Exact Algorithms via Monotone Local Search

Fedor Fomin[*]   Serge Gaspers[†‡]   Daniel Lokshtanov[*]   Saket Saurabh[*§]

Abstract

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 $c^k n^{\mathcal{O}(1)}$ time algorithm for the extension problem immediately yields a randomized algorithm for finding a solution of any size with running time $\mathcal{O}((2 - \frac{1}{c})^n)$.

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.

---

[*]University of Bergen, Norway. `{fomin|daniello}@ii.uib.no`
[†]The University of New South Wales, Sydney, Australia. `sergeg@cse.unsw.edu.au`
[‡]Data61 (formerly: NICTA), CSIRO, Sydney, Australia
[§]The Institute of Mathematical Sciences, Chennai, India. `saket@imsc.res.in`.

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 on n elements, the brute-force algorithm that tries all possible solutions has running time $2^n n^{\mathcal{O}(1)}$. Thus our goal is typically to design an algorithm with running time $c^n n^{\mathcal{O}(1)}$ for $c < 2$, and we try to minimize the constant c. We refer to the textbook of Fomin and Kratsch [20] for an introduction to the field.

In the area of parameterized algorithms (see [12]), 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 $c^k n^{\mathcal{O}(1)}$ for a constant c, which is often much larger than 2.

In this paper 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 $c^k n^{\mathcal{O}(1)}$ for any $c > 1$ immediately implies an exact algorithm with running time $\mathcal{O}((2 - \frac{1}{c})^{n+o(n)})$ 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 $c^k n^{\mathcal{O}(1)}$ for $c < 4$ and exact exponential-time algorithms beating $\mathcal{O}(2^n)$ 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 $\mathcal{O}(2^n)$ 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 $\binom{n}{k}n^{\mathcal{O}(1)}$ which is significantly faster than $\mathcal{O}(2^n)$. Thus, if there is an algorithm deciding whether there is a solution of size at most k in time $c^k n^{\mathcal{O}(1)}$ for some $c < 4$, we can deduce that the problem can be solved in time $\mathcal{O}((2 - \epsilon)^n)$ for an $\epsilon > 0$ that depends only on c. On the other hand, 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 $\mathcal{O}(2^n)$ 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 [8] and runs in time $8^k n^{\mathcal{O}(1)}$. 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. [4], and runs in time $\mathcal{O}((2 - \epsilon)^n)$ for $\epsilon < 10^{-20}$. On the other hand, 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.

Our results. We need some definitions in order to state our results precisely. We define an *implicit*

1

set system as a function Φ 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 Φ 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 paper are polynomial time computable, except for the minimal satisfying assignments of d-CNF formulas which are not polynomial time computable unless P=NP [44].

An implicit set system Φ 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 paper we will primarily focus on the first and last problems. Examples of implicit sets 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.

Φ-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 Φ 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.

Our results will rely on parameterized algorithms for a generalization of subset problems where we are also given as input a set $X \subseteq U_I$ and an integer k and the question is whether it is possible to obtain a set from $\mathcal{F}_I$ by adding at most k elements from U_I to X. We give a formal definition of such problems, which we call *extension problems*.

Φ-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 Φ-SUBSET based on single-exponential parameterized algorithms for Φ-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 $> \frac{1}{2}$.

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

Our second main result is that the algorithm of Theorem 1 can be derandomized at the cost of a subexponential factor in n in the running time.

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

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 $d^k \cdot n^{\mathcal{O}(1)}$.

Hence, by Theorem 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 $2.562^k n^{\mathcal{O}(1)}$ due to Kutzkov and Scheder [31]. Thus Min-Ones 3-SAT can be solved in time $\mathcal{O}(1.6097^n)$. 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, our theorem implies faster 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 $2^n n^{\mathcal{O}(1)}$ 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 [36]. The running time $\mathcal{O}(1.8899^n)$ of the algorithm from [36] was improved in [18] to $\mathcal{O}(1.7548^n)$. Then Xiao and Nagamochi [43] gave an algorithm with running time $\mathcal{O}(1.7356^n)$. Finally an algorithm of running time $\mathcal{O}(1.7347^n)$ was obtained in [21]. For the parameterized version of the problem there was also a chain of improvements [9, 10, 15, 25] resulting in a $3^k n^{\mathcal{O}(1)}$ time randomized algorithm [13] and a $3.591^k n^{\mathcal{O}(1)}$ time deterministic algorithm [28]. This, coupled with our main theorem, immediately gives us randomized and deterministic algorithms of running times $\mathcal{O}(1.6667^n)$ and $\mathcal{O}(1.7216^n)$, respectively.

More generally, let Π 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 Π there is a natural vertex deletion problem, that 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 $|S| \le k$ 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 Π yields a polynomial time algorithm to determine whether a set S is in $\mathcal{F}_I$, implying that Φ is polynomial time computable. Moreover, a $c^k N^{\mathcal{O}(1)}$ 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 Π is characterized by a set of forbidden induced subgraphs which all have at most d vertices, such as cographs ($d = 4$) and split graphs ($d = 5$), 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 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 2. Some of the results (for Weighted

Problem Name	Parameterized		New bound	Previous Bound	
FEEDBACK VERTEX SET	3^k (r)	[13]	1.6667^n (r)		
FEEDBACK VERTEX SET	3.592^k	[28]	1.7217^n	1.7347^n	[21]
SUBSET FEEDBACK VERTEX SET	4^k	[42]	1.7500^n	1.8638^n	[19]
FEEDBACK VERTEX SET IN TOURNAMENTS	1.6181^k	[30]	1.3820^n	1.4656^n	[30]
GROUP FEEDBACK VERTEX SET	4^k	[42]	1.7500^n	NPR	
NODE UNIQUE LABEL COVER	$|\Sigma|^{2k}$	[42]	$(2 - \frac{1}{|\Sigma|^2})^n$	NPR	
VERTEX (r,ℓ)-PARTIZATION $(r,\ell \le 2)$	3.3146^k	[3, 29]	1.6984^n	NPR	
INTERVAL VERTEX DELETION	8^k	[8]	1.8750^n	$(2 - \varepsilon)^n$ for $\varepsilon < 10^{-20}$	[4]
PROPER INTERVAL VERTEX DELETION	6^k	[40, 7]	1.8334^n	$(2 - \varepsilon)^n$ for $\varepsilon < 10^{-20}$	[4]
BLOCK GRAPH VERTEX DELETION	4^k	[1]	1.7500^n	$(2 - \varepsilon)^n$ for $\varepsilon < 10^{-20}$	[4]
CLUSTER VERTEX DELETION	1.9102^k	[5]	1.4765^n	1.6181^n	[17]
THREAD GRAPH VERTEX DELETION	8^k	[27]	1.8750^n	NPR	
MULTICUT ON TREES	1.5538^k	[26]	1.3565^n	NPR	
3-HITTING SET	2.0755^k	[41]	1.5182^n	1.6278^n	[41]
4-HITTING SET	3.0755^k	[17]	1.6750^n	1.8704^n	[17]
d-HITTING SET $(d \ge 3)$	$(d - 0.9245)^k$	[17]	$(2 - \frac{1}{(d-0.9245)})^n$		[11, 17]
MIN-ONES 3-SAT	2.562^k	[31]	1.6097^n	NPR	
MIN-ONES d-SAT $(d \ge 4)$	d^k		$(2 - \frac{1}{d})^n$	NPR	
WEIGHTED d-SAT $(d \ge 3)$	d^k		$(2 - \frac{1}{d})^n$	NPR	
WEIGHTED FEEDBACK VERTEX SET	3.6181^k	[1]	1.7237^n	1.8638^n	[18]
WEIGHTED 3-HITTING SET	2.168^k	[39]	1.5388^n	1.6755^n	[11]
WEIGHTED d-HITTING SET $(d \ge 4)$	$(d - 0.832)^k$	[17, 39]	$(2 - \frac{1}{d-0.932})^n$		[11]

Table 1: Summary of known and new results for different optimization problems. 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).

d-HITTING SET) follow from a variant that relies on algorithms for a permissive version of Φ-EXTENSION; see Subsection 2.3.

We also extend the technique developed for decision problems 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. 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 Φ be an implicit set system and $c \ge 1$ be a real-valued constant. We say that Φ is *c-uniform* if, for every instance I, set $X \subseteq U_I$, and integer $k \le n - |X|$, 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 $c^k n^{\mathcal{O}(1)}$. The next theorem will provide new combinatorial upper bounds for collections generated by c-uniform implicit set systems.

Theorem 3. *Let $c > 1$ and Φ be an implicit set system. If Φ is c-uniform, then $|\mathcal{F}_I| \le \left(2 - \frac{1}{c}\right)^n n^{\mathcal{O}(1)}$ for every instance I.*

We say that an implicit set system Φ 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 $c^k N^{\mathcal{O}(1)}$. In this case, we can enumerate $\mathcal{F}_I$ in the same time, up to a subexponential factor in n.

Problem Name	c-uniform	New bound	Previous Bound
Minimal FVSs in Tournaments	3	1.6667^n	1.6740^n [23]
Minimal 3-Hitting Sets	3	1.6667^n	1.6755^n [11]
Minimal 4-Hitting Sets	4	1.7500^n	1.8863^n [11]
Minimal 5-Hitting Sets	5	1.8000^n	1.9538^n [11]
Minimal d-Hitting Sets	d	$(2-\frac{1}{d})^n$	$(2-\epsilon_d)^n$ with $\epsilon_d < 1/d$ [11]
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 [24]
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

Table 2: Summary of known and new results for different combinatorial bounds. 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.

Theorem 4. *Let $c > 1$ and Φ be an implicit set system. If Φ is efficiently c-uniform, then there is an algorithm that given as input I enumerates $\mathcal{F}_I$ in time $\left(2-\frac{1}{c}\right)^{n+o(n)} N^{\mathcal{O}(1)}$.*

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

By a classical theorem of Moon from 1971 [33], the number of maximal transitive subtournaments in an n-vertex tournament does not exceed 1.7170^n. In [23], Gaspers and Mnich improved this bound to 1.6740^n. Our approach yields immediately a better bound of $\mathcal{O}(1.6667^n)$ since every directed 3-cycle needs to be hit. Similarly, in chordal graphs, a set is a feedback vertex set (FVS) 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 paper Schöning [38] 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 $\mathcal{O}((2-2/d)^n)$ and it was shown by Moser and Scheder [34] how to derandomize it in almost the same running time, see also [14].

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, 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 $d^r \cdot n^{\mathcal{O}(1)}$. 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 $f(r) \cdot n^{\mathcal{O}(1)}$ for any function f would imply that $\mathsf{FPT} = \mathsf{W}[1]$. 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 [16].

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 $c^{k-t} \cdot n^{\mathcal{O}(1)}$. Another option is to pick a *random* vertex x from $U \setminus X$, add x to X and then proceed. We succeed if x is in $S \setminus X$, so the probability of success is $(k - t)/(n - t)$. 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 is simple, the analysis is a bit more involved. At a first glance it is not at all obvious that a $100^k \cdot n^{\mathcal{O}(1)}$ 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

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

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 $p = \binom{k}{t}/\binom{n}{t}$. In order to get constant success probability, we run the algorithm $1/p$ times, taking time $c^{k-t} \cdot n^{\mathcal{O}(1)}$ for each run.

In order 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 $\mathcal{F}$ lends ideas from Naor et al. [35], however their methods are not directly applicable to our setting.

The main technical contribution of this paper 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 1–4, that 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 and 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. The theorem will follow from the following lemma, which gives a new randomized algorithm for Φ-EXTENSION.

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

Proof. Let $\mathcal{B}$ be an algorithm for Φ-EXTENSION with running time $c^k N^{\mathcal{O}(1)}$. 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 $\mathcal{A}$. Its running time is clearly upper bounded by $c^{k'-t} N^{\mathcal{O}(1)}$.

If $\mathcal{A}$ returns yes for (I, X, k'), 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

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

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 $\min_{k' \leq k}\{1 - (1 - p(k'))^{1/p(k')}\} \geq 1 - \frac{1}{e} > \frac{1}{2}$. Next we upper bound the running time of the main algorithm, which is

$$\sum_{k' \leq k} \frac{1}{p(k')} \cdot c^{k'-t} N^{\mathcal{O}(1)} \leq \max_{k' \leq k} \frac{\binom{n-|X|}{t}}{\binom{k'}{t}} \cdot c^{k'-t} N^{\mathcal{O}(1)} \leq \max_{k \leq n-|X|} \frac{\binom{n-|X|}{t}}{\binom{k}{t}} \cdot c^{k-t} N^{\mathcal{O}(1)}.$$

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

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

We upper bound the expression in (1) by $\left(2 - \frac{1}{c}\right)^{n-|X|} N^{\mathcal{O}(1)}$ in Lemma 2.2. The running time of the algorithm is thus upper bounded by $\left(2 - \frac{1}{c}\right)^{n-|X|} N^{\mathcal{O}(1)}$, completing the proof. $\qquad\square$

Remark 1. 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 2.2. *Let $c > 1$ be a fixed constant, and let n and $k \leq n$ be non-negative integers. Then,*

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

Proof. Setting $\mu = \frac{k}{n}$ and $\alpha = \frac{t}{n}$, we have that

$$\max_{0 \leq k \leq n} \left\{ \min_{0 \leq t \leq k} \left\{ \frac{\binom{n}{t}}{\binom{k}{t}} c^{k-t} \right\} \right\} = \max_{0 \leq \mu \leq 1} \left\{ \min_{0 \leq \alpha \leq \mu} \left\{ \frac{\binom{n}{\lceil \alpha n \rceil}}{\binom{\lceil \mu n \rceil}{\lceil \alpha n \rceil}} c^{(\mu-\alpha)n} \right\} \right\} \cdot \mathcal{O}(1)$$

The right hand side of the equation above is upper bounded by picking a concrete value of α for every value of μ, rather than minimizing over all α. We set $\alpha = \max\left(0, \frac{1-c\mu}{1-c}\right)$. One can show that this choice of α minimizes the internal expression. In particular, this basically guarantees that $\frac{k-t}{n-t} = \frac{1}{c}$, and this is the natural threshold for when to stop sampling (see the discussion in the introduction).

First, consider the case where $\alpha = 0$. The expression is upper bounded by $c^{n/c}$ since $1 - c\mu \geq 0$. To show that $\left(2 - \frac{1}{c}\right)^n \geq c^{n/c}$, it suffices to show that $2 - \frac{1}{c} - c^{1/c} \geq 0$. But this is so because $2 - \frac{1}{c} - c^{1/c} = 0$ when $c = 1$ and it is increasing with c when $c > 1$.

From now on, we assume $\alpha = \frac{1-c\mu}{1-c} > 0$. Thus,

$$\max_{0 \leq \mu \leq 1} \left\{ \min_{0 \leq \alpha \leq \mu} \left\{ \frac{\binom{n}{\lceil \alpha n \rceil}}{\binom{\lceil \mu n \rceil}{\lceil \alpha n \rceil}} c^{(\mu-\alpha)n} \right\} \right\} \leq \max_{\substack{0 \leq \mu \leq 1 \\ \alpha = \frac{1-c\mu}{1-c}}} \left\{ \frac{\binom{n}{\lceil \alpha n \rceil}}{\binom{\lceil \mu n \rceil}{\lceil \alpha n \rceil}} c^{(\mu-\alpha)n} \right\}. \tag{2}$$

We will also use the following well known bounds on binomial coefficients to simplify our expressions,

$$\frac{1}{n^{\mathcal{O}(1)}}\left[\left(\frac{k}{n}\right)^{-\frac{k}{n}}\left(1-\frac{k}{n}\right)^{\frac{k}{n}-1}\right]^n \le \binom{n}{k} \le \left[\left(\frac{k}{n}\right)^{-\frac{k}{n}}\left(1-\frac{k}{n}\right)^{\frac{k}{n}-1}\right]^n. \tag{3}$$

Using the upper bound in Equation 2 we obtain the following.

$$\begin{aligned}
\frac{\binom{n}{\lceil \alpha n \rceil}}{\binom{\lceil \mu n \rceil}{\lceil \alpha n \rceil}} c^{(\mu-\alpha)n} &\le \left(\frac{\alpha^{-\alpha}(1-\alpha)^{\alpha-1}}{(\frac{\alpha}{\mu})^{-\alpha}(1-\frac{\alpha}{\mu})^{\alpha-\mu}}c^{(\mu-\alpha)}\right)^n n^{\mathcal{O}(1)} \\
&= \left(\frac{\alpha^{-\alpha}(1-\alpha)^{\alpha-1}}{\alpha^{-\alpha}\mu^{\alpha}(\mu-\alpha)^{\alpha-\mu}\mu^{(\mu-\alpha)}}c^{(\mu-\alpha)}\right)^n n^{\mathcal{O}(1)} \\
&= \left((1-\alpha)^{\alpha-1}(\mu-\alpha)^{\mu-\alpha}\mu^{-\mu}c^{(\mu-\alpha)}\right)^n n^{\mathcal{O}(1)} \tag{4}
\end{aligned}$$

Substituting the value of α in Equation 4, we get the following as the base of the exponent.

$$\begin{aligned}
&\left(\frac{c(1-\mu)}{c-1}\right)^{\frac{c(1-\mu)}{1-c}}\left(\frac{1-\mu}{c-1}\right)^{\frac{1-\mu}{c-1}}\mu^{-\mu}c^{\frac{1-\mu}{c-1}} \\
&= \left(\frac{c}{c-1}\right)^{\frac{c(1-\mu)}{1-c}+\frac{1-\mu}{c-1}}(1-\mu)^{\frac{c(1-\mu)}{1-c}+\frac{1-\mu}{c-1}}\mu^{-\mu} \\
&= \left(\frac{c}{c-1}\right)^{\mu-1}(1-\mu)^{\mu-1}\mu^{-\mu} \tag{5}
\end{aligned}$$

The last assertion in Equation 5 follows from the following simplification.

$$\frac{c(1-\mu)}{1-c}+\frac{1-\mu}{c-1}=\frac{c(1-\mu)-(1-\mu)}{1-c}=\frac{(c-1)(1-\mu)}{1-c}=\mu-1.$$

To summarize the above discussion, we have upper bounded the expression in the statement of the lemma by

$$\max_{0\le k\le n}\left\{\min_{0\le t\le k}\left\{\frac{\binom{n}{t}}{\binom{k}{t}}c^{k-t}\right\}\right\} \le \left[\max_{0\le \mu\le 1}f(\mu)\right]^n \cdot n^{\mathcal{O}(1)},$$

where $f(\mu)=\left(\frac{c}{c-1}\right)^{\mu-1}(1-\mu)^{\mu-1}\mu^{-\mu}$.

We now turn to upper bounding the maximum of $f(\mu)$. Clearly, f is continuous and differentiable on the interval $[0,1]$ and thus f achieves its maximum at $\mu \in \{0,1\}$ or at a point where the derivative vanishes. Setting μ to 0 and to 1, we get that $f(0)=\frac{c-1}{c}$ and $f(1)=1$, respectively. Next we differentiate f with respect to μ. The product rule for differentiation yields

$$f'(\mu) = f(\mu)\cdot\left(\ln\left(\frac{c}{c-1}\right)+(\ln(1-\mu)+1)-1-\ln\mu\right).$$

Since $f(\mu)\ne 0$, we have that $f'(\mu)=0$ if and only if

$$\ln\left(\frac{c}{c-1}\right)+\ln(1-\mu)-\ln\mu=0,$$

and the unique solution to this equation is $\mu=\frac{c}{2c-1}$. Substituting this value of $\mu=\frac{c}{2c-1}$ in f we get $f(\mu)=2-\frac{1}{c}$. Thus the maximum value f can attain on $\mu\in[0,1]$ is the maximum of 1, $1-\frac{1}{c}$ and $2-\frac{1}{c}$. Since, $c>1$ this implies that $f(\mu)\le 2-\frac{1}{c}$, completing the proof. $\qquad\square$

By running the algorithm from Lemma 2.1 with $X=\emptyset$ and for each value of $k\in\{0,\dots,n\}$, we obtain an algorithm for Φ-SUBSET, and this proves Theorem 1.

2.2 Derandomization

In this subsection we prove Theorem 2 by derandomizing the algorithm of Theorem 1, at the cost of a subexponential factor in the running time. 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 2.1. *Let U be a universe of size n and let $0 \le q \le p \le n$. 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 $\kappa(n, p, q) = \binom{n}{q}/\binom{p}{q}$. In Section 3 (Theorem 6) 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 $\kappa(n, p, q) \cdot 2^{o(n)}$.

The proof of Theorem 2 is now almost identical to the proof of Theorem 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 6. 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, our results rely on algorithms for permissive variants of the Φ-EX-TENSION problem. Permissive problems were introduced in the context of local search algorithms [32] 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 [22].

PERMISSIVE Φ-EXTENSION
Input: An instance I, a set $X \subseteq U_I$, and an integer k.
Output: If there is a subset $S \subseteq (U_I \setminus X)$ such that $S \cup X \in \mathcal{F}_I$ and $|S| \le k$, 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 2.3. *If there exists a constant $c > 1$ and an algorithm for PERMISSIVE Φ-EXTENSION with running time $c^k N^{\mathcal{O}(1)}$, then there exists a randomized algorithm for PERMISSIVE Φ-EXTENSION with running time $(2 - \frac{1}{c})^{n - |X|} N^{\mathcal{O}(1)}$.*

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, 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 5. *If there is an algorithm for* PERMISSIVE Φ-EXTENSION *with running time $c^k N^{\mathcal{O}(1)}$ then there is an algorithm for* DECISION Φ-SUBSET *with running time $(2 - \frac{1}{c})^{n+o(n)} N^{\mathcal{O}(1)}$. Moreover, if the algorithm for* PERMISSIVE Φ-EXTENSION *computes a certificate whenever it answers yes, then there is an algorithm for Φ-SUBSET with running time $(2 - \frac{1}{c})^{n+o(n)} N^{\mathcal{O}(1)}$.*

2.4 Enumeration and Combinatorial Upper Bounds

In this subsection, we prove Theorems 3 and 4 on combinatorial upper bounds and enumeration algorithms.

Theorem 3. *Let $c > 1$ and Φ be an implicit set system. If Φ is c-uniform, then $|\mathcal{F}_I| \le \left(2 - \frac{1}{c}\right)^n n^{\mathcal{O}(1)}$ for every instance I.*

Proof. Let I be an instance and $k \le n$. We prove that the number of sets in $\mathcal{F}_I$ of size exactly k is upper bounded by $\left(2 - \frac{1}{c}\right)^n n^{\mathcal{O}(1)}$. 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.

$$
\begin{aligned}
\Pr[E_Z] &= \Pr[X \subseteq Z \wedge S = Z \setminus X] \\
&= \Pr[X \subseteq Z] \times \Pr[S = Z \setminus X \mid X \subseteq Z] \\
&= \frac{\binom{k}{t}}{\binom{n}{t}} \times \frac{1}{|\mathcal{F}_{I,X}^{k-t}|}
\end{aligned}
\tag{6}
$$

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

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

We are now ready to discuss the choice of t in the random process. The integer t is chosen such that the above expression for $\Pr[E_Z]$ is maximized (or, in other words, it's reciprocal is minimized). By Lemma 2.2 we have that for every $k \le n$ there exists a $t \le k$ such that

$$
\frac{\binom{k}{t}}{\binom{n}{t}} c^{-(k-t)} \ge (2 - \frac{1}{c})^{-n} \cdot n^{-\mathcal{O}(1)}.
$$

Hence $\Pr[E_Z] \ge (2 - \frac{1}{c})^{-n} \cdot n^{-\mathcal{O}(1)}$ 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

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

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 $\left(2 - \frac{1}{c}\right)^n n^{\mathcal{O}(1)}$, completing the proof. $\square$

If the implicit set system Φ is efficiently c-uniform then the proof of Theorem 3 can be made constructive by replacing the sampling step by a construction of an (n, k, t)-set-inclusion-family $\mathcal{C}$ using Theorem 6. For each $X \in \mathcal{C}$ the algorithm uses the fact that Φ 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 $\mathcal{C}$ instead of sampling X incurs a $2^{o(n)}$ overhead in the running time of the algorithm. In order 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 4. *Let $c > 1$ and Φ be an implicit set system. If Φ is efficiently c-uniform, then there is an algorithm that given as input I enumerates $\mathcal{F}_I$ in time $\left(2 - \frac{1}{c}\right)^{n+o(n)} N^{\mathcal{O}(1)}$.*

3 Efficient Construction of Set-Inclusion-Families

In this section we give the promised construction of set-inclusion-families. 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 $\kappa(n, p, q) = \binom{n}{q}/\binom{p}{q}$.

Lemma 3.1. *There is an algorithm that given n, p and q outputs an (n, p, q)-set-inclusion-family $\mathcal{C}$ of size at most $\kappa(n, p, q) \cdot n^{\mathcal{O}(1)}$ in time $\mathcal{O}(3^n)$.*

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

By construction, the size of $\mathcal{C}$ is within the required bounds. We now argue that with positive probability $\mathcal{C}$ 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 $\mathcal{C}$, 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 $C_i \in \mathcal{C}$ satisfies $C_i \subseteq A$ is upper bounded by $\frac{1}{n^{p+1}} \cdot n^p = \frac{1}{n}$.

To construct $\mathcal{C}$ 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 $\mathcal{S}$ of subsets of $\mathcal{U}$. The objective is to find a minimum sized sub-collection $\mathcal{S}' \subseteq \mathcal{S}$ such that the union of elements of the sets in $\mathcal{S}'$ is $\mathcal{U}$. It is known that SET COVER admits a polynomial time approximation algorithm with factor $\mathcal{O}(\log |\mathcal{U}|)$. 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 $\mathcal{S}$ contains F_B for each choice of $B \in \binom{U}{q}$. Given a sub-collection $\mathcal{S}' \subseteq \mathcal{S}$ we construct the family $\mathcal{C}(\mathcal{S}')$ by taking the sets $B \in \binom{U}{q}$ such that $F_B \in \mathcal{S}'$. Clearly, any $\mathcal{C}(\mathcal{S}')$ corresponding to a sub-collection $\mathcal{S}' \subseteq \mathcal{S}$ 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 $\mathcal{S}' \subseteq \mathcal{S}$ covering $\mathcal{U}$. We run the known $\mathcal{O}(\log |\mathcal{U}|)$-factor approximation algorithm on our instance and obtain a sub-collection $\mathcal{S}' \subseteq \mathcal{S}$

covering $\mathcal{U}$. Let $\mathcal{C} = \mathcal{C}(\mathcal{S}')$. By discussions above we know that $\mathcal{C}$ is an (n, p, q)-set-inclusion-family. Clearly, the size of $\mathcal{C}$ is upper bounded by

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

It is well known that one can implement the approximation algorithm for SET COVER to run in time $\mathcal{O}(|\mathcal{U}| + \sum_{S \in \mathcal{S}} |S|) = \mathcal{O}(\binom{n}{p} + \binom{n}{q}\binom{n-q}{p-q}) = \mathcal{O}(3^n)$. This concludes the proof. $\qquad\square$

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 3.1 ([2]). *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 $\mathcal{O}(n^2)$.*

Using Proposition 3.1 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.

Theorem 6. *There is an algorithm that given n, p and q outputs an (n, p, q)-set-inclusion-family $\mathcal{C}$ 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 3.1 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 $\mathcal{C}_f^{(i,s)}$ using Lemma 3.1. We now describe the family $\mathcal{C}$ output by the construction. Each set $Y \in \mathcal{C}$ 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 $Y_i \in \mathcal{C}_f^{(i, p_i)}$,
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.

First we analyze the running time of the construction. Constructing the set $\mathcal{X}$ takes polynomial time by Proposition 3.1. For each good f, $i \in [b]$ and $s \leq n_f^i$, constructing $\mathcal{C}_f^{(i,s)}$ using Lemma 3.1 takes time $2^{o(n)}$ because $n_i \leq \frac{n}{b} + \sqrt{n} \cdot \log n = \mathcal{O}(\frac{n}{\log n})$. There are $\mathcal{O}(n^2)$ choices for f, at most $\mathcal{O}(\log n)$ choices for i and $\mathcal{O}(\frac{n}{\log n})$ choices for s. Thus, the overall time of the construction is $2^{o(n)}$ plus the time to output $\mathcal{C}$. Outputting $\mathcal{C}$ can be done spending polynomial time for each set

13

$Y \in \mathcal{C}$ 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 $2^{o(n)} + |\mathcal{C}| \cdot n^{\mathcal{O}(1)}$. It remains to upper bound $|\mathcal{C}|$.

The size of $\mathcal{C}$ is upper bounded by the number of tuples $(f, p_1, \ldots, p_b, Y_1, \ldots, Y_b, D)$. There are $\mathcal{O}(n^2)$ choices for f, at most n^b choices for $p_1, \ldots, p_b$ and $n^{\mathcal{O}(b)}$ 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 $\mathcal{C}_f^{(i,p_i)}$, so there are $\kappa(n_f^i, p_i, \lceil \beta p_i \rceil) \cdot n^{\mathcal{O}(1)}$ choices for Y_i. It follows that the total number of choices for $Y_1, \ldots, Y_b$ is upper bounded by

$$\prod_{i \le b} \kappa(n_f^i, p_i, \lceil \beta p_i \rceil) \cdot n^{\mathcal{O}(1)} \le 2^{o(n)} \cdot \prod_{i \le b} \frac{\binom{n_f^i}{\lceil \beta p_i \rceil}}{\binom{p_i}{\lceil \beta p_i \rceil}}. \tag{7}$$

Now, we have that

$$\prod_{i \le b} \binom{n_f^i}{\lceil \beta p_i \rceil} \le \prod_{i \le b} \left[\binom{\lceil n/b \rceil}{\lceil p/b \rceil} \cdot n^{\mathcal{O}(\sqrt{n} \log n)} \right] \le \binom{n}{p} \cdot 2^{o(n)} \tag{8}$$

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,

$$
\begin{aligned}
\prod_{i \le b} \binom{p^i}{\lceil \beta p_i \rceil} &\ge \prod_{i \le b} \left[\binom{\lceil p/b \rceil}{\lceil \beta(p/b) \rceil} \cdot n^{-\mathcal{O}(\sqrt{n} \log n)} \right] \\
&\ge \left[\left(\beta^{-\beta}(1-\beta)^{\beta-1} \right)^{(p/b)} \cdot n^{-\mathcal{O}(1)} \right]^b \cdot 2^{-o(n)} \\
&\ge \binom{p}{q} \cdot 2^{-o(n)}
\end{aligned}
\tag{9}
$$

Here the two last transitions use Equation 3. Inserting the bounds from (8) and (9) into (7) 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, $|\mathcal{C}| \le \kappa(n, p, q) \cdot 2^{o(n)}$ as well.

All that remains is to argue that $\mathcal{C}$ 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

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

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

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

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 above 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

14

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

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

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 $i \leq b$, let $S_i = |U_f^i \cap S|$ and $p_i = |S_i|$. Again, by the choice of f we have that $|p_i - \frac{p}{b}| \leq \sqrt{n} \cdot b$. Since $\mathcal{C}_f^{(i,p_i)}$ is an $(n_f^i, p_i, \lceil \beta p_i \rceil)$-set-inclusion-family, there exists a set $Y_i \in \mathcal{C}_f^{(i,p_i)}$ 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 $\mathcal{C}_f^{(i,p_i)}$. Finally let D be any subset of $\bigcup_{i \leq b} Y_i$ of size $\sum_{i \leq b} |Y_i| - q$. Note that $|Y_i| \leq \beta p_i + 1$, thus $\sum_{i \leq b} |Y_i| - q \leq b$, so $|D| \leq b$. Consider finally the tuple $(f, p_1 \ldots p_b, Y_1, \ldots Y_b, D)$. 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 $\mathcal{C}$. However, $Y_i \subseteq S_i$ for all i, so $Y \subseteq S$, proving that $\mathcal{C}$ is a (n, p, q)-set-inclusion-family. $\qquad \square$

4 Conclusion and Discusison

In this paper we have shown that for many subset problems, an algorithm that finds a solution of size k in time $c^k n^{\mathcal{O}(1)}$ directly implies an algorithm with running time $\mathcal{O}((2 - \frac{1}{c})^{n+o(n)})$. We also show that often, an upper bound of $c^k n^{\mathcal{O}(1)}$ on the number of sets of size at most k in a family $\mathcal{F}$ can yield an upper bound of $\mathcal{O}((2 - \frac{1}{c})^{n+o(n)})$ 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 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 the algorithm needs exponential space in order to store a trie 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 [23], and its running time is $O(1.6667^n)$ by our combinatorial upper bound.

Our analysis also reveals that in order to obtain a $(2 - \epsilon)^n$ time algorithm with $\epsilon > 1$ for a subset problem, it is sufficient to get a $\mathcal{O}(c^k \binom{n-|X|}{k}^{1-\delta})$ 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 $\mathcal{O}(1.9977^n)$ time algorithm [37] for DIRECTED FEEDBACK VERTEX SET by designing a $\mathcal{O}(c^k \binom{n-|X|}{k}^{1-\delta})$ time algorithm.

References

[1] A. Agrawal, D. Lokshtanov, S. Kolay, and S. Saurabh, *A faster FPT algorithm and a smaller kernel for block graph vertex deletion*, in Proceedings of the 12th Latin American Theoretical Informatics Symposium (LATIN 2016), Lecture Notes in Computer Science, Springer, to appear. Available as arXiv CoRR abs/1510.08154. 4

[2] N. Alon, L. Babai, and A. Itai, *A fast and simple randomized parallel algorithm for the maximal independent set problem*, J. Algorithms, 7 (1986), pp. 567–583. 13

[3] J. Baste, L. Faria, S. Klein, and I. Sau, *Parameterized complexity dichotomy for* (r, ℓ)-*vertex deletion*, Tech. Report abs/1504.05515, arXiv CoRR, 2015. 4

[4] I. Bliznets, F. V. Fomin, M. Pilipczuk, and Y. Villanger, *Largest chordal and interval subgraphs faster than* 2^n, in Proceedings of the 21st Annual European Symposium on Algorithms (ESA), vol. 8125 of Lecture Notes in Comput. Sci., Springer, 2013, pp. 193–204. 1, 4

[5] A. Boral, M. Cygan, T. Kociumaka, and M. Pilipczuk, *A fast branching algorithm for cluster vertex deletion*, in Proceedings of the 9th International Computer Science Symposium in Russia (CSR), vol. 8476 of Lecture Notes in Computer Science, Springer, 2014, pp. 111–124. 4

[6] A. Brandstädt, V. B. Le, and J. P. Spinrad, *Graph Classes: A Survey*, Discrete Mathematics and Applications, SIAM, 1999. 21

[7] Y. Cao, *Unit interval editing is fixed-parameter tractable*, in Proceedings of the 42nd International Colloquium of Automata, Languages and Programming (ICALP), vol. 9134 of Lecture Notes in Comput. Sci., Springer, 2015, pp. 306–317. 4

[8] Y. Cao, *Linear recognition of almost interval graphs*, in Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016), to appear. Available as arXiv CoRR abs/1403.1515. 1, 4

[9] Y. Cao, J. Chen, and Y. Liu, *On feedback vertex set: New measure and new structures*, Algorithmica, 73 (2015), pp. 63–86. 3

[10] J. Chen, F. V. Fomin, Y. Liu, S. Lu, and Y. Villanger, *Improved algorithms for feedback vertex set problems*, J. Computer and System Sciences, 74 (2008), pp. 1188–1198. 3

[11] M. Cochefert, J.-F. Couturier, S. Gaspers, and D. Kratsch, *Faster algorithms to enumerate hypergraph transversals*, in Proceedings of the 12th Latin American Theoretical Informatics Symposium (LATIN 2016), Lecture Notes in Computer Science, Springer, to appear. Available as arXiv CoRR abs/1510.05093. 4, 5

[12] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer, 2015. 1

[13] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. M. M. van Rooij, and J. O. Wojtaszczyk, *Solving connectivity problems parameterized by treewidth in single exponential time*, in Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS), IEEE, 2011, pp. 150–159. 3, 4

[14] E. Dantsin, A. Goerdt, E. A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, and U. Schöning, *A deterministic $(2 - 2/(k + 1))^n$ algorithm for k-SAT based on local search*, Theoretical Computer Science, 289 (2002), pp. 69–83. 5

[15] F. K. H. A. Dehne, M. R. Fellows, M. A. Langston, F. A. Rosamond, and K. Stevens, *An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem*, Theory of Computing Systems, 41 (2007), pp. 479–492. 3

[16] M. R. Fellows, F. V. Fomin, D. Lokshtanov, F. A. Rosamond, S. Saurabh, and Y. Villanger, *Local search: Is brute-force avoidable?*, J. Comput. Syst. Sci., 78 (2012), pp. 707–719. 6

[17] F. V. Fomin, S. Gaspers, D. Kratsch, M. Liedloff, and S. Saurabh, *Iterative compression and exact algorithms*, Theor. Comput. Sci., 411 (2010), pp. 1045–1053. 4

[18] F. V. Fomin, S. Gaspers, A. V. Pyatkin, and I. Razgon, *On the minimum feedback vertex set problem: Exact and enumeration algorithms*, Algorithmica, 52 (2008), pp. 293–307. 3, 4

[19] F. V. Fomin, P. Heggernes, D. Kratsch, C. Papadopoulos, and Y. Villanger, *Enumerating minimal subset feedback vertex sets*, Algorithmica, 69 (2014), pp. 216–231. 4

[20] F. V. Fomin and D. Kratsch, *Exact Exponential Algorithms*, Springer, 2010. An EATCS Series: Texts in Theoretical Computer Science. 1

[21] F. V. Fomin, I. Todinca, and Y. Villanger, *Large induced subgraphs via triangulations and CMSO*, SIAM J. Comput., 44 (2015), pp. 54–87. 3, 4

[22] S. Gaspers, E. J. Kim, S. Ordyniak, S. Saurabh, and S. Szeider, *Don't be strict in local search!*, in Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI), AAAI Press, 2012. 10

[23] S. Gaspers and M. Mnich, *Feedback vertex sets in tournaments*, Journal of Graph Theory, 72 (2013), pp. 72–89. 5, 15

[24] P. A. Golovach, P. van 't Hof, and D. Paulusma, *Obtaining planarity by contracting few edges*, Theoretical Computer Science, 476 (2013), pp. 38–46. 5

[25] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke, *Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization*, J. Computer and System Sciences, 72 (2006), pp. 1386–1396. 3

[26] I. A. Kanj, G. Lin, T. Liu, W. Tong, G. Xia, J. Xu, B. Yang, F. Zhang, P. Zhang, and B. Zhu, *Algorithms for cut problems on trees*, in Proceedings of the 8th International Conference on Combinatorial Optimization and Applications (COCOA), vol. 8881 of Lecture Notes in Computer Science, Springer, 2014, pp. 283–298. 4

[27] M. M. KANTÉ, E. J. KIM, O. KWON, AND C. PAUL, *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 2015), vol. 43 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pp. 138–150. 4

[28] T. KOCIUMAKA AND M. PILIPCZUK, *Faster deterministic feedback vertex set*, Inf. Process. Lett., 114 (2014), pp. 556–560. 3, 4

[29] S. KOLAY AND F. PANOLAN, *Parameterized algorithms for deletion to (r, ℓ)-graphs*, in Proceedings of FSTTCS 2015, to appear. Available as arXiv CoRR abs/1504.08120. 4

[30] M. KUMAR AND D. LOKSHTANOV, *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 2016), to appear. Available as arXiv CoRR abs/1510.07676. 4

[31] K. KUTZKOV AND D. SCHEDER, *Using CSP to improve deterministic 3-SAT*, CoRR, abs/1007.1166 (2010). 3, 4

[32] D. MARX AND I. SCHLOTTER, *Stable assignment with couples: Parameterized complexity and local search*, Discrete Optimization, 8 (2011), pp. 25–40. 10

[33] J. W. MOON, *On maximal transitive subtournaments*, Proc. Edinburgh Math. Soc. (2), 17 (1971), pp. 345–349. 5

[34] R. A. MOSER AND D. SCHEDER, *A full derandomization of Schöning's k-SAT algorithm*, in Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC), ACM, 2011, pp. 245–252. 5

[35] M. NAOR, L. J. SCHULMAN, AND A. SRINIVASAN, *Splitters and near-optimal derandomization*, in Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS), IEEE, 1995, pp. 182–191. 7

[36] I. RAZGON, *Exact computation of maximum induced forest*, in Proceedings of the 10th Scandinavian Workshop on Algorithm Theory (SWAT 2006), vol. 4059 of Lecture Notes in Comput. Sci., Berlin, 2006, Springer, pp. 160–171. 3

[37] ——, *Computing minimum directed feedback vertex set in $o(1.9977^n)$*, in Proceedings of the 10th Italian Conference on Theoretical Computer Science (ICTCS), 2007, pp. 70–81. 15

[38] U. SCHÖNING, *A probabilistic algorithm for k-SAT and constraint satisfaction problems*, in 40th Annual Symposium on Foundations of Computer Science (New York, 1999), IEEE Computer Soc., Los Alamitos, CA, 1999, pp. 410–414. 5

[39] H. SHACHNAI AND M. ZEHAVI, *A multivariate approach for weighted FPT algorithms*, in Proceedigs of the 23rd Annual European Symposium on Algorithms (ESA), vol. 9294 of Lecture Notes in Computer Science, Springer, 2015, pp. 965–976. 4

[40] P. VAN 'T HOF AND Y. VILLANGER, *Proper interval vertex deletion*, Algorithmica, 65 (2013), pp. 845–867. 4

[41] M. WAHLSTRÖM, *Algorithms, measures and upper bounds for satisfiability and related problems*, PhD thesis, Linköping University, Sweden, 2007. 4

[42] ——, *Half-integrality, LP-branching and FPT algorithms*, in Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2014, pp. 1762–1781. 4

[43] M. XIAO AND H. NAGAMOCHI, *Exact algorithms for maximum independent set*, in Proceedings of the 24th International Symposium on Algorithms and Computation (ISAAC), vol. 8283 of Lecture Notes in Computer Science, Springer, 2013, pp. 328–338. See also arXiv CoRR abs/1312.6260. 3, 20

[44] T. YATO AND T. SETA, *Complexity and completeness of finding another solution and its application to puzzles*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E86-A (2003), pp. 1052–1060. 2

A   Problem definitions

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

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 Γ be a finite group with identity element 1_Γ. A Γ-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 Γ, a Γ-labelled graph (G, λ), 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 Σ, an integer k, and for each edge $e \in E$ and each of its endpoints v a permutation $\psi_{e,v}$ of Σ 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, ℓ)-*graph* if the vertex set $V(G)$ can be partitioned into r independent sets and ℓ cliques.

VERTEX (r, ℓ)-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, ℓ)-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, ℓ)-PARTIZATION can be solved in $O(1.1996^{(r+\ell) \cdot n})$ by taking r copies of the input graph, ℓ 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 [43]. 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 [6].

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 $\mathcal{S}$ 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 $\mathcal{S}$?

WEIGHTED d-HITTING SET   **Parameter:** k
Input: A family $\mathcal{S}$ 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 $\mathcal{S}$?

MIN-ONES d-SAT   **Parameter:** k
Input: A propositional formula F in conjunctive normal form 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 propositional formula F in conjunctive normal form 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?