Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Discrete Algorithms 4 (2006) 209–214
JOURNAL OF DISCRETE ALGORITHMS
www.elsevier.com/locate/jda

A note on the complexity of minimum dominating set

Fabrizio Grandoni

Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany

Available online 13 April 2005

Abstract

The currently (asymptotically) fastest algorithm for minimum dominating set on graphs of n nodes is the trivial $\Omega(2^n)$ algorithm which enumerates and checks all the subsets of nodes. In this paper we present a simple algorithm which solves this problem in $O(1.81^n)$ time.

Keywords: Dominating set; Set cover; Exact algorithms

1. Introduction

Since the seminal work of Tarjan and Trojanowski [16], a lot of effort has been devoted to develop faster and faster (exponential-time) exact algorithms for NP-hard and NP-complete problems, such as *maximum independent set* [3,10,14,15], *vertex cover* [1,5,12], (*maximum*) *satisfiability* [2,7,11,13], 3-*coloring* [4,8] and many others.

Minimum dominating set is one of the most basic NP-hard problems [9]. The currently (asymptotically) fastest algorithm to solve this problem on graphs of n nodes is the trivial $\Omega(2^n)$ algorithm which enumerates and checks all the subsets of nodes.

In this paper we present a $O(1.3424^k)$ algorithm for *minimum set cover*, where k, the *dimension* of the problem, is the sum of the number of sets available and of the number of elements which need to be covered. Minimum dominating set can be formulated as a

E-mail address: grandoni@disp.uniroma2.it (F. Grandoni).

1570-8667/$ – see front matter
doi:10.1016/j.jda.2005.03.002

minimum set cover problem of dimension $k = 2n$. It follows that minimum dominating set can be solved in $O(1.3424^{2n}) = O(1.8021^n)$ time.

1.1. Preliminaries

We use standard set notation. Let $\mathcal{S}$ be a collection of subsets of a given *universe* $\mathcal{U}$. For the sake of simplicity, we assume that $\mathcal{S}$ *covers* $\mathcal{U}$:

$$\mathcal{U} = \mathcal{U}_{\mathcal{S}} = \bigcup_{S \in \mathcal{S}} S.$$

A *set cover* of $\mathcal{S}$ is a subset $\mathcal{S}'$ of $\mathcal{S}$ which *covers* $\mathcal{U}$. The *minimum set cover problem* consists in determining the minimum cardinality $msc(\mathcal{S})$ of a set cover of $\mathcal{S}$. Without loss of generality, we can assume that $\mathcal{S}$ does not contain the empty set (since it does not belong to any minimum set cover). We call *dimension* k of $\mathcal{S}$ the sum of the cardinalities of $\mathcal{S}$ and $\mathcal{U}$:

$$k = |\mathcal{S}| + |\mathcal{U}|.$$

Let R be a subset of $\mathcal{U}$. By $del(\mathcal{S}, R)$ we denote the collection which is obtained from $\mathcal{S}$ by removing the elements of R from each S in $\mathcal{S}$, and by eventually removing the empty sets obtained:

$$del(\mathcal{S}, R) = \{S' \neq \emptyset\colon\ S' = S \backslash R,\ \ S \in \mathcal{S}\}.$$

We use standard graph notation as for example in [6]. In particular, by $G = (V, E)$ we denote a (undirected) graph, where V is the set of *nodes* and E is the set of *edges* (pairs of distinct nodes). Two nodes u and v are *adjacent* if there is an edge $\{u, v\} \in E$. The *neighborhood* $N(v)$ of node v is the set of nodes adjacent to v. A *dominating set* of G is a subset V' of V such that every node $u \in V \backslash V'$ is adjacent to at least one node $v \in V'$, i.e.:

$$V \backslash V' \subseteq \bigcup_{v \in V'} N(v).$$

The *minimum dominating set problem* consists in determining the minimum cardinality of a dominating set of G. Minimum dominating set can be naturally formulated as a minimum set cover problem, in which there is a set $N(v) \cup \{v\}$ for each node $v \in V$ (the set of nodes *dominated* by v). Note that the dimension of the minimum set cover formulation of minimum dominating set is $k = 2n$, where n is the number of nodes of G (there is one set for each node and the set of elements which need to be covered is the set of nodes).

2. A polynomial-space algorithm

In this section we describe a $O(1.3803^k)$ polynomial-space recursive algorithm `MSC` for minimum set cover, where k is the *dimension* of the problem. It immediately follows that minimum dominating set can be solved in $O(1.3803^{2n}) = O(1.9053^n)$ time.

The algorithm is based on the following simple properties of set covers.

```
1   int MSC(S) {
2       if(|S| = 0) return 0;
3       if(∃S, R ∈ S: S ⊂ R) return MSC(S\{S});
4       if(∃s ∈ U_S ∃ a unique S ∈ S: s ∈ S) return 1 + MSC(del(S, S));
5       take S ∈ S of maximum cardinality;
6       return min{MSC(S\{S}), 1 + MSC(del(S, S))};
7   }
```

Fig. 1. Recursive algorithm for minimum set cover.

Lemma 1. *Let $\mathcal{S}$ be an instance of minimum set cover. The following properties hold*:

(1) *If there is a set S in $\mathcal{S}$ which is (properly) included in another set R in $\mathcal{S}$ ($S \subset R$), then there is a minimum set cover which does not contain S. In particular*:

$$msc(\mathcal{S}) = msc\big(\mathcal{S}\backslash\{S\}\big).$$

(2) *If there is an element s of $\mathcal{U}$ which belongs to a unique $S \in \mathcal{S}$, then S belongs to every set cover. In particular*:

$$msc(\mathcal{S}) = 1 + msc\big(del(\mathcal{S}, S)\big).$$

(3) *For all the remaining $S \in \mathcal{S}$, the following holds*:

$$msc(\mathcal{S}) = \min\Big\{msc\big(\mathcal{S}\backslash\{S\}\big), 1 + msc\big(del(\mathcal{S}, S)\big)\Big\}.$$

Note that the sets of cardinality one satisfy exactly one of the properties (1) and (2) of Lemma 1.

A basic version of the algorithm is described in Fig. 1. The base case (line 2) is when $|\mathcal{S}| = 0$. In that case, $msc(\mathcal{S}) = 0$. Otherwise (lines 3 and 4), the algorithm tries to reduce the dimension of the problem without branching, by applying one of the properties (1) and (2) of Lemma 1. If none of the two properties above applies, the algorithm simply takes (line 5) a set $S \in \mathcal{S}$ of maximum cardinality and branches (line 6) according to property (3) of Lemma 1.

Theorem 1. *Algorithm* `MSC` *solves minimum set cover in time* $O(1.3803^k)$*, where k is the dimension of the problem.*

Proof. The correctness of the algorithm is a straightforward consequence of Lemma 1.

Let $N_h(k)$ denote the number of subproblems of dimension h solved by the algorithm to solve a problem of dimension k. Clearly, $N_h(k) = 0$ for $h > k$ (the subproblems are of dimension lower than the dimension of the original problem). Moreover, $N_k(k) = 1$ (considering the original problem as one of the subproblems). Consider the case $h < k$ (which implies $|\mathcal{S}| \neq 0$). If one of the conditions of lines 3 and 4 is satisfied, the algorithm generates a unique subproblem of dimension at most $k - 1$. Thus:

$$N_h(k) \leqslant N_h(k-1).$$

Otherwise, the algorithm takes a set S of maximum cardinality ($|S| \geqslant 2$), and it branches on the two subproblems $\mathcal{S}_1 = \mathcal{S} \backslash \{S\}$ and $\mathcal{S}_2 = del(\mathcal{S}, S)$. The dimension of $\mathcal{S}_1$ is $k-1$ (one set removed from $\mathcal{S}$). If $|S| \geqslant 3$, the dimension of $\mathcal{S}_2$ is at most $k-4$ (one set removed from $\mathcal{S}$ and at least three elements removed from $\mathcal{U}$). Thus:

$$N_h(k) \leqslant N_h(k-1) + N_h(k-4).$$

Otherwise (all the sets in $\mathcal{S}$ have cardinality two), the dimension of $\mathcal{S}_2$ is $k-3$ (one set removed from $\mathcal{S}$ and two elements removed from $\mathcal{U}$). Moreover, $\mathcal{S}_2$ must contain a set S' of cardinality one. Then the algorithm has to execute one of the lines 3 and 4 to solve the subproblem $\mathcal{S}_2$. Thus:

$$N_h(k) \leqslant N_h(k-1) + N_h(k-3-1) = N_h(k-1) + N_h(k-4).$$

A valid upper bound on $N_h(k)$ is $N_h(k) \leqslant c^{k-h}$, where $c = 1.3802\ldots < 1.3803$ is the (unique) positive root of the polynomial $(x^4 - x^3 - 1)$. This implies that the total number $N(k)$ of subproblems solved is:

$$N(k) = \sum_{h=0}^{k} N_h(k) \leqslant \sum_{h=0}^{k} c^{k-h} = \mathrm{O}(c^k).$$

The cost of solving a problem of dimension $h \leqslant k$, excluding the cost of solving the corresponding subproblems (if any), is upper bounded by a polynomial $p(k)$ of k. It follows that the time complexity of the algorithm is $\mathrm{O}(c^k p(k)) = \mathrm{O}(1.3803^k)$. □

Corollary 1. *There is an algorithm which solves minimum dominating set in* $\mathrm{O}(1.9053^n)$ *time, where n is the number of nodes in the graph.*

In line 5 of `MSC` a set of maximum cardinality is taken. More sophisticated selection criteria may lead to improved time bounds.

3. An exponential-space algorithm

In this section we show how to reduce the time complexity of `MSC` via *dynamic programming*, at the cost of an exponential space complexity.

The technique is similar to the one used by Robson in the context of maximum independent set [14]. While solving a problem $\mathcal{S}$, the same subproblem $\mathcal{S}'$ can appear many times. The idea is then to store the solutions of all the subproblems solved in a database. Whenever a new subproblem is generated, this database is checked to see whether the solution of that subproblem is already available. This way, one ensures that a given subproblem is solved at most once. The database can be implemented in such a way that the *query time* is logarithmic in the number of solutions stored.

Theorem 2. *Algorithm* `MSC`*, modified as above, solves minimum set cover in time* $\mathrm{O}(1.3424^k)$*, where k is the dimension of the problem.*

Proof. The correctness of the algorithm follows from Lemma 1.

Let $N_h(k)$ denote the number of subproblems of dimension $h \in \{0, 1, \ldots, k\}$ solved by the algorithm to solve a problem of dimension k. From the proof of Theorem 1, $N_h(k) \leqslant c^{k-h}$, where $c = 1.3802\ldots < 1.3803$ is the positive root of the polynomial $(x^4 - x^3 - 1)$. Each subproblem $\mathcal{S}'$ is obtained by removing some sets from $\mathcal{S}$ and some elements from $\mathcal{U}$. In other words:

$$\mathcal{S}' = del(\mathcal{S}\backslash\mathcal{S}^*, \mathcal{U}^*),$$

for some $\mathcal{S}^* \subseteq \mathcal{S}$ and $\mathcal{U}^* \subseteq \mathcal{U}$. This implies that $N_h(k) \leqslant \binom{k}{h}$. Let h' be the largest integer in $\{0, 1, \ldots, \lfloor k/2 \rfloor\}$ such that $\binom{k}{h'} < c^{k-h'}$. The total number $N(k)$ of subproblems solved is:

$$N(k) = \sum_{h=0}^{k} N_h(k) \leqslant \sum_{h=0}^{h'} \binom{k}{h} + \sum_{h=h'+1}^{k} c^{k-h} = \mathrm{O}(c^{k-h'}).$$

It follows from Stirling's approximation that $N(k)$ is $\mathrm{O}(c^{(1-\alpha)k})$, where α satisfies:

$$c^{1-\alpha} = \frac{1}{\alpha^{\alpha}(1-\alpha)^{1-\alpha}}.$$

The cost of each query to the database is polynomial in k. Thus the cost of solving a problem of dimension $h \leqslant k$, excluding the cost of solving the corresponding subproblems (if any), is upper bounded by a polynomial $p(k)$ of k. It follows that the time complexity of the algorithm is $\mathrm{O}(c^{(1-\alpha)k} p(k)) = \mathrm{O}(1.3424^k)$. □

Corollary 2. *There is an algorithm which solves minimum dominating set in* $\mathrm{O}(1.8021^n)$ *time, where n is the number of nodes in the graph.*

References

[1] R. Balasubramanian, M. Fellows, V. Raman, An improved fixed-parameter algorithm for vertex cover, Inform. Process. Lett. 65 (1998) 163–168.

[2] N. Bansal, V. Raman, Upper bounds for MaxSat: further improved, in: International Symposium on Algorithms and Computation (ISAAC), 1999.

[3] R. Beigel, Finding maximum independent sets in sparse and general graphs, in: ACM-SIAM Symposium on Discrete Algorithms (SODA), 1999, pp. 856–857.

[4] R. Beigel, D. Eppstein, 3-coloring in time $\mathrm{O}(1.3446^n)$: a no-MIS algorithm, in: IEEE Symposium on Foundations of Computer Science (FOCS), 1995, pp. 444–452.

[5] J. Chen, I. Kanj, W. Jia, Vertex cover: further observations and further improvements, J. Algorithms 41 (2001) 280–301.

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press/McGraw-Hill, Cambridge, MA/New York, 1990.

[7] E. Dantsin, A. Goerdt, E.A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, U. Schöning, A deterministic $(2 - 2/(k+1))^n$ algorithm for k-SAT based on local search, Theoret. Comput. Sci. 289 (1) (2002) 69–83.

[8] D. Eppstein, Improved algorithms for 3-coloring, 3-edge-coloring, and constraint satisfaction, in: ACM-SIAM Symposium on Discrete Algorithms (SODA), 2001, pp. 329–337.

[9] M.R. Garey, D.S. Johnson, Computers and Intractability. A Guide to the Theory of NP-Completeness, Freemann, New York, 1979.
[10] T. Jian, An $O(2^{0.304n})$ algorithm for solving maximum independent set problem, IEEE Trans. Comput. 35 (9) (1986) 847–851.
[11] R. Niedermeier, P. Rossmanith, New upper bounds for maximum satisfiability, J. Algorithms 36 (1) (2000) 63–88.
[12] R. Niedermeier, P. Rossmanith, On efficient fixed-parameter algorithms for weighted vertex cover, J. Algorithms 47 (2) (2003) 63–77.
[13] R. Paturi, P. Pudlak, M.E. Saks, F. Zane, An improved exponential-time algorithm for k-SAT, in: IEEE Symposium on Foundations of Computer Science (FOCS), 1998, pp. 628–637.
[14] J.M. Robson, Algorithms for maximum independent sets, J. Algorithms 7 (3) (1986) 425–440.
[15] J.M. Robson, Finding a maximum independent set in time $O(2^{n/4})$, Technical Report 1251-01, LaBRI, Université Bordeaux I, 2001.
[16] R. Tarjan, A. Trojanowski, Finding a maximum independent set, SIAM J. Comput. 6 (3) (1977) 537–546.