# Inclusion–Exclusion Algorithms for Counting Set Partitions[*]

Andreas Björklund and Thore Husfeldt
Department of Computer Science
Lund University, Sweden
andreas.bjorklund@anoto.com, thore.husfeldt@cs.lu.se

## Abstract

*Given a set $U$ with $n$ elements and a family of subsets $\mathcal{S} \subseteq 2^U$ we show how to count the number of $k$-partitions $S_1 \cup \cdots \cup S_k = U$ into subsets $S_i \in \mathcal{S}$ in time $2^n n^{O(1)}$. The only assumption on $\mathcal{S}$ is that it can be enumerated in time $2^n n^{O(1)}$.*

*In effect we get exact algorithms in time $2^n n^{O(1)}$ for several well-studied partition problems including Domatic Number, Chromatic Number, Bounded Component Spanning Forest, Partition into Hamiltonian Subgraphs, and Bin Packing.*

*If only polynomial space is available, our algorithms run in time $3^n n^{O(1)}$ if membership in $\mathcal{S}$ can be decided in polynomial time. For Chromatic Number, we present a version that runs in time $O(2.2461^n)$ and polynomial space. For Domatic Number, we present a version that runs in time $O(2.8718^n)$.*

*Finally, we present a family of polynomial space approximation algorithms that find a number between $\chi(G)$ and $\lceil(1+\epsilon)\chi(G)\rceil$ in time $O(1.2209^n + 2.2461^{e^{-\epsilon}n})$.*

## 1. Introduction

We can view graph colouring as a set covering problem: A graph has chromatic number $\leq k$ if and only if its vertices can be covered with $k$ stable sets. Replacing 'stable sets' with any family $\mathcal{S}$ of subsets we arrive at the following general problem: Given a set $U$ of $n$ elements and a family $\mathcal{S}$ of subsets of $U$, decide if $U$ can be partitioned into $k$ disjoint subsets $S_1 \cup S_2 \cup \cdots \cup S_k = U$, $S_i \in \mathcal{S}$.

Typically, $\mathcal{S}$ is defined implicitly by a polynomial time computable predicate. Besides graph colouring, another example is to let $\mathcal{S}$ be the dominating sets of a graph, i.e., the sets $S$ such that every vertex has distance at most one to $S$. Finding the largest $k$ such that the graph can be partitioned into $k$ sets from $\mathcal{S}$ is the Domatic Number problem. Table 2 shows some other graph problems included in this framework.

| Time $O(c^n)$ | Problem | Reference |
|---|---|---|
| $c$ =2.4423 | Find $\chi$ | Lawler [23] |
| 2.4151 | Find $\chi$ | Eppstein [11] |
| 2.4023 | Find $\chi$ | Byskov [7] |
| 2.3236 | Find $\chi$ | Björklund and Husfeldt [5] |
| 2.2590 | Decide $\chi \leq 5$ | Beigel and Eppstein [3] |
| 2.1592 | Decide $\chi \leq 5$ | Byskov [7] |
| 2.1020 | Decide $\chi \leq 5$ | Byskov and Eppstein [9] |
| 2.1809 | Decide $\chi \leq 6$ | *ibid.* |
| 2.9416 | Decide $\delta \geq 3$ | Riege and Rothe [27] |
| 2.8718 | Find $\delta$ | Fomin *et al.* [15] |
| 2.6949 | Decide $\delta \geq 3$ | Riege *et al.* [28] |

**Table 1. Previous algorithms for Chromatic Number $\chi$ and Domatic Number $\delta$.**

A way to solve this problem that goes back at least to Lawler [23], is to use dynamic programming over the subsets of $U$: Build a table $T(X, m)$ with entries for every $X \subseteq U$ and $m \leq k$. Iterate over the subsets in order of increasing size and use $T(X, m) = \sum_{S \in \mathcal{S}} T(X - S, m - 1)$, to check for each $m \leq k$ whether $X$ can be covered by $m$ of the subsets. Clearly, the algorithm's running time is bounded by $|\mathcal{S}|2^n n^{O(1)}$, and it is never worse than within a polynomial factor of $\sum_{S \in \mathcal{S}} 2^{n-|S|} \leq \sum_{i=0}^{n} \binom{n}{i} 2^i = 3^n$. Ingenious ways to enumerate and bound the size of the family $\mathcal{S}$ (corresponding to minimal dominating sets in the case of Domatic Number or to maximal stable sets in the case of Chromatic Number) have resulted in the time bounds $O(2.8718^n)$ for Domatic Number [15] and $O(2.4022^n)$ for Chromatic Number [7]. Reducing these constants towards two has been a perpetual algorithmic challenge (see Table 1), and the possibility of ever arriving within a polynomial factor of time $2^n$, for example for Chromatic Number, has been a well-known open problem [33].

[*]A preliminary version of this paper appeared as *Inclusion-Exclusion Based Algorithms for Graph Colouring*, ECCC report TR06-044, March 28, 2006.

## 1.1 Main result

We punctuate this history of successive improvements by solving the general problem in time $2^n n^{O(1)}$ using no properties of $\mathcal{S}$ at all, other than being enumerable within that time bound. The algorithm and its analysis are short, self-contained and elementary.

The idea is to express the the number of $k$-covers as an inclusion–exclusion formula over the subsets of $U$. In its simplest form, it says that $U$ can be covered with $k$ sets from $\mathcal{S}$ if

$$\sum_{X \subseteq U} (-1)^{|X|} s[X]^k \qquad (1)$$

is nonzero, where $s[X]$ denotes the number of sets in $\mathcal{S}$ not intersecting $X$. To evaluate the summands quickly we show how to first build a table containing $s[X]$ for all $X \subseteq U$ in time $2^n n^{O(1)}$.

In fact, our algorithm counts the number of such covers. We have results both for the case where the covers are disjoint and where they are overlapping. As a consequence we can compute the chromatic polynomial in time $2^n n^{O(1)}$.

## 1.2. Applications

Perhaps the simplest application of our result is [SR1] Bin Packing, where we are given a weight $w(u)$ for each $u \in U$ and $\mathcal{S}$ consists of the subsets $S \subseteq U$ satisfying $\sum_{u \in S} w(u) \leq B$. But we may consider much more constrained partitions. Most notably the theorem applies to some well-known NP-complete problems on (hyper)graphs that ask for the optimum number of parts in a vertex partition where every part satisfies a given property.

Table 2 shows some examples. These properties are all polynomial time checkable, except for [GT13], which is NP-hard. However, we can enumerate all subsets $S \subseteq U$ such that $G[S]$ is Hamiltonian in time $2^n n^{O(1)}$ [20], which suffices for our purposes.

The most obvious problem is of course [SP5] Minimum Set Cover and its many variants, but this may be a misleading example. In those problems, the set $\mathcal{S}$ is given explicitly as part of the input, and is often small compared to $n$; for example the clauses of a monotone satisfiability problem or the edges of a sparse (hyper)graph. Our algorithms apply to this problem as well, and become interesting when $\mathcal{S}$ is large compared to $n$.

We also solve the colouring problem Chromatic Sum in the same time bound.

## 1.3. Further results

We note that (1) immediately yields an $|\mathcal{S}|2^n n^{O(1)}$ time, *polynomial space* algorithm for our problem. Until very recently [5, 6], no polynomial space algorithm for e.g. Chromatic Number running in time $O(c^n)$ for any positive constant $c$ was known, an open problem observed in [8, 24, 34]. For the case where $\mathcal{S}$ is given as a polynomial time computable predicate, the running time becomes $3^n n^{O(1)}$, in polynomial space.

We take a closer look at polynomial space algorithms for Chromatic and Domatic Number. Using the fastest currently known algorithm in the literature for counting stable sets [16] to compute $s[X]$, the total running time to evaluate (1) becomes $O(2.2461^n)$. For Domatic Number, we need a more complicated argument that can be seen as an extension of our main result, together with a recent algorithm to count the number of minimal dominating sets [15] and arrive at total time $O(2.8718^n)$. Both of these algorithms are the fastest polynomial space algorithms known for these problems, in fact they are faster than the best *exponential* space algorithms known prior to this paper.

Finally, we derive a family of exponential-time approximation algorithms based on first removing large stable sets and then applying our ideas on the remaining graph. For instance, we can approximate $\chi(G)$ within a factor 2 in time $O(1.3467^n)$ and polynomial space. The approximability of the chromatic number is very well studied; the best known polynomial time algorithm guarantees only an approximation ratio of $O(n \log^{-3} n \log\log^2 n)$ [19], and $\chi(G)$ is NP-hard to approximate within $n^{1-o(1)}$ [35].

Our inclusion–exclusion formulas themselves provide characterizations of well-studied graph numbers. For example, for the chromatic polynomial we arrive at

$$P(G;k) = \sum_{r=1}^{k} \binom{k}{r} \Bigg( \sum_{X \subseteq V} (-1)^{|X|} a_r(X) \Bigg), \qquad (2)$$

where $a_r(X)$ denotes the number of ways to choose $r$ stable sets $S_1, \ldots, S_r \subseteq V - X$, such that $|S_1| + \cdots + |S_r| = n$. To the best knowledge of the authors, these characterizations are new and might be of independent combinatorial interest; in any case, their proofs are elementary.

## 1.4. Previous work and discussion

The first non-trivial algorithm for finding the chromatic number, by Christofides [10] in 1971, runs in time $n!n^{O(1)}$ and can be seen to require only polynomial space. Then, a series of exponential time and space algorithms began in 1976 with Lawler's algorithm [23], see Table 1, all of which are based on finding maximal stable sets and owing their running time ultimately to the fact that there are only $3^{n/3}$ maximal stable sets in a graph [25].

Our algorithms beat the running time of previous algorithms that decide $k$-colourability for small values of $k$. The exceptions are 3- and 4-colourability, which can be decided in time $O(1.3289^n)$ [3] and $O(1.7504^n)$ [7], respectively, well beyond the reach of our constructions.

| Name [17] | Property of $S \in \mathcal{S}$ |
|---|---|
| GT3, Domatic Number | $S$ is a dominating set in $G$ |
| GT5, Chromatic Number | $S$ is a stable set in $G$ |
| GT13, Partition into Hamiltonian Subgraphs | $G[S]$ is Hamiltonian |
| GT14, Partition into Forests | $G[S]$ is a forest |
| GT16, Partition into Perfect Matchings | $G[S]$ has a perfect matching |
| ND10, Bounded Component Spanning Forest | $G[S]$ connected, $\sum_{v\in S} w(v) \leq B$ |

**Table 2. Some exact partition problems on graphs** $G = (U, E)$

.

For polynomial space, Feder and Motwani [12] gave a randomised linear space algorithm with running time $O\big((\chi/e)^n\big)$, improving Christofides' result for small values of $\chi$. The running time of an algorithm by Angelsmark and Thapper [1] can be given as $O\big((2 + \log\chi)^n\big)$, an asymptotic improvement over Christofides' result for all values of $\chi$. Very recently, running times of the form $O(c^n)$ have appeared; Bodlaender and Kratsch [6] achieve $O(5.283^n)$ and in a precursor to the present paper [5], the authors arrived at $O(8.3203^n)$ and $O(2.4423^n)$. The bound given in the present paper, $O(2.2416^n)$ will improve whenever the running time for counting stable sets is improved, but there is little hope that this approach will ever reach $2^n n^{O(1)}$ in polynomial space, since counting stable sets is $\#P$-complete [31, 18]. The existence of such an algorithm remains open.

For Domatic Number, exponential space algorithms that are faster than $3^n n^{O(1)}$ have appeared only recently. [15] shows an $O(2.8718^n)$ time algorithm for deciding the domatic number. [28] recently presented an $O(2.6949^n)$ time polynomial space algorithm for deciding if the domatic number is at least three. No prior polynomial space algorithm for the general problem is known to the authors.

Anthony [2] surveys and compares previous methods for computing the chromatic polynomial, see also [4, 32]. The *Whitney expansion*,

$$P(G; k) = \sum_{H\subseteq E} (-1)^{|H|} k^{n-r\langle H\rangle}, \qquad (3)$$

where $r\langle H\rangle$ is the rank of the subgraph induced by the edge set $H$, requires time $2^m$. On some instances, a faster way is the *deletion–contraction method*, based on the recurrence

$$P(G; k) = P(G - e, k) + P(G/e, k),$$

where $G - e$ and $G/e$ are constructed by deleting or contracting edge $e$, which runs within a polynomial factor of $\big(\frac{1}{2}(1 + \sqrt{5})\big)^{n+m} = O(1.6180^{n+m})$. Finally, the relation between $P$ and the *Tutte polynomial* $\sum t_{ij} x^i y^j$,

$$P(G; k) = (-1)^{n-1} k \sum_{i=1}^{n-1} t_{i0}(1 - k)^i$$

leads to an algorithm that runs within a polynomial factor of $\binom{m}{n-1}$. All these algorithms can be seen to run in polynomial space.

The principle of inclusion–exclusion has been used before to solve combinatorial problems on graphs. For instance the most effective way known to date to count the number of matchings in a bipartite graph exactly is to apply the Ryser formula for the permanent [29]. A concise paper by Karp [21] counts the number of Hamiltonian circuits in a graph in polynomial space and time $2^n n^{O(1)}$ using the principle. Both examples count covers of the vertices by graph edges. The contribution of the present paper is to observe that the technique can be almost as powerful when counting covers assembled from an *exponential* number of larger subsets of the vertex set. An earlier paper by the authors [5] already tentatively explores this idea, presenting an inclusion–exclusion formula for graph colouring, but the constructions there are still based on maximal stable sets, much slower, and more complicated. Independently, Koivisto [22] uses inclusion–exclusion in a more general framework that subsumes our results for exponential space.

## 2. Results

*Notation.* In this section, $U$ is a set of size $n$ and $\mathcal{S}$ is a family of subsets of $U$, enumerable in time $2^n n^{O(1)}$. We write $\mathcal{S}[X] = \{\, S \in \mathcal{S} \colon S \cap X = \varnothing \,\}$, the subfamily avoiding $X$, and let $s[X]$ denote the cardinality of $\mathcal{S}[X]$. We write $\mathcal{S}^{(i)} = \{\, S \in \mathcal{S} \colon |S| = i \,\}$ for the subfamily of $i$-sets. Finally $s^{(i)}[X]$ is the cardinality of $\mathcal{S}^{(i)}[X]$, the number of $i$-sets avoiding $X$.

We present two versions of our main result. One counts the number of covers, possibly overlapping, and the other counts the number of partitions. The first result is somewhat simpler and suffices for many of our applications.

### 2.1. Covers

For a positive integer $k \leq n$ let $c_k = c_k(\mathcal{S})$ denote the number of (possibly overlapping) $k$-covers, that is the number of ways to choose $S_1, \ldots, S_k \in \mathcal{S}$ with replacement such that

$$S_1 \cup S_2 \cup \cdots \cup S_k = U. \tag{4}$$

**Theorem 1** *The number of k-covers $c_k$ can be computed in time and space $2^n n^{O(1)}$.*

We establish the theorem in the rest of this subsection. First, we present an inclusion–exclusion formula for $c_k$:

**Lemma 1**

$$c_k = \sum_{X \subseteq U} (-1)^{|X|} s[X]^k. \tag{5}$$

*Proof.* For any $X$, the term $s[X]^k$ counts the number of ways to pick $k$ sets $S_1, \ldots, S_k \in \mathcal{S}[X]$ with replacement. There are two cases.

If $S_1 \cup \cdots \cup S_k = U$, they especially cover $X$, so when $X$ is nonempty, at least one $S_i$ must intersect $X$. Since all $S_i$ were chosen from $\mathcal{S}[X]$ this means that $X$ is empty. In other words, every cover $S_1, \ldots, S_k$ contributes only to the term $(-1)^0 s[\varnothing]^k$.

If $S_1 \cup \cdots \cup S_k = V \neq U$ then the $S_i$ might all have been chosen from $\mathcal{S}[U - V]$, meaning that they contribute (among others) to the term corresponding to $X = U - V$. In fact, they contribute to every subset of $U - V$ as well, including the empty set, so the total contribution of $S_1, \cdots, S_k$ is

$$\sum_{X \subseteq U - V} (-1)^{|X|} = 0.$$

The sum vanishes because every nonempty set has as many even-sized subsets as odd ones.

In summary, the only contribution to $c_k$ is from the choices satisfying (4). ∎

It remains to build a table with $2^n$ entries containing $s[X]$ for all $X \subseteq U$, after which we evaluate (5) in time $2^n n^{O(1)}$. Such a table can be built in several ways. For instance, for every subset $W \subseteq U$ disjoint from $X$ let $s_W[X]$ denote the number of sets $S \in \mathcal{S}$ such that $W \subseteq S$ and $S \cap X = \varnothing$. We seek $s[X] = s_\varnothing[X]$. Since

$$s_W[X] = s_W[X \cup \{v\}] + s_{W \cup \{v\}}[X]$$

holds for all mutually disjoint $X$, $W$, and $v$, we can calculate $s_\varnothing[X]$ recursively in time $O(2^n n^2)$ and space $O(2^n n)$ by peeling off the elements one by one. The factor $n$ reflects the fact that the table entries $s_W[X]$ are $O(n)$-bit numbers. This completes the proof of Thm. 1.

## 2.2. Partitions

For a positive integer $k \leq n$ let $p_k = p_k(\mathcal{S})$ denote the number of $k$-partitions, that is the number of ways to choose $S_1, \ldots, S_k \in \mathcal{S}$ with replacement such that

$$S_1 \cup S_2 \cup \cdots \cup S_k = U, \qquad S_i \cap S_j = \varnothing \quad (i \neq j). \tag{6}$$

Note that the partitions $S_i \cup S_j$ and $S_j \cup S_i$ for $i \neq j$ are here considered different.

**Theorem 2** *The number of k-partitions $p_k$ can be computed in time and space $2^n n^{O(1)}$.*

The proof follows the same melody as that for $c_k$.

**Lemma 2**

$$p_k = \sum_{X \subseteq U} (-1)^{|X|} a_k(X), \tag{7}$$

*where $a_k(X)$ denotes the number of ways to choose k sets $S_1, \ldots, S_k \in \mathcal{S}[X]$, possibly overlapping, such that*

$$|S_1| + \cdots + |S_k| = n. \tag{8}$$

*Proof.* A collection $S_1, \ldots, S_k$ that satisfies (6) will also satisfy (8). As before, such an exact cover will avoid no vertices, so if all sets are chosen from $\mathcal{S}[X]$ then $X$ must be empty. Hence, this collection is counted in $(-1)^0 a_k(\varnothing)$, and only there.

On the other hand, a collection $S_1, \ldots, S_k$ that fails to satisfy (6) contributes nothing to the sum, by exactly the same argument as in the previous proof. ∎

It remains to compute $a_k(X)$. For this we need tables for $s^{(i)}[X]$ instead of just $s[X]$. Letting $s_W^{(i)}[X]$ denote the number of $i$-set $S \in \mathcal{S}^{(i)}$ with $W \subseteq S$ and $S \cap X = \varnothing$, we again observe

$$s_W^{(i)}[X] = s_W^{(i)}[X \cup \{v\}] + s_{W \cup \{v\}}^{(i)}[X],$$

whenever $X$, $W$, and $\{v\}$ are disjoint. Thus we can compute $s^{(i)}[X] = s_\varnothing^{(i)}[X]$ as before, in time $O(2^n n^2)$ and space $O(2^n n)$. We need tables for every $i = 1, \ldots, n$, filling space $O(2^n n^2)$ in total.

To obtain $a_k(X)$ from these, we build yet another table, although this time of polynomial size, using dynamic programming. Let $A(l, m, X)$ denote the number of ways to choose $l$ sets $S_1, \ldots, S_l \in \mathcal{S}[X]$ with replacement such that $|S_1| + \cdots + |S_l| = m$. Then $a_k(X) = A(k, n, X)$. To compute $A(l, m, X)$ we use dynamic programming for $l = 1, \ldots, k$, observing $A(1, m, X) = s^{(m)}[X]$ and

$$A(l, m, X) = \sum_{i=1}^{m-1} s^{(m-i)}[X] A(l-1, i, X).$$

Finally, we sum the $a_k(X)$ according to (7). This completes the proof of Thm. 2.

## 2.3. Polynomial space

Provided that $\mathcal{S}$ itself can be enumerated in polynomial space, algorithms for $c_k$ and $p_k$ in polynomial space and time $c^n$ for some $c \leq 4$ are immediate from our inclusion–exclusion formulas. The precise value depends on how fast we can decide membership in $\mathcal{S}$:

**Theorem 3** *The number of $k$-covers $c_k$ and $k$-partitions $p_k$ can be computed in polynomial space and*

1. *time $2^n|\mathcal{S}|n^{O(1)}$ if $\mathcal{S}$ can be enumerated in polynomial space with polynomial delay,*
2. *time $3^n n^{O(1)}$ if membership in $\mathcal{S}$ can be decided in polynomial time,*
3. *time $\sum_{i=0}^{n}\binom{n}{i}T_{\mathcal{S}}(i)$, where $T_{\mathcal{S}}(i)$ is the time to count the number of elements from $\mathcal{S}$ in an arbitrary $i$-subset of $U$ in polynomial space.*

*Proof.* We give the proof for $c_k$. To evaluate (5) we iterate over all $X$, adding the value of $(-1)^{|X|}s[X]^k$ to a running total. The difficulty is to compute $s[X]$ fast, the number of $S \in \mathcal{S}$ not intersecting $X$. For the first part of the theorem, we enumerate all of $\mathcal{S}$, checking $X \cap S$ for every $S \in \mathcal{S}$. For the second part, we test all $2^{n-|X|}$ sets in the complement of $X$ for membership in $\mathcal{S}$. This amounts to total running time $\sum_{i=0}^{n}\binom{n}{i}2^i n^{O(1)} = 3^n n^{O(1)}$. Finally, if there actually exists a faster polynomial space algorithm to compute $s[X]$ we use that instead. ∎

## 2.4. Extensions

The fact that 'unwanted' combinations of $S_1, \dots, S_k$ cancel in our inclusion–exclusion formulas means that we could put further constraints on these collections, other than just being a cover. In Sec. 3.1 below, we give a concrete example involving their weighted sum $|S_1| + 2|S_2| + 3|S_3| + \cdots$. In the interest of generality, one can give a formulation of our main result that abstracts such constructions using certain predicates $Q(S_1, \dots, S_k)$, but we will be content with the example in Sect. 3.1 to illustrate the idea. Such a general and elegant formulation was recently given by Koivisto [22].

Another extension is given as Thm. 4, which we need for Domatic Number.

Furthermore, a family of polynomial time approximation algorithms follows from Thm. 3 for certain $\mathcal{S}$. We present this for the most interesting and well-studied example, graph colouring, in Sect. 3.2, and then explain under which circumstances it applies to the general case.

## 3. Graph colouring

The *chromatic number* $\chi(G)$ of a graph $G = (V, E)$, $|V| = n$ is the smallest integer $k \leq n$ such that there is a mapping $V \to \{1, \dots, k\}$ (a '$k$-colouring') that gives different values ('colours') to neighbouring vertices. The *chromatic polynomial* is defined by letting $P(G; k)$ denote the number of valid $k$-colourings of $G$.

In this section we choose for $\mathcal{S}$ the family of stable (independent) sets of a graph $G$ on vertices $V$. Notation is simplified by deciding that $\varnothing$ is not stable.

**Lemma 3** $\chi(G) = \min\{k : c_k(\mathcal{S}) > 0\}$.

*Proof.* A legal $k$-colouring is a covering with $k$ stable sets, so if it exists, $c_k(\mathcal{S}) > 0$. On the other hand, if $S_1, \dots, S_k$ cover $V$ (possibly non-distinct and non-disjoint) then $C(v) = \min\{r : v \in S_r\}$ is a legal colouring of size at most $k$. ∎

**Proposition 1** *The chromatic number can be found in time $2^n n^{O(1)}$ and space $2^n n$.*

*Proof.* To find the least $k$ for which $c_k$ is nonzero we perform a binary search among the $c_k$, each of which is found using Thm. 1. ∎

For a fast polynomial space algorithm we can turn to a rich literature about counting stable sets. Very recently, continuing a line of improvements, [16] showed that the stable sets in a $n$-vertex graph can be counted in time bounded by $T(n) = O(1.2461^n)$. This fits into Thm. 3:

**Proposition 2** *The chromatic number can be found in time $\sum_{i=0}^{n}\binom{n}{i}T(i) = O(2.2461^n)$ and polynomial space.*

We turn to the chromatic polynomial.

**Proposition 3** *The number $P(G; k)$ of $k$-colourings of an $n$-vertex graph $G$ can be found in time and space $2^n n^{O(1)}$.*

*Proof.* Every partition into $r$ non-empty stable sets corresponds to $(k)_r = k(k-1)(k-2)\cdots(k-r+1)$ different $k$-colourings, so

$$P(G; k) = \sum_{r=1}^{k} \frac{k!}{(k-r)!}\,\frac{p_r(\mathcal{S})}{r!} = \sum_{r=1}^{k}\binom{k}{r}p_r(\mathcal{S}),$$

which can be computed using Thm. 2. ∎

Since $P(G; \cdot)$ is known to be a degree $n$ polynomial we can recover its coefficients by computing $P(G; k)$ at $k = 0, 1, 2, \dots, n$ and interpolating the unique polynomial through these points. This representation then allows us to evaluate the chromatic polynomial at other points, such as computing $P(G; -1)$, the number of acyclic orientations of $G$ [30].

## 3.1. Chromatic Sum

The Chromatic Sum problem (also called Minimum Colour Sum) is to find a colouring $C : V \to \{1, \ldots, n\}$ that minimises $\sum_{v \in V} C(v)$. An algorithm for Chromatic Sum is not a direct consequence of our main theorems, but we can still solve it with the same techniques:

**Proposition 4** *The chromatic sum of a graph can be determined in time and space $2^n n^{O(1)}$.*

*Proof.* We count the number of solutions $q_{k,l}$ having chromatic sum at most $k$ using $l$ colours; for each $l \in [1 \ldots n]$ we do a binary search over the range of possible chromatic sums $k \in [1 \ldots l^2]$.

To compute $q_{k,l}$ we need another inclusion–exclusion formula:

$$q_{k,l} = \sum_{X \subseteq V} (-1)^{|X|} b_{k,l}(X), \tag{9}$$

where $b_{k,l}(X)$ is the number of ways to choose $l$ stable sets $S_1, \ldots, S_l \in \mathcal{S}[X]$, such that

$$|S_1| + 2|S_2| + \cdots + l|S_l| \leq k.$$

The proof of (9) is similar to that of Lemma 1. Again, $b_{k,l}(X)$ can be found in polynomial time by dynamic programming. ∎

## 3.2. Approximating the chromatic number

Combining our exact algorithm with the well-known technique of successively removing the largest stable sets we arrive at a fast approximation algorithm for the chromatic number.

**Proposition 5** *For every $\epsilon > 0$, the chromatic number $\chi$ of a graph on $n$ vertices can be approximated by a value $\bar{\chi}$ obeying $\chi \leq \bar{\chi} \leq \lceil (1+\epsilon)\chi \rceil$ which can be found in polynomial space and time $O(1.2209^n + 2.2461^{e^{-\epsilon}n})$.*

*Proof.* Fix some $\epsilon > 0$. We will perform the following operation a number of times:

Find the largest stable set and remove it from the graph. Repeat until the graph has at most $e^{-\epsilon}n$ vertices. Let $s$ be the number of thus removed stable sets. We run the exact algorithm in Prop. 2 for the resulting graph to find its chromatic number $\chi_0$. Our approximation is $\bar{\chi} = \chi_0 + s$.

We need to argue $\bar{\chi}$ is not far from the actual chromacity. First note that $\bar{\chi} \geq \chi$ since the subgraph obtained after removing a stable set has chromacity at least $\chi - 1$. Second, $\chi_0 \leq \chi$ since a subgraph cannot have larger chromacity than its host graph. We note that $s \leq t$ for integer $t$ obeying

$$(1 - 1/\chi)^t \leq e^{-\epsilon}$$

since every graph with chromacity $\chi$ has stable set consisting of at least a fraction $1/\chi$ of its vertex set. Furthermore, $(1 - 1/\chi)^t \leq e^{-t/\chi}$ and thus $s \leq \lceil \epsilon\chi \rceil$.

Turning to the running time, we note that the fastest known polynomial space algorithm finding a largest stable set in a graph runs in time $O(1.2209^n)$ [14]. ∎

*Discussion.* The above approximation idea translates to the general case of finding a minimal covering provided $\mathcal{S}$ has the following properties:

1. there is a fast algorithm to find the largest $S \in \mathcal{S}$.
2. $\mathcal{S}$ is hereditary, that is $S \subset T \in \mathcal{S}$ implies $S \in \mathcal{S}$.

An example of an interesting family of sets that is *not* hereditary is given by the induced trees of a graph. On the other hand, the induced forests *are* a hereditary family. In fact, some recent algorithms [26, 13] find a maximum induced forest in time $O(1.7548^n)$, satisfying also the first requirement. Thus our constructions give a good approximation algorithm for finding a small partition into induced forests.

## 3.3. Finding an optimal colouring

Because our algorithms are based on evaluating a formula, they return the chromatic number without ever constructing a corresponding colouring. Of course, we can iteratively contract certain vertex pairs, computing $\chi(G)$ at each step to guide our search, and recover an optimal colouring within the same time bounds. This is standard and included only for completeness.

Pick a vertex $v \in V$ and enumerate the vertices $u_1, \ldots, u_k$ not incident to $v$. For $1 \leq i \leq k$, consider the graphs $G_i$ formed by adding the missing edges,

$$V(G_i) = V(G), \quad E(G_i) = E(G) \cup \{vu_1, \ldots, vu_i\}.$$

Observe that the sequence of chromatic numbers $\chi(G) = \chi(G_0), \chi(G_1), \ldots, \chi(G_k)$ cannot decrease, and increases by at most one at each step. If $\chi(G) = \chi(G_k)$ then there is an $\chi(G)$-colouring of $G$ in which $v$ has a different colour than the rest of the vertices; we can remove it and its incident edges from $G$ and look for a $(\chi(G) - 1)$-colouring in the resulting graph. If $\chi(G) < \chi(G_k)$ we can find the smallest $i$ such that $\chi(G_i) = \chi(G) + 1$ using binary search. We infer from this that in some optimal colourings of $G_{i-1}$ (and $G$), the vertices $v$ and $u_i$ received the same colour. Hence we can contract $vu_i$ in $G_i$ and continue in the resulting graph.

Each iteration removes a vertex and incurs $O(\log n)$ computations of $\chi$. Let $T_\chi(n)$ denote the running time of our chromatic number algorithm, then the total running time is $O\big((T_\chi(n) + T_\chi(n-1) + T_\chi(n-2) + \cdots + 1)\log n\big) = O\big(T_\chi(n)\log n\big)$, since our algorithm is exponential.

## 4. Domatic Number

The *domatic number* $\delta(G)$ of a graph $G = (V, E)$, $|V| = n$ is the largest integer $k$ such that there is a partition $V_1 \cup V_2 \cup \cdots \cup V_k = V$ into pairwise disjoint subsets $V_i$ that dominate $G$. A vertex set $S \subseteq V$ *dominates* the graph if each vertex in $V$ has distance at most one to a vertex in $S$. Note that for maximization partition problems, the simpler $c_k$ cannot be used. We need to use $p_k$:

**Proposition 6** *The domatic number can be found in time and space* $2^n n^{O(1)}$.

*Proof.* To find the largest $k$ for which $p_k$ is nonzero we perform a binary search among the $p_k$, each of which is found using Thm. 2, with $\mathcal{S}$ being the dominating sets of $G$. ∎

A polynomial space algorithm for domatic number time $3^n n^{O(1)}$ is immediate from Thm. 3. We can do better by considering a sparser family of sets $\mathcal{S}$ that can be quickly counted.

**Proposition 7** *The domatic number can be found in time* $O(2.8718^n)$ *and polynomial space.*

The result is established in the remainder of this section.

A dominating set $W$ is *minimal* if removing any of its vertices destroys the dominance property. Fomin *et al.* [15] observed that enumerating all minimal dominating sets in a graph can be done much faster than $2^n$ (They derive the upper bound $1.7170^n$). This suggests an approach to get a faster algorithm since $\delta \geq k$ if and only if there are $k$ pairwise disjoint minimal dominating sets $W_1, \cdots, W_k$. Unfortunately, their union need not exhaust all of $V$, so Lem. 2 does not apply directly. We need the more general result below. As before, $U$ is an $n$-element set and $\mathcal{S}$ is a family of subsets of $U$.

For positive numbers $k, m \leq n$, let $l_{k,m} = l_{k,m}(\mathcal{S})$ denote the number of ways to choose $S_1, \ldots, S_k \in \mathcal{S}$ such that

$$\begin{aligned} |S_1| + |S_2| + \cdots + |S_k| &= n - m, \\ S_i \cap S_j &= \varnothing \quad (i \neq j). \end{aligned} \tag{10}$$

**Theorem 4** *The number $l_{k,m}$ of partitions satisfying* (10) *can be computed in time*

$$n^{O(1)} \sum_{i=0}^{n} \binom{n}{i} \sum_{j=0}^{n} T_{\mathcal{S}}(i, j)$$

*and polynomial space, where $T_{\mathcal{S}}(i, j)$ is the time to count the number of sets from $\mathcal{S}$ of size $j$ in an arbitrary $i$-subset of $U$.*

The proof is a bit different from the previous cases of $k$-covers and $k$-partitions in that the solution is not given by an inclusion–exclusion formula, but as another weighted sum. Rather than giving the formula explicitly, we describe it implicitly through a linear equation system:

Let $b_{k,m,j}$ be the number of ways to choose subsets $S_1, \cdots, S_k \in \mathcal{S}$ with replacement such that $|\bigcup_i S_i| = n - j$ and $\sum_i |S_i| = n - m$. Let

$$d_{k,m,j} = \sum_{X \subseteq U, |X| = j} a_{k,m}(X),$$

where $a_{k,m}(X)$ is the number of ways to choose $k$ subsets $S_1, \cdots, S_k \in \mathcal{S}[X]$ such that $\sum_i |S_i| = n - m$.

**Lemma 4** *The two $(n+1) \times 1$ column vectors $\hat{b} = (b_{k,m,\cdot})$ and $\hat{d} = (d_{k,m,\cdot})$ fulfil the linear equation system $A\hat{b} = \hat{d}$, where $A$ is a $(n+1) \times (n+1)$-matrix of full rank over the reals.*

*Proof.* We need some additional notation. Let $\mathcal{C}$ be the family of *candidate* $k$-subsets $C = \{S_1, \cdots, S_k\} \subseteq \mathcal{S}$, obeying $\sum_i |S_i| = n - m$. For each candidate $C \in \mathcal{C}$ we define its *remainder*, denoted $R(C)$, as $\left|U - \left(\bigcup_{S \in C} S\right)\right|$, i.e. the number of elements not covered by any $S \in C$. We can partition the candidates $\mathcal{C}$ in equivalence classes according to there remainder as $\mathcal{C} = \mathcal{C}_0 \cup \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_{n-1}$ where $\mathcal{C}_i = \{C \colon C \in \mathcal{C}, R(C) = i\}$ and we observe that $b_{k,m,j} = |\mathcal{C}_j|$. Next note that every $C \in \mathcal{C}_i$ is counted $\binom{i}{j}$ times each in $d_{k,m,j}$. Thus the matrix element at row $i$ and column $j$ is $A_{i,j} = \binom{i}{j}$. In particular, $A_{i,i} = 1$ whereas for $i < j$, $A_{i,j} = 0$. Consequently, $A$ is lower triangular with non-zero elements on the diagonal and hence it has full rank. ∎

The previous lemma tells us how to compute all of the numbers $l_{k,m} = b_{k,m,m}$ in polynomial time by solving the linear equation system once the numbers $d_{k,m,j}$ are known. The latter numbers can be computed by dynamic programming in polynomial time as in the case of $k$-partitions from the numbers $s^{(i)}[X]$. Calculating these in their turn requires time $\sum_{i=0}^{n} \binom{n}{i} \sum_{j=0}^{n} T_{\mathcal{S}}(i, j)$. This finishes the proof of Thm. 4.

Turning back to the domatic number, we note that $s^{(i)}[X]$ for a subset $X \subseteq V$ refers to the number of minimal dominating sets of size $i$ of $G$ contained in the vertex set $V - X$. Counting all of these takes time at most $\lambda^{n+\alpha_4(n-|X|)}$ for $\lambda < 1.1487$ and $\alpha_4 = 2.9248$ using the enumeration algorithm and notation from [15, proof of Thm. 5.1] (However, as the authors point out, it is unclear exactly how close this bound is from the true running time of their branching algorithm). By using their bound in theorem 4 we finally arrive at Prop. 7.

## References

[1] O. Angelsmark and J. Thapper. Partitioning based algorithms for some colouring problems. In *Recent Advances in Constraints*, volume 3978 of *LNAI*, pages 44–58. Springer Verlag, Berlin, 2005.

[2] M. H. G. Anthony. Computing chromatic polynomials. *Ars Combinatorica*, 29(C):216–220, 1990.

[3] R. Beigel and D. Eppstein. 3-coloring in time $o(1.3289^n)$. *J. Algorithms*, 54(2):168–204, 2005.

[4] N. Biggs. *Algebraic graph theory*. Cambridge University Press, 2nd edition, 1993.

[5] A. Björklund and T. Husfeldt. Exact algorithms for exact satisfiability and number of perfect matchings. In *Proc. 33rd ICALP*, LNCS volume 4051, pages 548–559, 2006.

[6] H. L. Bodlaender and D. Kratsch. An exact algorithm for graph coloring with polynomial memory. Technical Report UU-CS-2006-015, Utrecht University, 2006.

[7] J. M. Byskov. Enumerating maximal independent sets with applications to graph colouring. *Operations Research Letters*, 32:547–556, 2004.

[8] J. M. Byskov. *Exact algorithms for graph colouring and exact satisfiability*. PhD thesis, University of Aarhus, 2004.

[9] J. M. Byskov and D. Eppstein. An algorithm for enumerating maximal bipartite subgraphs. Manuscript, 2004.

[10] N. Christofides. An algorithm for the chromatic number of a graph. *Computer J.*, 14:38–39, 1971.

[11] D. Eppstein. Small maximal independent sets and faster exact graph coloring. *J. Graph Algorithms and Applications*, 7(2):131–140, 2003.

[12] T. Feder and R. Motwani. Worst-case time bounds for coloring and satisfiability problems. *J. Algorithms*, 45(2):192–201, 2002.

[13] F. V. Fomin, S. Gaspers, and A. V. Pyatkin. Finding a minimum feedback vertex set in time $o(1.7548^n)$. In *Proc. 2nd IWPEC*, volume 4169 of *LNCS*, pages 184–191, 2006.

[14] F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: A simple $o(2^{0.288n})$ Independent Set algorithm. In *Proc. 17th SODA*, pages 18–25, 2006.

[15] F. V. Fomin, F. Grandoni, A. V. Pyatkin, and A. A. Stepanov. Combinatorial bounds via measure and conquer: Boundings minimal dominating sets and applications. Prelim.version in *Proc. 16th ISAAC*, pages 573–582, 2006.

[16] M. Fürer and S. P. Kasiviswanathan. Algorithms for counting 2-SAT solutions and colorings with applications. Technical Report 05-033, Elect. Coll. Comput. Compl., 2005.

[17] M. Garey and D. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman, San Francisco, 1979.

[18] C. Greenhill. The complexity of counting colourings and independent sets in sparse graphs and hypergraphs. *Computational Complexity*, 9:52–73, 2000.

[19] M. M. Halldórsson. A still better performance guarantee for approximate graph coloring. *Information Processing Letters*, 45:19–23, 1993.

[20] M. Held and R. Karp. A dynamic programming approach to sequencing problems. *SIAM J. Appl. Math.*, 10:196–210, 1962.

[21] R. M. Karp. Dynamic programming meets the principle of inclusion-exclusion. *Oper. Res. Lett.*, 1:49–51, 1982.

[22] M. Koivisto. An $O(2^n)$ algorithm for graph coloring and other partitioning problems via inclusion-exclusion. In *Proc. 47th FOCS*, 2006.

[23] E. L. Lawler. A note on the complexity of the chromatic number problem. *Information Processing Letters*, 5(3):66–67, 1976.

[24] B. A. Madsen. An algorithm for exact satisfiability analysed with the number of clauses as parameter. *Information Processing Letters*, 97(1):28–30, 2006.

[25] J. W. Moon and L. Moser. On cliques in graphs. *Israel J. Math.*, 1965.

[26] I. Razgon. Exact computation of maximum induced forest. In *Proc. 10th SWAT*, LNCS volume 4059, pages 160–171, 2006.

[27] T. Riege and J. Rothe. An exact $2.9416^n$ algorithm for the three domatic number problem. In *Proc. 30th MFCS*, LNCS, 2005.

[28] T. Riege, J. Rothe, H. Spakowski, and M. Yamamoto. An improved exact algorithm for the domatic number problem. In *Proc. 2nd ICTTA*, 2006.

[29] H. J. Ryser. *Combinatorial Mathematics*. Number 14 in Carus Math. Monographs. Math. Assoc. America, 1963.

[30] R. P. Stanley, Acyclic Orientations of Graphs. *Disc. Math.* 5, pages 171-178, 1973.

[31] S. Vadhan. The complexity of counting, 1995.

[32] H. S. Wilf. *Algorithms and complexity*. Prentice–Hall, 1986.

[33] G. J. Woeginger. Exact algorithms for NP-hard problems: a survey. In *Combinatorial optimization: Eureka, you shrink!*, pages 185–207. Springer, 2003.

[34] G. J. Woeginger. Space and time complexity of exact algorithms: Some open problems. In *Proc. 1st IWPEC*, LNCS volume 3162, pages 281–290, 2004.

[35] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. Technical Report 05-100, Elect. Coll. Comput. Compl., 2005.