Small Maximal Independent Sets and Faster Exact Graph Coloring

David Eppstein

Dept. of Information and Computer Science, UC Irvine
Irvine, CA 92697-3425, USA
eppstein@ics.uci.edu
http://www.ics.uci.edu/~eppstein/

Abstract. We show that, for any n-vertex graph G and integer parameter k, there are at most $3^{4k-n}4^{n-3k}$ maximal independent sets $I \subset G$ with $|I| \leq k$, and that all such sets can be listed in time $\mathcal{O}(3^{4k-n}4^{n-3k})$. These bounds are tight when $n/4 \leq k \leq n/3$. As a consequence, we show how to compute the exact chromatic number of a graph in time $\mathcal{O}((4/3 + 3^{4/3}/4)^n) \approx 2.4150^n$, improving a previous $\mathcal{O}((1 + 3^{1/3})^n) \approx 2.4422^n$ algorithm of Lawler (1976).

1 Introduction

One of the earliest works in the area of worst-case analysis of NP-hard problems is a 1976 paper by Lawler [5] on graph coloring. It contains two results: an algorithm for finding a 3-coloring of a graph (if the graph is 3-chromatic) in time $\mathcal{O}(3^{n/3}) \approx 1.4422^n$, and an algorithm for finding the chromatic number of an arbitrary graph in time $\mathcal{O}((1+3^{1/3})^n) \approx 2.4422^n$. Since then, the area has grown, and there has been a sequence of papers improving Lawler's 3-coloring algorithm [1,2,4,7], with the most recent algorithm taking time $\approx 1.3289^n$. However, there has been no improvement to Lawler's chromatic number algorithm.

Lawler's algorithm follows a simple dynamic programming approach, in which we compute the chromatic number not just of G but of all its induced subgraphs. For each subgraph S, the chromatic number is found by listing all maximal independent subsets $I \subset S$, adding one to the chromatic number of $S \setminus I$, and taking the minimum of these values. The $\mathcal{O}((1+3^{1/3})^n)$ running time of this technique follows from an upper bound of $3^{n/3}$ on the number of maximal independent sets in any n-vertex graph, due to Moon and Moser [6]. This bound is tight in graphs formed by a disjoint union of triangles.

In this paper, we provide the first improvement to Lawler's algorithm, using the following ideas. First, instead of removing a maximal independent set from each induced subgraph S, and computing the chromatic number of S from that of the resulting subset, we add a maximal independent set of $G \setminus S$ and compute the chromatic number of the resulting superset from that of S. This reversal does not itself affect the running time of the dynamic programming algorithm, but it allows us to constrain the size of the maximal independent sets we consider

to at most $|S|/3$. We show that, with such a constraint, we can improve the Moon-Moser bound: for any n-vertex graph G and integer parameter k, there are at most $3^{4k-n}4^{n-3k}$ maximal independent sets $I \subset G$ with $|I| \leq k$. This bound then leads to a corresponding improvement in the running time of our chromatic number algorithm.

2 Preliminaries

We assume as given a graph G with vertex set $V(G)$ and edge set $E(G)$. We let $n = |V(G)|$ and $m = |E(G)|$. A *proper coloring* of G is an assignment of colors to vertices such that no two endpoints of any edge share the same color. We denote the chromatic number of G (the minimum number of colors in any proper coloring) by $\chi(G)$.

If $V(G) = \{v_0, v_1, \ldots v_{n-1}\}$, then we can place subsets $S \subseteq V(G)$ in one-to-one correspondence with the integers $0, 1, \ldots 2^n - 1$:

$$S \leftrightarrow \sum_{v_i \in S} 2^i.$$

Subsets of vertices also correspond to induced subgraphs of G, in which we include all edges between vertices in the subset. We make no distinction between these three equivalent views of a vertex subset, so e.g. we will write $\chi(S)$ to indicate the chromatic number of the subgraph induced by set S, and $X[S]$ to indicate a reference to an array element indexed by the number $\sum_{v_i \in S} 2^i$. We write $S < T$ to indicate the usual arithmetic comparison between two numbers, and $S \subset T$ to indicate the usual (proper) subset relation between two sets. Note that, if $S \subset T$, then also $S < T$, although the reverse implication does not hold.

A set S is a *maximal k-chromatic subset* of T if $S \subseteq T$, $\chi(S) = k$, and $\chi(S') > k$ for every $S \subset S' \subseteq T$. In particular, if $k = 1$, S is a *maximal independent subset* of T.

For any vertex $v \in V(G)$, we let $N(v)$ denote the set of neighbors of v, including v itself. If S and T are sets, $S \setminus T$ denotes the set-theoretic difference, consisting of elements of S that are not also in T. K_i denotes the complete graph on i vertices. We write $\deg(v, S)$ to denote the degree of vertex v in the subgraph induced by S.

We express our pseudocode in a syntax similar to that of C, C++, or Java. In particular this implies that array indexing is zero-based. We assume the usual RAM model of computation, in which a single cell is capable of storing an integer large enough to index the memory requirements of the program (thus, in our case, n-bit values are machine integers), and in which arithmetic and array indexing operations on these values are assumed to take constant time.

3 Small Maximal Independent Sets

Theorem 1. *Let G be an n-vertex graph, and k be a nonnegative number. Then the number of maximal independent sets $I \subset V(G)$ for which $|I| \le k$ is at most $3^{4k-n}4^{n-3k}$.*

Proof. We use induction on n; in the base case $n = 0$, there is one (empty) maximal independent set, and for any $k \ge 0$, $1 \le 3^{4k}4^{-3k} = (81/64)^k$. Otherwise, we divide into cases according to the degrees of the vertices in G, as follows:

- If G contains a vertex v of degree three or more, then each maximal independent set I either contains v (in which case $I \setminus \{v\}$ is a maximal independent set of $G \setminus N(v)$) or it does not contain v (in which case I itself is a maximal independent set of $G \setminus \{v\}$). Thus, by induction, the number of maximal independent sets of cardinality at most k is at most

$$\begin{aligned} &3^{4k-(n-1)}4^{(n-1)-3k} + 3^{4(k-1)-(n-4)}4^{(n-4)-3(k-1)} \\ &\quad = (\frac{3}{4} + \frac{1}{4})3^{4k-n}4^{n-3k} = 3^{4k-n}4^{n-3k} \end{aligned}$$

  as was to be proved.
- If G contains a degree-one vertex v, let its neighbor be u. Then each maximal independent set contains exactly one of u or v, and removing this vertex from the set produces a maximal independent set of either $G \setminus N(u)$ or $G \setminus N(v)$. If the degree of u is d, this gives us by induction a bound of

$$\begin{aligned} &3^{4(k-1)-(n-2)}4^{(n-2)-3(k-1)} + 3^{4(k-1)-(n-d-1)}4^{(n-d-1)-3(k-1)} \\ &\quad \le \frac{8}{9}\,3^{4k-n}4^{n-3k} \end{aligned}$$

  on the number of maximal independent sets of cardinality at most k.
- If G contains an isolated vertex v, then each maximal independent set contains v, and the number of maximal independent sets of cardinality at most k is at most

$$3^{4(k-1)-(n-1)}4^{(n-1)-3(k-1)} = \frac{16}{27}\,3^{4k-n}4^{n-3k}.$$

- If G contains a chain u-v-w-x of degree two vertices, then each maximal independent set contains u, contains v, or does not contain u and contains w. Thus in this case the number of maximal independent sets of cardinality at most k is at most

$$2{\cdot}3^{4(k-1)-(n-3)}4^{(n-3)-3(k-1)} + 3^{4(k-1)-(n-4)}4^{(n-4)-3(k-1)} = \frac{11}{12}\,3^{4k-n}4^{n-3k}.$$

- In the remaining case, G consists of a disjoint union of triangles, all maximal independent sets have exactly $n/3$ vertices, and there are exactly $3^{n/3}$ maximal independent sets. If $k \ge n/3$, then $3^{n/3} \le 3^{4k-n}4^{n-3k}$. If $k < n/3$, there are no maximal independent sets of cardinality at most k.

Thus in all cases the number of maximal independent sets is within the claimed bound. □

Croitoru [3] proved a similar bound with the stronger assumption that all maximal independent sets have $|I| \leq k$. When $n/4 \leq k \leq n/3$, our result is tight, as can be seen for a graph formed by the disjoint union of $4k-n$ triangles and $n-3k$ K_4's.

Theorem 2. *There is an algorithm for listing all maximal independent sets smaller than k in an n-vertex graph G, in time $\mathcal{O}(3^{4k-n}4^{n-3k})$.*

Proof. We use a recursive backtracking search, following the case analysis of Theorem 1: if there is a high-degree vertex, we try including it or not including it; if there is a degree-one vertex, we try including it or its neighbor; if there is a degree-zero vertex, we include it; and if all vertices form chains of degree-two vertices, we test whether the parameter k allows any small maximal independent sets, and if so we try including each of a chain of three adjacent vertices. The same case analysis shows that this algorithm performs $\mathcal{O}(3^{4k-n}4^{n-3k})$ recursive calls.

Each recursive call can easily be implemented in time polynomial in the size of the graph passed to the recursive call. Since our $3^{4k-n}4^{n-3k}$ bound is exponential in n, even when $k=0$, this polynomial overhead at the higher levels of the recursion is swamped by the time spent at lower levels of the recursion, and does not appear in our overall time bound. □

A more detailed pseudocode description of the algorithm is shown in Figure 1. The given pseudocode may generate non-maximal as well as maximal independent sets, because (when we try not including a high degree vertex) we do not make sure that a neighbor is later included. This will not cause problems for our chromatic number algorithm, but if only maximal independent sets are desired one can easily test the generated sets and eliminate the non-maximal ones. The pseudocode also omits the data structures necessary to implement each recursive call in time polynomial in $|S|$ instead of polynomial in the number of vertices of the original graph.

4 Chromatic Number

We are now ready to describe our algorithm for computing the chromatic number of graph G. We use an array X, indexed by the 2^n subsets of G, which will (eventually) hold the chromatic numbers of certain of the subsets including $V(G)$ itself. We initialize this array by testing, for each subset S, whether $\chi(S) \leq 3$; if so, we set $X[S]$ to $\chi(S)$, but otherwise we set $X[S]$ to ∞.

Next, we loop through the subsets S of $V(G)$, in numerical order (or any other order such that all proper subsets of each set S are visited before we visit S itself). When we visit S, we first test whether $X[S] \geq 3$. If not, we skip over S without doing anything. But if $X[S] \geq 3$, we loop through the small independent

```
// List maximal independent subsets of S smaller than a given parameter.
//      S is a set of vertices forming an induced subgraph in G,
//      I is a set of vertices to be included in the MIS (initially zero), and
//      k bounds the number of vertices of S to add to I.
// We call processMIS(I) on each generated set. Some non-maximal sets may be
// generated along with the maximal ones, but all generated sets are independent.

void smallMIS (set S, set I, int k)
{
    if (S = 0 or k = 0) processMIS(I);
    else if (there exists v ∈ S with deg(v, S) ≥ 3)
    {
        smallMIS (S \ {v}, I, k);
        smallMIS (S \ N(v), I ∪ {v}, k − 1);
    }
    else if (there exists v ∈ S with deg(v, S) = 1)
    {
        let u be the neighbor of v;
        smallMIS (S \ N(u), I ∪ {u}, k − 1);
        smallMIS (S \ N(v), I ∪ {v}, k − 1);
    }
    else if (there exists v ∈ S with deg(v, S) = 0)
        smallMIS (S \ {v}, I ∪ {v}, k − 1);
    else if (some cycle in S is not a triangle or k ≥ |S|/3)
    {
        let u, v, and w be adjacent degree-two vertices, such that (if possible) u and w are nonadjacent;
        smallMIS (S \ N(u), I ∪ {u}, k − 1);
        smallMIS (S \ N(v), I ∪ {v}, k − 1);
        smallMIS (S \ ({u} ∪ N(w)), I ∪ {w}, k − 1);
    }
}
```

Fig. 1. Algorithm for listing all small maximal independent sets.

```
int chromaticNumber (graph G)
{
    int X[2^n];
    for (S = 0; S ≤ 2^n; S++)
    {
        if (χ(S) ≤ 3) X[S] = χ[S];
        else X[S] = ∞;
    }
    for (S = 0; S ≤ 2^n; S++)
    {
        if (3 ≤ X[S] < ∞)
        {
            for (each maximal independent set I of G \ S with |I| ≤ |S|/X[S])
                X[S ∪ I] = min(X[S ∪ I], X[S] + 1);
        }
    }
    return X[V(G)];
}
```

Fig. 2. Algorithm for computing the chromatic number of a graph.

sets of $G \setminus S$, limiting the size of each such set to $|S|/X[S]$, using the algorithm of the previous section. For each independent set I, we set $X[S \cup I]$ to the minimum of its previous value and $X[S] + 1$.

Finally, after looping through all subsets, we return the value in $X[V(G)]$ as the chromatic number of G. Pseudocode for this algorithm is shown in Figure 2.

Lemma 1. *Throughout the course of the algorithm, for any set S, $X[S] \geq \chi(S)$.*

Proof. Clearly this is true of the initial values of X. Then for any S and any independent set I, we can color $S \cup I$ by using a coloring of S and another color for each vertex in I, so $\chi(S \cup I) \leq \chi(S) + 1 \leq X[S] + 1$, and each step of our algorithm preserves the invariant. □

Lemma 2. *Let M be a maximal $k+1$-chromatic subset of G, and let (S, I) be a partition of M into a k-chromatic subset S and an independent subset I, maximizing the cardinality of S among all such partitions. Then I is a maximal independent subset of $G \setminus S$ with $|I| \leq |S|/k$, and S is a maximal k-chromatic subset of G.*

Proof. If we have any $(k+1)$-coloring of G, then the partition formed by separating the largest k color classes from the smallest color class satisfies the inequality $|I| \leq |S|/k$, so clearly this also is true when (S, I) is the partition maximizing $|S|$. If I were not maximal, due to the existence of another independent set $I \subset I' \subset G \setminus S$, then $S \cup I'$ would be a larger $(k+1)$-chromatic graph, violating the assumption of maximality of M.

Similarly, suppose there were another k-chromatic set $S \subset S' \subset G$. Then if $S' \cap I$ were empty, $S' \cup I$ would be a $(k+1)$-chromatic superset of M, violating the assumption of M's maximality. But if $S' \cap I$ were nonempty, $(S', I \setminus S')$ would be a better partition than (S, I), so in either case we get a contradiction. □

Lemma 3. *Let M be a maximal $k+1$-chromatic subset of G. Then, when the outer loop of our algorithm reaches M, it will be the case that $X[M] = \chi(M)$.*

Proof. Clearly, the initialization phase of the algorithm causes this to be true when $\chi(M) \leq 3$. Otherwise, let (S, I) be as in Lemma 2. By induction on $|M|$, $X[S] = \chi(S)$ at the time we visit S. Then $X[S] \geq 3$, and $|I| \leq |S|/X[S]$, so the inner loop for S will visit I and set $X[M]$ to $X[S] + 1 = \chi(M)$. □

Theorem 3. *We can compute the chromatic number of a graph G in time $\mathcal{O}((4/3 + 3^{4/3}/4)^n)$ and space $\mathcal{O}(2^n)$.*

Proof. $V(G)$ is itself a maximal $\chi(G)$-chromatic subset of G, so Lemma 3 shows that the algorithm correctly computes $\chi(G) = X[V(G)]$. Clearly, the space is bounded by $\mathcal{O}(2^n)$. It remains to analyze the algorithm's time complexity.

First, we consider the time spent initializing X. Since we perform a 3-coloring algorithm on each subset of G, this time is

$$\sum_{S \subset V(G)} \mathcal{O}(1.3289^{|S|}) = \mathcal{O}\Big(\sum_{i=0}^{n} \binom{n}{i} 1.3289^i\Big) = \mathcal{O}(2.3289^i).$$

Finally, we bound the time in the main loop of the algorithm. We may possibly apply the algorithm of Theorem 2 to generate small independent subsets of each set $G \setminus S$. In the worst case, $X[S] = 3$ and we can only limit the size of the generated independent sets to $|S|/3$. We spend constant time adjusting the value of $X[S \cup I]$ for each generated set. Thus, the total time can be bounded as

$$\sum_{S \subset V(G)} \mathcal{O}(3^{4\frac{|S|}{3} - |G \setminus S|} 4^{|G \setminus S| - 3\frac{|S|}{3}}) = \mathcal{O}\Big(\sum_{i=0}^{n} \binom{n}{i} 3^{\frac{7i}{3} - n} 4^{n-2i}\Big) = \mathcal{O}\Big((\frac{4}{3} + \frac{3^{4/3}}{4})^n\Big).$$

This final term dominates the overall time bound. □

5 Finding a Coloring

Although the algorithm of the previous section finds the chromatic number of G, it is likely that an explicit coloring is desired, rather than just this number. The usual method of performing this sort of construction task in a dynamic programming algorithm is to augment the dynamic programming array with back pointers indicating the origin of each value computed in the array, but since storing 2^n chromatic numbers is likely to be the limiting factor in determining how large a graph this algorithm can be applied to, it is likely that also storing 2^n set indices will severely reduce its applicability.

```
void color (graph G)
{
    compute array X as in Figure 2;
    S = V(G);
    for (T = 2^n − 1; T ≥ 0; T−−)
    {
        if (T ⊂ S and X[S \ T] = 1 and X[T] = X[S] − 1)
        {
            color all vertices in S \ T with the same new color;
            S = T;
        }
    }
}
```

Fig. 3. Algorithm for optimally coloring a graph.

Instead, we can simply search backwards from $V(G)$ until we find a subset S that can be augmented by an independent set to form $V(G)$, and that has chromatic number $\chi(S) = \chi(G) - 1$ as indicated by the value of $X[S]$. We assign the first color to $G \setminus S$. Then, we continue searching for a similar subset $T \subset S$, etc., until we reach the empty set. Although not every set S may necessarily have $X[S] = \chi(S)$, it is guaranteed that for any S we can find $T \subset S$ with $S \setminus T$ independent and $X[T] = X[S] - 1$, so this search procedure always finds a correct coloring.

Theorem 4. *After computing the array X as in Theorem 3, we can compute an optimal coloring of G in additional time $\mathcal{O}(2^n)$ and no additional space.*

We omit the details of the correctness proof and analysis. A pseudocode description of the coloring algorithm is shown in Figure 3.

6 Conclusions

We have shown a bound on the number of small independent sets in a graph, shown how to list all small independent sets in time proportional to our bound, and used this algorithm in a new dynamic programming algorithm for computing the chromatic number of a graph as well as an optimal coloring of the graph.

Our bound on the number of small independent sets is tight for $n/4 \leq k \leq n/3$, and an examination of the analysis of Theorem 3 shows that independent set sizes in this range are also the ones leading to the worst case time bound for our coloring algorithm. Nevertheless, it would be of interest to find tight bounds on the number of small independent sets for all ranges of k. It would also be of interest to find an algorithm for listing all small maximal independent sets in time proportional to the number of generated sets rather than simply proportional to the worst case bound on this number.

Our worst case analysis of the chromatic number algorithm assumes that, every time we call the procedure for listing small maximal independent sets, this procedure achieves its worst case time bound. But is it really possible for all sets $G \setminus S$ to be worst case instances for this procedure? If not, perhaps the analysis of our coloring algorithm can be improved.

Can we prove a bound smaller than $\binom{n}{i}$ on the number of i-vertex maximal k-chromatic induced subgraphs of a graph G? If such a bound could be proven, even for $k = 3$, we could likely improve the algorithm presented here by only looping through the independent subgraphs of $G \setminus S$ when S is maximal.

An alternative possibility for improving the present algorithm would be to find an algorithm for testing whether $\chi(G) \leq 4$ in time $o(1.415^n)$. Then we could test the four-colorability of all subsets of G before applying the rest of our algorithm, and avoid looping over maximal independent subsets of $G \setminus S$ unless $X[S] \geq 4$. This would produce tighter limits on the independent set sizes and therefore reduce the number of independent sets examined. However such a time bound would be significantly better than what is currently known for four-coloring algorithms.

References

1. R. Beigel and D. Eppstein. 3-coloring in time $\mathcal{O}(1.3446^n)$: a no-MIS algorithm. *Proc. 36th Symp. Foundations of Computer Science*, pp. 444–453. IEEE, October 1995, ftp://ftp.eccc.uni-trier.de/pub/eccc/reports/1995/TR95-033/index.html.
2. R. Beigel and D. Eppstein. 3-coloring in time $\mathcal{O}(1.3289^n)$. ACM Computing Research Repository, June 2000, cs.DS/0006046.
3. C. Croitoru. On stables in graphs. *Proc. 3rd Coll. Operations Research*, pp. 55–60. Babes-Bolyai Univ., Cluj-Napoca, Romania, 1979.
4. D. Eppstein. Improved algorithms for 3-coloring, 3-edge-coloring, and constraint satisfaction. *Proc. 12th Symp. Discrete Algorithms*, pp. 329–337. ACM and SIAM, January 2001, cs.DS/0009006.
5. E. L. Lawler. A note on the complexity of the chromatic number problem. *Inf. Proc. Lett.* 5(3):66–67, August 1976.
6. J. W. Moon and L. Moser. On cliques in graphs. *Israel J. Math.* 3:23–28, 1965.
7. I. Schiermeyer. Deciding 3-colourability in less than $\mathcal{O}(1.415^n)$ steps. *Proc. 19th Int. Worksh. Graph-Theoretic Concepts in Computer Science*, pp. 177–182. Springer-Verlag, Lecture Notes in Comp. Sci. 790, 1994.