An exact algorithm for graph coloring with polynomial memory

Hans L. Bodlaender

Dieter Kratsch

Department of Information and Computing Sciences,
Utrecht University

Technical Report UU-CS-2006-015

www.cs.uu.nl

ISSN: 0924-3275

An exact algorithm for graph coloring with polynomial memory

Hans L. Bodlaender[*] Dieter Kratsch[†]

Abstract

In this paper, we give an algorithm that computes the chromatic number of a graph in $O(5.283^n)$ time and polynomial memory.

1 Introduction

During the last years the interest in designing and analysing exact exponential-time algorithms for NP-hard problems has been growing significantly. Despite many other approaches to attack NP-hard problems these algorithms really aim at coping with NP-hardness, i.e. they compute an exact solution for all possible inputs. When dealing with intractable problems, exact algorithms are not expected to have polynomial running time.

See the surveys by Woeginger [6, 7] for an introduction to the area of exponential time algorithms.

Exponential worst case running time implies that such algorithms cannot solve inputs of large size in reasonable time. Some exponential-time algorithms use exponential memory and some need only polynomial memory. For instance, Branch & Reduce algorithms typically use polynomial memory while dynamic programming algorithms often need exponential memory.

Clearly, having two algorithms of same running time one would prefer one using polynomial memory. In fact memory requirements of an algorithm can be such an obstacle for its use on larger inputs, that polynomial memory could be desirable even when paying the price of higher running time. We refer to [7] for discussions of polynomial vs. exponential memory. This motivated us to study a polynomial memory algorithm for a classical NP-hard problem – the graph coloring problem.

All known exponential-time algorithms to compute the chromatic number of a graph (and an optimal coloring) need exponential memory, more precisely $O^*(2^n)$ memory, and they all are based on a dynamic programming approach and the use of maximal independent

[*]Department of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, the Netherlands, hansb@cs.uu.nl.

[†]Laboratoire d'Informatique Théorique et Appliquée, Université Paul Verlaine - Metz, 57045 Metz Cedex 01, France, kratsch@univ-metz.fr.

sets. The first one had been published by Lawler in *Information Processing Letters* 30 years ago [4]. Its running time is $O^*((1+\sqrt[3]{3})^n) = O(2.4423^n)$. This algorithm had not been improved for 25 years. Then the combination of improved upper bounds on the number of maximal independent sets of size at most k and changes in the way to fill the table of subsolutions in the dynamic programming algorithm led to better algorithms. Eppstein established an $O^*((4/3 + \frac{3^{4/3}}{4})^n) = O(2.4151^n)$ time algorithm. Finally Byskov provided an $O(2.4023^n)$ algorithm to compute the chromatic number of a graph. Faster algorithms exist when the number of colorings is fixed. For small fixed numbers of colors, faster algorithms are known. The currently best known bounds for c-coloring are: $O^*(1.3289^n)$ for $c = 3$ (Beigel and Eppstein [1]), $O^*(1.7504^n)$ for $c = 4$ (Byskov [2]), $O^*(2.1020^n)$ for $c = 5$ (Byskov and Eppstein, see [3]), and $O^*(2.3289^n)$ for $c = 6$ (Byskov [2]). Each of these algorithms uses polynomial memory. Byskov and Eppstein also give an algorithm using $O^*(2^n)$ memory and $O^*(2.1809^n)$ time for 6-coloring, see [3]. We refer to the PhD thesis of Byskov for a nice summary on exponential-time coloring algorithms [3].

As a first answer to the natural question about the best possible worst case running time of an exact coloring algorithm using only polynomial memory, we present an $O(5.283^n)$ time and polynomial space algorithm to compute the chromatic number of any graph.

2 Definitions

We consider undirected, simple and finite graphs $G = (V, E)$ with vertex set V and edge set E. A vertex subset $I \subseteq V$ is an *independent set* if any two vertices x and y of I are non adjacent. A (proper) coloring c of a graph $G = (V, E)$ assigns a color $c(v)$ to each vertex v of G such that $c(u) \neq c(v)$ if $uv \in E$. The smallest possible number of colors used by a coloring of G is the *chromatic number* of G, denoted by $\chi(G)$. A coloring of G using $\chi(G)$ colors is an *optimal coloring* of G. Notice that a coloring of a graph G induces a partition of its vertex set into independent sets $c = (I_1, I_2, \ldots, I_k)$.

3 An exact algorithm for graph coloring

In this section, we describe our algorithm. The algorithm is based upon the following simple lemma.

Lemma 1 *Let $0 < \alpha < 1$, and let $G = (V, E)$ be a graph with n vertices. The chromatic number of G is the minimum of the following two terms.*

- *The minimum over all maximal independent sets I with $|I| \geq \alpha \cdot n$ in G of $1 + \chi(G[V - I])$.*
- *The minimum over all sets of vertices $S \subseteq V$ with $(n - \alpha \cdot n)/2 \leq |S| \leq n/2$ of $\chi(G[S]) + \chi(G[V - S])$.*

Proof. Consider an optimal coloring $c = (I_1, I_2, \ldots, I_k)$ of G. Suppose there is an I_j, $j \in \{1, 2, \ldots, k\}$, such that $|I_j| \geq \alpha \cdot n$. Then there is an optimal coloring of G, such that one color class is I'_j with $I_j \subset I'_j$. Hence the chromatic number is equal to the first term.

Otherwise, in the optimal coloring c, for all $j \in \{1, 2, \ldots, k\}$, $|I_j| < \alpha \cdot n$. Thus there is a subset $S \subseteq V$ such that each color class I_j is either a subset of S or a subset of $V - S$ satisfying $(n - \alpha \cdot n)/2 \leq |S| \leq n/2$. Hence the chromatic number is equal to the second term. □

We use the lemma to obtain a recursive algorithm. To obtain the best time bound in our analysis, we set $\alpha = 0.19903$. The pseudo code of the algorithm is given below.

```
• Let $n$ be the number of vertices of the graph. Set *best* $= n$.
• Enumerate all subsets $S \subseteq V$. For each such set $S$, do:
    – If $S$ is a maximal independent set and $|S| \geq \alpha \cdot n$, then
        ∗ Recursively, compute $\chi(G[V - S])$.
        ∗ Set *best* $= \min(best, 1 + \chi(G[V - S])$.
    – If $(n - \alpha \cdot n)/2 \leq |S| \leq (n + \alpha \cdot n)$, then
        ∗ Recursively, compute $\chi(G[S])$.
        ∗ Recursively, compute $\chi(G[V - S])$.
        ∗ Set *best* $= \min(best, \chi(G[S]) + \chi(G[V - S])$.
• Return *best*.
```

From Lemma 1, it directly follows that this procedure computes $\chi(G)$. As we can enumerate all subsets of a set in polynomial space, the other steps also can be done in polynomial space, and recursion depth is bounded by a polynomial in n (or, more precisely, by $O(\log n)$), the algorithm takes polynomial space.

4 Time analysis of the algorithm

In this section, we show that the algorithm uses $O^*(5.283^n)$ time. We show this with induction. First, we look at the steps where we recursively compute $\chi(G[V - S])$ for a maximal independent set S with $|S| \geq \alpha \cdot n$. Write $T(n)$ for the maximum time used by the algorithm on a graph with n vertices. We use the following result.

Theorem 2 (Byskov [3]) *Let $0 < \beta < 1$. Let $d \in \mathbf{N}$ and $d \geq 1$. The number of maximal independent sets of size $\beta \cdot n$ in a graph with n vertices is $O^*(d^{((d+1)\cdot\beta-1)n} \cdot (d+1)^{(1-d\cdot\beta)n})$.*

Let $d = 5$. Write $A(\beta) = 5^{6\cdot\beta-1} \cdot 6^{1-5\beta} \cdot 5.283^{1-\beta}$. It follows that the total time spend on computing $\chi(G[V - S])$ for all maximal independent sets S with $|S| = \beta n \geq \alpha \cdot n$ is

$$O^*(5^{6\cdot\beta-1} \cdot 6^{1-5\beta}) \cdot T((1 - \beta) \cdot n) = O^*(A(\beta)^n)$$

Lemma 3 $5.283 > A(\alpha) = \max_{\alpha \leq \beta \leq 1} A(\beta)$.

Proof. Direct computation shows that $A(\alpha)$ is slightly less than 5.283. Furthermore the function A is monotone decreasing in the interval $[0, 1]$. $\square$

Thus, the time spend for computing the values $\chi(G[V-S])$ for all maximal independent sets S is bounded by

$$\sum_{i=\alpha\cdot n}^{n} A(i/n)^n < n \cdot A(\alpha^n) = O^*(5.283^n)$$

Now, we count the time spend on computing $\chi(G[S])$ and $\chi(G[V-S])$ for the sets S of size between $(n-\alpha\cdot n)/2$ and $n/2$. Each set R with $(n-\alpha\cdot n)/2 \leq |R| \leq (n+\alpha\cdot n)/2$ will play at most twice the role of either S or $V-S$ in one call of the procedure. Write $B(\beta) = (\beta^{-\beta}\cdot(1-\beta)^{\beta-1})\cdot 5.283^{\beta}$.

The following lemma is folklore (see e.g. [5]).

Lemma 4 *Let V be a set with n elements, and let $0 < \beta < 1$ with $\beta \cdot n \in \mathbf{N}$. There are $O^*((\beta^{-\beta}\cdot(1-\beta)^{\beta-1})^n)$ subsets of V of size $\beta\cdot n$.*

Thus, for $(n-\alpha\cdot n)/2 \leq \beta\cdot n \leq (n+\alpha\cdot n)/2$, the total time of the computation of $\chi(G[S])$ and $\chi(G[V-S])$ of sets S with $|S| = \beta\cdot n$ is bounded by

$$O^*((\beta^{-\beta}\cdot(1-\beta)^{\beta-1})^n\cdot T(\beta)) = O^*(B(\beta)^n)$$

Lemma 5 $5.283 > B((1+\alpha)/2) = \max_{(1-\alpha)/2 \leq \beta \leq (1+\alpha)/2} B(\beta)$.

Proof. Direct computation shows that $B((1+\alpha)/2)$ is slightly smaller than 5.283. Furthermore the function B is monotone increasing in the interval $[0, 0.67]$ and $(1+\alpha)/2 < 0.6$. $\square$

So, the total time bounded by the second type of recursive calls is at most

$$\sum_{i=\lfloor(1-\alpha)/2\cdot n\rfloor}^{i=\lceil(1+\alpha)/2\cdot n\rceil} O^*(B(\beta^n)) < n\cdot O^*(B((1+\alpha)/2)^n) = O^*(5.283^n)$$

As in both cases, we have the required time bound, we can conclude our main result.

Theorem 6 *There is an algorithm that computes the chromatic number of a graph in $O^*(5.283^n)$ time and polynomial memory.*

With simple addition of bookkeeping, the algorithm can also find an optimal coloring of the input graph, within same time and memory bounds.

5 Conclusions

We have provided a first exact coloring algorithm using only polynomial memory, that has a running time of $O(5.283^n)$. We think that our technique can also be applied to other NP-hard graph problems, in particular partition problems.

Unfortunately our algorithm is not practical. It is an interesting question to improve it and to obtain an $O(t^n)$ time polynomial memory coloring algorithm such that t is as close as possible to 2.4023 (respectively the base of the running time of the best known coloring algorithm).

Polynomial vs. exponential memory is an important issue in the design and analysis of exact exponential-time algorithms. In general, it would be desirable to develop techniques transforming exponential-time algorithms needing exponential space into those needing only polynomial memory while keeping the increase of the (base of) the running time small.

References

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

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

[3] J. M. Byskov. *Exact Algorithms for Graph Colouring and Exact Satisfiability.* Phd thesis, University of Aarhus, Aarhus, Denmark, August, 2005.

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

[5] U. Schöning. Algorithmics in exponential time. In *Proceedings of the 22nd International Symposium on Theoretical Aspects of Computer Science (STACS 2005)*, pages 36–43. Springer Verlag, Lecture Notes in Computer Science, vol. 3404, 2005.

[6] G. J. Woeginger. Exact algorithms for NP-hard problems: A survey. In *Combinatorial Optimization: "Eureka, you shrink"*, pages 185–207, Berlin, 2003. Springer Lecture Notes in Computer Science, vol. 2570.

[7] G. J. Woeginger. Space and time complexity of exact algorithms: some open problems (invited talk). In *Proceedings 1st International Workshop on Parameterized and Exact Computation*, pages 281–290, Berlin, 2004. Springer Lecture Notes in Computer Science, vol. 3162.