Exact algorithms for dominating clique problems

Nicolas Bourgeois, Federico Della Croce, Bruno Escoffier, Vangelis Paschos

UNIVERSITE PARIS DAUPHINE

Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision

CNRS UMR 7024

CAHIER DU LAMSADE

285

Septembre 2009

Exact algorithms for dominating clique problems

N. Bourgeois, F. Della Croce, B. Escoffier, V. Th. Paschos

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Exact algorithms for dominating clique problems

N. Bourgeois[1] F. Della Croce[2] B. Escoffier[1] V. Th. Paschos[1]

[1]LAMSADE, CNRS and Université Paris-Dauphine, France
{bourgeois,escoffier,paschos}@lamsade.dauphine.fr
[2]D.A.I., Politecnico di Torino, Italy
{federico.dellacroce}@polito.it

September 7, 2009

Abstract

We handle in this paper three dominating clique problems, namely, the decision problem itself when one asks if there exists a dominating clique in a graph G and two optimization versions where one asks for a maximum- and a minimum-size dominating clique, if any. For the three problems we propose optimal algorithms with provably worst-case upper bounds improving existing ones by (D. Kratsch and M. Liedloff, *An exact algorithm for the minimum dominating clique problem*, Theoretical Computer Science 385(1-3), pp. 226–240, 2007). We then settle all the three problems in sparse and dense graphs also providing improved upper running time bounds.

1 Introduction

Given a graph $G(V, E)$, a dominating clique is a clique which is also a dominating set for G. Determining whether there exists a dominating clique or not is known to be **NP**-hard ([5]) and so are the two related problems MIN DOMINATING CLIQUE and MAX DOMINATING CLIQUE, where we are asked for a dominating clique of minimum and of maximum size, respectively. These problems can of course be solved by enumerating all the subsets of V. So, an interesting problem is to devise algorithms able to optimally solve the three problems EXISTING DOMINATING CLIQUE, MIN DOMINATING CLIQUE and MAX DOMINATING CLIQUE within time $O(2^{c|V|}p(|V|))$, where c is a constant lower than 1 and p some polynomial function. Notice that, compared to the slightest improvement of c, p is non relevant. So, from now on, we use notation $O^*(2^{c|V|})$ in order to omit polynomial factors.

Regarding EXISTING DOMINATING CLIQUE, trivial $O^*(2^{|V|})$ bound has been initially broken by [6] down to $O^*(3^{|V|/3}) = O^*(1.443^{|V|})$ using a result by [8], namely that the number of maximal (for inclusion) independent sets in a graph is at most $3^{|V|/3}$. Recently, [7] have proposed a branching algorithm that, according to a measure and conquer analysis [2], solves MIN DOMINATING CLIQUE with polynomial space and running time $O^*(1.3387^{|V|})$, and another one that requires $O^*(1.3234^{|V|})$ time and space. Naturally, these algorithms also solve EXISTING DOMINATING CLIQUE.

In this paper, we first devise simple branching algorithms improving [7] for EXISTING DOMINATING CLIQUE (Section 2). The first one runs with tight running time $O^*(2^{2|V|/5}) = O^*(1.3196^{|V|})$ using polynomial space; we then further improve it to produce a second algorithm that solves EXISTING DOMINATING CLIQUE in $O^*(2^{0.35|V|}) = O^*(1.2740^{|V|})$ still using polynomial space. Finally, using memorization techniques, we propose a third algorithm working within $O^*(2^{0.329|V|}) = O^*(1.2556^{|V|})$ running time and space. In Section 3 we settle MAX DOMINATING CLIQUE and propose an algorithm with tight running time $O^*(2^{2|V|/5})$ using polynomial space. Allowing exponential space, we then decrease running time to $O^*(2^{0.372|V|}) = O^*(1.2937^{|V|})$. In Section 4 we settle MIN DOMINATING CLIQUE and propose an algorithm with tight running time $O^*(2^{0.4058|V|}) = O^*(1.3248^{|V|})$ using polynomial space. Allowing exponential space, we then decrease running time to $O^*(2^{0.3762|V|}) = O^*(1.298^{|V|})$. Table 1 summarizes our results in Sections 2, 3 and 4.

In Section 5, we restrict ourselves to sparse and dense graphs, and produce parameterized algorithms depending on minimum, maximum and average degree. These results are summarized in Table 2 (some of them being obvious), where $D(G) = 2|E|/(|V|(|V| - 1))$ is the density of the graph G. For instance,

	Former result	Our result
EXISTING DOMINATING CLIQUE	$O^*(1.3387^{\|V\|})$	$O^*(1.2740^{\|V\|})$
- exponential space allowed	$O^*(1.3234^{\|V\|})$	$O^*(1.2556^{\|V\|})$
MAX DOMINATING CLIQUE	$O^*(1.4423^{\|V\|})$	$O^*(1.3196^{\|V\|})$
- exponential space allowed		$O^*(1.2937^{\|V\|})$
MIN DOMINATING CLIQUE	$O^*(1.3387^{\|V\|})$	$O^*(1.3248^{\|V\|})$
- exponential space allowed	$O^*(1.3234^{\|V\|})$	$O^*(1.298^{\|V\|})$

Table 1: Results of Sections 2, 3 and 4.

	MIN DOMINATING CLIQUE	MAX DOMINATING CLIQUE
Max degree Δ	2^{Δ}	$3^{\Delta/3}$
Min degree $\delta \geqslant \|V\|/2$	$\binom{\|V\|}{\|V\|-\delta}$	$1.22^{\|V\|}\binom{\|V\|}{\|V\|-\delta}$
$D(G) \geqslant 3/4$	$\binom{\|V\|}{\sqrt{1-D(G)}\|V\|}$	$1.22^{\|V\|}\binom{\|V\|}{\sqrt{1-D(G)}\|V\|}$
$D(G) \leqslant 1/4$	$\binom{\|V\|}{\sqrt{D(G)}\|V\|}$	$(1+\mu/\sqrt{D(G)})^{\sqrt{D(G)}}$

Table 2: Results of Section 5 where $\mu \in [0,1]$ is an increasing function of $D(G)$.

we show in Section 5 that if $n-\delta = o(n)$, or if $D(G) = 1-o(1)$, MIN DOMINATING CLIQUE can be solved in subexponential time and MAX DOMINATING CLIQUE within roughly the same running time as MAX CLIQUE. On the other hand, if $D(G) = o(1)$ both MIN and MAX DOMINATING CLIQUE can be also solved in subexponential time.

It is easy to see that no polynomial time approximation algorithm can exist for MIN DOMINATING CLIQUE and MAX DOMINATING CLIQUE, since any such algorithm should first solve EXISTING DOMINATING CLIQUE that is **NP**-complete. For this reason, in Section 6, we present some approximation results for MIN DOMINATING CLIQUE and MAX DOMINATING CLIQUE by exponential time algorithms that run faster than the corresponding exact algorithms for these problems. Recall that the approximation ratio of an algorithm `A` supposed to solve an **NP**-hard problem Π is defined by $\max\{m(I,S)/\operatorname{opt}(I), \operatorname{opt}(I)/m(I,S)\}$ where, for any instance I of Π, $m(I,S)$ is the value of the solution S computed by `A` in I and $\operatorname{opt}(I)$ is the value of an optimal solution of I.

In what follows, given a graph $G(V,E)$ and a vertex $v \in V$, the neighborhood $N(v)$ of v is the set of vertices that are adjacent to v and the set $N[v] = N(v) \cup \{v\}$ is called the closed neighborhood of v. For the degree of v, we use the notation $d(v) = |N(v)|$; the anti-degree of v is the quantity $\bar{d}(v) = |V \setminus N[v]|$. For any set $H \subset V$, we write $N_H(v) = N(v) \cap H$, $d_H(v) = |N_H(v)|$ and $\bar{d}_H(v) = |H \setminus N_H[v]|$; $G[H]$ is the subgraph induced by H in G. Finally, for $v \in V$, we set $f(v) = 1 - |N(v)|/|V|$.

For simplicity, we set $n = |V|$ and $m = |E|$. For any function $\mathcal{T}$, $T(n)$ stands for the maximum running time the algorithm requires to compute $\mathcal{T}$ on a graph containing at most n vertices.

Due to limits in paper's length some of the results are given without proofs that can be found in [**?**].

2 EXISTING DOMINATING CLIQUE

We first make some very simple remarks, that have already been stated, for instance, in [7], but that one should keep in mind in order to understand how our algorithms work.

Remark 1. If there exists some dominating clique K^*, then any clique that contains K^* is a dominating clique. ∎

Remark 2. For any $v \in V$, any dominating clique containing v contains only vertices from $N[v]$. ∎

Remark 3. For any $v \in V$, if there exists some dominating clique K containing v, then there exists a dominating clique $K' \subseteq K$ with $|K'| \leqslant |V \setminus N(v)|$. ∎

Clique K' claimed by Remark 3 could be, for instance, obtained by simply taking one neighbor in V for any vertex in $V \setminus N(v)$.

Based on these remarks, we can now describe a first simple algorithm whose running time is worse than the one in [7], but which is very simple to analyse. *It decides whether there exists or not a dominating clique in G with running time $O^*(1.3870^n)$* and works as follows:

> for any $v \in V$: if $f(v) > 0.10686$, then compute any maximal (for inclusion) clique in $N(v)$ and check if one of them is a dominating set in G, otherwise, compute any subset of $N(v)$ of size $nf(v)$ or less and check if one of them is a dominating clique in G; stop at the time a dominating clique is found, or return a negative answer.

Indeed, assume there exists a dominating clique K, and fix some $v \in K$. As pointed out in Remark 2, $K \subseteq N[v]$. According to Remark 1, there exists a maximal dominating clique and according to Remark 3, there exists a dominating clique of size $nf(v)$ or less. Then, regardless of the value of $f(v)$, the algorithm above computes a dominating clique.

Let us now analyse the complexity of the algorithm. Consider first the case $f(v) \geqslant 0.10686$. From [8, 6], it is possible to compute any maximal independent set, thus any maximal clique, with running time:

$$O^*\left(3^{\frac{|N(v)|}{3}}\right) = O^*\left(3^{\frac{1-f(v)}{3}n}\right) = O^*\left(1.3870^n\right)$$

On the other hand, if $f(v)$ is small, then every subset of $N(v)$ whose size is bounded above by $nf(v)$ can be computed with running time:

$$n \times \binom{N(v)}{nf(v)} = n \times \binom{n(1-f(v))}{f(v)n} = O^*\left(\left(\frac{(1-f(v))^{1-f(v)}}{(1-2f(v))^{1-2f(v)}f(v)^{f(v)}}\right)^n\right) = O^*\left(1.3870^n\right)$$

2.1 A tight $O^*(2^{2n/5})$ branching algorithm

We use the same notations as in [7]: S is the set of vertices we have added to the solution, D is the set of vertices we have discarded, $A = \cap_{s\in S} N(s) \setminus D$ is the set of vertices still available and $F = V \setminus (\bigcup_{s\in S} N(s))$ is the set of free vertices, i.e., the set of vertices that still remain to be dominated; $T(S, D, A, F)$ is a boolean function that returns TRUE if and only if there exists a dominating clique in G contained in A and with no vertex from D. As pointed out in [7], for MIN DOMINATING CLIQUE, it is equivalent to solve EXISTING DOMINATING CLIQUE or to determine if there exists $v \in V$ such that $T(\{v\}, \emptyset, N(v), V \setminus N[v]) =$ TRUE.

We devise in this section, an improved algorithm, called `EDC` that computes $T(\{v\}, \emptyset, N(v), V \setminus N[v])$ (for each vertex v) as follows:

1. if $F = \emptyset$, then $T(S, D, A, F) =$ TRUE;
2. if $\exists u \in F$, such that $d_A(u) = 0$, then $T(S, D, A, F) =$ FALSE;
3. if $\exists u \in A$, such that $d_F(u) = 0$, then $T(S, D, A, F) = T(S, D \cup \{u\}, A \setminus \{u\}, F)$
4. otherwise, fix $u \in A$ such that $d_A(u)$ is maximal and set:

$$T(S, D, A, F) = \bigvee_{w \in A \setminus N_A(u)} T\left(S \cup \{w\}, D \cup (A \setminus N_A[w]) \cup N_F(w), N_A(w), F \setminus N_F(w)\right)$$

Rules 1, 2 and 3 are straightforward, so we only need to prove correctness of the branching rule. If we add some vertex u to S, we must discard from A any vertex that is not neighbor of u. On the other hand, assume that every vertex in $A \setminus N(u)$ have been discarded. Then, any vertex still available is in $N_A(u)$, so we can safely add u to S. Hence, in any case we take one vertex from $A \setminus N_A(u)$, leading to the recurrence of the branching rule. Notice that, because of this insertion of u in the last case, the clique computed by Algorithm `EDC` may be not a minimum dominating clique.

Proposition 1. *Algorithm `EDC` decides whether there exists a dominating clique, or not, with running time $O^*(2^{2n/5}) = O^*(1.3196^n)$. This bound is tight.*

Proof. We simply count the number of vertices in the remaining graph having $n = |A \cup F|$ vertices. Fix $\delta = \min_{a \in A}\{|A \setminus N_A(a)|\} = |A \setminus N_A(u)|$. By definition of u we have:

$$T(n) \leqslant \sum_{w \in A \setminus N_A(u)} T\left(|N_A(w)| + |F| - d_F(w)\right) \leqslant \delta \times T(|A| - \delta + |F| - 1) \leqslant \delta \times T(n - \delta - 1)$$

Then, a simple recursion shows that $T(n) \leqslant \max_{\delta \in \mathbb{N}}\{\delta^{n/(\delta+1)}\} = 4^{n/5}$.

Tightness is shown in the following graph. Consider a collection G_p of p stars of size 5 (1 center plus 4 outer vertices). Set A is the set of the outer vertices, and set F is the set of the centers. Any outer vertex u is adjacent to any other outer vertex in any other star; so, $d_A(v) = 4p - 4$. When the algorithm branches on any of these outer vertices, it has to solve EXISTING DOMINATING CLIQUE on four identical copies of G_{p-1} and, in this case $T(n) = \Omega(4^{n/5})$. ∎

2.2 Improvement of EDC

We now refine algorithm EDC in such a way that when branching on a vertex u (of maximum degree $d_A(u)$), instead of removing for each of the δ branches $\delta + 1$ vertices, we remove in the worst case $\delta + 2$ vertices in $\delta - 1$ branches, and $\delta + 1$ vertices only in one branch. To do this, we say that a vertex w is *good* if:

- either $|A \setminus N_A(w)| \geqslant \delta + 1$,
- or w is adjacent to at least two vertices in F,
- or, finally, w is adjacent to only one vertex $v \in F$ and this vertex v has another neighbor w' in $N(w)$.

Otherwise, we say that w is *bad*.

Suppose first that there exists a vertex u such that any vertex in $A \setminus N_A[u]$ is good. We branch on u, as in EDC. For each $w \in A \setminus N_A[u]$, in the branch we take w, we remove at least $\delta + 2$ vertices. Indeed, in the two first cases this is trivial. In the third one, if we take w we can safely discard w' (and we also remove $\delta + 2$ vertices) since, otherwise, if we take w' then w does not cover new elements so w is useless (this case being handled in some other branch).

Assume now there exist at least two non adjacent bad vertices u and u'. Suppose first that u is adjacent to v in F and u' is adjacent to $v' \neq v$. Let $u_1, u_2, \ldots, u_\delta$ the vertices in $A \setminus N_A[u]$ (with $u_\delta = u'$). When branching on u_1, we consider the cases of taking u_1 or not, u_2 or not, ... Then, in the branch where all the other vertices in $A \setminus N_A[u]$ but $u' = u_\delta$ have been discarded, v has degree 1 and then we have to take u in the solution (without branching on u'). In this way, we get $\delta - 1$ branches where we remove $\delta + 1$ vertices.

Suppose now $v' = v$. Then, in the branch where all the other vertices in $A \setminus N_A[u]$ but u' have been discarded, $d(v) = 2$ (since u is bad); so, we have to take either u or u', but it is not interesting to take u' (take u instead). So we can keep u in the solution (without branching on u'). In this way, we get also $\delta - 1$ branches where we remove $\delta + 1$ vertices.

In all, we get either $T(n) \leqslant (\delta - 1)T(n - \delta - 2) + T(n - \delta - 1)$, or $T(n) \leqslant (\delta - 1)T(n - \delta - 1)$. The worst case of these recurrence relations is $T(n) \leqslant 3T(n - 6) + T(n - 5)$, leading to a running time $O^*(1.2740^n) = O^*(2^{0.35n})$. Denoting by EDC' this improved version of EDC, we get the following.

Proposition 2. *Algorithm EDC' decides whether there exists a dominating clique, or not, with running time* $O^*(2^{0.35n}) = O^*(1.2740^n)$.

2.3 Memorization: trading space for time

The principle of memorization, as it has been explained for example in [3], is quite simple. Before running the algorithm on the main graph, we run it on every induced subgraph of size at most αn and store the results in a table[1]. Notice that, thanks to the recurrence defined above at the end of Section 2.2, we only need to call a finite number of subproblems of size $k - 1$ or less in order to compute a given subproblem

[1] Note that, in the algorithms, available vertices never become free (or *vice-versa*), hence the number of subproblems of size αn to consider is $\binom{n}{\alpha n}$.

of size k. Then, using a classical bottom up technique, the total computation time (and space) for all the subproblems of size k, say $S(k, G)$, is at most:

$$O^*\left(\binom{n}{k} + \sum_{i \leqslant k-1} S(i, G)\right)$$

from what we get (for $\alpha \leqslant 1/2$):

$$\sum_{k \leqslant \alpha n} S(k, G) \in O^*\left(\binom{n}{\alpha n}\right)$$

Now, we run the main algorithm until the remaining graph has size αn or less; then a polynomial-time query in the storage table allows us to conclude. Thus, the total running time and space is:

$$O^*\left(\max\left\{\binom{n}{\alpha n}, 1.2740^{(1-\alpha)n}\right\}\right)$$

This value is minimal for an α solution of the equation $1.2740^{1-\alpha} = 1/(\alpha^\alpha(1-\alpha)^{1-\alpha})$, leading to $T(n) = O^*(1.2556^n)$.

3 MAX DOMINATING CLIQUE

The function $T(S, D, A, F)$ is now an integer function that returns the cardinality of a maximum dominating clique in G contained in set A and with no vertex from set D, if any, and $-\infty$ otherwise. Once again, solving MAX DOMINATING CLIQUE is equivalent to finding $\max_{v \in V}\{T(\{v\}, \emptyset, N(v), V \setminus N[v])\}$. Note that now we cannot discard as in Algorithm `EDC` any vertex that has degree 0 in F, because we could be led to a solution that is not a maximum one; however, we claim that we can compute a solution with the same running time by the following algorithm called `MDC1` that works as follows:

1. if $A = F = \emptyset$, then $T(S, D, A, F) = |S|$;
2. if $\exists u \in F$, such that $d_A(u) = 0$, then $T(S, D, A, F) = -\infty$;
3. fix $w \in A$ such that $d_A(w)$ is maximum, and $d_F(w)$ is maximum among vertices of maximum d_A. If there $u \in A \setminus N[w]$, such that $d_F(u) = 0$, then

   $$T(S, D, A, F) = \max_{u' \in A \setminus (N_A(w) \cup \{u\})} \{T\left(S \cup \{u'\}, D \cup (A \setminus N_A[u']) \cup N_F(u'), N_A(u'), F \setminus N_F(u')\right)\}$$

   otherwise

   $$T(S, D, A, F) = \max_{u \in A \setminus N_A(w)} \{T\left(S \cup \{u\}, D \cup (A \setminus N_A[u]) \cup N_F(u), N_A(u), F \setminus N_F(u)\right)\}$$

To prove correctness, just consider the following three facts:

1. if we add some vertex to S, we must discard any vertices from A that are not its neighbors;
2. if every vertices in $A \setminus N(w)$ have been discarded, then any vertex still available is in $N_A(w)$, so we can safely add w to S;
3. if every vertex in $A \setminus N(w)$ but u has been discarded, then only one among u and w may be added; furthermore, if $d_F(u) = 0$, we can safely discard u and add w to S.

Proposition 3. *Algorithm `MDC` computes a maximum-size dominating clique, if any, with running time $O^*(2^{2n/5})$. This bound is tight.*

Proof of Proposition 3 is a straightforward consequence of the following Lemma 1

Lemma 1. *If $\bar{d}_A(w) = \alpha$, for some $\alpha \in \mathbb{N}$, then:*

$$T(n) \leqslant \max\{\alpha \times T(n-\alpha-1), (\alpha+1) \times T(n-\alpha-2), T(n-\alpha) + \alpha \times T(n-\alpha-3)\}$$

Proof. Let $(u_i)_{i\leqslant\alpha}$ be the α neighbors of u. If $d_F(u_1) = 0$, then $(u_2, \ldots, u_\delta \in D) \Rightarrow w \in S$; so, $T(n) \leqslant \delta T(n-\delta-1)$. Otherwise, $\forall i, d_F(u_i) \geqslant 1$. In this case, if $\bar{d}_A(u_1) = \delta$ then, by the fact that $d_F(w)$ is maximum (Step 3 of Algorithm `MDC1`), $d_F(w) \geqslant 1$, and $T(n) \leqslant (\alpha+1)T(n-\alpha-2)$. Finally, in any other case, for any i, $\bar{d}_A(u_i) + d_F(u_i) \geqslant \alpha+2$. Thus, $T(n) \leqslant T(n-\alpha) + \alpha T(n-\alpha-3)$. ∎

To conclude the proof of Proposition 3, it suffices to observe that $T(n) = 2^{2n/5}$ verifies all recurrences stated in Lemma 1. The tightness example is the same as for `EDC`.

Proposition 4. *Using memorization, Algorithm `MDC` computes a maximum-size dominating clique, if any, with running time and space $O^*(1.2937^n)$.*

Proof. As previously, we run `MDC` on every subgraph of size at most βn and store the results in a table, with time and space at most:

$$S(\beta, n) \in O^*\left(\binom{n}{\beta n}\right)$$

Now, we run the main algorithm until the remaining graph has size βn or less; then a polynomial-time query in the storage table allow us to conclude. Thus, total running time and space is:

$$O^*\left(\max\left\{\binom{n}{\beta n}, 2^{2(1-\beta)n/5}\right\}\right)$$

This value is minimal for a β solution of the equation $2^{2(1-\beta)/5} = 1/(\beta^\beta(1-\beta)^{1-\beta})$, that leads to $T(n) \leqslant O^*(1.2937^n)$. ∎

4 MIN DOMINATING CLIQUE

The function $T(S, D, A, F)$ is now an integer function that returns the cardinality of a minimum dominating clique in G contained in set A and with no vertex from set D, if any, and ∞ otherwise. Once again, solving MIN DOMINATING CLIQUE is equivalent to finding $\min_{v\in V}\{T(\{v\}, \emptyset, N(v), V \setminus N[v])\}$. We denote in this section by $d_A(v)$ the degree of v within the set A of available vertices and by $\bar{d}_A(v)$ the anti-degree of v always within set A. The following remarks hold.

Remark 4. Each vertex $j \in A$ is adjacent to at least one vertex $i \in F$, or else j can be discarded as it cannot be part of the dominating clique. ∎

Remark 5. There exists at least one vertex $j \in A$ such that $\bar{d}_A(j) \geqslant 1$, or else a pure set covering problem where the set F corresponds to the universe U of elements and the set A corresponds to the collection S of the (nonempty) subsets of U and the aim is to determine a minimum cardinality subcollection $S' \subseteq S$ which covers U. This set covering problem is known to be solvable to optimality in $O^*(1.2301^{|A|+|F|})$ time ([2]). But, as $|A| + |F| \leqslant n$ this is not superior to $O^*(1.2301^n)$ time. ∎

Remark 6. Each vertex $i \in F$ is adjacent to at least three vertices $j, k, l \in A$, or else a low exponential complexity immediately holds in the worst case. Indeed, if $i \in F$ is adjacent only to the vertex $j \in A$, then no branch occurs, j is included in S dominating i which is removed from F; alternatively, $i \in F$ is adjacent to vertices $j, k \in A$ and either j or k must be included in S to dominate i. Then: if $d_F(j) \geqslant 2$, either j is included in S and at least 3 vertices are removed, or j is discarded, k is included and i is covered, *i.e.*, 3 vertices are removed. If $d_F(k) = d_F(k) = 1$, $\bar{d}_A(j) \geqslant 1$ holds, or else k could be discarded without branching. But then, in both cases at least three vertices are fixed leading to $T(n) \leqslant 2T(n-3)$ corresponding to $O^*(1.2599)$ time. ∎

We claim that we can solve MIN DOMINATING CLIQUE with running time $O^*(1.3248^n)$ by the following algorithm called `MINDC1` that works as follows:

1. if $A = F = \emptyset$, then $T(S, D, A, F) = |S|$;
2. else, if $\exists i \in F$, such that $d_A(i) = 0$, then $T(S, D, A, F) = \infty$;
3. else, if $\forall j \in A$ $\bar{d}_A(j) = 0$, then solve the problem as a minimum set covering problem according to Remark 5 with complexity not superior to $O^*(1.2301^n)$;

4. else, if $\exists i \in F$, such that $2 \geqslant d_A(i) \geqslant 1$, then branch on the vertices adjacent to i according to Remark 6 with complexity not superior to $O^*(1.2599^n)$;

5. else, if $\forall i \in F\ d_A(i) \geqslant 3$, then select $j \in A$ such that $\bar{d_A}(j)$ is maximum and branch according to the following exhaustive cases

   (a) $\bar{d_A}(j) \geqslant 1$ with vertex $j \in A$ adjacent to only one vertex $i \in F$ with $d_A(i) = 3$.

   (b) $\bar{d_A}(j) \geqslant 1$ with vertex $j \in A$ adjacent to only one vertex $i \in F$ with $d_A(i) \geqslant 4$.

   (c) $\bar{d_A}(j) = 1$ with vertex $j \in A$ non adjacent to vertex $k \in A$ and adjacent to exactly two vertices $h, i \in F$ with $d_A(i) \geqslant d_A(h) = 3$.

   (d) $\bar{d_A}(j) = 1$ with vertex $j \in A$ non adjacent to vertex $k \in A$ and adjacent to exactly two vertices $h, i \in F$ with $d_A(i) \geqslant d_A(h) = 4$.

   (e) $\bar{d_A}(j) = 1$ with vertex $j \in A$ non adjacent to vertex $k \in A$ and adjacent to exactly two vertices $h, i \in F$ with $d_A(i) \geqslant d_A(h) \geqslant 5$.

   (f) $\bar{d_A}(j) = 1$ with vertex $j \in A$ non adjacent to vertex $k \in A$ and adjacent to at least three vertices $g, h, i \in F$.

   (g) $\bar{d_A}(j) \geqslant 2$ with vertex $j \in A$ non adjacent to vertices $k, m \in A$ and adjacent to at least two vertices $h, i \in F$.

Proposition 5. *Algorithm* `MINDC` *computes a minimum-size dominating clique, if any, with running time* $O^*(1.3248^n)$.

Proof. The correctness of the algorithm is straightforward. For the complexity we consider all subcases of case 5.

- If case 5a holds, let j, k, l the three vertices $\in A$ adjacent to i. If j is selected, it must dominate i, hence k, l must be discarded. Alternatively, j is discarded inducing $d_A(i) = 2$ and the applicability of Remark 6. Overall, we have $T(n) \leqslant T(n-4) + 2T(n-4) = 3T(n-4)$ inducing as complexity $O^*(1.3196^n)$.

- If case 5b holds, there are at least 4 vertices $j, k, l, m \in A$ adjacent to i. If j is selected, it must dominate i, hence k, l, m must be discarded. Alternatively, j is discarded. Hence, either 1 vertex or 5 vertices are fixed, namely we have $T(n) \leqslant T(n-1)+T(n-5)$ inducing as complexity $O^*(1.3248^n)$.

- If case 5c holds, then either j is selected, $k \in A$ non adjacent to j is discarded and h, i are dominated, or j is discarded inducing $d_A(h) = 2$ and the applicability of Remark 6. Overall, we have $T(n) \leqslant T(n-4) + 2T(n-4) = 3T(n-4)$ inducing as complexity $O^*(1.3196^n)$.

- If case 5d holds, then j is non adjacent to $k \in A$ and is adjacent to $h, i \in F$ with $d_A(i) \geqslant d_A(h) = 4$. If $\exists\ l \in A$ with $\bar{d_A}(l) = 0$ which is adjacent to both h and i, then j is dominated by l and can be discarded. Also, if l is adjacent to just one among h and i, namely, w.l.o.g., adjacent to h and non adjacent to i, then all other vertices adjacent to i must be discarded when selecting j or else j would again be dominated by l: but this, induces a recurrence $T(n) \leqslant T(n-1) + T(n-6)$ with complexity $O^*(1.286^n)$. Similar considerations hold for vertex k. If k is adjacent to both h and i, then j is dominated by k and can be discarded. Else, if k is adjacent to just one among h and i, namely, w.l.o.g., adjacent to h and non adjacent to i, then all other vertices adjacent to i must be discarded when selecting j or else j would be dominated by k: once again, this, induces a recurrence $T(n) \leqslant T(n-1) + T(n-6)$ with complexity $O^*(1.286^n)$. Alternatively both k and l are non adjacent to h nor to i. Let α, β and $\gamma \in A$ be the available vertices adjacent to h. Notice that $\bar{d_A}(\alpha) = \bar{d_A}(\beta) = \bar{d_A}(\gamma) = 1$ as $\bar{d_A}(j) = 1$ j being the vertex with maximum anti-degree within set A and must have antidegree within set A strictly greater than 0 are they are adjacent to h. Then, there are at least 2 edges between vertices α, β and γ, say wlog. edges $(\alpha\beta)$ and $(\alpha\gamma)$. Consequently, a branch on the vertices dominating h holds, where either j is selected or j is discarded and α is selected or j, α are discarded and β is selected or j, β, γ are discarded and γ is selected inducing a recurrence $T(n) \leqslant T(n-4) + T(n-5) + T(n-6) + T(n-6)$ with complexity $O^*(1.3086^n)$.

- If case 5e holds, then j is non adjacent to $k \in A$ and is adjacent to $h, i \in F$ with $d_A(i) \geqslant d_A(h) \geqslant 5$. Similarly to subcase 5d, if $\exists\ l \in A$ with $\bar{d_A}(l) = 0$ which is adjacent to either h or i or both, or if k is adjacent to either h or i or both, then a recurrence $T(n) \leqslant T(n-1) + T(n-7)$ would hold (now $d_A(i) \geqslant d_A(h) \geqslant 5$) with complexity $O^*(1.2555^n)$. Alternatively, a branch on j holds where, if j is selected, then k is discarded, h, i are dominated and either all other vertices $\in A$ and adjacent to h are discarded, or all other vertices $\in A$ and adjacent to i are discarded. This induces a recurrence $T(n) \leqslant T(n-1) + 2T(n-8)$ with complexity $O^*(1.307^n)$.

- If case 5f holds, vertex j is non adjacent to vertex $k \in A$ and is adjacent to at least 3 vertices $g, h, i \in F$. If j is selected, k is discarded and g, h, i are dominated. Alternatively, j is discarded. Hence, either 1 vertex or 5 vertices are fixed, namely we have $T(n) \leqslant T(n-1) + T(n-5)$ inducing as complexity $O^*(1.3248^n)$.

- If case 5g holds, vertex j is non adjacent at least to vertices $k, m \in A$ and is adjacent at least to vertices $h, i \in F$. If j is selected, k, m are discarded and h, i are dominated. Alternatively, j is discarded. Hence, either 1 vertex or 5 vertices are fixed, namely we have $T(n) \leqslant T(n-1)+T(n-5)$ inducing as complexity $O^*(1.3248^n)$.

The overall worst-case complexity is then $O^*(1.3248^n)$. ∎

Proposition 6. *Using memorization, Algorithm* `MINDC` *computes a minimum-size dominating clique, if any, with running time and space* $O^*(1.2980^n)$.

Proof. As previously, we run `MINDC` on every subgraph of size at most γn and store the results in a table, with time and space at most:

$$S(\gamma, n) \in O^*\left(\binom{n}{\gamma n}\right)$$

Now, we run the main algorithm until the remaining graph has size γn or less; then a polynomial-time query in the storage table allow us to conclude. Thus, total running time and space is:

$$O^*\left(\max\left\{\binom{n}{\gamma n}, 1.3248^{(1-\gamma)n}\right\}\right)$$

This value is minimal for a γ solution of the equation $1.3248^{1-\gamma} = 1/(\gamma^\gamma(1-\gamma)^{1-\gamma})$, that leads to $T(n) = O^*(1.2980^n)$. ∎

5 Dense and sparse graphs

5.1 Graphs of fixed maximal or minimal degree

Notice that if a graph has maximum degree Δ, all the maximal cliques can be computed with running time $O^*(3^{\Delta/3})$ and all the cliques with running time $O^*(2^\Delta)$. In particular, dominating clique problems are polynomial if Δ is finite and subexponential if $\Delta = o(n)$.

In the case of high minimum degree δ, this does not remain true. Indeed, MAX CLIQUE easily reduces to MAX DOMINATING CLIQUE by adding to the graph instance G a new vertex adjacent to any vertex in G, and MAX CLIQUE is well known to be **NP**-hard even in graphs of minimum degree $n-4$ (MAX INDEPENDENT SET being **NP**-hard in graphs of maximum degree 3,[5]). Then, even in graphs with minimum degree $\delta \geqslant n - \overline{\delta}$ for some constant $\overline{\delta}$ MAX DOMINATING CLIQUE is not polynomial (if **P**≠ **NP**). However, there are some interesting results.

By definition, for any $v \in V$, $nf(v) \leqslant n - \delta$. Thanks to Remark 3, if there exists a dominating clique, then there exists a dominating clique of size at most $nf(v)$. Thus, EXISTING, and MIN DOMINATING CLIQUE can be computed with running time $O^*\left(\binom{n}{n-\delta}\right)$. In particular, EXISTING, and MIN DOMINATING CLIQUE are polynomial if $n - \delta$ is finite and subexponential if $n - \delta = o(n)$.

We now exhibit Algorithm `MDC2` that solves MAX DOMINATING CLIQUE in any graph, but it is interesting only for dense graphs:

1. for every $v \in V$, form the collection $\mathcal{S}(v)$ of every subset of $N[v]$ of size at most $nf(v)$ that is a dominating clique;

2. for every $S \in \mathcal{S}(v)$, compute a maximum clique in $G[\bigcap_{s\in S} N[s]]$;

3. return the maximum-size clique among those computed in Step 2.

Proposition 7. *Algorithm* `MDC2` *solves* MAX DOMINATING CLIQUE.

Proof. Let K^* be an optimal solution for MAX DOMINATING CLIQUE. According to Remark 3, for any $v \in K^*$, there exists $K \subset K^*$ such that K is a dominating clique and $|K| \leqslant nf(v) \leqslant n - \delta$. Thus, K belongs to $\mathcal{S}(u)$, for some u. Let K' the maximum clique in $G[\bigcap_{s\in K} N[s]]$. According to Remark 1, K' is a dominating clique, while $K^* \subseteq \bigcap_{s\in K} N[s]$. Then, by maximality, $|K'| = |K^*|$. So, Algorithm `MDC2` computes a maximum-size dominating clique. ∎

For $\delta \geqslant n/2$, the size of the collection computed at step 1 is at most $n^2\binom{n}{n-\delta}$; thus, Algorithm `MDC2` has running time $O^*\left(c^n\binom{n}{n-\delta}\right)$ if the algorithm used to solve MAX CLIQUE has complexity $O^*(c^n)$. In particular, MAX DOMINATING CLIQUE can be computed with the same exponential bound on running time as MAX CLIQUE in graphs such that $n - \delta = o(n)$; this bound cannot be improved (thanks to the reduction from MAX CLIQUE mentioned above).

Note that the best worst-case complexity bound for MAX CLIQUE is, to our knowledge, the $O^*(1.1889^n)$ bound claimed by [9] in his unpublished technical report, or the $O^*(1.2210^n)$ algorithm by [4], that is the best published result.

5.2 Graphs of small average degree

More generally, the density of a graph can be defined as $D(G) = 2m/(n(n-1)) = d/(n-1)$ where d is the average degree of the graph. It can be easily seen that, if $D(G) = o(1)$, the size of a maximum clique is $o(n)$ and then we can enumerate all cliques in subexponential time.

We now focus ourselves on the case where $D(G) \notin o(1)$ but is bounded above by some constant $\lambda \in [0, 1]$. In this case, average degree is at most $\lambda(n-1)$, hence:

$$m \leqslant \frac{\lambda n(n-1)}{2} \leqslant \frac{\lambda n^2}{2} \tag{1}$$

By enumeration of subsets of size at most $\sqrt{D(G)}n$, we trivially find if there is a dominating clique (and return the minimal and the maximal one) with running time $O^*\left(\binom{n}{\sqrt{\lambda}n}\right)$. This is only interesting for rather small values of λ; for example, if $\sqrt{\lambda} = 1/20$, running time is $O^*(1.22^n)$.

As far as EXISTING DOMINATING CLIQUE and MAX DOMINATING CLIQUE are concerned (not MIN DOMINATING CLIQUE), we can greatly improve this result. Notice that, for any dominating clique K we have the following inequality:

$$m \geqslant \frac{|K|(|K|-1)}{2} + \sum_{v\in K} d_{V\setminus K}(v)$$

Assume first that there exists a vertex $v \in K$ such that $d_{V\setminus K}(v) < \mu n + 1/2$, for a given μ whose value will be fixed later. Then $n(1 - f(v)) < \mu n + 1/2 + |K|$.

In [1] it is established that in a graph of size n, the cliques of size k or less (where $n/k \geqslant 3$) can be enumerated within time $O^*((n/k)^k)$. Thus, we can enumerate all maximal cliques containing v with running time:

$$T(n) = O^*\left(\left(\frac{\mu n + \frac{1}{2} + |K|}{|K|}\right)^{|K|}\right) = O^*\left(2^{|K|\log_2(1+\mu n/|K|)}\right)$$

Since this function is increasing with K and, according to (1), $|K| \leqslant \sqrt{2m} + 1 \leqslant \sqrt{\lambda}n + 1$, this leads to:

$$T(n) = O^*\left(2^{\sqrt{\lambda}n\log_2(1+\mu/\sqrt{\lambda})}\right) \tag{2}$$

On the other hand, if for any $v \in K$, $d_{V\setminus K}(v) \geqslant \mu n + 1/2$, using also (1), we get:

$$m \geqslant \frac{|K|(|K|-1)}{2} + |K|\left(\mu n + \frac{1}{2}\right) = \frac{|K|^2}{2} + |K|\mu n \Longrightarrow |K| \leqslant \left(\sqrt{\lambda + \mu^2} - \mu\right)n$$

In this case, running time for enumerating all small subsets containing v is bounded above by:

$$T(n) = O^*\left(\left(\frac{n}{\left(\sqrt{\lambda+\mu^2}-\mu\right)n}\right)^{\left(\sqrt{\lambda+\mu^2}-\mu\right)n}\right) = O^*\left(2^{\left(\sqrt{\lambda+\mu^2}-\mu\right)n\left|\log_2\left(\sqrt{\lambda+\mu^2}-\mu\right)\right|}\right) \quad (3)$$

The running time given by (2) is increasing with μ while the one given by (3) is decreasing. So, the best value for μ is given when both are equal[2], i.e. when:

$$\left(\sqrt{\lambda+\mu^2}-\mu\right) \times \left|\log_2\left(\sqrt{\lambda+\mu^2}-\mu\right)\right| = \sqrt{\lambda}\log_2\left(1+\frac{\mu}{\sqrt{\lambda}}\right)$$

In Table 3, bounds on running time of the above method vs. running time of exhaustive search (both depending on parameter λ) are given.

$\sqrt{\lambda}$	1/4	1/6	1/8	1/10	1/20	1/50
Optimal μ	0.326	0.248	0.200	0.169	0.097	0.045
Running time of our algorithm	1.233^n	1.164^n	1.127^n	1.104^n	1.056^n	1.046^n
Running time of exhaustive search	1.755^n	1.570^n	1.458^n	1.385^n	1.220^n	1.104^n

Table 3: Running time of our algorithm vs. running time of exhaustive search for some values of λ.

5.3 Graphs of high average degree

In this section, we deal with graphs of density $D(G) \geqslant 3/4$. As previously, it is easy to see that in such graphs MAX DOMINATING CLIQUE is harder than MAX CLIQUE.

Let us define Algorithms `mDC1` and `MDC3` for MIN DOMINATING CLIQUE and MAX DOMINATING CLIQUE, respectively. The former, `mDC1`, works as follows:

> fix $\epsilon = \sqrt{1-D(G)}$, compute any subset of size at most ϵn and return a smallest one that is a dominating clique if any.

On the other hand, Algorithm `MDC3` for MAX DOMINATING CLIQUE works as follows:

1. fix $\epsilon = \sqrt{1-D(G)}$ and compute any subset of size at most ϵn; let K_0 be a largest one that is a dominating clique, if any;
2. for every $v \in V$ such that $nf(v) \leqslant \epsilon n$, form the collection $\mathcal{S}(v)$ of every subset of size at most $nf(v)$ that is a dominating clique;
3. for every $S \in \mathcal{S}(v)$, compute a maximum clique in $G[\bigcap_{s\in S} N[s]]$;
4. let K_1 be a clique of maximum size among those computed in Step 3; output $\max\{K_0, K_1\}$.

Obviously, if there exists no dominating clique in the graph, both algorithms return nothing. We now prove that, if some dominating clique exists, Algorithm `mDC1` computes a minimum dominating clique, while Algorithm `MDC3` computes a maximum dominating clique.

Fix K^* a maximum dominating clique. If $|K^*| \leqslant \epsilon n$, it is clear that both algorithms work correctly. Assume $|K^*| > \epsilon n$ and suppose that, for any $v \in K^*$, $f(v) \geqslant \epsilon$. Then:

$$\frac{n(n-1)}{2} - m \geqslant \frac{1}{2}\sum_{v\in K^*} nf(v) \geqslant \frac{\epsilon^2}{2}n^2 > \frac{1-D(G)}{2}n(n-1)$$

Plugging $m = D(G)n(n-1)/2$ in the previous inequality leads to a contradiction.

Consequently, there exists a vertex $v \in K^*$ with $nf(v) < \epsilon n$. Since there exists a dominating clique of size at most $nf(v)$, according to Remark 3, Algorithm `mDC1` returns a minimum dominating

[2] under the condition that $n/k \geqslant 3$, *i.e.* $\sqrt{\lambda+\mu^2}-\mu \leqslant 1/3$, which is verified is the sequel.

clique K_0. Furthermore, K^* has the same size as a maximum clique in $G[\bigcap_{s\in K_0} N[s]]$, which means that Algorithm `MDC3` returns a maximum dominating clique.

To conclude, notice that $D(G) = 1 - o(1)$ implies $\epsilon = o(1)$, and then

$$\binom{n}{\epsilon n} = O^*\left(\left(\frac{1}{\epsilon^\epsilon(1-\epsilon)^{1-\epsilon}}\right)^n\right) = 2^{o(n)}$$

Consequently, it is clear that in this case Algorithm `mDC1` has subexponential running time, while the complexity of `MDC3` is the running time of the MAX CLIQUE-algorithm called in Step 3, with a subexponential multiplicative factor. Again, no improvement in the exponential basis of the running time seems possible thanks to the reduction from MAX CLIQUE to MAX DOMINATING CLIQUE mentioned above.

6 Moderately exponential approximation

Since any approximation algorithm for MIN DOMINATING CLIQUE or MAX DOMINATING CLIQUE solves EXISTING DOMINATING CLIQUE that is **NP**-complete, it is impossible to devise any polynomial approximation algorithm for MIN- and MAX DOMINATING CLIQUE. Thus, it seems interesting to see if it is possible to compute solutions for the two optimization problems that guarantee a good approximation ratio with moderately exponential running time; interesting running times of approximation algorithms lie between the best known complexity for solving EXISTING DOMINATING CLIQUE ($O^*(1.2740)$ with our algorithm) and the best known complexity for solving the corresponding optimization problem. We propose in what follows two algorithms `mMOD` and `MMOD` that do this for MIN DOMINATING CLIQUE and MAX DOMINATING CLIQUE, respectively. Obviously, in what follows we assume that we handle graphs admitting dominating cliques (otherwise our algorithm detects the .

Algorithm `mMOD`(ρ) (i.e., parameterized by the ratio ρ that one wishes to attain) for MIN DOMINATING CLIQUE works as follows:

- run algorithm `EDC'` and let K_0 be the dominating clique computed;
- compute all the subsets of V whose size is at most n/ρ; let K_1 be a dominating clique of minimum size among them; if none is found, then set $K_1 = V$;
- return $\text{argmin}\{|K_0|, |K_1|\}$.

Proposition 8. *If $G(V, E)$ has a dominating clique then, for any ρ, $2 \leqslant \rho \leqslant 15.24$, it is possible to compute a ρ-approximation to* MIN DOMINATING CLIQUE *with polynomial space and running time $O^*\left(\binom{n}{n/\rho}\right)$. This is faster than the exact (polynomial space) algorithm for $\rho \geqslant 11.71$.*

Proof. Observe first that Algorithm `mMOD` has running time (for $2 \leqslant \rho \leqslant 15.24$):

$$O^*\left(1.2740^n + \binom{n}{n/\rho}\right) = O^*\left(\binom{n}{n/\rho}\right)$$

Now, let K_m^* be a minimum dominating clique of G. If $|K_m^*| \leqslant n/\rho$, then $|K_1| = |K_m^*|$ and `mMOD` is optimal. Otherwise:

$$\frac{|K_0|}{|K_m^*|} \leqslant \frac{n}{\frac{n}{\rho}} = \rho$$

A simple algebra fixes ρ as claimed. ∎

The same proof applied to an algorithm with memorization leads to the following result.

Proposition 9. *If $G(V, E)$ has a dominating clique, then, for any ρ, $2 \leqslant \rho \leqslant 16.60$, it is possible to compute a ρ-approximation to* MIN DOMINATING CLIQUE *with running time and space $O^*\left(\binom{n}{n/\rho}\right)$. This is faster than the exact (exponential space) algorithm for $\rho \geqslant 12.40$.*

Unfortunately, this strategy does not work as far as MAX DOMINATING CLIQUE is concerned. Indeed, consider the following instance: $K = (k_i)_{i\leqslant n/3}$ is a clique and $S(s_i)_{i\leqslant n/3}$, $T(t_i)_{i\leqslant n/3}$ are two independent sets. Add an edge for each pair (k_i, s_i), (k_i, t_i) and (s_i, t_i). All the dominating cliques but K have size

exactly 3. Thus, searching for all small and/or large cliques leads to an unbounded approximation ratio. However, it is possible to get some approximation result if we observe that *algorithm* `EDC'` *is able not only to find a dominating clique, but also, given a subset S, to find a dominating clique containing S.*

Based upon this remark we devise Algorithm `MMOD`(ρ) for MAX DOMINATING CLIQUE that works as follows:

- compute all the subsets K of V whose size is at most n/ρ; if K is a clique, then find $T(K, N[K] \setminus \bigcap_{v\in K} N[v], \bigcap_{v\in K} N(v), V \setminus N[K])$, according to algorithm `EDC'` (where $N[K] = \cup_{v\in K} N[v]$);
- return the largest dominating clique found, denoted by K.

Proposition 10. *If $G(V, E)$ has a dominating clique, then, for any $\rho \geqslant 2$ it is possible to compute a ρ-approximation to* MAX DOMINATING CLIQUE *with polynomial space and running time:*

$$O^*\left(\binom{n}{n/\rho}1.2740^{n(1-1/\rho)}\right)$$

This is faster than the exact (polynomial space) algorithm for $\rho \geqslant 168$.

Proof. Observe first that Algorithm `MMOD` has running time:

$$O^*\left(1.2740^{|V\setminus N[K]|}\binom{n}{|K|}\right) = O^*\left(1.2740^{n(1-1/\rho)}\binom{n}{n/\rho}\right)$$

Now, let K^*_M be a maximum dominating clique of G. If $|K^*_M| \leqslant n/\rho$, then `MMOD` computes an optimal solution. Otherwise, it computes at least a dominating clique containing a subclique of size n/ρ thus guaranteeing $|K^*_M|/|K| \leqslant n/(n/\rho) = \rho$, as claimed. ∎

Once again, the same proof applies to the algorithm with memorization and leads to the following.

Proposition 11. *If $G(V, E)$ has a dominating clique, then, for any $\rho \geqslant 2$ it is possible to compute a ρ-approximation to* MAX DOMINATING CLIQUE *with running time and space:*

$$O^*\left(\binom{n}{n/\rho}1.2556^{n(1-1/\rho)}\right)$$

This is faster than the exact (exponential space) algorithm for $\rho \geqslant 204$.

References

[1] J. M. Byskov. Enumerating maximal independent sets with applications to graph colouring. *Oper. Res. Lett.*, 32(6):547–556, 2004.

[2] F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: domination – a case study. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Proc. ICALP'05*, volume 3580 of *Lecture Notes in Computer Science*, pages 191–203. Springer-Verlag, 2005.

[3] F. V. Fomin, F. Grandoni, and D. Kratsch. Some new techniques in design and analysis of exact (exponential) algorithms. Bulletin of the European Association for Theoretical Computer Science 87, pp. 47–77, 2005.

[4] F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: a simple $O(2^{0.288n})$ independent set algorithm. In *Proc. Symposium on Discrete Algorithms, SODA'06*, pages 18–25, 2006.

[5] M. R. Garey and D. S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness.* W. H. Freeman, San Francisco, 1979.

[6] D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. *Inform. Process. Lett.*, 27:119–123, 1988.

[7] D. Kratsch and M. Liedloff. An exact algorithm for the minimum dominating clique problem. *Theoret. Comput. Sci.*, 385(1-3):226–240, 2007.

[8] J. W. Moon and L. Moser. On cliques in graphs. *Israel J. of Mathematics*, 3:23–28, 1965.

[9] J. M. Robson. Finding a maximum independent set in time $O(2^{n/4})$. Technical Report 1251-01, LaBRI, Université de Bordeaux I, 2001.