Average-case complexity of a branch-and-bound algorithm for MIN DOMINATING SET

Tom Denat, Ararat Harutyunyan, Vangelis Th. Paschos

Université Paris-Dauphine, PSL* Research University, CNRS UMR 7243, LAMSADE, Paris 75016, France
{tom.denat,ararat.harutyunyan,vangelis.paschos}@lamsade.dauphine.fr

December 28, 2021

Abstract

The average-case complexity of a branch-and-bound algorithm for MIN DOMINATING SET problem in random graphs in the $\mathcal{G}(n,p)$ model is studied. We identify phase transitions between subexponential and exponential average-case complexities, depending on the growth of the probability p with respect to the number n of nodes.

1 Introduction

Given a graph $G = (V, E)$ of order n, a dominating set $S \subseteq V$ is a subset of V such that any $v_i \in V$ is either included in S or connected to a vertex of S by an edge of E. The MIN DOMINATING SET problem consists of finding a minimum-size dominating set in G. MIN DOMINATING SET is a very well-known **NP**-hard problem completely equivalent (from both complexity and polynomial approximation points of view) to MIN SET COVER problem.

Dealing with the exact solution of MIN DOMINATING SET, besides the obvious $O(2^n)$ algorithm which considers the power set of V and chooses the smallest one that also forms a dominating set, several moderately exponential algorithms have been proposed mainly during the last fifteen years. To the best of our knowledge, the fastest one is the $O(1.4969^n)$ algorithm due to [7].

The main purpose of this paper is the study of the average case complexity of branch-and-bound algorithms for the MIN DOMINATING SET problem in random graphs in the $\mathcal{G}(n,p)$ model. This model represents graphs on n vertices where each of the possible $\binom{n}{2}$ edges appears independently with probability p. For an extensive treatment of random graphs, we refer the reader to the monograph [3].

Even though branch-and-bound is one of the best known and most widely used techniques for exactly solving **NP**-hard problems, there has been little systematic study of its worst- or average-case complexity. Also, even though mathematical tools for average case analysis of algorithms have existed for decades [6] and have much advanced in sophistication [5], we do not know of many results on the average case complexity of exact algorithms for **NP**-hard problems. The only works known to us are the ones of [2] where the authors study the complexity of a "pruning the search-tree algorithm" for MAX INDEPENDENT SET (the worst-case complexity of this algorithm is $O(1.1996^n)$, [8]) under the $\mathcal{G}(n,p)$ model, the one of [1], where the same algorithm is studied under the $\mathcal{G}(n,m)$ model and, finally, the one in [4] where the average-case complexity of a branch-and-bound algorithm for MAX INDEPENDENT SET is studied under the $\mathcal{G}(n,p)$ model. Here our goal is to study the complexity of a (rather intuitive and simple, with respect to the bounds in the nodes of the branch-and-bound-tree), branch-and-bound-algorithm for MIN DOMINATING SET. Even if, the obtained complexity-bounds are worse than the ones obtained for the problem by other more evolved methods (inclusion-exclusion, dynamic programming, etc.), we think that it is interesting to obtained some formal complexity results for the branch-and-bound method (the most used one for the exact solution of **NP**-hard problems).

In what follows, in Section 2 we specify the branch-and-bound algorithm the complexity of which is then analysed in Section 3. Finally, in Section 4 we study the complexity of simple exhaustive search algorithm

which, starting from the whole vertex-set V of the input graph, produces a minimum dominating set by considering all the subsets of V and finally returns the smallest one that is a dominating set.

2 The branch-and-bound algorithm

Let $G = (V, E)$ be a graph; set $n = |V|$ and fix an order $v_1, v_2, \ldots, v_n$ on V. The type of branch-and-bound algorithms for MIN DOMINATING SET studied here works by building a branch-and-bound binary tree, nodes of which are associated with a vector $\vec{x} \in \{0,1\}^n$ and a depth δ in the binary tree. Obviously, $x_i = 1$ means that vertex v_i has been taken in the solution under construction and $x_i = 0$ means that v_i has not been taken. For a tree-node at level δ only vertices $v_1, v_2, \ldots, v_\delta$ have been explored, i.e., only $x_1, x_2, \ldots, x_\delta$ have been assigned definite values. At this point the values for $x_{\delta+1}, \ldots, x_n$ are, for the moment, equal to 1. We remark that the superset of a dominating set is also a dominating set.

The root of the branch-and-bound tree T corresponds to the trivial dominating set including all the vertices ($\vec{x} = (1, 1, ..., 1)$) at the depth 0 and is initially visited. The left child $(\vec{x}, \delta)_l = (\vec{x}, \delta + 1)$ of a node $n_i = (\vec{x}, \delta)$ at level δ of T, has exactly the same vector $\vec{x}$ as n_i but with the value of $x_{\delta+1} = 1$ now determined, i.e., the partial solution represented by n_i is extended by putting $v_{\delta+1}$ in the solution under construction. The right child $(\vec{x}, \delta)_r$ of $(\vec{x}, \delta)$ corresponds to changing the partial solution represented by n_i by putting $x_{\delta+1} = 0$.

At each new step of the algorithm a new node will be explored. In order to be explorable a node $(\vec{x}, \delta)$ must correspond to a dominating set in G and be either the left or the right child of an already visited node.

Therefore, at each step of the algorithm, the nodes can be divided in four categories:

1. the already visited nodes which correspond to dominating sets (feasible solutions);
2. the nodes that correspond to vertex-sets that are not dominating sets in G (the infeasible solutions);
3. the explorable solutions which correspond to dominating sets in G and are either the left or the right child of an already visited node;
4. the currently "hidden" feasible solutions which correspond to dominating sets in G but are not the left nor the right child of an already visited node.

For a node at level δ with vector $\vec{x}$, we define its score $u(\vec{x}, \delta)$ as the number of vertices that currently must be included in the solution. Formally:

$$u(\vec{x}, \delta) = \left|\vec{x}\right| - n + \delta$$

where $|\vec{x}| = \sum_{i=1,2,\ldots n} x_i$. Score $u(\vec{x}, \delta)$ can be seen as an optimistic prediction of the value of the optimal solution since it implies that no other vertex will be added to the feasible solution corresponding to the node at hand.

The choice of the next node to be explored among the explorable nodes (solutions) is made by minimizing a score $u(\vec{x}, \delta)$, named potential in what follows, associated with each node of the branch-and-bound tree.

The explorable solutions corresponding to tree-nodes of depth n constitute feasible dominating sets for the whole G. It is easy to see that the rightmost among them corresponds to a minimum dominating set of G.

As an example, the graph G in figure 1 contains three vertices A, B and C. Therefore, the branch-and-bound tree contains 3 levels. The branch and bound algorithm starts by exploring the root $\{1, 1, 1\}, 0$. Now, the explorable nodes are $\{1, 1, 1\}, 1$ and $\{0, 1, 1\}, 1$ with respective potentials 1 and 0. Therefore, the node $\{0, 1, 1\}, 1$ is explored next. Then, the two explorable nodes are $\{1, 1, 1\}, 1$ and $\{0, 1, 1\}, 2$, both with potential 1. The next node to be explored is chosen randomly. Assume that the next explored node is $\{1, 1, 1\}, 1$ and so on. Finally, the complete solution $\{0, 1, 0\}, 3$ is found. As its depth is 3, this solution $\{B\}$ is the minimum dominating set.

In what follows, we denote by $\mathbb{T}(n, p)$ the average complexity of branch-and-bound in a binomial random graph $G = (V, E)$ with parameters (n, p), that is, the total number of leaves expansions, and by $\mathbb{E}[\#S(G)]$ the expectation of the number of dominating sets in a binomial random graph G.

3 Analysis of the branch-and-bound algorithm

Notice first that any child xb of a node x of the branch and bound tree, where $b \in \{0, 1\}$, has $u(xb) \geqslant u(x)$.

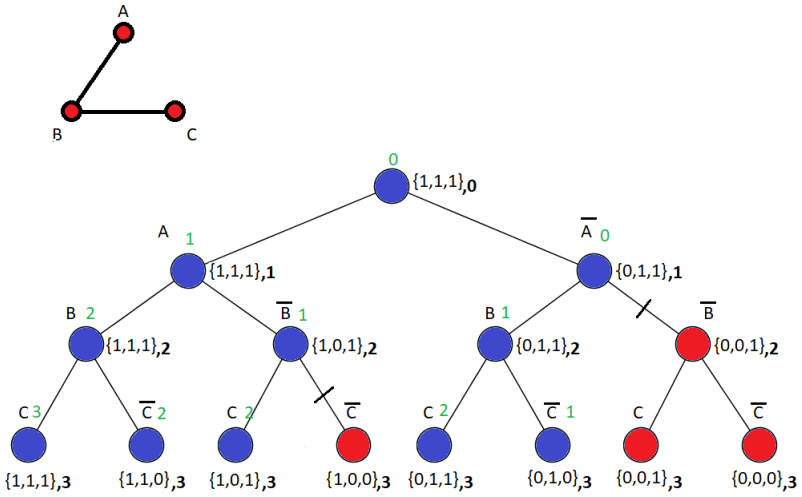

Figure 1: Illustration of the branch-and-bound algorithm for the minimum dominating set problem. The instance G appears on the upper left corner of the figure. In the branch-and-bound tree, the nodes representing feasible solutions are coloured in deep blue. The green number above each node represents the score function $u(\vec{x})$.

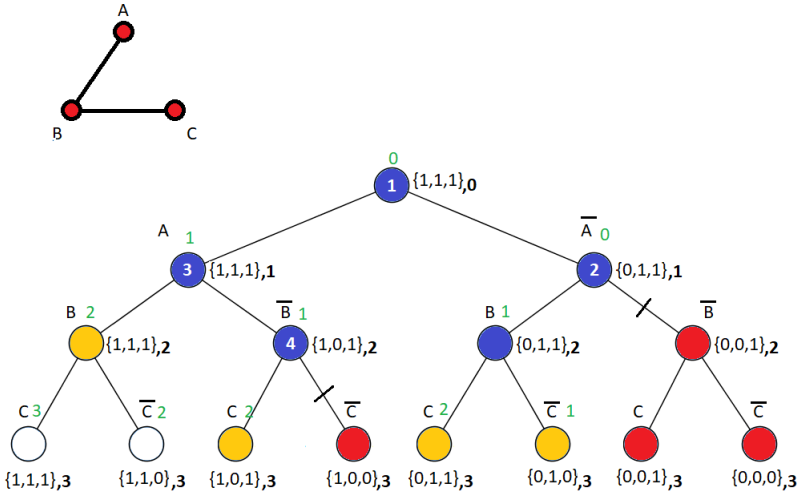

Figure 2: Illustration of the fifth step of the algorithm. Here, the explored nodes are in deep blue, the explorable solutions are in yellow, the hidden solutions in white and the infeasible solutions are in red.

So, let x be the first leaf at the nth level — that is with $|x| = n$ — expanded by the algorithm; obviously $u(x) = |S_x|$. Any leaf that has not been expanded yet must have a bound that is at least $|S_x|$ (otherwise they would have been handled before). Since all other unexpanded leaves have bounds no smaller than $|S_x|$, S_x must be a minimum dominating set, and the algorithm terminates.

It follows that, during the running of the algorithm, a leaf x is expanded only if $u(x) \leqslant |S^*|$, where S^* is the minimum dominating set. Thus, the expanded leaves x with $|x| = i$ are dominating subsets $S \subseteq \{1, \ldots, i\}$

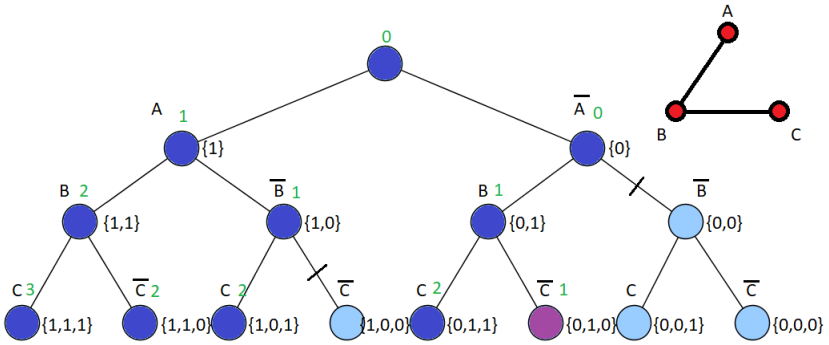

Figure 3: Illustration of the whole execution of branch-and-bound algorithm for the minimum dominating set problem. The optimal solution is coloured in purple.

satisfying $n - |S| \leqslant |S^*|$. We can therefore write:

$$\begin{aligned}
\mathbb{T} &\leqslant \sum_{i=0}^{n} \sum_{S \subseteq \{1,2,\ldots,i\}} \mathbb{P}\left((S \text{ is a dominating set}) \cap (n - |S| \leqslant |S^*|)\right) \Rightarrow \\
\Rightarrow \mathbb{T} &\leqslant \sum_{i=0}^{n} \sum_{S \subseteq \{1,2,\ldots,i\}} \mathbb{P}\left(S \text{ is a dominating set}\right) \quad \wedge \quad \mathbb{T} \leqslant \sum_{i=0}^{n} \sum_{S \subseteq \{1,2,\ldots,i\}} \mathbb{P}\left(n - |S| \leqslant |S^*|\right) \\
&\leqslant n \cdot \max_{1 \leqslant k \leqslant n} \binom{n}{k} \Pr[\gamma > n - k] \qquad (1) \\
& \qquad (2)
\end{aligned}$$

Thus, we need to upper-bound the quantity $M := \binom{n}{k} \Pr[\gamma > n - k]$ for all $1 \leqslant k \leqslant n$.

3.1 Upper bounds

The following theorem provides upper bounds for the complexity of the branch-and-bound algorithm presented in Section 2 for random graphs in the $\mathcal{G}(n,p)$ model.

Theorem 1. *The following two facts hold:*

(a) If $pn \to \infty$, then the branch-and-bound algorithm takes subexponential time.

(b) If $pn = c$, where $c \geqslant 20$ is a constant, then the branch-and-bound algorithm takes time $(2-\epsilon)^n$, where $\epsilon \geqslant 0.01$ is some constant.

Proof. We first focus on the case where p is fixed. We show that in this case, M is subexponential. We use the fact that $\gamma(G) \leqslant \alpha(G)$ for every graph G, where $\alpha(G)$ denotes the stability number of graph G (indeed, a maximal independent set is a dominating set) and the union bound.

We recall that $1 - x \leqslant e^{-x}$ for all values of x (this follows from the fact that $1 - x$ is the tangent line of e^{-x} at $x = 0$). We remark that $\alpha(G) > x$ for some x, implies that one of the $\binom{n}{x}$ subsets of vertices of size x induces an independent set in G. Thus:

$$M \leqslant \binom{n}{k} \Pr[\alpha > n - k] \leqslant \binom{n}{k}\binom{n}{n-k}(1-p)^{\binom{n-k}{2}} \leqslant \left(\binom{n}{n-k}\right)^2 e^{-p(n-k)(n-k-1)/2} = \binom{n}{x}^2 e^{-px(x-1)/2}$$

where $x = n - k$. We consider x as a function of n, setting $x := f(n)$. If $f(n) = o(n)$, then $\binom{n}{x}^2 < (en/x)^{2x}$ is clearly subexponential. Thus, we may assume $f(n) = \Theta(n)$. Quantity M clearly satisfies:

$$M \leqslant \left(\left(\frac{en}{x}\right)^2 e^{-p(x-1)/2}\right)^x$$

Since $f(n) = \Theta(n)$ and p is fixed, clearly, $(en/x)^2 e^{-p(x-1)/2} < 1$ for n sufficiently large. In fact, with a slightly more careful analysis, we can obtain that M is subexponential in the regime $pn \to \infty$. Indeed, we may suppose

as before that $x = \Theta(n)$ and now it follows that $(en/x)^2$ is bounded by a constant and since $p = \omega(1)$ we have that $e^{-p(x-1)/2} \to 0$, as $n \to \infty$. Therefore, $(en/x)^2 e^{-p(x-1)/2} < 1$ for n sufficiently large and the claim follows.

It remains to consider the case (b), where $p = c/n$, where $c \geqslant 20$ is some absolute constant. As before, we consider the quantity $\binom{n}{x}^2 e^{-px(x-1)/2}$. We may assume as before that $x = \Omega(n)$, i.e. $\lim_{x\to\infty} x/n = \epsilon$, for some $\epsilon > 0$. Indeed, otherwise the time is subexponential. Thus, we need to show that $\binom{n}{\epsilon n}^2 (1-p)^{\epsilon^2 n^2/2}$ is bounded for any $\epsilon > 0$. We use the fact that $\binom{n}{\epsilon n} = 2^{(1+o(1))H(\epsilon)n}$, where $H(\epsilon)$ is the binary entropy function. Thus,

$$\binom{n}{\epsilon n}^2 (1-p)^{\epsilon^2 n^2/2} \leqslant \left(\left[\left(\epsilon^{-\epsilon} (1-\epsilon)^{-(1-\epsilon)} \right)^{(1+o(1))} \right]^2 e^{-c\epsilon^2/2} \right)^n$$

We remark that $(\epsilon^{-\epsilon}(1-\epsilon)^{-(1-\epsilon)})$ is increasing on the interval $(0, 1/2)$ and decreasing on $(1/2, 1)$, with its maximum at $\epsilon = 1/2$, since $H(\epsilon)$ is the logarithm of this function. Thus, there is some constant $\epsilon_0 < 1/2$ such that, for all $\epsilon \leqslant \epsilon_0$, $(\epsilon^{-\epsilon}(1-\epsilon)^{-(1-\epsilon)}) < \sqrt{2}$. It is easy to check that we can take $\epsilon_0 = 1/10$. Thus, we may assume that $\epsilon > 1/10$. Since $c \geqslant 20$, it suffices to show that:

$$\left[\left(\epsilon^{-\epsilon} (1-\epsilon)^{-(1-\epsilon)} \right)^{(1+o(1))} \right]^2 e^{-10\epsilon^2} < 2$$

for all $\epsilon \in [1/10, 1/2]$. Since the first of the two products is increasing and the second is decreasing with ϵ, we will have to dominate each separately. To this end, we refine the intervals of ϵ. First consider the interval $\epsilon \in [1/10, 1/8]$. To bound our product, it is sufficient to substitute $1/8$ in the first term and $1/10$ in the second. By doing this, we obtain that the product is less than, say, 1.99. It is easily verified that we can repeat this argument on the following intervals, thus finishing the theorem. In each case, we obtain a bound of less than 1.99. The precise bounds are given below, where $f(\epsilon) := [(\epsilon^{-\epsilon}(1-\epsilon)^{-(1-\epsilon)})^{(1+o(1))}]^2 e^{-10\epsilon^2}$:

- $\epsilon \in [1/8, 1/7]$; $f(\epsilon) < 1.943$;
- $\epsilon \in [1/7, 0.15]$; $f(\epsilon) < 1.9$;
- $\epsilon \in [0.15, 0.17]$; $f(\epsilon) < 1.988$;
- $\epsilon \in [0.17, 0.19]$; $f(\epsilon) < 1.981$;
- $\epsilon \in [0.19, 0.21]$; $f(\epsilon) < 1.95$;
- $\epsilon \in [0.21, 0.25]$; $f(\epsilon) < 1.982$;
- $\epsilon \in [0.25, 0.35]$; $f(\epsilon) < 1.955$;
- $\epsilon \in [0.35, 0.5]$; $f(\epsilon) < 1.2$.

The proof of the theorem is now completed. □

3.2 Lower bounds

The following result shows that the upper bound on complexity of the algorithm given by Item (b) of Theorem 1 cannot be drastically improved in order that a subexponential bound is taken.

Theorem 2. *Let $p = c/n$, where c is a positive fixed constant. Then, the branch-and-bound algorithm takes at least $(1/\epsilon)^{\epsilon n}$ time for $G(n,p)$, where $\epsilon := \max\{0.99, 1 - (1/10c)\}$*

Proof. Note that for any k, $\binom{n}{k} \Pr[\gamma > n - k]$ is a lower bound on the complexity of the algorithm. Thus, it is sufficient to show that:

$$\binom{n}{\epsilon n} \Pr[\gamma > n - \epsilon n] = \Omega\left(\left(\frac{1}{\epsilon} \right)^{\epsilon n} \right)$$

for n sufficiently large.

We will prove that $\Pr[\gamma \leqslant n - \epsilon n]$ is arbitrarily small. This is clearly sufficient. Let A be the event that a fixed set S of size $n - \epsilon n$ is a dominating set. Then, $\Pr[\gamma \leqslant n - \epsilon n] \leqslant \binom{n}{n-\epsilon n} \Pr[A] < 2^n \Pr[A]$. We use the fact that $1 - x \geqslant e^{-2x}$ for all $x \in (0, 1/2)$; this can be seen, for example, by noticing that e^{-2x}

is a convex function and that $1 - x = e^{-2x}$ has two solutions at $x = 0$ and at some $x \in (0.5, 1)$. Now, $\Pr[A] = (1 - (1-p)^{n-\epsilon n})^{\epsilon n} \leqslant (1 - e^{-2c(1-\epsilon)})^{\epsilon n}$. Thus:

$$\Pr[\gamma \leqslant n - \epsilon n] \leqslant \left(2\left(1 - e^{-2c(1-\epsilon)}\right)^{\epsilon}\right)^{n}$$

Note that by definition of ϵ, $2c(1-\epsilon) < 1/5$. Thus, $(1 - e^{-2c(1-\epsilon)})^{\epsilon} < (1 - e^{-1/5})^{0.99} < 1/2$.
It follows that:

$$\binom{n}{\epsilon n}\Pr[\gamma > n - \epsilon n] = \Omega\left(\left(\frac{1}{\epsilon}\right)^{\epsilon n}\right)$$

as required. □

4 Analysis of the simple exhaustive search algorithm

We conclude the paper by studying in this section a simple exhaustive search algorithm which, starting from the whole vertex-set V of the input graph produces a minimum dominating set by considering all the subsets of V and finally returns the smallest one that is a dominating set. In the remaining part of this section, we prove the following proposition.

Proposition 1. *Consider a random (n,p)-binomial graph G. If $p = c/n$, for some constant c, then:*

$$\mathbb{T}(n,p) \leqslant \max\left\{1.99^n, \left(2\left(1 - e^{-2c}\right)^{1/3}\right)^n\right\}$$

Proof. We can clearly suppose $n/3 < k < 2n/3$; otherwise $\binom{n}{k} \leqslant 1.99^n$ and we are done. Using the fact that $1 - x \geqslant e^{-2x}$ for $x \in (0, 0.5)$, one can deduce:

$$\begin{aligned}\mathbb{T}(n,p) &\leqslant 2^n n^2 \max_{n/3 \leqslant k \leqslant 2n/3}\left\{\binom{n}{k}\left(1 - (1-p)^{n-k}\right)^k\right\} \\ &\leqslant 2^n n^2\left(1 - \left(1 - \frac{c}{n}\right)^{2n/3}\right)^{n/3} \leqslant 2^n n^2\left(1 - e^{-4c/3}\right)^{n/3} \leqslant n^2\left(2\left(1 - e^{-4c/3}\right)^{1/3}\right)^n\end{aligned}$$

and the result follows immediately. □

5 Conclusion

We have studied in this paper the average-case complexity of a branch-and-bound algorithm for MIN DOMINATING SET in random graphs under the $\mathcal{G}(n,p)$ model. It has been proved that this complexity is: (a) *subexpontial* when $p = f(n)/n$, for any function $f \to \infty$ with n; (b) *exponential* when $p = c/n$. For the latter case it was proved that the smaller the constant c the closer to 2^n average case complexity of the algorithm. Then the complexity of a naive exhaustive search algorithm has been studied. Here, for $p = c/n$, for some constant c, its complexity becomes exponential, tending, for very large values of c, to 2^n.

References

[1] C. Banderier, H. Hwang, V. Ravelomanana, and V. Zacharovas. Average case analysis of NP-complete problems: Maximum independent set and exhaustive search algorithms. International Meeting on Probabilistic, Combinatorial, and Asymptotic Methods in the Analysis of Algorithms, AofA09, 2009.

[2] C. Banderier, H. Hwang, V. Ravelomanana, and V. Zacharovas. Analysis of an exhaustive search algorithm in random graphs and the $n^{c \log n}$-asymptotics. SIAM J. Disc. Math., 28(1):342–371, 2014.

[3] B. Bollobás. Random graphs. Academic Press, London, 1985.

[4] N. Bourgeois, R. Catellier, T. Denat, and V. Th. Paschos. Average-case complexity of a branch-and-bound algorithm for maximum independent set, under the G(n, p) random model. CoRR, abs/1505.04969, 2015.

[5] Ph. Flajolet and R. Sedgewick. Analytic combinatorics. Cambridge University Press, 2008.

[6] D. E. Knuth. The art of computer programming: fundamental algorithms, volume 1. Addison-Wesley, Reading MA, 1969.

[7] J. M.M. van Rooij and H. L. Bodlaender. Exact algorithms for dominating set. Discrete Appl. Math., 159(17):2147–2164, 2011.

[8] M. Xiao and H. Nagamochi. Exact algorithms for maximum independent set. Information and Computation, 255:126 – 146, 2017.