Exact Algorithms via Multivariate Subroutines*

Serge Gaspers†[1] **and Edward J. Lee**[2]

2 **The University of New South Wales, Sydney, Australia; and Data61, CSIRO, Sydney, Australia**
sergeg@cse.unsw.edu.au

2 **The University of New South Wales, Sydney, Australia; and Data61, CSIRO, Sydney, Australia**
e.lee@unsw.edu.au

Abstract

We consider the family of Φ-Subset problems, where the input consists of an instance I of size N over a universe U_I of size n and the task is to check whether the universe contains a subset with property Φ (e.g., Φ could be the property of being a feedback vertex set for the input graph of size at most k). Our main tool is a simple randomized algorithm which solves Φ-Subset in time $(1+b-\frac{1}{c})^n N^{O(1)}$, provided that there is an algorithm for the Φ-Extension problem with running time $b^{n-|X|}c^k N^{O(1)}$. Here, the input for Φ-Extension is an instance I of size N over a universe U_I of size n, a subset $X \subseteq U_I$, and an integer k, and the task is to check whether there is a set Y with $X \subseteq Y \subseteq U_I$ and $|Y \setminus X| \leq k$ with property Φ. We also derandomize this algorithm at the cost of increasing the running time by a subexponential factor in n, and we adapt it to the enumeration setting where we need to enumerate all subsets of the universe with property Φ. This generalizes the results of Fomin et al. [STOC 2016] who proved them for the case $b = 1$. As case studies, we use these results to design faster deterministic algorithms for

- checking whether a graph has a feedback vertex set of size at most k,
- enumerating all minimal feedback vertex sets,
- enumerating all minimal vertex covers of size at most k, and
- enumerating all minimal 3-hitting sets.

We obtain these results by deriving new $b^{n-|X|}c^k N^{O(1)}$-time algorithms for the corresponding Φ-Extension problems (or the enumeration variant). In some cases, this is done by simply adapting the analysis of an existing algorithm, in other cases it is done by designing a new algorithm. Our analyses are based on Measure and Conquer, but the value to minimize, $1+b-\frac{1}{c}$, is unconventional and leads to non-convex optimization problems in the analysis.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory

Keywords and phrases enumeration algorithms, exponential time algorithms, feedback vertex set, hitting set

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.69

1 Introduction

In exponential-time algorithmics [8], the aim is to design algorithms for NP-hard problems with the natural objective to minimize their running times. In this paper, we consider a

* For a full version of the paper see http://arxiv.org/abs/1704.07982 [11].

† Serge Gaspers is the recipient of an Australian Research Council (ARC) Future Fellowship (FT140100048) and acknowledges support under the ARC's Discovery Projects funding scheme (DP150101134).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 69; pp. 69:1–69:13

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

broad class of subset problems, where for an input instance I on a universe U_I, the question is whether there is a subset S of the universe satisfying certain properties. For example, in the Feedback Vertex Set problem, the input instance consists of a graph $G = (V, E)$ and an integer k, the universe is the vertex set and the property to be satisfied by a subset S is the conjunction of "$|S| \leq k$" and "$G - S$ is acyclic".

More formally, and using definitions from [5], an *implicit set system* is 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 *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.

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

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 $\lvert S\rvert \leq k$?

In recent work, Fomin et al. [5] showed that $c^k N^{O(1)}$ time algorithms ($c \in O(1)$) for Φ-Extension lead to competitive exponential-time algorithms for many Φ-Subset problems. The main tool was a simple randomized algorithm which solves Φ-Subset in time $(2 - \frac{1}{c})^n N^{O(1)}$ if there is an algorithm that solves Φ-Extension in time $c^k N^{O(1)}$. A derandomization was also given, turning the randomized algorithm into a deterministic one at the cost of a $2^{o(n)}$ factor in the running time. The method was also adapted to enumeration algorithms and combinatorial upper bounds. This framework, together with a large body of work in parameterized algorithmics [3], where $c^k N^{O(1)}$ time algorithms are readily available for many subset problems, led to faster algorithms for around 30 decision and enumeration problems.

In this paper, we extend the results of Fomin et al. [5] and show that a $b^{n-|X|}c^k N^{O(1)}$ time algorithms ($b, c \in O(1)$) for Φ-Extension lead to randomized $(1 + b - \frac{1}{c})^n N^{O(1)}$ time algorithms for Φ-Subset. Our result can be similarly derandomized and adapted to the enumeration setting. Observe that for $b = 1$, the results of [5] coincide with ours, but that ours have the potential to be more broadly applicable and to lead to faster running times. The main point is that if we use a $c^k N^{O(1)}$ time algorithm as a subroutine to design an algorithm exponential in n, we might as well allow a small exponential factor in n in the running time of the subroutine.

Similar as in [5], the Φ-Extension problem can often be solved by preprocessing the elements in X in a simple way and then using an algorithm for a subset problem. In the case of Feedback Vertex Set, the vertices in X can simply be deleted from the input graph. Whereas the literature is rich with $c^k N^{O(1)}$ time algorithms for subset problems, algorithms with running times of the form $b^n c^k N^{O(1)}$ with $b > 1$ are much less common.[1] One issue is

[1] One notable exception is by Eppstein [4], who showed that all maximal independent sets of size at most k in a graph on n vertices can be enumerated in time $(4/3)^n (81/64)^k n^{O(1)}$.

that there is, in general, no obviously best trade-off between the values of b and c for such algorithms. However, the present framework gives us a precise objective: we should aim for values of b and c that minimize the base of the exponent, $(1 + b - \frac{1}{c})$.

Our applications consist of three case studies centered around some of the most fundamental problems considered in [5], feedback vertex sets and hitting sets. For the first case study, we considered the Feedback Vertex Set problem: given a graph G and an integer k, does G have a feedback vertex set of size at most k? For this problem, we re-analyze the running time of the algorithm from [6]. In [6, 10], the algorithm was analyzed using Measure and Conquer: using a measure that is upper bounded by αn and aiming for a running time of $2^{\alpha n} n^{O(1)}$ the analysis of the branching cases led to constraints lower bounding the measure and the objective was to minimize α subject to these constraints. In our new analysis, we add an additive term $w_k \cdot k$ to the measure and adapt the constraints accordingly. If all constraints are satisfied, we obtain a running time of $2^{\alpha n + w_k k} n^{O(1)}$. Our framework naturally leads us to minimize $2^{\alpha} - 2^{-w_k}$. This approach leads to a $O(1.5422^n \cdot 1.2041^k)$ time algorithm, which, combined with our framework gives a deterministic $O(1.7117^n)$ time algorithm for Feedback Vertex Set. This improves on previous results giving $O(1.8899^n)$ [13], $O(1.7548^n)$ [6], $O(1.7356^n)$ [14], $O(1.7347^n)$ [9], and $O(1.7216^n)$ [5] time algorithms for the problem. We note that adapting the analysis of other existing exact and parameterized algorithms did not give faster running times. Also, if we allow randomization, the $O(1.6667^n)$ time algorithm by [5] (which can also be achieved using our framework) remains fastest.

Our second case study is more involved. Simply using an existing algorithm and adapting the measure was not sufficient to improve upon the best known enumeration algorithms (and combinatorial upper bounds) for minimal feedback vertex sets. Here, the task is, given a graph G, to output each feedback vertex set that is not contained in any other feedback vertex set. We design a new algorithm for enumerating all minimal feedback vertex sets. We also need a new combinatorial upper bound for the number of minimal vertex covers of size at most k to handle one special case in the enumeration of minimal feedback vertex sets.[2] We obtain a $O(1.7183^n \cdot 1.1552^k)$ time algorithm for enumerating all minimal feedback vertex sets. Our framework thus leads to a running time of $O(1.8527^n)$, improving on the previous best bound of $O(1.8638^n)$ [6]. The current best lower bound for the number of minimal feedback vertex sets is $O(1.5926^n)$ [6]. We would like to highlight that the enumeration of minimal feedback vertex sets is completely out of scope for the more restricted framework of [5]: the number of minimal feedback vertex sets of size at most k cannot be upper bounded by $c^k n^{O(1)}$ for any $c \in O(1)$, as evidenced by a disjoint union of k cycles of length n/k.

Our last case study gives a new algorithm for enumerating all minimal 3-hitting sets, also known as minimal transversals of rank-3 hypergraphs. These are minimal sets S of vertices of a hypergraph where each hyperedge has size at most 3 such that every hyperedge contains at least one vertex of S. We re-analyze an existing algorithm [2] for this enumeration problem, adapting the measure in a similar way as in the first case study, and we obtain a multivariate running time of $O(1.5135^n \cdot 1.1754^k)$, leading to an $O(1.6627^n)$ time enumeration algorithm. This breaks the natural time bound of $O(1.6667^n)$ of the previously fastest algorithm [5]. The current best lower bound gives an infinite family of rank-3 hypergraphs with $\Omega(1.5848^n)$ minimal transversals [2].

Lastly, all random selection is done from a uniform distribution and all randomized algorithms in this paper are Monte Carlo algorithms with one-sided error. On No-instances

[2] Previous work [1, 4] focused on small maximal independent sets, whose bounds were insufficient for us. We need better bounds on large maximal independent sets or small minimal vertex covers.

they always return No, and on Yes-instances they return Yes (or output a certificate) with probability $> \frac{1}{2}$.

2 Results

Our first main result gives exponential-time randomized algorithms for Φ-Subset based on single-exponential multivariate algorithms for Φ-Extension with parameter k.

▶ **Theorem 1.** *If there is an algorithm for* Φ-Extension *with running time* $b^{n-|X|}c^k N^{O(1)}$ *then there is a randomized algorithm for* Φ-Subset *with running time* $(1+b-\frac{1}{c})^n N^{O(1)}$.

The next main result derandomizes the algorithm of Theorem 1 at a cost of a subexponential factor in n in the running time.

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

We require the following notion of *(b,c)-uniform* to describe our enumeration algorithms. Let $c, b \geq 1$ be real valued constants and Φ be an implicit set system. Then Φ is *(b,c)-*uniform if for every instance I, set $X \subseteq U_I$, and integer $k \leq n - |X|$, the cardinality of the collection $\mathcal{F}^k_{I,X} = \{S \subseteq U_I \backslash X : |S| = k \text{ and } S \cup X \in \mathcal{F}_I\}$ is at most $b^{n-|X|}c^k n^{O(1)}$. Then the following theorem provides new combinatorial bounds for collections generated by (b, c)-uniform implicit set systems.

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

We say that an implicit set system is *efficiently* (b, c)*-uniform* if there exists an algorithm that given I, X and k enumerates all elements of $\mathcal{F}^k_{X,I}$ in time $b^{n-|X|}c^k N^{O(1)}$. In this case, we enumerate $\mathcal{F}_I$ in the same time, up to a subexponential factor in n.

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

3 Random Sampling and Multivariate Subroutines

In this section, we prove Theorem 1. To do this, we first need the following lemmas.

▶ **Lemma 5.** *If* $b, c \geq 1$ *then* $b \cdot c^{\frac{1}{bc}} \leq 1 + b - \frac{1}{c}$

Proof. As both sides of the inequality are positive, it suffices to show that $\log(bc^{\frac{1}{bc}}) \leq \log(1+b-1/c)$. So we let $y = \log(1+b-1/c) - \log b - \frac{1}{bc}\log c$ and prove that $y \geq 0$ for all $b, c \geq 1$. When $c = 1$ we have that $y = 0$ for all b. We will show that for any fixed $b \geq 1$ we have that $y \geq 0$ by showing that y increases with $c \geq 1$. For fixed b, the partial derivative with respect to c is $\frac{\partial y}{\partial c} = \frac{(bc+c-1)\log c - c + 1}{bc^2(bc+c-1)}$. When $c = 1$ then for all b, $\frac{\partial y}{\partial c} = 0$. As the denominator is positive for $b, c \geq 1$ it is sufficient to show that the numerator $z = (bc + c - 1)\log c - c + 1$ is non-negative. To show that $z \geq 0$, we consider the partial derivative again with respect to c: $\frac{\partial z}{\partial c} = (b+1)\log c + b - \frac{1}{c}$ For $b, c \geq 1$, we have that $b - \frac{1}{c} \geq 0$ and $(b+1)\log(c) \geq 0$. Since $\frac{\partial z}{\partial c} \geq 0$, we conclude that z is increasing and non-negative which implies y is also increasing and non-negative, for all $b, c \geq 1$. This proves the lemma. ◀

The proof of the next lemma follows the proof of Lemma 2.2 from [5], who proved it for $b = 1$.

▶ **Lemma 6.** *Let $b, c \geq 1$, n and $k \leq n$ be non-negative integers. Then, there exists $t \geq 0$ such that*

$$\frac{\binom{n}{t}}{\binom{k}{t}} b^{n-t} c^{k-t} = \left(1 + b - \frac{1}{c}\right)^n n^{O(1)}, \quad \textit{specifically when } t = \frac{cbk - n}{cb - 1}.$$

Proof. We consider two cases. First suppose $k \leq \frac{n}{bc}$. Then for $t = 0$ the LHS (left-hand side) is at most $b^n c^k \leq b^n c^{\frac{n}{bc}} \leq (1 + b - 1/c)^n$ by Lemma 5. Now if $k > \frac{n}{bc}$ then we rewrite the LHS as

$$\frac{\binom{n}{t}}{\binom{k}{t}} b^{n-t} c^{k-t} = \frac{\binom{n}{k} b^{n-k}}{\binom{n-t}{k-t} \left(\frac{1}{bc}\right)^{k-t}}.$$

Let us lower bound the denominator. For any $x \geq 0$ and an integer $m \geq 0$,

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

by a known generating function. For $m = n - k$ and $x = \frac{1}{bc}$, the summand at $i = k - t$ equals the denominator $\binom{n-t}{k-t} \left(\frac{1}{bc}\right)^{k-t}$. Since $\frac{n}{k} < bc$ we have that $\frac{m+k}{k} < \frac{1}{x}$ and the terms of this sum decay exponentially for $i > k$. Thus, the maximum term $\frac{(m+i)(m+i-1)\ldots(m+1)}{i(i-1)\ldots 1} x^i$ for this sum occurs for $i \leq k$, and its value is $\Omega\left(\left(\frac{1}{1-x}\right)^m\right)$ up to a lower order factor of $O(k)$. So by the binomial theorem the expression is at most

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

Specifically, the maximum term for Equation (1) occurs when $\frac{m+i}{i} = \frac{1}{x}$, that is when $\frac{n-t}{k-t} = cb$, and therefore, $t = \frac{cbk-n}{cb-1}$. ◀

▶ **Lemma 7.** *If there exist constants $b, c \geq 1$ and an algorithm for* Φ-Extension *with running time $b^{n-|X|} c^k N^{O(1)}$ then there exists a randomized algorithm for* Φ-Extension *with running time $\left(1 + b - \frac{1}{c}\right)^{n-|X|} N^{O(1)}$.*

Proof. Our proof is similar to Lemma 2.1 in [5]. Let $\mathcal{B}$ be an algorithm for Φ-Extension with running time $b^{n-|X|} c^k N^{O(1)}$. We now present a randomized algorithm $\mathcal{A}$, for the same problem for an input instance (I, X, k') with $k' \leq k$.

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

Algorithm $\mathcal{A}$ has a running time upper bounded by $b^{n-|X|-t} c^{k'-t} N^{O(1)}$. Algorithm $\mathcal{A}$ returns yes for (I, X, k') when $\mathcal{B}$ returns yes for $(I, X \cup Y, k' - t)$. In this case there exists a set $S \subseteq U_I \backslash (X \cup Y)$ of size at most $k' - t \leq k - t$ such that $S \cup X \cup Y \in \mathcal{F}_I$. This, $Y \cup S$ witnesses that (I, X, k) is indeed a yes-instance.

Next we lower bound the probability that $\mathcal{A}$ returns yes if there exists a set $S \subseteq U_I \backslash X$ 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 $U_I \backslash X$. 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 $S \subseteq U_I \backslash X$ of size k' such that $X \cup S \in \mathcal{F}_I$, we have that

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

Let $p(k') = \binom{k'}{t}/\binom{n-|X|}{t}$. For each $k' \in \{0, ..., k\}$, our main algorithm runs $\mathcal{A}$ independently $\frac{1}{p(k')}$ 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

$$\min_{k' \leq k} \left\{ 1 - (1 - p(k'))^{\frac{1}{p(k')}} \right\} \geq 1 - \frac{1}{e} > \frac{1}{2}.$$

Next we upper bound the running time of the main algorithm, which is

$$\sum_{k' \leq k} \frac{1}{p(k')} b^{n-|X|-t} c^{k'-t} N^{O(1)} \leq \max_{k' \leq k} \frac{\binom{n-|X|}{t}}{\binom{k'}{t}} b^{n-|X|-t} c^{k'-t} N^{O(1)} \tag{2}$$

$$\leq \max_{k' \leq n-|X|} \frac{\binom{n-|X|}{t}}{\binom{k}{t}} b^{n-|X|-t} c^{k-t} N^{O(1)}. \tag{3}$$

The choice of t in algorithm $\mathcal{A}$ is chosen to minimize the value of $\frac{\binom{n-|X|}{t}}{\binom{k}{t}} b^{n-|X|-t} c^{k-t}$. For fixed n and $|X|$ the running time of the algorithm is upper bounded by

$$\max_{0 \leq k \leq n-|X|} \left\{ \min_{0 \leq t \leq k} \left\{ \frac{\binom{n-|X|}{t}}{\binom{k}{t}} b^{n-|X|-t} c^{k-t} N^{O(1)} \right\} \right\}. \tag{4}$$

By application of Lemma 6 we choose $t = \frac{cbk-(n-|X|)}{cb-1}$ to obtain the upper bound $\left(1 + b - \frac{1}{c}\right)^{n-|X|} (n - |X|)^{O(1)}$, which, combined with $n < N$, completes the proof. ◀

Running algorithm $\mathcal{A}$ with $X = \emptyset$ and for each value of $k \in \{0, ..., n\}$ results in an algorithm for Φ-Subset with running time $\left(1 + b - \frac{1}{c}\right)^n N^{O(1)}$, proving Theorem 1.

4 Derandomization

In this section we prove Theorem 2, by derandomizing the algorithm in Theorem 1.

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

Given a set U and an integer $q \leq |U|$ let $\binom{U}{q}$ represent the set of sets which contain q elements of U. From [5] we define a pseudo-random object, the *set-inclusion-family*, as well as an almost optimal sub-exponential construction of these objects.

▶ **Definition 8.** 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 is a set $Y \in \mathcal{C}$ such that $Y \subseteq S$.

Let $\kappa(n, p, q) = \binom{n}{q}/\binom{p}{q}$. We also make use of the following theorem.

▶ **Theorem 9** ([5])**.** *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* $\kappa(n, p, q) \cdot 2^{o(n)}$.

We are now ready to prove Lemma 10, by a very similar proof to Lemma 7.

▶ **Lemma 10.** *If there exists constants* $b, c \geq 1$ *and an algorithm for* Φ-Extension *with running time* $b^{n-|X|} c^k N^{O(1)}$ *then there exists a deterministic algorithm for* Φ-Extension *with running time* $\left(1 + b - \frac{1}{c}\right)^{n-|X|} \cdot 2^{o(n)} \cdot N^{O(1)}$.

Proof. Let $\mathcal{B}$ be an algorithm for Φ-Extension with running time $b^{n-|X|}c^k N^{O(1)}$. We can then adapt Algorithm $\mathcal{A}$ from the proof of Lemma 7. Let $\mathcal{A}'$ be a new algorithm which has an input instance (I, X, k') with $k' \leq k$. Choose $t = \frac{cbk' - (n-|X|)}{cb-1}$.

1. Compute an $(n-|X|, k', t)$-set-inclusion-family $\mathcal{C}$ using the algorithm from Theorem 9 of size at most $\kappa(n-|X|, k', t) \cdot 2^{o(n)}$, in $\kappa(n-|X|, k', t) \cdot 2^{o(n)}$ time.
2. For each set Y in the set-inclusion-family $\mathcal{C}$ run algorithm $\mathcal{B}$ on the instance $(I, X \cup Y, k'-t)$ and return Yes of at least one returns Yes and No otherwise.

The running time of $\mathcal{A}'$ is upper bounded by $\kappa(n-|X|, k', t) \cdot 2^{o(n)} \cdot b^{n-|X|-t}c^{k'-t}N^{O(1)}$, a term encountered in Equation 2 with a new subexponential factor in n,

$$\max_{k' \leq k} \frac{\binom{n-|X|}{t}}{\binom{k'}{t}} \cdot b^{n-|X|-t}c^{k'-t}N^{O(1)} \cdot 2^{o(n)}.$$

From here the proof follows that of Lemma 7. ◀

The proof of Theorem 2 follows by inclusion of the factor $2^{o(n)}$.

5 Enumeration

We now proceed to prove Theorems 3 and 4 on combinatorial upper bounds and enumeration algorithms. Consider the following random process.

1. Choose an integer t based on b, 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 output the empty set.

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

Proof. Let I be an instance, $k \leq n$. We will prove that the number of sets in $\mathcal{F}_I$ of size exactly k is upper bounded by $|\mathcal{F}_I| \leq \left(1+b-\frac{1}{c}\right)^n n^{O(1)}$, where k is chosen arbitrarily. We follow the random process described above, which picks a set W of size k from $\mathcal{F}_I$.

For each set $Z \in \mathcal{F}_I$ of size exactly k, let E_Z denote the event that the set W output in step 2 is equal to Z. We then have the following lower bound on the probability of the event E_Z:

$$\Pr[E_Z] = \Pr[X \subseteq Q \wedge S = Z \backslash X] = \Pr[X \subseteq Z] \times \Pr[S = Z \backslash Z | X \subseteq Z] = \frac{\binom{k}{t}}{\binom{n}{t}} \cdot \frac{1}{|\mathcal{F}_{I,X}^{k-t}|}$$

Since Φ is (b,c)-uniform then $\left|\mathcal{F}_{I,X}^{k-t}\right| \leq b^{n-|X|}c^k n^{O(1)}$ and X is selected such that $|X| = t$, this results in the lower bound

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

A choice of t is made to minimize the lower bound, and this choice is given by Lemma 6 which states that for every $k \leq n$ there exists a $t \leq k$ such that we obtain a new lower bound

$$\Pr[E_Z] \geq \left(1+b-\frac{1}{c}\right)^{-n} \cdot n^{O(1)}$$

for every $Z \in \mathcal{F}_I$ of size k. For every individual set $Z \in \mathcal{F}_I$, the event E_Z occurs disjointly, and we have that $\sum_{Z \in \mathcal{F}_I, |Z|=k} \Pr[E_Z] \leq 1$. This fact with the lower bound of $\Pr[E_Z]$ implies an upper bound on the number of sets in $\mathcal{F}_I$ of $(1+b-\frac{1}{c})^n n^{O(1)}$, completing the proof. ◀

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

Proof. We alter the random process used to prove Theorem 3 to a deterministic one:
1. Construct a (n, k, t)-set inclusion family $\mathcal{C}$ using Theorem 6 from [5]. Loop over $X \in \mathcal{C}$.
2. For each $X \in \mathcal{C}$, loop over all sets $S \in \mathcal{F}_{I,X}^{k-t}$.

Then we output $W = X \cup S$ from these two loops. Looping over $\mathcal{C}$ instead of random sampling for X incurs a $2^{o(n)}$ overhead in the running time. As Φ is efficiently (b, c)-uniform, the inner loop requires $(1 + b - \frac{1}{c})^n N^{O(1)}$ time. In order to avoid enumerating duplicates, we save the sets which have been output in a trie and check first in linear time if a set has already been output. The product of the running times for these two nested loops results in the running time claimed by the theorem statement. ◀

6 Case Studies

This section briefly outlines case studies which used Theorem 2 and Theorem 4 in order to design faster deterministic algorithms.

6.1 Preliminaries

Let $G = (V, E)$ be a graph with a set of vertices V and a set of edges $E \subseteq \{uv : u, v \in V\}$. The *degree* $d(u)$ of a vertex u is the number of neighbors of u in G. The *degree* of a graph $\Delta(G)$ is the maximum $d(u)$ across all $u \in V$. A graph $G' = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$ and G' is an *induced subgraph* of G if, in addition, G has no edge uv with $u, v \in V'$ but $uv \notin E'$. In this case, we also denote G' by $G[V']$. A forest is an acyclic graph. A subset $F \subseteq V$ is acyclic if $G[F]$ is a forest. An acyclic subset $F \subseteq V$ is *maximal* in G if it is not a subset of any other acyclic subset. For an acyclic subset $F \subseteq V$, we denote the set of maximal acyclic supersets of F as $\mathcal{M}_G(F)$ and the set of maximum (i.e., largest) acyclic supersets of F as $\mathcal{M}_G^*(F)$.

Let T be a subgraph of G. Define $\mathrm{Id}(T, t)$ as an operation on G which contracts all edges of T into one vertex t, removing induced loops. This may create multiedges in G. Define $\mathrm{Id}^*(T, t)$ as the operation $\mathrm{Id}(T, t)$ followed by removing all vertices connected to t by multiedges. A *non-trivial* component of a graph G is a connected component on at least two vertices. The following propositions from [6] will be useful.

▶ **Proposition 11.** *[6] Let $G = (V, E)$ be a graph, $F \subseteq V$ be an acyclic subset of vertices and T be a non-trivial component of $G[F]$. Denote by G' the graph obtained from G by the operation $Id^*(T, t)$ and let $F' = F \cup \{t\} \backslash T$. Then for $X' = X \cup \{t\} \backslash T$ where $X, X' \subseteq V$*
- *$X \in \mathcal{M}_G(F)$ if and only if $X' \in \mathcal{M}_{G'}(F')$, and*
- *$X \in \mathcal{M}_G^*(F)$ if and only if $X' \in \mathcal{M}_{G'}^*(F')$.*

Using operation Id^* on each non-trivial component of $G[F]$, results in an independent set F'.

▶ **Proposition 12.** *[6] Let $G = (V, E)$ be a graph and F be an independent set in G such that $V \backslash F = N(t)$ for some $t \in F$. Consider the graph $G' = G[N(t)]$ and for every pair of vertices $u, v \in N(t)$ having a common neighbor in $F \backslash \{t\}$ add an edge uv to G'. Denote the obtained graph by H and let $I \subseteq N(t)$. Then $F \cup I \in \mathcal{M}_G(F)$ if and only if I is a maximal independent set in H . In particular, $F \cup I \in \mathcal{M}_G^*(F)$ if and only if I is a maximum independent set in H.*

For an acyclic subset F, a so-called *active* vertex $t \in F$ and a neighbor $v \in N(t) \setminus F$, we will now define the concept of generalized neighbors of v, also known as (v)-*generalized neighbors*. Denote by K the set of vertices of F adjacent to v other than t. Let G' be the graph obtained after the operation $\mathrm{Id}(K \cup \{v\}, u)$. A vertex $w \in V(G') \setminus \{t\}$ is a (v)-generalized neighbor in G if w is a neighbor of u in G'. Denote by $\mathrm{gd}(v)$ the *generalized degree* of v which is the number of (v)-generalized neighbors for a given v.

6.2 Feedback Vertex Set

First we describe the extension variant of Feedback Vertex Set

Feedback Vertex Set Extension	
Input:	A graph $G = (V, E)$, vertex subset $X \subseteq V$ and an integer k
Question:	Does there exist subset $S \subseteq V \setminus X$ such that $S \cup X$ is a FVS and $\lvert S \rvert \leq k$?

Instead of directly finding the feedback vertex set in a graph, we present algorithm $\texttt{mif}(G, F, k)$ [6] which computes for a given graph G and an acylic set F the maximum size of an induced forest F' containing F with $|F'| \geq n - k$. This means that $G - F$ is a minimal feedback vertex set of size at most k. This algorithm can easily be turned into an algorithm computing at least one such set.

During the execution of `mif` one vertex $t \in F$ is called an *active vertex*. Algorithm `mif` then branches on a chosen neighbor of t. Let $v \in N(t)$. Let k be the set of all vertices of $F \setminus \{t\}$ that are adjacent to v and parameter k which represents a bound on the size of the feedback vertex set.

As well as describing the algorithm we simultaneously perform the running time analysis which uses the Measure and Conquer framework and Lemma 13 at its core.

▶ **Lemma 13.** *[10] Let A be an algorithm for a problem P, B be an algorithm for a class C of instances of P, $c \geq 0$ and $r > 1$ be constants, and $\mu(\cdot), \mu_B(\cdot), \eta(\cdot)$ be measures for P, such that for any input instance I from C, $\mu_B(\cdot) \leq \mu(I)$, and for any input instance I, A either solves P on $I \in C$ by invoking B with running time $O(\eta(I)^{c+1} r^{\mu B(I)})$, or reduces I to k instances $I_1, ..., I_k$, solves these recursively, and combines their solutions to solve I, using time $O(\eta(I)^c)$ for the reduction and combination steps (but not the recursive solves),*

$$(\forall i) \quad \eta(I_i) \leq \eta(I) - 1, \qquad \text{and} \qquad \sum_{i=1}^{k} r^{\mu(I_i)} \leq r^{\mu(I)}. \tag{5}$$

Then A solves any instance I in time $O(\eta(I)^{c+1} r^{\mu(I)})$.

Branching constraints of the form $\sum_{i=1}^{j} 2^{-\delta_i} \leq 1$ are given as branching vectors $(\delta_1, ..., \delta_j)$.

6.2.1 Measure

To upper bound the exponential time complexity of the algorithm `mif` we use the measure

$$\mu = |N(t) \setminus F| w_1 + |V \setminus (F \cup N(t))| w_2 + k \cdot w_k.$$

In other words, each vertex in F has weight 0, each vertex in $N(t)$ has weight w_1, each other vertex has weight w_2 and each unit of budget for the feedback vertex set has weight w_k, in the measure with an active vertex t.

6.2.2 Algorithm

The description of `mif` consists of a sequence of cases and subcases. The first case which applies is used, and inside a given case the hypotheses of all previous cases are assumed to be false. Preprocessing procedures come before main procedures.

Preprocessing

1. If G consists of $j \geq 2$ connected components $G_1, G_2, ..., G_j$, then the algorithm is called on each component. For $F_i = G_i \cap F$ for all $i \in \{1, 2, ..., j\}$ and $\sum_{i=1}^{j} k_i \leq k$ then

$$\texttt{mif}(G, F, k) = \sum_{i=1}^{j} \texttt{mif}(G_i, F_i, k_i)\,.$$

2. If F is not independent, then apply operation $\mathrm{Id}^*(T, v_T)$ on an arbitrary non-trivial component T of F. If T contains the active vertex then v_T becomes active. Let G' be the resulting graph and let F' be the set of vertices of G' obtained from F. Then

$$\texttt{mif}(G, F, k) = \texttt{mif}(G', F', k) + |T| - 1\,.$$

Main Procedures

1. If $k < 0$ then $\texttt{mif}(G, F, k) = 0$.
2. If $F = \emptyset$ and $\Delta(G) \leq 1$ then $\mathcal{M}_G(F) = \{V\}$ and $\texttt{mif}(G, F, k) = |V|$.
3. If $F = \emptyset$ and $\Delta(G) \geq 2$ then the algorithm chooses a vertex $t \in G$ of degree at least 2. Then t is either contained in a maximum induced forest or not. The algorithm branches on two subproblems and returns the maximum:

$$\texttt{mif}(G, F, k) = \max\{\texttt{mif}(G, F \cup \{t\}, k), \texttt{mif}(G \backslash \{t\}, F, k-1)\}.$$

   The first branch reduces the weight of t to zero, as it is in F, and at least 2 neighbors have a reduced degree from w_2 to w_1. In the second branch we remove t from the graph, meaning it will be in the feedback vertex set. We thus also gain a reduction of w_k in the measure. Hence this rule induces the branching constraint

$$(w_2 + 2(w_2 - w_1), w_2 + w_k).$$

4. If F contains no active vertex then choose an arbitrary vertex $t \in F$ as an active vertex. Denote the active vertex by t from now on.
5. If $V \backslash F = N(t)$ then the algorithm constructs the graph H from Proposition 12 and computes a maximum independent set I in G of maximum size $n - k$. Then

$$\texttt{mif}(G, F, k) = |F| + |I|.$$

6. If there is $v \in N(t)$ with $\mathrm{gd}(v) \leq 1$ then add v to F.

$$\texttt{mif}(G, F, k) = \texttt{mif}(G, F \cup \{v\}, k).$$

7. If there is $v \in N(t)$ with $\mathrm{gd}(v) \geq 4$ then either add v to F or remove v from G.

$$\texttt{mif}(G, F, k) = \max\{\texttt{mif}(G, F \cup \{v\}, k), \texttt{mif}(G \backslash \{v\}, F, k-1)\}.$$

   The first case adds v to F reducing the measure by w_1, and a minimum of $4(w_2 - w_1)$ for all the (v)-generalized neighbors. The other case removes v this decreasing the measure by w_k. Hence this rule induces the branching constraint

$$(w_1 + 4(w_2 - w_1), w_1 + w_k).$$

8. If there is $v \in N(t)$ with $\text{gd}(v) = 2$ then denote the (v)-generalized neighbors by u_1 and u_2. Either add v to F or remove v from G but add u_1 and u_2 to F. If adding u_1 and u_2 to F induces a cycle, we just ignore the last branch.

$$\texttt{mif}(G, F, k) = \max\{\texttt{mif}(G, F \cup \{v\}, k), \texttt{mif}(G \backslash \{v\}, F \cup \{u_1, u_2\}, k-1)\}.$$

Let $i \in \{0, 1, 2\}$ be the number of vertices adjacent to v with weight w_2. The first case adds v to F, and hence all i w_2-weight neighbors of v reduce to w_1, and the other $2-i$ vertices of weight w_1 induce a cycle, hence we remove them from G and reduce the measure by $(2-i)w_k$. The second case removes v and adds both u_1 and u_2 to F. This causes a reduction of iw_2 for the relevant vertices and $(2-i)w_1$ for the other vertices, and a single w_k reduction due to the removal of v. This rule induces the branching constraint

$$(w_1 + i(w_2 - w_1) + (2-i)w_1 + (2-i)w_k, w_1 + iw_2 + (2-i)w_1 + w_k).$$

9. If all vertices in $N(t)$ have exactly three generalized neighbors then at least one of these vertices must have a generalized neighbor outside $N(t)$, since the graph is connected and the condition of the case Main 6 does not hold. Denote such a vertex by v and its generalized neighbors by u_1, u_2 and u_3 in such a way that $u_1 \notin N(t)$. Then we either add v to F; or remove v from G but add u_1 to F; or remove v and u_1 from G and add u_2 and u_3 to F. Similar to the previous case, if adding u_2 and u_3 to F induces a cycle, we just ignore the last branch.

$$\begin{aligned}\texttt{mif}(G, F) = \max\{&\texttt{mif}(G, F \cup \{v\}, k), \texttt{mif}(G \backslash \{v\}, F \cup \{u_1\}, k-1),\\ &\texttt{mif}(G \backslash \{v, u_1\}, F \cup \{u_2, u_3\}, k-2)\}.\end{aligned}$$

Let $i \in \{1, 2, 3\}$ be the number of vertices adjacent to v with weight w_2. The first and last cases are analogous to the analysis done in Main 8. The second case removes v from the forest hence adding it to the minimum feedback vertex set and reducing the measure by $w_1 + w_k$. A reduction of w_2 is gained by adding u_1 to F. Then this rule induces the branching constraint

$$(w_1 + i(w_2 - w_1) + (3-i)w_1 + (3-i)w_k, w_1 + w_2 + w_k, w_1 + iw_2 + (3-i)w_1 + 2w_k).$$

6.2.3 Results

▶ **Theorem 14.** *Let G be a graph on n vertices. Then a minimal feedback vertex set in G can be found in time $O(1.7117^n)$.*

Proof. Using the algorithm above along with the measure μ, the following values of weights can be shown to satisfy all the branching vector constraints generated above.

$$w_1 = 0.2775 \qquad w_2 = 0.6250 \qquad w_k = 0.2680$$

These weights result in an upper bound for the running time of `mif` as $O(1.5422^n \cdot 1.2041^k)$ for computing a maximally induced forest of size a least $n-k$, and hence we have the running time for Feedback Vertex Set Extension of $O(1.5422^{n-|X|} \cdot 1.2041^k)$. By Theorem 2 this results in a $O(1.7117^n)$ algorithm for computing a minimal feedback vertex set. ◀

6.3 Minimal Vertex Covers

The following result on minimal vertex covers of size at most k is needed in the next section.

▶ **Theorem 15.** *Let γ be a constant with $0.169925 \approx 2\log_2 3 - 3 \leq \gamma \leq 1$. For every $n \geq k \geq 0$, and every graph G on n vertices, the number of minimal vertex covers of size at most k of G is at most $2^{\beta n+\gamma k}$, where $\beta = (1-\gamma)/2$.*

For $\gamma = \frac{1}{3}$, this implies that G has at most $2^{(n+k)/3}$ minimal vertex covers of size at most k.

6.4 Minimal Feedback Vertex Sets

We apply a similar methodology to enumerating minimal feedback vertex sets on an undirected graph with n vertices. The algorithm is similar in construction to one used in [6] yet a large amount of case analysis was added, along with potential functions in combination with the Measure and Conquer framework.

▶ **Theorem 16.** *For a graph G with n vertices, all minimal feedback vertex sets can be enumerated in time $O(1.8527^n)$.*

6.5 Minimal Hitting Sets

Based on [2] we once again apply a multivariate analysis to enumerating all minimal hitting sets on a hypergraph of rank 3.

▶ **Theorem 17.** *For a hypergraph H with n vertices and rank 3, all minimal hitting sets can be enumerated in time $O(1.6627^n)$.*

7 Conclusion

The main contribution of this paper is a framework allowing us to turn many $b^n c^k N^{O(1)}$ time algorithms for subset and subset enumeration problems into $(1 + b - \frac{1}{c})^n N^{O(1)}$ time algorithms, generalizing a recent framework of Fomin et al. [5].

The main complications in using the framework are, firstly, that new algorithms or running-time analyses are often needed, and, secondly, that such analyses need solutions to non-convex programs in the Measure and Conquer framework. In the usual Measure and Conquer analyses [7],the objective is to upper bound a single variable (α) which upper bounds the exponential part of the running time ($2^{\alpha n}$) subject to convex constraints. Thus, it is sufficient to solve a convex optimization problem to minimize the running time [10, 12] resulting from the constraints derived from the analysis. Here, the objective function ($2^{\alpha} - 2^{-w_k}$) is non-convex. While experimenting with a range of solvers (Couenne, IPOPT, MINOS, SNOPT), either guaranteeing to find a global optimum (slow and used for optimality checks) or only a local optimum (faster and used mainly in the course of the algorithm design), we experienced on one hand that the local optima found by solvers are often the global optimum, but on the other hand that weakening non-tight constraints can sometimes lead to a better globally optimum solution.

Acknowledgements. We thank Daniel Lokshtanov, Fedor V. Fomin, and Saket Saurabh for discussions inspiring some of this work.

References

1 Jesper Makholm Byskov. Enumerating maximal independent sets with applications to graph colouring. *Oper. Res. Lett.*, 32(6):547–556, 2004. doi:10.1016/j.orl.2004.03.002.

2 Manfred Cochefert, Jean-François Couturier, Serge Gaspers, and Dieter Kratsch. Faster algorithms to enumerate hypergraph transversals. In *Latin American Symposium on Theoretical Informatics*, pages 306–318. Springer, 2016.

3 Rodney G. Downey and Michael R. Fellows. *Fundamentals of parameterized complexity*, volume 4. Springer, 2013.

4 David Eppstein. Small maximal independent sets and faster exact graph coloring. *J. Graph Algorithms Appl.*, 7(2):131–140, 2003.

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

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

7 Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. A measure & conquer approach for the analysis of exact algorithms. *Journal of the ACM*, 56(5), 2009. doi:10.1145/1552285.1552286.

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

9 Fedor V. Fomin, Ioan Todinca, and Yngve Villanger. Large induced subgraphs via triangulations and cmso. *SIAM Journal on Computing*, 44(1):54–87, 2015.

10 Serge Gaspers. *Exponential Time Algorithms – Structures, Measures, and Bounds*. VDM, 2010. URL: http://amzn.com/3639218256.

11 Serge Gaspers and Edward Lee. Exact algorithms via multivariate subroutines, April 2017. arXiv e-prints. URL: https://arxiv.org/abs/1704.07982, arXiv:1704.07982.

12 Serge Gaspers and Gregory B. Sorkin. A universally fastest algorithm for Max 2-Sat, Max 2-CSP, and everything in between. *Journal of Computer and System Sciences*, 78(1):305–335, 2012. doi:10.1016/j.jcss.2011.05.010.

13 Igor Razgon. Exact computation of maximum induced forest. In *Scandinavian Workshop on Algorithm Theory*, pages 160–171. Springer, 2006.

14 Mingyu Xiao and Hiroshi Nagamochi. Exact algorithms for maximum independent set. In *International Symposium on Algorithms and Computation*, pages 328–338. Springer, 2013.