REPORTS
IN
INFORMATICS

ISSN 0333-3590

Measure and Conquer: A Simple $O(2^{0.288n})$ Independent Set Algorithm

Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch

REPORT NO 313 **November 2005**

Department of Informatics

UNIVERSITY OF BERGEN

Bergen, Norway

This report has URL
http://www.ii.uib.no/publikasjoner/texrap/ps/2005-313.ps

Reports in Informatics from Department of Informatics, University of Bergen, Norway, is available at http://www.ii.uib.no/publikasjoner/texrap/.

Requests for paper copies of this report can be sent to:

Department of Informatics, University of Bergen, Høyteknologisenteret,
P.O. Box 7800, N-5020 Bergen, Norway

Measure and Conquer:
A Simple $O(2^{0.288n})$ Independent Set Algorithm

Fedor V. Fomin* Fabrizio Grandoni† Dieter Kratsch‡

9th November 2005

Abstract

For more than 30 years Davis-Putnam-style exponential-time backtracking algorithms have been the most common tools used for finding exact solutions of NP-hard problems. Despite of that, the way to analyze such recursive algorithms is still far from producing tight worst case running time bounds.

The "Measure and Conquer" approach is one of the recent attempts to step beyond such limitations. The approach is based on the choice of the measure of the subproblems recursively generated by the algorithm considered; this measure is used to lower bound the progress made by the algorithm at each branching step. A good choice of the measure can lead to a significantly better worst case time analysis.

In this paper we apply "Measure and Conquer" to the analysis of a very simple backtracking algorithm solving the well-studied maximum independent set problem. The result of the analysis is striking: the running time of the algorithm is $O(2^{0.288n})$, which is competitive with the current best time bounds obtained with far more complicated algorithms (and naive analysis).

Our example shows that a good choice of the measure, made in the very first stages of exact algorithms design, can have a tremendous impact on the running time bounds achievable.

Keywords: Algorithms and data structures, exponential-time exact algorithms, NP-hard problems, independent set problem.

1 Introduction

Recently, there has been a growing interest in the design and analysis of exact exponential time algorithms for NP-hard problems such as satisfiability [6, 12, 16], coloring [2, 3], maximum independent set [1, 5], max-cut [19], minimum dominating set [8], treewidth [9] and many others. (See the survey [20].) There are several explanations to that:

- There are certain applications that require exact solutions of NP-hard problems, although this might only be possible for moderate input sizes.

- Approximation algorithms are not always satisfactory. For example, maximum independent set is hard to approximate within $n^{1-\varepsilon}$ [11].

*Department of Informatics, University of Bergen, N-5020 Bergen, Norway, fomin@ii.uib.no. Supported by Norges forskningsråd project 160778/V30.

†Dipartimento di Informatica, Università di Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italy, grandoni@di.uniroma1.it. Supported by EC Project DELIS, and project WebMinds of the Italian Ministry of University and Research (MIUR).

‡LITA, Université de Metz, 57045 Metz Cedex 01, France, kratsch@univ-metz.fr

- A reduction of the base of the exponential running time, say from $O(2^n)$ to $O(2^{0.9n})$, increases the size of the instances solvable within a given amount of time by a constant *multiplicative* factor; running a given exponential algorithm on a faster computer can enlarge the mentioned size only by a constant *additive* factor.

- The design and analysis of exact algorithms leads to a better understanding of NP-hard problems and initiates interesting new combinatorial and algorithmic challenges.

Related Work. The design of exponential algorithms for the maximum independent set problem has a long history starting from the first algorithm due to Tarjan [17] breaking the trivial $O(2^n)$ bound. The first published algorithm is due to Tarjan and Trojanowski (1977): it has running time $O(2^{n/3})$ and polynomial space complexity [18]. In 1986 Jian published a polynomial-space algorithm of running time $O(2^{0.304n})$ [13] and Robson came out with a polynomial space algorithm of running time $O(2^{0.296n})$ and with an exponential space algorithm using time $O(2^{0.276n})$ [14]. In a recent technical report [15] Robson claims better running times, both in polynomial and in exponential space. The description of his new algorithm, which is partially computer generated, takes almost 18 pages. A significant amount of research was also devoted to solve the maximum independent set problem on sparse graphs [1, 4, 5].

All mentioned exact algorithms for the maximum independent set problem are search-tree based. They use a branch and reduce paradigm with a (long) list of reduction and branching rules. Often, this list is derived from a somewhat tedious and complicated case analysis. The running time analysis leads to a linear recurrence for each branching rule that can be solved using standard techniques. The classical worst case running time analysis of such algorithms is based on the assumption that a "natural" size of a problem (instance) is associated to each node of the search tree. For graph algorithms the natural measure is the number of vertices or edges in the corresponding (remaining) graph and it is not surprising that usually this measure is used for maximum independent set.

The interest on exact algorithms for the independent set problem seems to have vanished in the nineties. This is partially due to the fact that the current best algorithms for the problem are already very complicated: adding a few more branching rules in order to slightly reduce their running time might seem a waste of time.

Fortunately the growing interest on exact algorithms for NP-hard problems has led to important new insights. One of those is the idea to use non-standard measures for the size of a problem (instance). Using this approach, that they called "Measure and Conquer", Fomin et al. obtained the currently fastest $O(2^{0.598n})$ algorithm for the minimum dominating set problem [8].

New Results. In this paper we continue our previous work on the "Measure and Conquer" method [8]. To study the power of the method it is natural to test it at a problem into which a lot of effort was put with the classical analysis — the maximum independent set problem. To stress the power of the method, we developed a very simple algorithm: it can be described in a few lines and analyzed in a few pages (though we needed a computer to find a good measure for it). Our polynomial-space algorithm has running time $O(2^{0.288n})$. Currently there is no better published polynomial-space algorithm.

The algorithm uses reduction rules based on *domination* and *folding* which are known in the literature. We also introduce the useful notion of *mirroring*, which allows to greatly simplify the description of the algorithm. Intuitively, when we branch by discarding a node, we can discard its *mirrors* as well. This handy tool might be of independent interest.

The current methods of worst case running time analysis for search-tree algorithms do not seem to provide tight upper bounds. This motivates the study of lower bounds for such algorithms. We prove a lower bound of $\Omega(2^{0.166n})$ for the running time of our algorithm.

2 Preliminaries

Let $G = (V, E)$ be an n-nodes undirected graph. By $N^d(v)$ we denote the set of nodes at distance (number of hops) d from v. In particular, $N^1(v) = N(v)$ is the *neighborhood* of v. We let $d(v) = |N(v)|$ be the degree of v, and $N[v] = N(v) \cup \{v\}$. Given a subset V' of nodes, $G[V']$ is the graph induced by V', and $G - V' = G[V \setminus V']$. A *maximum independent set* of G is a maximum cardinality subset of pair-wise non-adjacent nodes. Our maximum independent set algorithm branches by imposing that some nodes belong or do not belong to the maximum independent set computed: we call the nodes of the first kind *taken*, and the other nodes *discarded*. Throughout this paper we use a modified big-Oh notation that suppresses all polynomially bounded factors. For functions f and g we write $f(n) = O^*(g(n))$ if $f(n) = O(g(n)poly(n))$, where $poly(n)$ is a polynomial.

A node v is *foldable* if $N(v) = \{u_1, u_2 \dots u_{d(v)}\}$ contains no anti-triangles. *Folding* of a given node v of G is the process of transforming G into a new graph $\widetilde{G} = \widetilde{G}(v)$ by:

(1) adding a new node u_{ij} for each anti-edge $u_i u_j$ in $N(v)$;

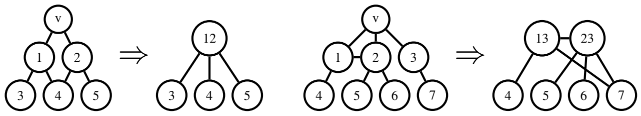
(2) adding edges between each u_{ij} and the nodes in $N(u_i) \cup N(u_j)$;

(3) adding one edge between each pair of new nodes;

(4) removing $N[v]$.

Note that when we fold a node v of degree either zero or one, we simply remove $N[v]$ from the graph. Examples of folding are given in Figure 1. Let $\alpha(G)$ denote the size of a

Figure 1 Folding of a node v.



maximum independent set of a graph G.

Lemma 1 *For any graph G,*

- **(Connected components)** *If G properly contains a connected component C,*

$$\alpha(G) = \alpha(C) + \alpha(G - C).$$

- **(Dominance)** *If there are two nodes v and w such that $N[w] \subseteq N[v]$ (w* dominates *v),*

$$\alpha(G) = \alpha(G - \{v\}).$$

- **(Folding)** *For any foldable node v,*

$$\alpha(G) = 1 + \alpha(\widetilde{G}(v)).$$

Proof. The first two properties are trivial.

Consider folding. Let S be a maximum independent set of G. If $v \in S$, $S \setminus \{v\}$ is an independent set of $\widetilde{G} = \widetilde{G}(v)$. Otherwise, S contains at least one node of $N(v)$ (since it is of maximum cardinality). If $N(v) \cap S = \{u\}$, $S \setminus \{u\}$ is an independent set of $\widetilde{G}$. Otherwise, it must be $N(v) \cap S = \{u_i, u_j\}$, for two non-adjacent nodes u_i and u_j (since $N(v)$ does not contain any anti-triangle by assumption). In this case $S \cup \{u_{ij}\} \setminus \{u_i, u_j\}$ is an independent set of $\widetilde{G}$. It follows that $\alpha(G) \le 1 + \alpha(\widetilde{G})$. A similar argument shows that $\alpha(G) \ge 1 + \alpha(\widetilde{G})$. $\square$

Figure 2 Example of mirrors: u is a mirror of v.

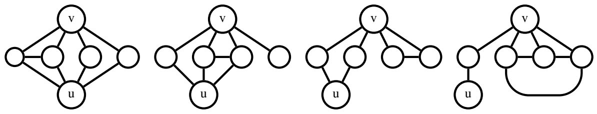


Figure 3 A recursive algorithm for maximum independent set.

int mis(G) {
(0) **if**($G = \emptyset$) **return** 0;
(1) **if**($\exists$ component $C \subset G$) **return** mis(C)+mis($G - C$);
(2) **if**($\exists$ nodes v and w: $N[w] \subseteq N[v]$) **return** mis($G - \{v\}$);
(3) **if**($\exists$ a foldable node v: $d(v) \leq 4$ and $N(v)$ contains at most 3 anti-edges) {
take one such node v of minimum degree;
return 1+mis($\widetilde{G}(v)$);
}
(4) select a node v of maximum degree;
return max{mis($G - \{v\} - M(v)$), 1+mis($G - N[v]$)};
}

We introduce the following useful notion of *mirror*. Given a node v, a *mirror* of v is a node $u \in N^2(v)$ such that $N(v) \setminus N(u)$ is a (possibly empty) clique. We denote by $M(v)$ the set of mirrors of v. Examples of mirrors are given in Figure 2. Intuitively, when you discard a node v, you can discard its mirrors as well without modifying the maximum independent set size. This intuition is formalized in the following lemma.

Lemma 2 (Mirroring) *For any graph G and for any node v of G,*

$$\alpha(G) = \max\{\alpha(G - \{v\} - M(v)), 1 + \alpha(G - N[v])\}.$$

Proof. Node v can either belong to a maximum independent set or not, from which we obtain the trivial equation

$$\alpha(G) = \max\{\alpha(G - \{v\}), 1 + \alpha(G - N[v])\}.$$

Thus it is sufficient to show that, if v is not contained in any maximum independent set, the same holds for its mirrors $M(v)$. Following the proof of Lemma 1, if no maximum independent set contains v, every maximum independent set contains at least two nodes in $N(v)$. Consider a mirror $u \in M(v)$. Since every independent set contains at most one node in $N(v) \setminus N(u)$ (which is a clique by assumption), it must contain at least one node in $N(v) \cap N(u) \subseteq N(u)$. It follows that u is not contained in any maximum independent set. □

3 The Algorithm

All the previous exact algorithms for the maximum independent set problem are analyzed by using the number of nodes as a measure of the size of the problem (in [1] the number of edges is considered, but that approach works only for sparse graphs). According to this standard analysis, simple algorithms perform quite bad. Thus it is not surprising that the currently fastest algorithms for maximum independent set, including Robson's $O^*(2^{0.296n})$ algorithm, are rather complicated, with up to several hundreds of branching rules. In this paper we use a different approach to the problem: instead of developing even more complicated algorithms, we analyze a very simple algorithm with a more careful choice of the measure for the size of a problem instance.

Our simple algorithm, which we call mis, works as follows (see also Figure 3). If G is (0) the empty graph, $mis(G) = 0$. Otherwise, mis tries to reduce the size of the problem without branching, by applying Lemma 1. Specifically, (1) if G contains a proper connected component C, the algorithm recursively solves the subproblems induced by C and $G-C$ separately, and sums the solutions obtained

$$mis(G) = mis(C) + mis(G-C).$$

Else, (2) if there are two (adjacent) nodes v and w, with $N[w] \subseteq N[v]$, `mis` discards v:

$$mis(G) = mis(G-\{v\}).$$

If none of the conditions above holds, and (3) there is a foldable node v of degree $d(v) \leq 3$, or a foldable node of degree $d(v) = 4$ with at most three anti-edges in $N(v)$, the algorithm selects one such node v of minimum degree and folds it:

$$mis(G) = 1 + mis(\widetilde{G}(v)).$$

The reason of the restrictions on the nodes we fold will be clearer from the analysis. Intuitively, we want to avoid that folding increases the "size" of the problem.

As a last choice, (4) the algorithm greedily selects a node v of maximum degree, and branches according to Lemma 2:

$$mis(G) = max\{mis(G-\{v\}-M(v)), 1 + mis(G-N[v])\}.$$

Notice that with simple modifications, the algorithm can also provide one maximum independent set (besides its cardinality).

To emphasize the importance of a good choice of the measure, we sketch the analysis of *mis* according to the standard measure $k = k(G) = n$. Let $P[k]$ be the number of leaves in the search tree generated by the algorithm to solve a problem of size k. Of course, $P[0] = 1$. If one of the conditions of steps (1), (2), and (3) is satisfied, $P[k] \leq P[k-1]$. Otherwise consider the node v at which we branch. Note that $d(v) \geq 3$. If $d(v) = 3$, when we discard v, we either discard a mirror of v or we fold a neighbor w of v in the following step (since $d(w) = 2$ after removing v). In both cases, we decrease the number of nodes by at least two. When we take v, we remove $N[v]$, where $|N[v]| = 4$. This leads to $P[k] \leq P[k-2] + P[k-4]$. Assume now $d(v) \geq 4$. In the worst case, v has no mirrors ($M(v) = \emptyset$). When we discard or take v, we remove at least one or five nodes, respectively. Thus $P[k] \leq P[k-1] + P[k-5]$. We conclude that $P[k] \leq \alpha^k$, where $\alpha = 1.3247\ldots < 2^{0.406}$ is the largest root of polynomials $(x^5 - x^4 - 1)$ and $(x^4 - x^2 - 1)$. Since in each step the size of the graphs generated decreases by at least one, the depth of the search tree is at most n. Moreover, solving each subproblem, not considering the possible recursive calls, takes polynomial time. Thus the time complexity of the algorithm is $O^*(P[n]) = O^*(2^{0.406n})$.

In next section we will show how to refine the running time analysis of `mis` to $O^*(2^{0.288n})$ via a more careful choice of the measure $k(G)$ (without modifying the algorithm!).

4 The Analysis

When we measure the size of a maximum independent set instance with the number of nodes, we do not take into account the fact that decreasing the degree of a node v has a positive impact on the progress of the algorithm (even if we do not immediately remove v from the graph). In fact, decreasing the degree of a node pays off on long term, since the nodes of degree at most two can be filtered out without branching.

This suggests the idea to give a different "weight" to nodes of different degree. In particular, let n_i ($n_{\geq i}$) denote the number of nodes of degree i (at least i) in the graph considered. We will use the following measure $k = k(G)$ of the size of G:

$$k(G) = \sum_{i\geq 0} w_i n_i \leq n,$$

where the weights $w_i \in [0,1]$ will be fixed in the following. In order to simplify the running time analysis, we make the following assumptions:

- First of all, $w_0 = w_1 = w_2 = 0$. The reason for this assumption is that nodes of degree at most two can always be removed without branching during steps (1), (2), and (3) of the algorithm. Thus their presence contributes to the running time only via a polynomial factor.
- Second, $w_i = 1$ for $i \geq 7$. This way, we have to compute only a finite (small) number of weights.
- When the degree of a node decreases from i to $i-1$, its weight decreases by $\Delta w_i = w_i - w_{i-1}$. We assume that $\Delta w_3 \geq \Delta w_4 \geq \Delta w_5 \geq \Delta w_6 \geq \Delta w_7 \geq 0$. In other words, the weights decrease from w_7 to w_2 at increasing speed. The reason for this assumption will be clearer from the analysis.

Theorem 3 *Algorithm* `mis` *solves the maximum independent set problem in time* $O^*(2^{0.298n})$.

Proof. The correctness of the algorithm immediately follows from Lemmas 1 and 2.

Let $P[k]$ be the total number of leaves in the search tree generated to solve a problem of size k. Also in this case we can assume $P[0] = 1$. In fact, when $k = 0$, the algorithm solves the problem without branching (in polynomial time). From the discussion of previous section, it is sufficient to show that $P[k] \leq 2^{0.298k} \leq 2^{0.298n}$. We break the running time analysis in different parts, one for each step of the algorithm.

(1) Connected components. Suppose there is a connected component C of size k_1. The size of $G - C$ is trivially $k_2 = k - k_1$. Thus

$$P[k] \leq P[k_1] + P[k_2]. \tag{1}$$

(2) Dominance. When we apply dominance, we generate a unique subproblem. Thus, from the point of view of the bound on $P[k]$, it is sufficient to guarantee that the size of the problem does not increase. This is trivially true in the case of dominance, since, when we remove a node v, the size of the problem decreases by

$$w_{d(v)} + \sum_{u \in N(v)} \Delta w_{d(u)} \geq 0.$$

(3) Folding. By basically the same arguments as in the dominance case, also in the case of folding it is sufficient to guarantee that the size of the problem does not increase. This property is trivially true when we fold a node v of degree zero or one (since we simple remove $N[v]$ from the graph). Unfortunately, this is not always the case when $d(v) \geq 2$. In fact in this case, besides removing nodes, we also introduce new nodes, possibly of larger degree (and weight). Hence we need to enforce some further constraints on the weights.

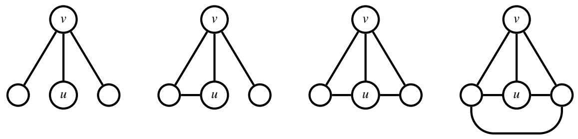
Let $N(v) = \{u_1, u_2, \ldots, u_d\}$, with $d_i = d(u_i)$. We distinguish different cases, depending on the value of $d \in \{2,3,4\}$.

(3.a) case $\mathbf{d = 2}$. Node u_1 and u_2 cannot be adjacent (otherwise v dominates them). Thus, by folding v, we remove $N[v]$ and we introduce a unique node u_{12} of degree $d(u_{12}) \leq (d_1 - 1) + (d_2 - 1) = d_1 + d_2 - 2$ (which implies that the weight of this new node is at most $w_{d_1+d_2-2}$). Hence, for all $d_1, d_2 \geq 2$,

$$w_2 + w_{d_1} + w_{d_2} - w_{d_1+d_2-2} = w_{d_1} + w_{d_2} - w_{d_1+d_2-2} \geq 0. \tag{2}$$

(3.b) case d $= 3$. By a simple case analysis (see Figure 4), $N(v)$ must contain exactly one edge, say $\{u_1, u_2\}$. Thus by folding v, we introduce two new nodes u_{13} and u_{23}, where

Figure 4 Possible graphs $N[v]$ for $d(v) = 3$. In the first case there is one anti-triangle in $N(v)$. In the last two cases v dominates u. The algorithm may fold v only in the second case.



$d(u_{i3}) \leq (d_i - 2) + (d_3 - 1) + 1 = d_i + d_3 - 2$. Hence, for all $d_1, d_2, d_3 \geq 3$,

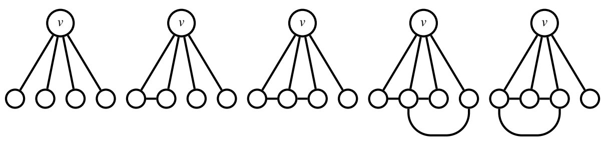
$$w_3 + w_{d_1} + w_{d_2} + w_{d_3} - w_{d_1+d_3-2} - w_{d_2+d_3-2} \geq 0. \tag{3}$$

(3.c) case d $= 4$. Assume $N(v)$ contains two non-incident edges, say $\{u_1, u_2\}$ and $\{u_3, u_4\}$. Since there cannot be four anti-edges in $N(v)$, there must be a third edge incident to both $\{u_1, u_2\}$ and $\{u_3, u_4\}$, say $\{u_2, u_3\}$. Note that $d_i \geq 4$ for all i, since otherwise we could fold a node of degree three. In the worst case (there are no other edges), when we fold v we introduce three nodes u_{13}, u_{24}, and u_{14} of degree at most $d_1 + d_3 - 3$, $d_2 + d_4 - 3$, and $d_1 + d_4 - 2$, respectively. Thus, for all $d_1, d_2, d_3, d_4 \geq 4$,

$$w_4 + \sum_{i=1}^{4} w_{d_i} - w_{d_1+d_3-3} - w_{d_2+d_4-3} - w_{d_1+d_4-2} \geq 0. \tag{4}$$

Now suppose the condition above does not hold (all the edges in $N(v)$ are incident to each other). Again by a simple case enumeration (see Figure 5), there must be exactly three edges which form a triangle, say $\{u_1, u_2\}$, $\{u_2, u_3\}$, and $\{u_1, u_3\}$. By the same argument

Figure 5 Possible graphs $N[v]$ for $d(v) = 4$, where all the edges in $N(v)$ are incident to each other. In the first four cases $N(v)$ contains one anti-triangle. Only in the last case v can be folded.



as above, the degree of u_1, u_2, and u_3 must be at least four. When we fold v, we introduce three nodes u_{14}, u_{24}, and u_{34}, where $d(u_{i4}) \leq (d_i - 3) + (d_4 - 1) + 2 = d_i + d_4 - 2$. Thus for all $d_1, d_2, d_3 \geq 4$ and $d_4 \geq 3$:

$$w_4 + \sum_{i=1}^{4} w_{d_i} - w_{d_1+d_4-2} - w_{d_2+d_4-2} - w_{d_3+d_4-2} \geq 0. \tag{5}$$

Note that it is sufficient to consider the (finite) subset of constraints where $d_i \leq 8$ for all i (each constraint with $d_i > 8$ is dominated by the same constraint with d_i replaced by 8).

(4) Branching. When we branch, we can assume that, for every node w: (i) $deg(w) \geq 3$; (ii) if $deg(w) \leq 4$, either w is not foldable or $N(w)$ contains more than three anti-edges; (iii) w is not dominated and does not dominate any other node.

Suppose we branch at a given node v, with $N(v) = \{u_1, u_2, \dots, u_d\}$ and $d_i = d(u_i)$. We denote by $m_i = m_i(v)$ the number of nodes of degree i in $N(v)$:

$$m_i = |\{u \in N(v) \, : \, d(u) = i\}|.$$

Let moreover $p_h = p_h(v)$ be the number of nodes in $N^2(v)$ which have exactly h neighbors in $N(v)$:

$$p_h = |\{w \in N^2(v) \, : \, |N(w) \cap N(v)| = h\}|.$$

Note that (at least) the nodes corresponding to p_{d-1} and p_d are mirrors of v. Additionally, the number of edges between $N(v)$ and $N^2(v)$ is

$$p = p(v) = \sum_{h=1}^{d} h \, p_h.$$

We also define $m_{\geq i} = \sum_{j \geq i} m_j$ and $p_{\geq h} = \sum_{j \geq h} p_j$.

Suppose we discard v. The size of the problem decreases by w_d because of the removal of v, by at least $p_d \, w_d + p_{d-1} \, w_{\max\{3,\, d-1\}}$ because of the removal of the mirrors of v, and by at least $\sum_{i=3}^{d} m_i \Delta w_i$ because of the reduction of the size of $N(v)$. Altogether the reduction is

$$\begin{aligned} \Delta_{OUT} &= \Delta_{OUT}(v) \\ &\geq w_d + p_d \, w_d + p_{d-1} \, w_{\max\{3,\, d-1\}} + \sum_{i=3}^{d} m_i \Delta w_i. \end{aligned}$$

Consider now the case we take v. The size of the problem decreases by w_d because of the removal of v, and by $\sum_{i=3}^{d} m_i w_i$ because of the removal of $N(v)$. The size of the problem further decreases because of the reduction of the size of $N^2(v)$. Consider a node $z \in N^2(v)$ with h neighbors in $N(v)$. Note that the reduction of the size of z is

$$w_{d(z)} - w_{d(z)-h} = \Delta w_{d(z)} + \Delta w_{d(z)-1} + \ldots + \Delta w_{d(z)-h+1}.$$

If $h = 1$, the minimum reduction of the size of z is achieved when z has the largest possible degree $d(z) = d$:

$$w_{d(z)} - w_{d(z)-1} = \Delta w_{d(z)} \geq \Delta w_d = w_d - w_{d-1}.$$

Now consider the case $h \geq 2$. Any case with $d(z) - h \geq 3$ is dominated by a case where the h edges considered are incident to h distinct nodes of degree d:

$$\begin{aligned} w_{d(z)} - w_{d(z)-h} &= \Delta w_{d(z)} + \Delta w_{d(z)-1} + \cdots + \Delta w_{d(z)-h+1} \\ &\geq h \Delta w_d = h(w_d - w_{d-1}). \end{aligned}$$

Thus we can assume $d(z) - h \leq 2$, and the reduction of the size of z is $w_{d(z)} - w_{d(z)-h} = w_{d(z)}$. It follows that in the worst case z has the minimum possible degree compatible with h: $d(z) = \max\{3, h\}$. Thus the reduction of the size of $N^2(v)$ is at least

$$p_1 \, \Delta w_d + \sum_{h=2}^{d} p_h \, w_{\max\{3,h\}}.$$

Each node in $N(v)$ has at least one neighbor in $N^2(v)$ (otherwise we could remove v by dominance). Moreover, the nodes of degree three in $N(v)$ have exactly two neighbors in $N^2(v)$ (otherwise we could fold them). Thus $p \geq m_{\geq 4} + 2m_3 = d + m_3$, the worst case occurring when $p = d + m_3$.

Altogether, the reduction of the size of the problem when we take v is

$$\Delta_{IN} = \Delta_{IN}(v) \geq w_d + \sum_{i=3}^{d} m_i w_i + p_1 \Delta w_d + \sum_{h=2}^{d} p_h w_{\max\{3,h\}}.$$

The following recurrence follows, for all m_i and p_h such that $\sum_{i=3}^{d} m_i = d$ and $\sum_{h=1}^{d} h\, p_h = d + m_3$:

$$P[k] \leq P[k - \Delta_{OUT}] + P[k - \Delta_{IN}]. \tag{6}$$

Observe that it is sufficient to consider only a finite number of recurrences. In fact, for $d \geq 8$, independently from the values of the m_i, the worst case value of the p_i is $p_1 = p$ and $p_i = 0$ for $i \neq 1$ (since $\Delta w_d = 0$ for $d \geq 8$). Recall that, for $d \geq 8$, $w_d = 1$. Thus all the worst case recurrences for $d \geq 8$ are dominated by a recurrence of the kind

$$P[k] \leq P[k - 1 - \sum_{i=3}^{7} m_i \Delta w_i] + P[k - 1 - \sum_{i=3}^{7} m_i w_i - m_{\geq 8}],$$

with $\sum_{i=3}^{7} m_i + m_{\geq 8} = 8$.

Consider an assignment of the weights w_3, w_4, w_5, and w_6 which satisfies the initial assumptions and the constraints (2), (3), (4), and (5). It turns out that $P[k] \leq \alpha^k$, where $\alpha > 1$ is the largest real root of the set of equations

$$\alpha^k = \alpha^{k-\Delta_{OUT}} + \alpha^{k-\Delta_{IN}}$$

corresponding to the recurrences of kind (6). Thus the estimation of the running time reduces to choosing the weights w minimizing $\alpha = \alpha(w)$ (we refer to Eppstein's work [7] on quasi-convex programming for a general treatment of such a problem). We numerically obtained

$$(w_3, w_4, w_5, w_6) = (0.5596, 0.8405, 0.9562, 0.9964),$$

which leads to $\alpha < 2^{0.298}$, and thus to the claimed running time $O^*(2^{0.298\,n})$.

□

The running time analysis can be further refined to $O^*(2^{0.288\,n})$, still without modifying the algorithm. We give the details in the appendix.

Theorem 4 *Algorithm* `mis` *solves the maximum independent set problem in time* $O^*(2^{0.288\,n})$.

5 A Lower Bound

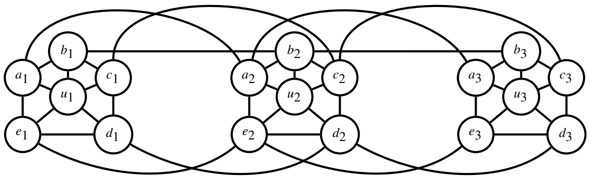
We have showed that a more careful choice of the measure leads to a much tighter bound on the running time of `mis`. Nonetheless, the bound achieved might still be pessimistic.

To estimate the possibilities for further improvements, it is natural to ask for lower bounds on such running times. (Notice that we are concerned with lower bounds on the complexity of a particular algorithm and not with lower bounds on the complexity of an algorithmic problem).

Theorem 5 *The worst case running time of* `mis` *is* $\Omega(2^{n/6}) = \Omega(2^{0.166\,n})$.

Proof. Consider the following connected graph G_ℓ, $\ell \geq 1$, of $n = 6\ell$ nodes: G_ℓ consists of ℓ blocks $B_1, B_2, \ldots, B_\ell$. Each block B_i, $i < \ell$, is formed by six nodes a_i, b_i, c_i, d_i, e_i, and u_i. For each $1 \leq i \leq \ell$, node u_i is adjacent to nodes a_i, b_i, c_i, d_i, e_i which form a chord-less cycle of length five. Also for each $i = 1, \ldots, \ell - 1$ there are edged $\{a_i, a_{i+1}\}$, $\{b_i, b_{i+1}\}$, $\{c_i, c_{i+1}\}$, $\{d_i, d_{i+1}\}$, and $\{e_i, e_{i+1}\}$. (See Figure 6 for an example).

Figure 6 Lower-bound graph G_ℓ for $\ell = 3$. Algorithm `mis` may branch at node u_1.



Let us apply Algorithm `mis` to graph G_ℓ. The graph is connected. Moreover, no node is foldable and dominance cannot be applied. Thus the algorithm branches at a node of maximum degree: so it can branch at u_1. If we take u_1, we remove nodes a_1, b_1, c_1, d_1, and e_1 arriving at graph $G_{\ell-1}$. If we discard u_1, the remaining vertices of B_1 are of degree three and are not foldable. So at the next step the algorithm can branch at u_2 and so on. Thus the algorithm can branch $\ell = n/6$ times, which implies a running time $\Omega(2^{n/6}) = \Omega(2^{0.166n})$. □

The large gap between upper and lower bound leaves room for improvement. Is is possible to further refine the analysis of algorithm `mis`, possibly via a further refined measure of the size of maximum independent set instances? Finding such measure is an interesting challenge.

References

[1] R. Beigel. Finding maximum independent sets in sparse and general graphs. Proceedings of the *10th ACM-SIAM Symposium on Discrete Algorithms (SODA 1999)*, pp. 856–857.

[2] R. Beigel and D. Eppstein, 3-coloring in time $O(1.3289^n)$, *J. Algorithms*, 54: 444–453, 2005.

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

[4] J. Chen, I. Kanj, and W. Jia. Vertex cover: further observations and further improvements. *Journal of Algorithms*, 41:280–301, 2001.

[5] J. Chen, I. A. Kanj, and G. Xia. Labeled search trees and amortized analysis: improved upper bounds for NP-hard problems. Proceedings of the *14th Annual International Symposium on Algorithms and Computation (ISAAC 2003)*, Springer LNCS vol. 2906, 2003, pp. 148–157.

[6] E. Dantsin, A. Goerdt, E. A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, and U. Schöning. A deterministic $(2-2/(k+1))^n$ algorithm for k-SAT based on local search. *Theoretical Computer Science*, 289(1):69–83, 2002.

[7] D. Eppstein. Quasiconvex analysis of backtracking algorithms. Proceedings of the *15th ACM-SIAM Symposium on Discrete Algorithms (SODA 2004)*, pp. 781–790.

[8] F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and Conquer: Domination - A Case Study, Proceedings of the *32nd International Colloquium on Automata, Languages and Programming (ICALP 2005)*, Springer LNCS vol. 3580, 2005, pp. 191–203.

[9] F. V. Fomin, D. Kratsch, and I. Todinca. Exact algorithms for treewidth and minimum fill-in. Proceedings of the *31st International Colloquium on Automata, Languages and Programming (ICALP 2004)*, Springer LNCS vol. 3142, 2004, pp. 568–580.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. Freemann, 1979.

[11] J. Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Math.* 182 (1):105–142, 1999.

[12] K. Iwama and S. Tamaki. Improved upper bounds for 3-SAT. Proceedings of the *15th ACM-SIAM Symposium on Discrete Algorithms (SODA 2004)*, p.328.

[13] T. Jian. An $O(2^{0.304n})$ algorithm for solving maximum independent set problem. *IEEE Transactions on Computers*, 35(9):847–851, 1986.

[14] J. M. Robson. Algorithms for maximum independent sets. *Journal of Algorithms*, 7(3):425–440, 1986.

[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] U. Schöning. A Probabilistic Algorithm for k-SAT and Constraint Satisfaction Problems. Proceedings of the *40th IEEE Symposium on Foundations of Computer Science (FOCS 1999)*, pp. 410-414.

[17] R. Tarjan. Finding a maximum clique. Technical Report 72-123, Computer Sci. Dept., Cornell Univ., Ithaca, NY, 1972.

[18] R. Tarjan and A. Trojanowski. Finding a maximum independent set. *SIAM Journal on Computing*, 6(3):537–546, 1977.

[19] R. Williams. A new algorithm for optimal constraint satisfaction and its implications. Proceedings of the *31st International Colloquium on Automata, Languages and Programming (ICALP 2004)*, Springer LNCS vol. 3142, 2004, pp. 1227–1237.

[20] G. J. Woeginger. Exact algorithms for NP-hard problems: A survey. *Combinatorial Optimization – Eureka, You Shrink*, Springer LNCS vol. 2570, 2003, pp. 185–207.

Appendix

Proof. (Theorem 4) We can refine the running time analysis given in the proof of Theorem 3 in the following way.

By a simple combinatorial property, if the sum of the degrees of $N[v]$ is even (odd), so it must be $p = p(v)$. Thus, if the lower bound $d+m_3$ on p is odd (even), we can replace it with $d+m_3+1$:

$$p \geq d+m_3+[(\sum_{u\in N[v]} d(v)+(d+m_3)) \pmod 2].$$

The lower bound on p can be further refined for $d=4$ (while for $d=3$, $p=d+m_3=6$ deterministically). If there are no edges in $N(v)$, $p=2m_3+3m_4$. If there is exactly one edge in $N(v)$, say $\{u_1,u_2\}$, the degree of u_1 and u_2 must be four (otherwise we would fold one of u_1 and u_2). Thus $p=2m_3+3m_4-2$, with $m_4 \geq 2$. It remains to consider the case there are at least two edges in $N(v)$. Following the analysis of case (3.c), if such edges are not incident, there are no other edges in $N(v)$ and all the nodes in $N(v)$ have degree four ($m_4=4$). Thus $p=2m_3+3m_4-4=8$. Otherwise there are at least two incident edges in $N(v)$, say $\{u_1,u_2\}$ and $\{u_2,u_3\}$, from which we obtain $p \geq 2m_3+3m_4-4$, with $m_4 \geq 3$. Altogether, $p \geq 7$ if $m_4=3$, and $p \geq 8$ in all the other cases.

Consider now the case $d = 3$. Without loss of generality, we can assume that $N^3(v)$ is not empty. In fact otherwise v would be contained in a small (constant-size) connected component, and thus it would be $P[k] = O(1)$ trivially. This has two consequences. First, we can assume $p_3 < 2$. Second, suppose all the nodes in $N(v)$ have at least one neighbor in $M(v)$. This surely happens if $p_3 > 0$ or $p_2 = 3$, and it may happen for $p_2 = 2$. After removing $\{v\} \cup M(v)$, nodes in $N(v)$ have degree at most one. Thus the nodes in $N(v)$ are taken, and all the remaining nodes in $N^2(v)$ are discarded (without branching). Hence the degree of all the nodes in $N^3(v)$ decreases, with a total extra reduction of the size of the problem by at least $\Delta w_3 = w_3$.

Suppose now $p_1 = p = 6$ (which implies that there are no mirrors). After removing v, the neighbors of v become nodes of degree two with two non-adjacent neighbors of degree three. One of them will be folded, with a reduction of the size of the problem by at least $w_2 + 2w_3 - w_4 = 2w_3 - w_4$ (which must be non-negative by the constraints on folding).

By considering the new (refined) set of recurrences, and by imposing $(w_3, w_4, w_5, w_6) = (0.5139, 0.7783, 0.9230, 0.9842)$, one obtains the claimed $O^*(2^{0.288n})$ running time. $\square$