we can give a much faster construction of an $(n, p, q)$-set-inclusion-family than the one in Lemma 3.1 at the cost of a subexponential overhead in the size of the family.

We will also use the following well known bounds on binomial coefficients to simplify our expressions in the proof of next theorem,

$$\frac{1}{n^{O(1)}}\left[\left(\frac{k}{n}\right)^{-\frac{k}{n}}\left(1-\frac{k}{n}\right)^{\frac{k}{n}-1}\right]^n \leq \binom{n}{k} \leq \left[\left(\frac{k}{n}\right)^{-\frac{k}{n}}\left(1-\frac{k}{n}\right)^{\frac{k}{n}-1}\right]^n. \tag{5}$$

Theorem 3.3. *There is an algorithm that given $n, p$ and $q$ outputs an $(n, p, q)$-set-inclusion-family $C$ of size at most $\kappa(n, p, q) \cdot 2^{o(n)}$ in time $\kappa(n, p, q) \cdot 2^{o(n)}$.*

Proof. The construction sets $\beta = q/p$, selects a number $b = \lceil \log n \rceil$ of *buckets* and applies Proposition 3.2 to construct a pairwise independent family $\mathcal{X}$ of functions from $U$ to $[b]$. For each function $f \in \mathcal{X}$ and integer $i \in [b]$, we set $U_f^i = \{u \in U : f(u) = i\}$ and $n_f^i = |U_f^i|$. Call a function $f$ *good* if, for every $i \in [b]$ we have that $|n_f^i - n/b| \leq \sqrt{n} \cdot b$. For every good function $f \in \mathcal{X}$, every $i \in [b]$, and every integer $s \leq n_f^i$, we construct an $(n_f^i, s, \lceil \beta s \rceil)$-set-inclusion-family $C_f^{(i,s)}$ using Lemma 3.1. We now describe the family $C$ output by the construction. Each set $Y \in C$ is defined by

(1) a good $f \in \mathcal{X}$,
(2) a sequence $p_1, \ldots, p_b$ of integers such that $|p_i - \frac{p}{b}| \leq \sqrt{n} \cdot b$,
(3) a sequence $Y_1, \ldots, Y_b$ of sets with $Y_i \in C_f^{(i,p_i)}$,
(4) a set $D \subseteq U$ of size at most $b$.

The set $Y$ defined by the tuple $(f, p_1, \ldots, p_b, Y_1, \ldots, Y_b, D)$ is set to $Y = (\bigcup_{i \leq b} Y_i) \setminus D$. This concludes the construction. Let $\mathcal{T}$ denote the set of tuples.

First, we analyze the running time of the construction. Constructing the set $\mathcal{X}$ takes polynomial time by Proposition 3.2. For each good $f$, $i \in [b]$ and $s \leq n_f^i$, constructing $C_f^{(i,s)}$ using Lemma 3.1 takes time $2^{o(n)}$, because $n_f^i \leq \frac{n}{b} + \sqrt{n} \cdot \log n = O(\frac{n}{\log n})$. There are $O(n^2)$ choices for $f$, at most $O(\log n)$ choices for $i$ and $O(\frac{n}{\log n})$ choices for $s$. Thus, the overall time of the construction is $2^{o(n)}$ plus the time to output $C$. Outputting $C$ can be done spending polynomial time for each set $Y \in \mathcal{T}$ by enumerating over all the tuples $(f, p_1, \ldots, p_b, Y_1, \ldots, Y_b, D)$. Thus, the running time of the construction is upper bounded by $2^{o(n)} + |\mathcal{T}| \cdot n^{O(1)}$.

Further, note that the size of $C$ is upper bounded by the number of tuples $(f, p_1, \ldots, p_b, Y_1, \ldots, Y_b, D)$. That is, $|C| \leq |\mathcal{T}|$. It remains to upper bound $|\mathcal{T}|$. There are $O(n^2)$ choices for $f$, at most $n^b$ choices for $p_1, \ldots, p_b$ and $n^{O(b)}$ choices for $D$. Thus, the number of tuples is upper bounded by $2^{o(n)}$ times the maximum number of choices for $Y_1 \ldots Y_b$ for any fixed choice of $f, p_1 \ldots p_b$ and $D$. For each $i$, we choose $Y_i$ from $C_f^{(i,p_i)}$, so there are $\kappa(n_f^i, p_i, \lceil \beta p_i \rceil) \cdot n^{O(1)}$ choices for $Y_i$. It follows that the total number of choices for $Y_1, \ldots, Y_b$ is upper bounded by

$$\prod_{i \leq b} \kappa\left(n_f^i, p_i, \lceil \beta p_i \rceil\right) \cdot n^{O(1)} \leq 2^{o(n)} \cdot \prod_{i \leq b} \frac{\binom{n_f^i}{\lceil \beta p_i \rceil}}{\binom{p_i}{\lceil \beta p_i \rceil}}. \tag{6}$$

Now, we have that

$$\prod_{i\le b}\binom{n_f^i}{\lceil\beta p_i\rceil}\le\prod_{i\le b}\left[\binom{\lceil n/b\rceil}{\lceil p/b\rceil}\cdot n^{O(\sqrt{n}\log n)}\right]\le\binom{n}{p}\cdot 2^{o(n)}. \tag{7}$$

In the last transition, we used that the number of ways to pick $b$ sets of size $\lceil p/b\rceil$, each from a universe of size $\lceil n/b\rceil$ is upper bounded by the number of ways to pick a set of size $b\cdot\lceil p/b\rceil$ from a universe of size $b\cdot\lceil n/b\rceil$. This in turn is upper bounded by $\binom{n}{p}\cdot 2^{o(n)}$. Furthermore,

$$\begin{aligned}\prod_{i\le b}\binom{p^i}{\lceil\beta p_i\rceil}&\ge\prod_{i\le b}\left[\binom{\lceil p/b\rceil}{\lceil\beta(p/b)\rceil}\cdot n^{-O(\sqrt{n}\log n)}\right]\\&\ge\left[\left(\beta^{-\beta}(1-\beta)^{\beta-1}\right)^{(p/b)}\cdot n^{-O(1)}\right]^b\cdot 2^{-o(n)}\\&\ge\binom{p}{q}\cdot 2^{-o(n)}\end{aligned}\tag{8}$$

Here the two last transitions use Equation (5). Inserting the bounds from (7) and (8) into (6) yields that the total number of choices for $Y_1,\ldots,Y_b$ is upper bounded by $\kappa(n,p,q)\cdot 2^{o(n)}$ and thus, $|C|\le\kappa(n,p,q)\cdot 2^{o(n)}$ as well.

All that remains is to argue that $C$ is in fact an $(n,p,q)$-set-inclusion-family. Towards this, consider any subset $S$ of $U$ of size exactly $p$. For any fixed $i\in[b]$, consider the process of picking a random function $f$ from $\mathcal{X}$. We are interested in the random variables $|U_f^i|$ and $|U_f^i\cap S|$. Using indicator variables for each element in $U$ it is easy to show that

$$\mathop{\mathrm{E}}_{f\in\mathcal{X}}\left[\left|U_f^i\right|\right]=\frac{n}{b}\text{ and }\mathop{\mathrm{E}}_{f\in\mathcal{X}}\left[\left|U_f^i\cap S\right|\right]=\frac{p}{b}.$$

Furthermore, $\mathcal{X}$ is pairwise independent, and therefore the covariance of any pair of indicator variables is 0. Thus, $\mathrm{Var}_{f\in\mathcal{X}}[|U_f^i|]\le n$ and $\mathrm{Var}_{f\in\mathcal{X}}[|U_f^i\cap S|]\le n$. By Chebyshev's inequality it follows that

$$\Pr\left[\left||U_f^i|-\frac{n}{b}\right|\ge\sqrt{n}\cdot b\right]\le\frac{1}{b^2}\qquad\text{and}$$
$$\Pr\left[\left||U_f^i\cap S|-\frac{p}{b}\right|\ge\sqrt{n}\cdot b\right]\le\frac{1}{b^2}.$$

Consider now the probability that at least one of the variables $|U_i|$ or $|U_i\cap S|$ deviates from its expectation by at least $\sqrt{n}\cdot b$. Combining the preceding inequalities with the union bound taken over all $i\in[b]$ yields that this probability is upper bounded by $2b\cdot\frac{1}{b^2}\le\frac{2}{b}$. Since $b=\log n>2$, we have that with non-zero probability, *all* the random variables $|U_1|,\ldots,|U_b|$ and $|U_1\cap S|,\ldots,|U_b\cap S|$ are within $\sqrt{n}\cdot b$ of their respective means. Thus, there exists a function $f\in\mathcal{X}$ such that for every $i\in[b]$, we have

$$\left||U_f^i|-\frac{n}{b}\right|\le\sqrt{n}\cdot b\text{ and }\left||U_f^i\cap S|-\frac{p}{b}\right|\le\sqrt{n}\cdot b.$$

In the remainder of the proof, let $f$ be such a function in $\mathcal{X}$.

The choice of $f$ implies that $f$ is a good function. For each $i\le b$, let $S_i=|U_f^i\cap S|$ and $p_i=|S_i|$. Again, by the choice of $f$, we have that $|p_i-\frac{p}{b}|\le\sqrt{n}\cdot b$. Since $C_f^{(i,p_i)}$ is an $(n_f^i,p_i,\lceil\beta p_i\rceil)$-set-inclusion-family, there exists a set $Y_i\in C_f^{(i,p_i)}$ such that $Y_i\subseteq S_i$ and $|Y_i|=\lceil\beta p_i\rceil$. For each $i\in[b]$ select such a $Y_i$ from $C_f^{(i,p_i)}$. Finally, let $D$ be any subset of $\bigcup_{i\le b}Y_i$ of size $\sum_{i\le b}|Y_i|-q$. Note that $|Y_i|\le\beta p_i+1$, thus $\sum_{i\le b}|Y_i|-q\le b$, so $|D|\le b$. Consider, finally, the tuple

$(f, p_1 \dots p_b, Y_1, \dots Y_b, D)$. We have just proved that this tuple satisfies all of the conditions for giving rise to a set $Y = \bigcup_{i \leq b} Y_i \setminus D$ in $C$. However, $Y_i \subseteq S_i$ for all $i$, so $Y \subseteq S$, proving that $C$ is a $(n, p, q)$-set-inclusion-family. □

## 4 CONCLUSION AND DISCUSSION

In this article, we have shown that for many subset problems, an algorithm that finds a solution of size $k$ in time $c^k n^{O(1)}$ directly implies an algorithm with running time $O((2 - \frac{1}{c})^{n+o(n)})$. We also show that often, an upper bound of $c^k n^{O(1)}$ on the number of sets of size at most $k$ in a family $\mathcal{F}$ can yield an upper bound of $O((2 - \frac{1}{c})^{n+o(n)})$ on the size of $\mathcal{F}$. Our results reveal an exciting new connection between parameterized algorithms and exponential-time algorithms. All of our algorithms have a randomized and a deterministic variant. The only down-side of using the deterministic algorithm rather than the randomized one is a $2^{o(n)}$ multiplicative factor in the running time, and an additional $2^{o(n)}$ space requirement. It is possible to reduce the space overhead to a much smaller (but still super-polynomial) term; however, this would make the presentation considerably more involved.

For the enumeration algorithm of Theorem 1.4, it is well worth noting that the algorithm only uses subexponential space if the algorithm is allowed to output the same set multiple times. If duplicates are not allowed, then the algorithm needs exponential space to store a trio of the sets that have already been output. Another approach is to use an output-sensitive algorithm. For example, there is a polynomial-delay polynomial-space algorithm enumerating all feedback vertex sets in a tournament (Gaspers and Mnich 2013), and its running time is $O(1.6667^n)$ by our combinatorial upper bound.

Our analysis also reveals that to obtain a $(2 - \epsilon)^n$ time algorithm with $\epsilon > 0$ for a subset problem, it is sufficient to get a $O(c^k \binom{n-|X|}{k}^{1-\delta})$ algorithm for any constant $c$ and $\delta > 0$ for the extension problem. This might be a promising route for obtaining better exact exponential-time algorithms for problems that currently do not have single-exponential-time parameterized algorithms. For example, it would be interesting to see whether it is possible to improve on Razgon's $O(1.9977^n)$ time algorithm (Razgon 2007) for Directed Feedback Vertex Set by designing a $O(c^k \binom{n-|X|}{k}^{1-\delta})$ time algorithm.

## APPENDIX

## A PROBLEM DEFINITIONS

We list the definitions of the problems considered in this article.

Feedback Vertex Set **Parameter:** $k$
**Input:** An undirected graph $G$ and a positive integer $k$.
**Question:** Does there exist a subset $S \subseteq V(G)$ of size at most $k$ such that $G - S$ is acyclic?

Weighted Feedback Vertex Set **Parameter:** $k$
**Input:** An undirected graph $G$, a positive integer $k$, a weight function $w : V(G) \to \mathbb{N}$, and a positive integer $W$.
**Question:** Is there a set $S \subseteq V(G)$ of size at most $k$ and weight at most $W$ such that $G - S$ is acyclic?

Subset Feedback Vertex Set **Parameter:** $k$
**Input:** An undirected graph $G$, a vertex subset $T \subseteq V(G)$, and a positive integer $k$.
**Question:** Does there exist a subset $S \subseteq V(G)$ of size at most $k$ such that $G - S$ has no cycle that contains a vertex from $T$?

Let $\Gamma$ be a finite group with identity element $1_\Gamma$. A $\Gamma$-labeled graph is a graph $G = (V, E)$ with a labeling $\lambda : E \to \Gamma$ such that $\lambda(u, v)\lambda(v, u) = 1_\Gamma$ for every edge $uv \in E$. For a cycle $C = (v_1, \ldots, v_r, v_1)$, define $\lambda(C) = \lambda(v_1, v_2) \cdot \cdots \cdot \lambda(v_r, v_1)$.

Group Feedback Vertex Set **Parameter:** $k$
**Input:** A group $\Gamma$, a $\Gamma$-labelled graph $(G, \lambda)$, and a positive integer $k$.
**Question:** Does there exist a subset $S \subseteq V(G)$ of size at most $k$ such that every cycle $C$ in $G - S$ has $\lambda(C) = 1_\Gamma$?

Node Unique Label Cover **Parameter:** $|\Sigma| + k$
**Input:** An undirected graph $G = (V, E)$, a finite alphabet $\Sigma$, an integer $k$, and for each edge $e \in E$ and each of its endpoints $v$ a permutation $\psi_{e,v}$ of $\Sigma$ such that if $e = xy$ then $\psi_{e,x} = \psi_{e,v}^{-1}$
**Question:** Is there a vertex subset $S \subset V$ of size at most $k$ and a function $\Psi : V \setminus S \to \Sigma$ such that for every edge $uv \in E(G - S)$, we have $(\Psi(u), \Psi(v)) \in \psi_{uv,u}$?

For fixed integers $r, \ell \geq 0$, a graph $G$ is called an *$(r, \ell)$-graph* if the vertex set $V(G)$ can be partitioned into $r$ independent sets and $\ell$ cliques.

Vertex $(r, \ell)$-Partization **Parameter:** $k$
**Input:** A graph $G$ and a positive integer $k$
**Question:** Is there a vertex subset $S \subseteq V(G)$ of size at most $k$ such that $G - S$ is an $(r, \ell)$-graph?

Several special cases of this problem are well known and have been widely studied. For example, $(2, 0)$- and $(1, 1)$-graphs correspond to bipartite graphs and split graphs, respectively. We note that Vertex $(r, \ell)$-Partization can be solved in $O(1.1996^{(r+\ell)\cdot n})$ by taking $r$ copies of the input graph, $\ell$ copies of its complement, making all the copies of a same vertex into a clique and computing a maximum independent set of this graph using the algorithm from Xiao and Nagamochi (2013). This is faster than $O(2^n)$ when $r + \ell \leq 3$. We improve on this algorithm for $r, \ell \leq 2$ and $r + \ell \geq 3$.

For the definition of graph classes, including interval graphs, proper interval graphs, block graphs, cluster graphs, we refer to Brandstädt et al. (1999).

Proper Interval Vertex Deletion **Parameter:** $k$
**Input:** An undirected graph $G$ and a positive integer $k$.
**Question:** Does there exist a subset $S \subseteq V(G)$ of size at most $k$ such that $G - S$ is a proper interval graph?

Interval Vertex Deletion **Parameter:** $k$
**Input:** An undirected graph $G$ and a positive integer $k$.
**Question:** Does there exist a subset $S \subseteq V(G)$ of size at most $k$ such that $G - S$ is an interval graph?

Block Graph Vertex Deletion **Parameter:** $k$
**Input:** An undirected graph $G$ and a positive integer $k$.
**Question:** Does there exist a subset $S \subseteq V(G)$ of size at most $k$ such that $G - S$ is a block graph?

Cluster Vertex Deletion **Parameter:** $k$
**Input:** An undirected graph $G$ and a positive integer $k$.
**Question:** Does there exist a subset $S \subseteq V(G)$ of size at most $k$ such that $G - S$ is a cluster graph?

Thread Graph Vertex Deletion **Parameter:** $k$
**Input:** An undirected graph $G$ and a positive integer $k$.
**Question:** Does there exist a subset $S \subseteq V(G)$ of size at most $k$ such that $G - S$ is of linear rank-width one?

Multicut on Trees **Parameter:** $k$
**Input:** A tree $T$ and a set $\mathcal{R} = \{\{s_1, t_1\}, \ldots, \{s_r, t_r\}\}$ of pairs of vertices of $T$ called terminals, and a positive integer $k$.
**Question:** Does there exist a subset $S \subseteq E(T)$ of size at most $k$ whose removal disconnects each $s_i$ from $t_i$, $i \in [r]$?

$d$-Hitting Set **Parameter:** $k$
**Input:** A family $\mathscr{S}$ of subsets of size at most $d$ of a universe $\mathcal{U}$ and a positive integer $k$.
**Question:** Does there exist a subset $S \subseteq \mathcal{U}$ of size at most $k$ such that $F$ is a hitting set for $\mathscr{S}$?

Weighted $d$-Hitting Set **Parameter:** $k$
**Input:** A family $\mathscr{S}$ of subsets of size at most $d$ of a universe $\mathcal{U}$, a weight function $w : \mathcal{U} \to \mathbb{N}$, and positive integers $k$ and $W$.
**Question:** Does there exist a subset $S \subseteq \mathcal{U}$ of size at most $k$ and weight at most $W$ such that $F$ is a hitting set for $\mathscr{S}$?

Min-Ones $d$-Sat **Parameter:** $k$
**Input:** A propositional formula $F$ in conjunctive normal form (CNF) where each clause has at most $d$ literals and an integer $k$.
**Question:** Does $F$ have a satisfying assignment with Hamming weight at most $k$?

Weighted $d$-Sat **Parameter:** $k$
**Input:** A CNF formula $F$ where each clause has at most $d$ literals, a weight function $w : var(F) \to \mathbb{Z}$, and integers $k$ and $W$.
**Question:** Is there a set $S \subseteq var(F)$ of size at most $k$ and weight at most $W$ such that $F$ is satisfied by the assignment that sets the variables in $S$ to 1 and all other variables to 0?

Tournament Feedback Vertex Set **Parameter:** $k$
**Input:** A tournament $G$ and a positive integer $k$.
**Question:** Does there exist a subset $S \subseteq V(G)$ of size at most $k$ such that $G - S$ is a transitive tournament?

Split Vertex Deletion **Parameter:** $k$
**Input:** An undirected graph $G$ and a positive integer $k$.
**Question:** Does there exist a subset $S \subseteq V(G)$ of size at most $k$ such that $G - S$ is a split graph?

Cograph Vertex Deletion **Parameter:** $k$
**Input:** An undirected graph $G$ and a positive integer $k$.
**Question:** Does there exist a subset $S \subseteq V(G)$ of size at most $k$ such that $G - S$ is a cograph?

Directed Feedback Vertex Set **Parameter:** $k$
**Input:** A directed graph $G$ and a positive integer $k$.
**Question:** Does there exist a subset $S \subseteq V(G)$ of size at most $k$ such that $G - S$ is directed acyclic graph?

## ACKNOWLEDGMENTS

Many thanks to Russell Impagliazzo, Meirav Zehavi, and anonymous reviewers for useful comments and insightful discussions. The final part of this work was done while the authors were visiting the Simons Institute for the Theory of Computing.

## REFERENCES

Akanksha Agrawal, Sudeshna Kolay, Daniel Lokshtanov, and Saket Saurabh. 2016. A faster FPT algorithm and a smaller kernel for block graph vertex deletion. In *Proceedings of the 12th Latin American Symposium on Theoretical Informatics (LATIN'16) (Lecture Notes in Computer Science)*, Vol. 9644. Springer, 1–13. DOI: https://doi.org/10.1007/978-3-662-49529-2_1

Noga Alon, László Babai, and Alon Itai. 1986. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algor.* 7, 4 (1986), 567–583.

Noga Alon and Shai Gutner. 2010. Balanced families of perfect hash functions and their applications. *ACM Trans. Algor.* 6, 3 (2010).

Julien Baste, Luerbio Faria, Sulamita Klein, and Ignasi Sau. 2015. *Parameterized Complexity Dichotomy for $(r, \ell)$-Vertex Deletion.* Technical Report abs/1504.05515. arXiv CoRR.

Ivan Bliznets, Fedor V. Fomin, Michal Pilipczuk, and Yngve Villanger. 2013. Largest chordal and interval subgraphs faster than $2^n$. In *Proceedings of the 21st Annual European Symposium on Algorithms (ESA'13) (Lecture Notes in Computer Science)*, Vol. 8125. Springer, 193–204.

Anudhyan Boral, Marek Cygan, Tomasz Kociumaka, and Marcin Pilipczuk. 2014. A fast branching algorithm for cluster vertex deletion. In *Proceedings of the 9th International Computer Science Symposium in Russia (CSR'14) (Lecture Notes in Computer Science)*, Vol. 8476. Springer, 111–124.

Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. 1999. *Graph Classes: A Survey.* SIAM.

Yixin Cao. 2015. Unit interval editing is fixed-parameter tractable. In *Proceedings of the 42nd International Colloquium of Automata, Languages and Programming (ICALP'15) (Lecture Notes in Computer Science)*, Vol. 9134. Springer, 306–317.

Yixin Cao. 2016. Linear recognition of almost interval graphs. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'16).* 1096–1115.

Yixin Cao, Jianer Chen, and Yang Liu. 2015. On feedback vertex set: New measure and new structures. *Algorithmica* 73, 1 (2015), 63–86. DOI: https://doi.org/10.1007/s00453-014-9904-6

Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. 2008. Improved algorithms for feedback vertex set problems. *J. Comput. Syst. Sci.* 74, 7 (2008), 1188–1198.

Manfred Cochefert, Jean-François Couturier, Serge Gaspers, and Dieter Kratsch. 2016. Faster algorithms to enumerate hypergraph transversals. In *Proceedings of the 12th Latin American Theoretical Informatics Symposium (LATIN'16).* Lecture Notes in Computer Science, Vol. 9644, Springer, 306–318.

Derek G. Corneil, H. Lerchs, and L. Stewart Burlingham. 1981. Complement reducible graphs. *Discrete Appl. Math.* 3, 3 (1981), 163–174. DOI: https://doi.org/10.1016/0166-218X(81)90013-5

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

Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. 2011. Solving connectivity problems parameterized by treewidth in single exponential time. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS'11)*. IEEE, 150–159.

Evgeny Dantsin, Andreas Goerdt, Edward A. Hirsch, Ravi Kannan, Jon Kleinberg, Christos Papadimitriou, Prabhakar Raghavan, and Uwe Schöning. 2002. A deterministic $(2 - 2/(k+1))^n$ algorithm for $k$-SAT based on local search. *Theor. Comput. Sci.* 289, 1 (2002), 69–83.

Frank K. H. A. Dehne, Michael R. Fellows, Michael A. Langston, Frances A. Rosamond, and Kim Stevens. 2007. An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem. *Theory Comput. Syst.* 41, 3 (2007), 479–492.

Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances A. Rosamond, Saket Saurabh, and Yngve Villanger. 2012. Local search: Is brute-force avoidable? *J. Comput. Syst. Sci.* 78, 3 (2012), 707–719. DOI: https://doi.org/10.1016/j.jcss.2011.10.003

Fedor V. Fomin, Serge Gaspers, Dieter Kratsch, Mathieu Liedloff, and Saket Saurabh. 2010. Iterative compression and exact algorithms. *Theor. Comput. Sci.* 411, 7–9 (2010), 1045–1053.

Fedor V. Fomin, Serge Gaspers, Daniel Lokshtanov, and Saket Saurabh. 2016. Exact algorithms via monotone local search. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC'16)*. ACM, 764–775. DOI: https://doi.org/10.1145/2897518.2897551

Fedor V. Fomin, Serge Gaspers, Artem V. Pyatkin, and Igor Razgon. 2008. On the minimum feedback vertex set problem: Exact and enumeration algorithms. *Algorithmica* 52, 2 (2008), 293–307.

Fedor V. Fomin, Pinar Heggernes, Dieter Kratsch, Charis Papadopoulos, and Yngve Villanger. 2014. Enumerating minimal subset feedback vertex sets. *Algorithmica* 69, 1 (2014), 216–231. DOI: https://doi.org/10.1007/s00453-012-9731-6

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

Fedor V. Fomin, Ioan Todinca, and Yngve Villanger. 2015. Large induced subgraphs via triangulations and CMSO. *SIAM J. Comput.* 44, 1 (2015), 54–87. DOI: https://doi.org/10.1137/140964801

Serge Gaspers, Eun Jung Kim, Sebastian Ordyniak, Saket Saurabh, and Stefan Szeider. 2012. Don't be strict in local search!. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI'12)*. AAAI Press.

Serge Gaspers and Matthias Mnich. 2013. Feedback vertex sets in tournaments. *J. Graph Theory* 72, 1 (2013), 72–89. DOI: https://doi.org/10.1002/jgt.21631

Petr A. Golovach, Pim van 't Hof, and Daniël Paulusma. 2013. Obtaining planarity by contracting few edges. *Theor. Comput. Sci.* 476 (2013), 38–46.

Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. 2006. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Comput. Syst. Sci.* 72, 8 (2006), 1386–1396.

Peter L. Hammer and Stéphane Földes. 1977. Split graphs. *Congress. Numerant.* 19 (1977), 311–315.

David S. Johnson. 1974. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.* 9, 3 (1974), 256–278. DOI: https://doi.org/10.1016/S0022-0000(74)80044-9

Stasys Jukna. 2011. *Extremal Combinatorics*. Springer.

Iyad A. Kanj, Guohui Lin, Tian Liu, Weitian Tong, Ge Xia, Jinhui Xu, Boting Yang, Fenghui Zhang, Peng Zhang, and Binhai Zhu. 2014. Algorithms for cut problems on trees. In *Proceedings of the 8th International Conference on Combinatorial Optimization and Applications (COCOA'14) (Lecture Notes in Computer Science)*, Vol. 8881. Springer, 283–298.

Mamadou Moustapha Kanté, Eun Jung Kim, O.-joung Kwon, and Christophe Paul. 2015. An FPT algorithm and a polynomial kernel for linear rankwidth-1 vertex deletion. In *Proceedings of the 10th International Symposium on Parameterized and Exact Computation (IPEC'15). CoRR* 43, 138–150. Retrieved from http://arxiv.org/abs/1504.05905

Tomasz Kociumaka and Marcin Pilipczuk. 2014. Faster deterministic feedback vertex set. *Inf. Process. Lett.* 114, 10 (2014), 556–560.

Sudeshna Kolay and Fahad Panolan. 2015. Parameterized algorithms for deletion to $(r, \ell)$-graphs. In *Proceedings of the 35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS'15) (LIPIcs)*, Vol. 45. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 420–433.

Bernhard Korte and Jens Vygen. 2012. *Combinatorial Optimization*. Springer.

Mithilesh Kumar and Daniel Lokshtanov. 2016. Faster exact and parameterized algorithm for feedback vertex set in tournaments. In *Proceedings of the 33rd International Symposium on Theoretical Aspects of Computer Science (STACS'16) (LIPIcs)*, Vol. 47. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 49:1–49:13.

Konstantin Kutzkov and Dominik Scheder. 2010. Using CSP to improve deterministic 3-SAT. *CoRR* abs/1007.1166 (2010). Retrieved from http://arxiv.org/abs/1007.1166.

Nikolai N. Kuzjurin. 2000. Explicit constructions of Rödl's asymptotically good packings and coverings. *Combinat. Prob. Comput.* 9, 3 (2000), 265–276. Retrieved from http://journals.cambridge.org/action/displayAbstract?aid=54643.

Dániel Marx and Ildikó Schlotter. 2011. Stable assignment with couples: Parameterized complexity and local search. *Discrete Optim.* 8, 1 (2011), 25–40.

J. W. Moon. 1971. On maximal transitive subtournaments. *Proc. Edinburgh Math. Soc. (2)* 17 (1971), 345–349.

Robin A. Moser and Dominik Scheder. 2011. A full derandomization of Schöning's $k$-SAT algorithm. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC'11)*. ACM, 245–252.

Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. 1995. Splitters and near-optimal derandomization. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*. IEEE, 182–191.

Igor Razgon. 2006. Exact computation of maximum induced forest. In *Proceedings of the 10th Scandinavian Workshop on Algorithm Theory (SWAT'06) (Lecture Notes in Computer Science)*, Vol. 4059. Springer, Berlin, 160–171.

Igor Razgon. 2007. Computing minimum directed feedback vertex set in $O(1.9977^n)$. In *Proceedings of the 10th Italian Conference on Theoretical Computer Science (ICTCS'07)*. 70–81.

Uwe Schöning. 1999. A probabilistic algorithm for $k$-SAT and constraint satisfaction problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*. IEEE Computer Soc., Los Alamitos, CA, 410–414.

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

Pim van 't Hof and Yngve Villanger. 2013. Proper interval vertex deletion. *Algorithmica* 65, 4 (2013), 845–867. DOI: https://doi.org/10.1007/s00453-012-9661-3

Magnus Wahlström. 2007. *Algorithms, Measures, and Upper Bounds for Satisfiability and Related Problems.* Ph.D. Dissertation. Linköping University, Sweden.

Magnus Wahlström. 2014. Half-integrality, LP-branching and FPT algorithms. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*. SIAM, 1762–1781.

Mingyu Xiao and Hiroshi Nagamochi. 2013. Exact algorithms for maximum independent set. In *Proceedings of the 24th International Symposium on Algorithms and Computation (ISAAC'13). CoRR* 8283, 328–338. Retrieved from http://arxiv.org/abs/1312.6260See also arXiv CoRR abs/1312.6260.

Takayuki Yato and Takahiro Seta. 2003. Complexity and completeness of finding another solution and its application to puzzles. *IEICE Trans. Fund. Electron. Commun. Comput. Sci.* E86-A, 5 (2003), 1052–1060.

Received May 2016; revised September 2018; accepted September 2018