5th International Workshop on Experimental Algorithms, WEA 2006,
Lecture Notes in Computer Science, Springer Verlag, Vol. 4007, pp. 231–240,

Kernels for the Vertex Cover Problem on the Preferred Attachment Model[⋆]

Josep Díaz, Jordi Petit, and Dimitrios M. Thilikos

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Campus Nord Ω-228
08034, Barcelona. Spain.

Abstract. We examine the behavior of two kernelization techniques for the vertex cover problem viewed as preprocessing algorithms. Specifically, we deal with the kernelization algorithms of Buss and of Nemhauser & Trotter. Our evaluation is applied to random graphs generated under the preferred attachment model, which is usually met in real word applications such as web graphs and others. Our experiments indicate that, in this model, both kernelization algorithms (and, specially, the Nemhauser & Trotter algorithm) reduce considerably the input size of the problem and can serve as very good preprocessing algorithms for vertex cover, on the preferential attachment graphs.

1 Introduction

Given a graph G and a non-negative integer k, the Vertex Cover problem asks whether k of the vertices of G are endpoints of all of its edges. The minimum k for which a graph G has a vertex cover of size k or less is called the *vertex cover* of G and is denoted as $\mathbf{vc}(G)$.

Vertex Cover was one of the first problems proven to be NP-complete [14] and, since then, a lot of efforts have been done in theory and in practice towards coping with its computational intractability. In this direction, there were several advances on the existence of approximation algorithms [15, 13], while lower bounds to its constant factor aproximability have been proposed in [12].

More recently, Vertex Cover has been extensively studied under the viewpoint of fixed parameter complexity. According to the parameterization approach, some part of the problem is declared as the *parameter* and reflects the part of the problem that is expected to be small in most of the "real word" instances. Parameterized complexity asks whether an algorithm of complexity $\mathrm{O}(f(k) \cdot n^{\mathrm{O}(1)})$ exists, where k is the parameter and n is the problem size. Such a *parameterized algorithm* classifies the parameterized problem in the class FPT and claims that the problem is tractable at least for small values of the parameter. For more details on the recent advances and challenges of parameterized

⋆ This research was supported by the EU 6th FP under contract 001907 (DELIS). The first author was partially supported by the *Distinció per a la Promoció de la Recerca de la GC, 2002.*

complexity we refer to [9, 8]. In the case of vertex cover, the parameter is k and Vertex Cover is known to be in FPT. There is a long time race towards obtaining the fastest parameterized algorithm for it; the current champion is [6] that runs in $O(kn + 1.2738^k)$ steps.

One of the main tools for the design of fast parameterized algorithms is *kernelization.* This technique consists in finding a polynomial-time reduction of a parameterized problem to itself in a way that the sizes of the instances of the new problem —we call it *kernel*— *only* depend on the parameter k. The function that bounds the size of the main part of the reduced problem determines the *size* of the kernel and is usually of polynomial (on k) size. If a kernel exists, then we can apply the fastest available exact algorithm for the problem to the (reduced size) kernel instead of the initial instance. Several kernelizations have been proposed for Vertex Cover. A simple kernel of size $O(k^2)$ was proposed by Buss [5]. So far, the smallest kernel has size $\leq 2k$ and its follows from the results of Nemhauser & Trotter in [16]. More recently a kernel of size $\leq 3k$ appeared, based on the "crown decomposition" technique in [10]. An experimental evaluation of these three kernelization techniques was done in [1]. The graphs used for the study in [1] were taken from open-source repositories of biological data. We stress that no random graph model was considered in [1].

Notice that kernelization can also be seen as a preprocessing algorithm. It essentially provides a way to preprocess the input of a problem and transform it to an equivalent one of size of provably bounded size. From a practical point of view, this approach may be useful not only when the parameter k is small, as it is presumed by the parameterized complexity. Moreover, in many cases of parameterized or exact algorithms, actual running times are much better than their theoretically proven bounds. This brings up the challenge of evaluating their performance on specific models of inputs. In this paper, we start such an evaluation with the Vertex Cover problem. A basic concern for our experiments was the model of graphs on which they should be applied.

At the end of the decade of 90's, several empirical studies showed that the degree distribution of large dynamic graphs seem to follow a power-law tail. Among those graphs studied were the WWW, Internet, metabolic networks and others (see, for example, [3] or the surveys [17, 2]). Those results gave a push to the theoretical study of new random graph models with power law tail, as the classical Erdős-Rényi-Gilbert model have a bell-shaped degree distribution, and an exponetially decreasing tail, which did not fit the experimental evidence of power law degree distribution. Recently, it has been showed that the same experimental techniques used on the WWW, namely *Traceroute sampling*, when applied to the classical $G_{n,p}$ model also follow a power law with exponent 1 [7]. In any case, the experimental work on power-law distribution graphs, gave a fruitfull research on dynamic random graph models to fit the experimental evidence, observed for power-law graphs, like the clustering coefficient and small diameter. The more basic model of heavy tailed degree distribution random graphs is the *preferential attachement model (PAM)* given implicitly in [3] and made rigorous in [4].

For the above reasons, we adopt here a model of random graphs generated using the preferred attachment criterion. In this model, a random graph is generated by adding vertices one by one, and the selection of their neighbors is done in a way that favorites vertices that have already *big degree*. In this sense, the vertices of the graph are organized around "clusters" of high degree vertices. Apart from the WWW, such models resemble graphs emerging by distinct applications such as collaboration networks (i.e. in biology, in science, or in film actors), word co-occurrences, or neural networks; see [17].

We base our experiment on the kernelizations of Buss and of Nemhauser & Trotter and we call them *Buss*- and *NT-kernelization* respectively. In both preprocessing algorithms the input is a pair (G, k) and the output is either a definite answer (`YES` or `NO`) or a new *equivalent* instance (G^*, k^*) of the problem. In case no definite answer is given by the kernelization, the returned equivalent instance (G^*, k^*) is the input for an exact algorithm for the vertex cover problem.

Let n denote the number of the vertices of the input graph. Currently, the fastest exact algorithm for Vertex Cover has running time $\mathrm{O}^*(1.19^n)$ [18]; A much simpler (and easier to implement) algorithm for the same problem was given in [11] and has running time $\mathrm{O}^*(1.221^n)$.

Clearly, a preprocessing algorithm is good if it achieves a good reduction of the input size. A measure of this reduction could be the ratio $|V(G^*)|/|V(G)|$. Instead, we suggest a more realistic measure based on the fact that $\mathbf{vc}(G) = \sum_{H \in \mathcal{G}(G)} \mathbf{vc}(H)$, where $\mathcal{G}(G)$ is the set of the connected components of a graph G. This implies that the size of the biggest component of G will dominate the running time of the (exponential time) search for a vertex cover of minimum size. Thus, it reasonable to choose as measure of the input size reduction the ratio $\max_{H \in \mathcal{G}(G^*)} |V(H)| / \max_{H \in \mathcal{G}(G)} |V(H)|$ where G^* is the output graph when we apply kernelization $\mathbf{K}$ to an input graph G. As the graphs generated by our model will always connected, we redefine the reduction measure as

$$\rho(\mathbf{K}, G) = \frac{\max_{H \in \mathcal{G}(G^*)} |V(H)|}{|V(G)|}$$

and we refer to it as the *component reduction* ratio for kernelization $\mathbf{K}$ applied to the graph G (in this paper $\mathbf{K} \in \{\mathrm{Buss}, \mathrm{NT}\}$).

In the following, we evaluate the algorithms for several values of k, with the following criteria:

1. The percentage of cases where the kernelization gives a definite answer.
2. The component reduction ratio (given that the kernelization does not give a definite answer).
3. The comparative performance of the two algorithms according to criteria 1 and 2.
4. The comparative performance of the two algorithms according to their running times.

The most remarkable result of our experiments is that both kernelization algorithms achieve a very good reduction on graphs generated under the preferential attachment model. In particular, we observe reductions approaching the 9% in the case of Buss-kernelization and the .35% in the case of NT-kernelization (for any value of $k \leq 100$ and for graphs with 2000 vertices). This indicates that the NT-kernelization does not only provide the smallest (so far) kernel for parameterized Vertex Cover but can also serve as a very good preprocessing algorithm for the same problem when applied to graphs emerging from the preferred attachment model.

The paper is organized as follows. In Section 2 we present the two kernelization algorithms and the way we implemented them. In Section 3 we detail the model of random graphs that we use. In Section 4 we present and comment the results of our experiments. Finally, in Section 5, we give some remarks and research directions.

2 Kernelizations for Vertex Cover

We consider undirected graphs without loops nor multiple edges. Given a graph G, its vertex and edge set are denoted as $V(G)$ and $E(G)$ respectively. A connected component of a graph is called non-trivial if it contains at least 2 vertices. Given a vertex $v \in V(G)$ we denote by $\deg_G(v)$ the degree of v in G, i.e. the number of vertices adjacent to v in G. We also denote by $G - v$ the graph obtained by G after we remove vertex v and all its incident edges. If $S \subseteq V(G)$ we define the *subgraph of G induced by S* as $G[S] = (S, \{e \in E(G) \mid e \subseteq S\})$. Finally, we use the notation $I(G)$ for the set of isolated vertices in G.

2.1 The kernelization of Buss

The first kernelization we study in this paper follows from the results in [5] and is sketched below:

```
def Buss(G, k) :
  1. if $k = 0$ and $I(G) = V(G)$:  return YES
  2. if $k = 0$ and $I(G) \neq V(G)$:  return NO1
  3. if $\exists v \in V(G)$ :  $\deg_G(v) > k$:  return Buss($G - v, k - 1$)
  4. if $|V(G)| > k(k + 1)$:  return NO2
  5. if $G$ has more than $k$ non-trivial connected components:  return NO3
  6. return $(G[V(G) - I(G)], k)$
```

The correctness of the algorithm is based on the three following facts: a) any vertex of degree more than k should be in any vertex cover of size $\leq k$, b) a graph with max degree $\leq k$ and a vertex cover of size $\leq k$ cannot have more than $k + k^2$ non-isolated vertices, and c) any graph with more than k non-trivial connected components has no vertex cover of size less than k. The time complexity of **Buss**(G, k) is $O(k \cdot |V(G)|)$.

In order to refine the analysis of our experiments, we distinguish between the three ways that the algorithm may return a negative answer. The first negative answer (`NO1`) appears is the case when, after the deletion of k high degree vertices, there still remain edges in the graph. The second (`NO2`) appears when no high degree vertices exist and the graph has big size (more than $k + k^2$ vertices). The last one (`NO3`) is when the current graph has more than k non-trivial connected components.

We would like to stress that Step 6 (corresponding to `NO3`) is not a part of the classic kernelization of Buss. However, we added it because in our experiments it behaves rather well as a filter of `NO`-instances.

2.2 The kernelization of Nemhauser & Trotter

The kernelization of Nemhauser & Trotter follows from the results in [16] and is sketched bellow:

```
def $\mathbf{NT}(G, k)$ :
 1. let $U_1 = V(G)$ and let $U_2$ be a new set of vertices where $|U_1| = |U_2|$
 2. let $\sigma : U_1 \to U_2$ be a bijection from $U_2$ to $U_1$
 3. let $H = (U_1 \cup U_2, \{\{x, y\} \mid x \in U_1, y \in U_2, \text{ and } \{x, \sigma(y)\} \in E(G)\})$
 4. let $S$ be a vertex cover of $H$
 5. let $V_1 = \{x \mid x \in S \cap U_1 \text{ and } \sigma^{-1}(x) \in S\}$
 6. let $V_0 = \{x \mid x \in U_1 - S \text{ and } \sigma^{-1}(x) \notin S\}$
 7. let $V_{\frac{1}{2}} = U_1 - V_1 - V_0$
 8. if $|V_1| > k$: return NO1
 9. if $|V_1| = k$ and $E(G[V_{\frac{1}{2}}]) \neq \emptyset$: return NO2
10. if $|V_1| = k$ and $E(G[V_{\frac{1}{2}}]) = \emptyset$: return YES
11. if $|V_{\frac{1}{2}}| > 2(k - |V_1|)$: return NO3
12. return $(G[V_{\frac{1}{2}}], k - |V_1|)$
```

$\mathbf{NT}(G, k)$ first constructs a bipartite graph H where one of its parts has the vertices of G and the other copies of the vertices of G. An edge from one part to the other is placed if the corresponding vertices are adjacent to the original graph G. This construction takes place in steps **1**–**3**. In step **4**, the algorithm finds a minimal vertex cover S of H. Such a vertex cover can be computed in polynomial time because H is bipartite. Specifically, we have done so adding a source and sink vertex to H and computing a max-flow min-cut using the Edmonds-Karp algorithm (its running time is $\mathrm{O}(|V||E|)$. In steps **5**–**7** the algorithm partitions the vertices of G into three sets V_1, V_0, and $V_{\frac{1}{2}}$: those that both themselves and their copies belong in S, those than neither themselves nor their copies belong into S, and those that either themselves or their copies belong to S (but not both). From [16], it follows that a) $\mathbf{vc}(G) = \mathbf{vc}(G[V_{\frac{1}{2}}]) + |V_1|$ and b) $|V_{\frac{1}{2}}| \leq 2 \cdot \mathbf{vc}(G[V_{\frac{1}{2}}])$. These relations justify the answers provided in steps **8**–**12**. There are three possible ways for the algorithm to return `NO`: the first appears when the size of V_1 is bigger than k (`NO1`), the second appears when $|V_1| = k$

and there are edges between vertices in $V_{\frac{1}{2}}$ (NO2), and the third appears when $|V_{\frac{1}{2}}| > 2(k - |V_1|)$ (NO3).

We did not add the filter of Step 6 of the **Buss**(G, k) algorithm as our experiments showed that, for our graph generation model, it does not offer any additional filtering for the case of NT-kernelization.

3 The model

The graph generation algorithm is depicted bellow; we assume that m is a small constant ($m \ll n$):

```
def PrefAttach(n, m) :
 1. let $G$ be a cycle of $m$ vertices
 2. for $i = m + 1$ to $n$:
        add a new vertex $u$ to $G$
        add $m$ edges from $u$ to $v_1, \ldots, v_m$,
            where each $v_i$ is selected with probability $\frac{\deg_G(v_i)^2}{\sum_{v \in V(G-u)} \deg_G(v)^2}$
 3 return $G$
```

Function **Pref-Attach**(n, m) generates graphs incrementally adding one vertex at each step. Each new vertex gets m neighbors that are picked among the vertices of the already constructed graph. The selection of a neighbor is biased in a way that vertices of high degree in G are prefered against those with low degree. This makes the generated graphs look like clusters of vertices positioned in a way that resembles the adjacency of web graphs. If G is a graph generated by **Pref-Attach**(n, m), then $|V(G)| = n$ and $|E(G)| = m(n - m + 1)$.

Our implementation of the above procedure confirms the intuition that the vast majority of the vertices in G are of low degree while a small minority has big degree. In Figure 1 one can see the degree distribution of a graph generated by **Pref-Attach**$(2000, 4)$.

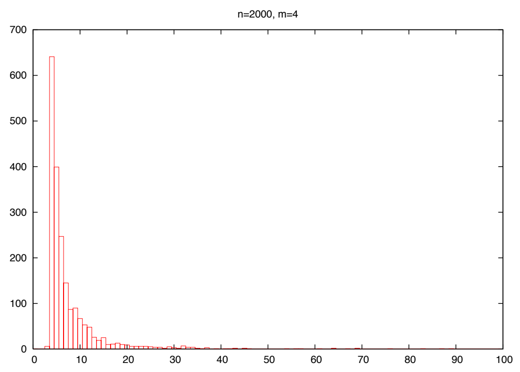

Fig. 1. Degree distribution on a **Pref-Attach**$(2000, 4)$ graph.

4 Experimental results

In this section we present some experimental results to evaluate the Buss and NT kernelization algorithms for the VERTEX COVER problem on the preferred attachment model. To do so, we have implemented both algorithms as well as the generation of the random graphs. All our code has been written in C++ using the STL and compiled with GCC 3.4.1 with the `-O3` option. The experiments have been performed on a machine with a Intel Pentium 4 CPU at 3.20GHz and with 3GB of memory running a Linux operating system.

We split the presentation of our results in three subsections. First, we present the distribution of the possible results of the two kernelization algorithms. Afterwards, we show the component reduction ratio for each one of them. Finally, we present some time measurements.

4.1 Distribution of the possible answers

In Figure 4.1, we present the percentages for the appearance of the possible outputs (`NO1`, `NO2`, `NO3`, and `DON'T KNOW`) of $\mathbf{Buss}(G, k)$ and $\mathbf{NT}(G, k)$ for $k \in \{1, \ldots, 100\}$. Our sample contained 500 graphs, generated by **Pref-Attach**(n, m) taking $n = 2000$ and $m = 4$. experiments. The curves for other values of n and m behave similary. We do not include the count of `YES` answers in the figures because they never appeared in our experiments.

With regard to the Buss kernelization, we can see that answers `NO1`, `NO2` and `NO3` play a complementary role as k increases: The `NO1` criterium is usefull when k is rather small and vanishes as it increases. At this point, the `NO2` criterium turns to be usefull to report negative answers but, again, vanishes as k keeps increasing. It is then the turn of the `NO3` criterium to report negative answers until a point where `DON'T KNOW` answers begin to appear, which happens in more than a half of the cases for $k > 33$ on our **Pref-Attach**$(2000, 4)$ graphs.

The NT kernelization exhibits a similar behaviour with respect to the complementary role of the different negative criteria as k increases. However, an immediate comparison of the two diagrams implies that the NT kernelization gives much more information than the Buss kernelization. For instance, we can report a negative answer on more than one half of the instances for values of k up to 65 (rather than 33 with Buss).

To have a clearear view of when certain percentages of `NO` answers can be expected, see Figure 4.1. The horizontal axis represents distinct values of n (the number of vertices of the generated graphs) and each of the lines represents the biggest k for which the percentage of the `DON'T KNOW` answer was grater than 0%, 20% or 50% for the Buss and the NT kernelizations. In this case, the experiment shows the averages computed over 200 graphs for each size.

Again it is clear that the NT-kernelization is more powerfull and scales better with the graph size than its Buss counterpart. No only that, Figure 4.1 also induces us to conjecture that if G is a random graph generated by **Pref-Attach**(m, n) then, with high probability, $\mathbf{vc}(G) \geq c_m \log^{O(1)} n$ where c_k is a constant depending on m.

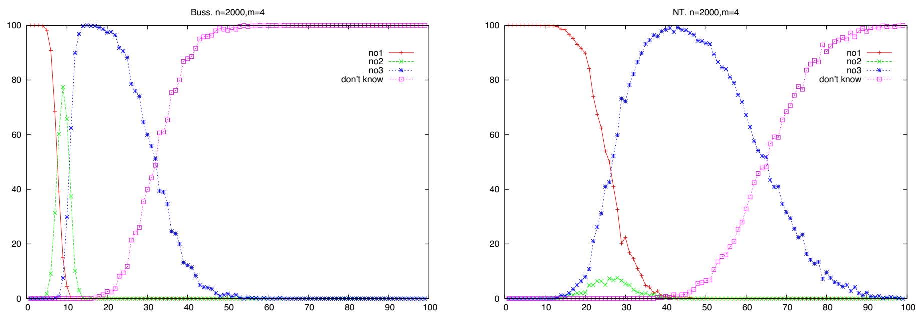

Fig. 2. Behaviour of Buss and NT kernalizations on Pref-Attach graphs with $n = 2000$ and $m = 4$ for $k \in \{1, \ldots, 100\}$.

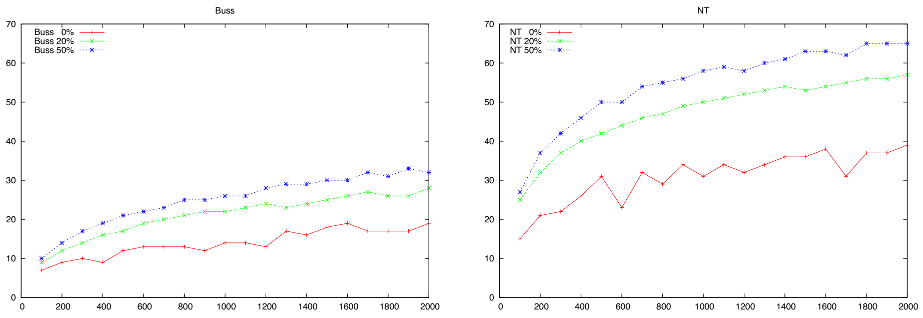

Fig. 3. Maximal value of k for which Buss and NT kernelizations can give a negative answer in $p\%$ of the cases for $p \in \{0, 20, 50\}$ on Pref-Attach graphs with $n \in \{100, \ldots, 2000\}$ and $m = 4$.

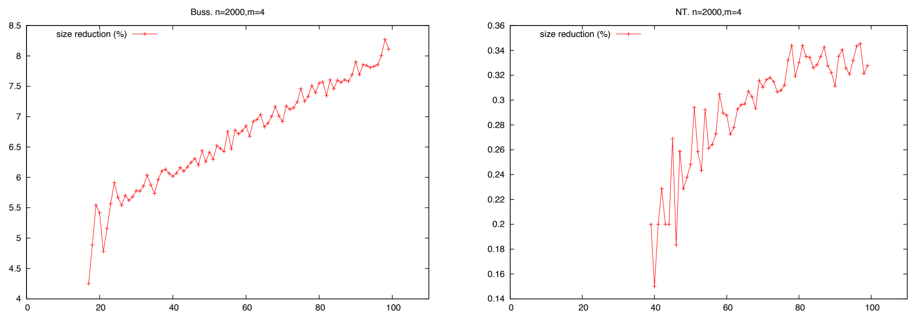

Fig. 4. Component reduction ratio

4.2 Component reduction ratio

We consider now the reduction that is achieved in the cases where the answer is unknown and a kernel is computed. We are interested in the component reduction.

Figure 4.1 presents the percentual values of $\rho(\mathbf{Buss}, k)$ and $\rho(\mathbf{NT}, k)$. Again, our sample contained 500 graphs, generated by $\mathbf{Pref\text{-}Attach}(n, m)$ for $n = 2000$ and $m = 4$. The horizontal axis represent the values of k up to 100, and the vertical axis represents the percentage of the component reduction. The curve is not shown for values of k where a definite answer was found.

The results are quite positive for the capacity of the Buss kernelization as a preprocesing algorithm: even when $k = 100$, the Buss kernelization achieves a reduction to equivalent instances where the maximum component has size that is no bigger than the 9% of the size of the original input.

News are even better in the case of the NT kernelization, because it achieves a maximum component size reduction arround a 0.35% for any $k \leq 100$. This means that the brute force algorithm shall only have to solve vertex covers for components with an average of 7 vertices. Such a performance clearly indicates that the NT kernelization is indeed an excellent preprocessing algorithm for graphs generated by the preferred attachment model.

4.3 Running times

So far NT kernelization beats the Buss kernelization by any means. However, we should mention that on the preferred attachment model, the opposite holds as far as the running time is concerned. Time measures for the previous experiments show that our implementation for NT is about 10 times slower than our implementation for Buss. However, the actual running times are so small (tenths of seconds for NT and hundredths of seconds for Buss) that these do not seem to matter. Moreover, we should stress that these are just preprocessing times: the main running times would still be dominated by the time required by the exact (exponential) algorithm. We thus conclude that the relative speed of both algorithms is not that big.

5 Discussion

This paper attempts a first study of the performance of kernelization algorithms when treated as preprocessing algorithms to hard problems, on the family of random graphs that we have used. Especially for the Vertex cover, it remains an open project to see whether the optimistic results of this paper can be extended for other models of random graphs generated by several proposed variations on the preferred attachment mechanism. Aa the PAM is a model for the web graph, it would be useful to confirm that the same results come up when the experiments are applied to a sufficiently large part of the main component of the web (or the internet graph). This could have a potential for a new way to identify big hubs in those graphs. Finally, we believe that the potential of the kernelization idea can offer experimental results that are really better than the formally proven bounds. It is an open project to examine this for kernelization algorithms provided by the parameterized complexity context for other problems as well.

References

1. Faisal N. Abu-Khzam, Rebecca L. Collins, Michael R. Fellows, Michael A. Langston, W. Henry Suters, and Christof T. Symons. Kernelization algorithms for the vertex cover problem: Theory and experiments. In *Sixth Workshop on Algorithm Engineering and Experiments and the First Workshop on Analytic Algorithmics and Combinatorics, New Orleans, LA, USA*, pages 62–69. SIAM, 2004.
2. Albert-László Barabási. Emergence of scaling in complex networks. In *Handbook of graphs and networks*, pages 69–84. Wiley-VCH, Weinheim, 2003.
3. Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.
4. Béla Bollobás, Oliver Riordan, Joel Spencer, and Gábor Tusnády. The degree sequence of a scale-free random graph process. *Random Structures Algorithms*, 18(3):279–290, 2001.
5. J.F. Buss and J. Goldsmith. Nondeterminism within p. *SIAM J. Computing*, 22:560–572, 1993.
6. Jianer Chen, Iyad A. Kanj, and Ge Xia. Simplicity is beauty: Improved upper bounds for vertex cover. Technical Report 05-008, Texas A&M University, Utrecht, the Netherlands, April 2005.
7. Aaron Clauset and Cristopher Moore. Accuracy and scaling phenomena in internet mapping. *Phys. Rev. Lett.*, 94, 2005.
8. R. G. Downey and M. R. Fellows. *Parameterized complexity*. Monographs in Computer Science. Springer-Verlag, New York, 1999.
9. Michael R. Fellows. Parameterized complexity: the main ideas and some research frontiers. In *Algorithms and computation (Christchurch, 2001)*, volume 2223 of *Lecture Notes in Comput. Sci.*, pages 291–307. Springer, Berlin, 2001.
10. Michael R. Fellows. Blow-ups, win/win's, and crown rules: Some new directions in fpt. In *Graph-Theoretic Concepts in Computer Science, 29th International Workshop, WG 2003, Elspeet, The Netherlands, June 19-21*, Lecture Notes in Computer Science, pages 1–12. Springer, 2003.
11. F. V. Fomin, F. Grandoni, and D. Kratsch. Large measure and conquer: A simple $O(2^{0.288\,n})$ independent set algorithm. In *17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006)*, page to appear. ACM and SIAM, New York, 2006.
12. Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859 (electronic), 2001.
13. George Karakostas. A better approximation ratio for the vertex cover problem. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, (ICALP),*, Lecture Notes in Computer Science, pages 1043–1050. Springer, 2005.
14. Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 85–103. Plenum, New York, 1972.

15. Burkhard Monien and Ewald Speckenmeyer. Ramsey numbers and an approximation algorithm for the vertex cover problem. *Acta Inform.*, 22(1):115–123, 1985.
16. G. L. Nemhauser and L. E. Trotter, Jr. Vertex packings: structural properties and algorithms. *Math. Programming*, 8:232–248, 1975.
17. Mark E. J. Newman. Random graphs as models of networks. In *Handbook of graphs and networks*, pages 35–68. Wiley-VCH, Weinheim, 2003.
18. J. M. Robson. Finding a maximum independent set in time $O(2^{n/4})$. manuscript, `http://dept-info.labri.fr/ robson/mis/techrep.html`, 2001.