# INCLUSION–EXCLUSION BASED ALGORITHMS FOR GRAPH COLOURING

ANDREAS BJÖRKLUND AND THORE HUSFELDT

Abstract. We present a deterministic algorithm producing the number of $k$-colourings of a graph on $n$ vertices in time $2^n n^{O(1)}$. We also show that the chromatic number can be found by a polynomial space algorithm running in time $O(2.2461^n)$. Finally, we present a family of polynomial space approximation algorithms that find a number between $\chi(G)$ and $(1+\epsilon)\chi(G)$ in time $O(1.2209^n + 2.2461^{e^{-\epsilon}n})$.

## 1. Introduction

The *chromatic number* 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 $G$.

Our main result is an algorithm that computes $\chi(G)$ in time $2^n n^{O(1)}$. This punctuates a history of successive improvements since Lawler's $O(2.4423^n)$ algorithm (Table 1) and answers in the affirmative an open question [22]. The algorithm is self-contained and elementary, the time and correctness analyses are from first principles.

### 1.1. **Inclusion–exclusion formulations.**

The algorithmic results in this paper are based on a characterisation of the chromatic number as the smallest $k$ for which

$$\sum_{X \subseteq V} (-1)^{|X|} s(X)^k \tag{1}$$

is nonzero. Here, $s(X)$ is the number of stable sets of $G$ not intersecting $X$. For the chromatic polynomial we use

$$p_G(k) = \sum_{r=1}^{n} \frac{k!}{(k-r)!} \left( \sum_{X \subseteq V} (-1)^{|X|} a_r(X) \right), \tag{2}$$

where $a_r(X)$ denotes the number of ways to choose $r$ stable sets $S_1, \dots, S_r \subseteq V - X$, such that $|S_1| + \cdots + |S_r| = n$.

To the best knowledge of the authors, these characterisations are new and might be of independent combinatorial interest; in any case, their proofs are elementary.

*Date*: January 31, 2006.

*Key words and phrases.* Graph algorithms, graph colouring, exact algorithms.

Lund University, Department of Computer Science. PO Box 118, SE-221 00 Lund, Sweden. Corresponding author: Thore.Husfeldt@cs.lu.se.

| Time $O(c^n)$ | Reference |
|---|---|
| $c = 2.4423$ | Lawler [16] |
| 2.4151 | Eppstein [10] |
| 2.4023 | Byskov [6] |
| 2.3236 | Björklund and Husfeldt [5] |
| 2.2590 | Beigel and Eppstein [3], randomised, $\chi(G) \leq 5$ |
| 2.1592 | Byskov [6], $\chi(G) \leq 5$ |
| 2.1020 | Byskov and Eppstein [4], $\chi(G) \leq 5$ |
| 2.1809 | *ibid.*, $\chi(G) \leq 6$ |

TABLE 1. Previous exponential space algorithms for finding $\chi(G)$, or deciding $\chi(G) \leq k$ for small constants $k$.

1.2. **Algorithms.** To find $\chi(G)$ we evaluate the $2^n$ terms of (1); the only challenge is to compute the $s(X)$ quickly. Through a recursion formula for $s(X)$ and memoization in an exponential-size table, we get the desired $2^n n^{O(1)}$ time bound, the main result of this paper.

If only polynomial space is available, the values $s(X)$ for $|X| = i$ can obviously be computed in time $O(2^i i)$ by a Gray-code enumeration of the subsets of $X$ , for a total running time within a polynomial factor of $\sum_{i=1}^{n} \binom{n}{i} 2^i = 3^n$. Until very recently [5], finding a polynomial space algorithm with running time $c^n$ for any $c$ was an open problem [7, 17]. Using instead the fastest currently known algorithm in the literature for counting stable sets to compute $s(X)$, the total running time to evaluate (1) becomes $O(2.2461^n)$, which is the best known for polynomial space. Note however, that the fastest algorithms for computing $s(X)$, and their analysis, are far from simple.

Because our construction is based on evaluating a formula, we learn little else from it than the value of $\chi(G)$. For example, the algorithm does not *construct* an optimal colouring. We show how to obtain one from the chromatic number algorithm. By using (2) instead, still in exponential time and space $2^n n^{O(1)}$, we show how to compute the number of $k$-colourings $p_G(k)$ of $G$ and how to solve the related Chromatic Sum problem.

Finally, we derive a family of exponential-time approximation algorithms based on the widely-used idea of iteratively removing large independent sets and applying our ideas on the remaining graph. For instance, we can approximate $\chi(G)$ within a factor 2 in time $O(1.3998^n)$ 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 $O(n \log^{-3} n \log\log^2 n)$ [15], and $\chi(G)$ is NP-hard to approximate within $n^{1-o(1)}$ [23].

1.3. **Previous work and discussion.** The first non-trivial algorithm for finding the chromatic number, by Christofides [8] 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 [16], see Table 1, all of which are based on finding maximal independent sets and owing their running time ultimately to the fact that there are only $3^{n/3}$ maximal independent sets in a graph [18].

For polynomial space, Feder and Motwani [11] 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 $\chi$. In a precursor to the present paper, the authors presented algorithms with running times $O(8.33^n)$ 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 independent sets is $\#P$-complete [20, 14]. The existence of such an algorithm remains open.

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 $O(1.7504^n)$ [6], respectively, well beyond the reach of our constructions.

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 [19]. Also, Bax [2] 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. We note that the technique can be almost as powerful when counting covers assembled from an *exponential* number of larger subsets of the vertex set. A precursor paper by the authors [5] already tentatively explores this idea, but the constructions there are still based on maximal independent sets, much slower, and more complicated.

Given the simplicity of our arguments it seems curious that they were not discovered before. Reconsidering the literature in the light of hindsight we find that in 1932, Whitney [21] proved

$$p_G(k) = \sum_{i=0}^{n-1} (-1)^i a_i k^{n-i}, \tag{3}$$

where $a_i$ is the number of $i$-subsets of $E(G)$ containing no *broken cycle*. (Given an ordering $E = \{e_1, \ldots, e_m\}$ of the edges, a broken cycle is a cycle missing the edge of highest index.) However, how to obtain $a_i$ more efficiently than explicitly check each of the $2^m$ subsets of $E(G)$ for broken cycles is unclear; it seems difficult to calculate these values in time close to $2^n$ for dense graphs. In any case, the resulting arguments, including the proof of (3) itself, would be more complicated.

## 2. Results

2.1. **Inclusion–exclusion formula.** Let $\mathcal{S}$ denote the family of stable (independent) sets of a graph $G$ on vertices $V$. (Notation is simplified by deciding that $\varnothing$ is not stable.)

For a set of vertices $X \subseteq V$ we write $\mathcal{S}[X] = \{\, S \in \mathcal{S} \colon S \subseteq X \,\}$ for its stable subsets. We write $s(X) = |\mathcal{S}[V - X]|$, the number of stable sets not intersecting $X$, and we let $s^{(i)}(X)$ be the number of stable sets in $\mathcal{S}[V - X]$ of size $i$. Let $c_k(G)$ denote the number of ways to cover $V$ with $k$ stable sets, possibly overlapping and non-distinct.

**Lemma 1.** $\chi(G) = \min\{\, k \colon c_k(G) > 0 \,\}$.

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

**Lemma 2.**

$$c_k(G) = \sum_{X \subseteq V} (-1)^{|X|} s(X)^k. \tag{4}$$

*Proof.* $s(X)^k$ counts the ways to pick $k$ sets from $\mathcal{S}[V - X]$ with replacement. Note that when a set of $k$ stable sets cover $V$ it is counted only in $s(\varnothing)$. Every other selection of $k$ sets avoids some vertices $U$ and is counted once in every $s(W)$ for every subset $W \subseteq U$ but in no other terms. Since every non-empty set has as many even-sized subsets as odd ones, the result follows. $\square$

## 2.2. **Algorithms for chromatic number.**

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

*Proof.* To compute (4) we need to evaluate $s(X)$ for every subset $X \subseteq V$.

We can compute $s(X)$ by the recursive formula

$$s(X) = s\big(X \cup \{v\}\big) + s\big(X \cup \{v\} \cup N(v)\big) + 1, \qquad v \notin X \tag{5}$$

where $N(v)$ is the set of neighbours of $v$. By storing computed values to avoid recomputing them later, we can get all $s(X)$ in time $O(2^n n)$; the linear factor accounts for $s(X)$ being an $n$-bit number.

To find the least $k$ for which (4) is nonzero we perform a binary search. $\square$

Without the exponential space needed to memorize previous values, we need to compute $s(X)$ anew for each term of (4). We can do this by applying (5) solely, but this will result in execution times as large as $2^{|X|}$ when the graph induced by $X$ is sparse. Very recently, [13] continuing a line of improvements, notes that by combining (5) by other strategies for sparse graphs, the time $T_{\text{stable}}(n)$ to count the stable sets in a $n$-vertex graph can be bounded by $O(1.2461^n)$.

**Proposition 2.** *The chromatic number can be found by a polynomial space algorithm in time*

$$\sum_{i=0}^{n} \binom{n}{i} T_{\text{stable}}(i) = O(2.2461^n).$$

*Proof.* To compute (4) we evaluate $s(X)$ for every subset $X \subseteq V$. $\square$

We note that the polynomial factors in Prop. 1 and hidden in the asymptotic bound in Prop. 2 can be reduced by repeatedly counting modulo a small integer instead of once for the full value.

**Lemma 3.**

$$c_k(G) \equiv \sum_{i=1}^{p-1} a_i i^k \pmod{p}$$

*where $a_i$ is the difference between the number of even sized subsets and odd sized subsets satisfying $s(X) \equiv i \pmod{p}$.*

*Proof.* Sum over the range of possible values for $s(X) \bmod p$ in (4). □

By calculating and storing $s(X) \mod p$ and evaluating Lem. 3 we save both time and space. However, we need to run the algorithm for several choices of $p$ to be certain to get the right value for $\chi$. If we are content with a randomized algorithm returning the correct value almost surely, we can settle for only one $p$ though.

Let $R$ be the set of the largest prime powers less than $n^4$ for each prime less than $n^4$. Note that each element of $R$ is larger than $n^2$. We pick a prime power $p \in R$ uniformly at random. The event that $p$ is bad in the sense that for some $k$, $c_k \equiv 0 \pmod p$ even though $c_k \neq 0$, is very unlikely. This is because $|R| \sim n^4/\log n$ by the prime number theorem, and $c_k < 2^{n^2}$ for all $k$. Thus each $c_k$ can be divisible by at most $n^2/\log n$ numbers in $R$, and hence the probability that we pick a bad $p$ is $O(1/n)$.

2.3. **Number of colourings.** The algorithm for chromatic number does not tell us the number $p_G(k)$ of $k$-colourings. To find those values, we need a slightly clumsier construction.

Let $\pi_G(k)$ denote the number of ways to partition $G$ into $k$ stable sets, i.e., the number of ways to choose stable sets $S_1, \ldots, S_k \in \mathcal{S}$ satisfying:

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

It is clear that every legal colouring induces such a partition, so $\chi(G) = \min\{\, k \colon \pi_G(k) > 0 \,\}$. Hence these numbers can be used to find the chromatic number as well.

**Lemma 4.** *For $X \subseteq V$ let $a_k(X)$ denote the number of ways to choose $k$ stable sets $S_1, \ldots, S_k \in \mathcal{S}[V - X]$, such that*

$$|S_1| + \cdots + |S_k| = n. \tag{7}$$

*Then*

$$\pi_G(k) = \sum_{X \subseteq V} (-1)^{|X|} a_k(X). \tag{8}$$

*Proof.* If $S_1, \ldots, S_k$ form a colouring then they are disjoint and cover the whole graph, so $|S_1| + \cdots + |S_k| = n$ and $S_i \cap X$ is empty (for all $i$) only for $X = \varnothing$. Thus every $k$-colouring contributes (once) to the term $a_k(\varnothing)$.

On the other hand, every non-covering $k$-set family avoids some vertices $U \subset V$ and thus contributes once to the terms corresponding to every subset $X \subseteq U$ but no other terms. These contributions cancel because every set has as many even sized subsets as it has odd ones. □

**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.* We compute $s^{(i)}(X)$ recursively, storing obtained values in a table as we go along. However, this time we need to build tables for all $s^{(i)}(X)$, instead of for $s(X)$. To obtain $a_k(X)$, we use dynamic programming. Let $A(l, m, X)$ denote the number of ways to choose $l$ stable sets $S_1, \ldots, S_l \in \mathcal{S}[V - X]$ with replacement such that

$$|S_1| + \cdots + |S_l| = m.$$

Then $a_k(X) = A(k, n, X)/k!$. We can compute $A(l, m, X)$ for $l = 1, \ldots, n$ by 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, every partition into $r$ non-empty independent sets corresponds to $(k)_r = k(k-1)(k-2)\cdots(k-r+1)$ different $k$-colourings, so

$$p_G(k) = \sum_{r=1}^{n} \pi_G(r)(k)_r.$$

$\square$

## 2.4. **Algorithm for chromatic sum.**

The Chromatic Sum problem (sometimes called Minimum Colour Sum) is to find a colouring $C : V \to \{1, \ldots, n\}$ that minimises

$$\sum_{v \in V} C(v)$$

(rather than $\max_{v \in V} C(v)$).

**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 $q_{k,l}(G)$ 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 $k \in [1 \ldots l^2]$, and calculate

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

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

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

Again, $b_{k,l}(X)$ can be found in polynomial time by dynamic programming. $\square$

## 2.5. **Finding an optimal colouring.**

Our algorithms only return the chromatic number or the chromatic sum, without ever constructing a corresponding colouring. For completeness, we outline how this can be done.

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,

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

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 $O\big((T_\chi(n) + T_\chi(n-1) + T_\chi(n-2) + \cdots + 1)\log n\big) = O(T_\chi(n)\log n)$, since our algorithm is exponential.

## 2.6. **Approximating the chromatic number.**

We can find a good approximation of the chromatic number much faster than the precise value.

**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 independent 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 independent 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 an independent 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 $t$ obeying

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

since every graph with chromacity $\chi$ has an independent 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 independent set in a graph runs in time $O(1.2209^n)$ [12]. $\square$

## References

[1] O. Angelsmark and J. Thapper. Partitioning based algorithms for some colouring problems. In *Proc. 5th Workshop on Constraint Solving and constraint logic programming (CSCLP)*, pages 28–42, 2005.

[2] E. T. Bax. Inclusion and exclusion algorithm for the Hamiltonian Path problem. *Inf. Process. Lett.*, 47(4):203–207, 1993.

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

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

[5] A. Björklund and T. Husfeldt Polynomial space algorithms for exact satisfiability and chromatic number. Manuscript, 2005.

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

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

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

[9] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, 1962.

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

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

[12] F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and Conquer: A Simple $O(2^{0.288n})$ Independent Set Algorithm. *SODA* , 2006.

[13] M. Fürer and S. P. Kasiviswanathan. Algorithms for Counting 2-SAT Solutions and Colorings with Applications. ECCC TR05-33, 2005.

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

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

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

[17] B. A. Madsen. An algorithm for exact satisfiability analysed with the number of clauses as parameter. *Information Processing Letters*, to appear:215–217, 2005.

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

[19] H. J. Ryser. Combinatorial Mathematics. The Carus Mathematical Monographs No. 14, Mathematical Association of America, 1963.

[20] S. Vadhan. The complexity of counting. Undergraduate thesis, Harvard University, 1995.

[21] H. Whitney A logical expansion in mathematics. Bull. Amer. Math. Soc. 38:572–579, 1932.

[22] G. Woeginger Exact algorithms for NP-hard problems: A survey. In *Combinatorial optimization – Eureka! You shrink*, LNCS 2570, 2003.

[23] D. Zuckerman Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. ECCC TR05-100, 2005.