Finding a Maximum Induced Degenerate Subgraph Faster Than 2^n

Marcin Pilipczuk[1,⋆] and Michał Pilipczuk[2,⋆⋆]

[1] Institute of Informatics, University of Warsaw, Poland
malcin@mimuw.edu.pl
[2] Department of Informatics, University of Bergen, Norway
michal.pilipczuk@ii.uib.no

Abstract. In this paper we study the problem of finding a maximum induced d-degenerate subgraph in a given n-vertex graph from the point of view of exact algorithms. We show that for any fixed d one can find a maximum induced d-degenerate subgraph in randomized $(2-\varepsilon_d)^n n^{\mathcal{O}(1)}$ time, for some constant $\varepsilon_d > 0$ depending only on d. Moreover, our algorithm can be used to sample inclusion-wise maximal induced d-degenerate subgraphs in such a manner that every such subgraph is output with probability at least $(2-\varepsilon_d)^{-n}$; hence, we prove that their number is bounded by $(2-\varepsilon_d)^n$.

1 Introduction

The theory of exact computations studies the design of algorithms for NP-hard problems that compute the answer optimally, however using possibly exponential time. The goal is to limit the exponential blow-up in the best possible running-time guarantee. For some problems, like Independent Set [1], Dominating Set [1, 2], and Bandwidth [3] the research concentrates on achieving better and better constants in the bases of exponents. However, for many important computational tasks designing even a routine faster than trivial brute-force solution or straightforward dynamic program is a challenging combinatorial question; the answer to this question can provide valuable insight into the structure of the problem. Perhaps the most prominent among recent developments in breaking trivial barriers is the algorithm for Hamiltonian Cycle of Björklund [4], but a lot of effort is put also into less fundamental problems, like Maximum Induced Planar Graph [5] or a scheduling problem $1|\text{prec}|\sum C_i$ [6], among many others [7–12]. However, many natural and well-studied problems still lack exact algorithms faster than the trivial ones; the most important examples are TSP, Permanent, Set Cover, #Hamiltonian Cycles and SAT. In particular, hardness of SAT is the starting point for the *Strong Exponential Time Hypothesis* of Impagliazzo and Paturi [13, 14], which is used as an argument that other problems are hard as well [15–18].

⋆ Partially supported by NCN grant N206567140 and Foundation for Polish Science.
⋆⋆ Partially supported by European Research Council (ERC) Grant "Rigorous Theory of Preprocessing", reference 267959.

D.M. Thilikos and G.J. Woeginger (Eds.): IPEC 2012, LNCS 7535, pp. 3–12, 2012.

A group of tasks we are particularly interested in in this paper are the problems that ask for a maximum size induced subgraph belonging to some class Π. If belonging to Π can be recognized in polynomial time, then we have an obvious brute-force solution working in $2^n n^{\mathcal{O}(1)}$ time that iterates through all the subsets of vertices checking which of them induce subgraphs belonging to Π. Note that the classical Independent Set problem can be formulated in this manner for Π being the class of edgeless graphs, while if Π is the class of forests then we arrive at the Maximum Induced Forest, which is dual to Feedback Vertex Set. For both these problems algorithms with running time of form $(2-\varepsilon)^n$ for some $\varepsilon > 0$ are known [1, 11, 12]. The list of problems admitting algorithms with similar complexities includes also Π being the classes of regular graphs [19], graphs of small treewidth [20], planar graphs [5], 2- or 3-colourable graphs [21], bicliques [22] or graphs excluding a forbidden subgraph [23].

The starting point of our work is the question raised by Fomin et al. in [5]. Having obtained an algorithm finding a maximum induced planar graph in time $\mathcal{O}(1.7347^n)$, they ask whether their result can be extended to graphs of bounded genus or even to H-minor-free graphs for fixed H. Note that all these graph classes are hereditary and consist of *sparse* graphs, i.e., graphs with the number of edges bounded linearly in the number of vertices. Moreover, for other hereditary sparse classes, such as graphs of bounded treewidth, algorithms with running time $(2-\varepsilon)^n$ for some $\varepsilon > 0$ are also known [20]. Therefore, it is tempting to ask whether the sparseness of the graph class can be used to break the 2^n barrier in a more general manner.

In order to formalize this question we study the problem of finding a maximum induced d-degenerate graph. Recall that a graph is called d-degenerate if each of its subgraphs contains a vertex of degree at most d. Every hereditary class of graphs with a number of edges bounded linearly in the number of vertices is d-degenerate for some d; for example, planar graphs are 5-degenerate, graphs excluding K_r as a minor are $\mathcal{O}(r\sqrt{\log r})$-degenerate, while the class of forests is equivalent to the class of 1-degenerate graphs. However, d-degeneracy does not impose any topological constraints; to see this, note that one can turn any graph into a 2-degenerate graph by subdividing every edge. Hence, considering a problem on the class of d-degenerate graphs can be useful to examine whether it is just sparseness that makes it more tractable, or one has to add additional restrictions of topological nature [24].

Our Results and Techniques. We make a step towards understanding the complexity of finding a maximum induced subgraph from a sparse graph class by breaking the 2^n-barrier for the problem of finding maximum induced d-degenerate subgraph. The main result of this paper is the following algorithmic theorem.

Theorem 1. *For any integer $d \geq 1$ there exists a constant $\varepsilon_d > 0$ and a polynomial-time randomized algorithm $\mathcal{A}_d$, which given an n-vertex graph G either reports an error, or outputs a subset of vertices inducing a d-degenerate subgraph. Moreover, for every inclusion-wise maximal induced d-degenerate subgraph, let X be its vertex set, the probability that $\mathcal{A}_d$ outputs X is at least $(2-\varepsilon_d)^{-n}$.*

Let X_0 be a set of vertices inducing a maximum d-degenerate subgraph. If we run the algorithm $(2-\varepsilon_d)^n$ times, we know that with probability at least $1/2$ in one of the runs the set X_0 will be found. Hence, outputting the maximum size set among those found by the runs gives the following corollary.

Corollary 2. *There exists a randomized algorithm which, given an n-vertex graph G, in $(2-\varepsilon_d)^n n^{\mathcal{O}(1)}$ time outputs a set $X \subseteq V(G)$ inducing a d-degenerate graph. Moreover, X is maximum with probability at least $\frac{1}{2}$.*

As the total probability that $\mathcal{A}_d$ outputs some set of vertices is bounded by 1, we obtain also the following corollary.

Corollary 3. *For any integer $d \geq 1$ there exists a constant $\varepsilon_d > 0$ such that any n-vertex graph contains at most $(2-\varepsilon_d)^n$ inclusion-wise maximal induced d-degenerate subgraphs.*

Let us elaborate briefly on the idea behind the algorithm of Theorem 1. Assume first that G has large average degree, i.e., $|E(G)| > \lambda d|V(G)|$ for some large constant λ. As d-degenerate graphs are sparse, i.e., the number of edges is less than d times the number of vertices, it follows that for any set X inducing a d-degenerate graph $G[X]$, only a tiny fraction of edges inside G are in fact inside $G[X]$. Hence, an edge uv chosen uniformly at random can be assumed with high probability to have at least one endpoint outside X. We can further choose at random, with probabilities $1/3$ each, one of the following decisions: $u \in X$, $v \notin X$ or $u \notin X$, $v \in X$, or $u, v \notin X$. In this manner we fix the status of two vertices of G and, if $\lambda > 4$, the probability that the guess is correct is larger than $1/4$. If this randomized step cannot be applied, we know that the average degree in G is at most λd and we can apply more standard branching arguments on vertices of low degrees.

Our algorithm is a polynomial-time routine that outputs an induced d-degenerate graph by guessing assignment of consecutive vertices with probabilities slightly better than $1/2$. We would like to remark that all but one of the ingredients of the algorithm can be turned into standard, deterministic branching steps. The only truly randomized part is the aforementioned random choice of an edge to perform a guess with enhanced success probability. However, to ease the presentation we choose to present the whole algorithm in a randomized fashion by expressing classical branchings as random choices of the branch.

Organization. In Section 2 we settle notation and give preliminary results on degenerate graphs. Section 3 contains the proof of Theorem 1. Section 4 concludes the paper.

2 Preliminaries

Notation. We use standard graph notation. For a graph G, by $V(G)$ and $E(G)$ we denote its vertex and edge sets, respectively. For $v \in V(G)$, its neighborhood $N_G(v)$ is defined as $N_G(v) = \{u : uv \in E(G)\}$. For a set $X \subseteq V(G)$ by $G[X]$ we denote the subgraph of G induced by X. For a set X of vertices or edges of G, by $G \setminus X$ we denote the graph with the vertices or edges of X removed; in case of vertex removal, we remove also all the incident edges.

Degenerate Graphs. For an integer $d \geq 0$, we say that a graph G is d-degenerate if every subgraph (equivalently, every induced subgraph) of G contains a vertex of degree at most d. Clearly, the class of d-degenerate graphs is closed under taking both subgraphs and induced subgraphs. Note that 0-degenerate graphs are independent sets, and the class of 1-degenerate graphs is exactly the class of forests. All planar graphs are 5-degenerate; moreover, every K_r-minor-free graph (in particular, any H-minor-free graph for $|V(H)| = r$) is $\mathcal{O}(r\sqrt{\log r})$-degenerate [25–27].

The following simple proposition shows that the notion of d-degeneracy admits greedy arguments.

Proposition 4. *Let G be a graph and v be a vertex of degree at most d in G. Then G is d-degenerate if and only if $G \setminus v$ is.*

Proof. As $G \setminus v$ is a subgraph of G, then d-degeneracy of G implies d-degeneracy of $G \setminus v$. Hence, we only need to justify that if $G \setminus v$ is d-degenerate, then so does G. Take any $X \subseteq V(G)$. If $v \in X$, then the degree of v in $G[X]$ is at most its degree in G, hence it is at most d. However, if $v \notin X$ then $G[X]$ is a subgraph of $G \setminus v$ and $G[X]$ contains a vertex of degree at most d as well. As X was chosen arbitrarily, the claim follows. □

Proposition 4 ensures that one can test d-degeneracy of a graph by in turn finding a vertex of degree at most d, which needs to exist due to the definition, and deleting it. If in this manner we can remove all the vertices of the graph, it is clearly d-degenerate. Otherwise we end up with an induced subgraph with minimum degree at least $d + 1$, which is a sufficient proof that the graph is not d-degenerate. Note that this procedure can be implemented in polynomial time. As during each deletion we remove at most d edges from the graph, the following proposition is straightforward.

Proposition 5. *Any n-vertex d-degenerate graph has at most dn edges.*

3 The Algorithm

In this section we prove Theorem 1. Let us fix $d \geq 1$, an n-vertex graph G and an inclusion-wise maximal set $X \subseteq V(G)$ inducing a d-degenerate graph.

The behaviour of the algorithm depends on a few constants that may depend on d and whose values influence the final success probability. At the end of this section we propose precise values of these constants and respective values of ε_d for $1 \leq d \leq 6$. However, as the values of ε_d are really tiny even for small d, when describing the algorithm we prefer to introduce these constants symbolically, and only argue that there exists their evaluation that leads to a $(2-\varepsilon_d)^{-n}$ lower bound on the probability of successfully sampling X.

The algorithm maintains two disjoint sets $A, Z \subseteq V(G)$, consisting of vertices about which we have already made some assumptions: we seek for the set X that contains A and is disjoint from Z. Let $Q = V(G) \setminus (A \cup Z)$ be the set of the remaining vertices, whose assignment is not yet decided.

We start with $A = Z = \emptyset$. The description of the algorithm consists of a sequence of *rules*; at each point, the lowest-numbered applicable rule is used. When applying a rule we assign some vertices of Q to the set A or Z, depending on some random decision. We say that an application of a rule is *correct* if, assuming that before the application we have $A \subseteq X$ and $Z \cap X = \emptyset$, the vertices assigned to A belong to X, and the vertices assigned to Z belong to $V(G) \setminus X$. In other words, a correct application assigns the vertices consistently with the fixed solution X.

We start with the randomized rule that is triggered when the graph is dense. Observe that, since $G[X]$ is d-degenerate, $G[X \cap Q]$ is d-degenerate as well and, by Proposition 5, contains less than $d|X \cap Q|$ edges. Thus, if $|E(G[Q])|/|Q|$ is significantly larger than d, then only a tiny fraction of the edges of $G[Q]$ are present in $G[X]$. Hence, an overwhelming fraction of edges of $G[Q]$ has at least one of the endpoints outside X, so having sampled an edge of $G[Q]$ uniformly at random with high probability we may assume that there are only three possibilities of the behaviour of its endpoints, instead of four. This observation leads to the following rule. Let $\lambda > 4$ be a constant.

Rule 1. If $|E(G[Q])| \geq \lambda d|Q|$, then:

1. choose an edge $uv \in E(G[Q])$ uniformly at random;
2. with probability 1/3 each, make one of the following decisions: either assign u to A and v to Z, or assign u to Z and v to A, or assign both u and v to Z.

Lemma 6. *Assume that $A \subseteq X$ and $Z \cap X = \emptyset$ before Rule 1 is applied. Then the application of Rule 1 is correct with probability at least $\frac{\lambda-1}{3\lambda}$.*

Proof. As $|E(G[Q])| \geq \lambda d|Q|$, but $|E(G[X \cap Q])| \leq d|X \cap Q| \leq d|Q|$ by Proposition 5, the probability that $uv \notin E(G[X])$ is at least $\frac{\lambda-1}{\lambda}$. Conditional on the assumption $uv \notin E(G[X])$, in the second step of Rule 1 we make a correct decision with probability 1/3. This concludes the proof. □

Note that the bound $\frac{\lambda-1}{3\lambda}$ is larger than 1/4 for $\lambda > 4$.

Equipped with Rule 1, we may focus on the case when $G[Q]$ has small average degree. Let us introduce a constant $\kappa > 2\lambda$ and let $S \subseteq Q$ be the set of vertices having degree less than κd in $G[Q]$. If Rule 1 is not applicable, then $|E(G[Q])| < \lambda d|Q|$. Hence we can infer that $|S| \geq \frac{\kappa-2\lambda}{\kappa}|Q|$, as otherwise by just counting the degrees of vertices in $Q \setminus S$ we could find at least $\frac{1}{2} \cdot \frac{2\lambda}{\kappa}|Q| \cdot \kappa d = \lambda d|Q|$ edges in $G[Q]$. Consider any $v \in S$. Such a vertex v may be of two types: it either has at most d neighbours in A, or at least $d+1$ of them. In the first case, we argue that we may perform a good guessing step in the closed neighbourhood of v, because the degree of v is bounded and when all the neighbours of v are deleted (assigned to Z), then one may greedily assign v to A. In the second case, we observe that we cannot assign too many such vertices v to A, as otherwise we would obtain a subgraph of $G[A]$ with too high average degree. Let us now proceed to the formal arguments.

Rule 2. Assume there exists a vertex $v \in Q$ such that $|N_G(v) \cap Q| < \kappa d$ and $|N_G(v) \cap A| \leq d$. Let $r = |N_G(v) \cap Q|$ and $v_1, v_2, \ldots, v_r$ be an arbitrary ordering of the neighbours of v in Q. Let $\gamma = \gamma(r) \geq 1$ be such that

$$\gamma^{-1} + \gamma^{-2} + \ldots + \gamma^{-r-1} = 1.$$

Randomly, make one of the following decisions:

1. for $1 \leq i \leq r$, with probability γ^{-i} assign $v_1, v_2, \ldots, v_{i-1}$ to Z and v_i to A;
2. with probability γ^{-r-1} assign all vertices $v_1, v_2, \ldots, v_r$ to Z and v to A.

Note that the choice of γ not only ensures that the probabilities of the options in Rule 2 sum up to one, but also that $\gamma(r) \leq \gamma(\lceil \kappa d \rceil - 1) < 2$. We now show a bound on the probability that an application of Rule 2 is correct.

Lemma 7. *Assume that $A \subseteq X$ and $Z \cap X = \emptyset$ before Rule 2 is applied. Then exactly one of the decisions considered in Rule 2 leads to a correct application. Moreover, if in the correct decision exactly i_0 vertices are assigned to $A \cup Z$, then the probability of choosing the correct one is equal to γ^{-i_0}.*

Proof. Firstly observe that the decisions in Rule 2 contradict each other, so at most one of them can lead to a correct application.

Assume that $(N_G(v) \cap Q) \cap X \neq \emptyset$ and let v_{i_0} be the vertex from $(N_G(v) \cap Q) \cap X$ with the smallest index. Then the decision, which assigns all the vertices of $N_G(v) \cap Q$ with smaller indices to Z and v_{i_0} to A leads to a correct application. Moreover, it assigns exactly i_0 vertices to $A \cup Z$ and the probability of choosing it is equal to γ^{-i_0}.

Assume now that $(N_G(v) \cap Q) \cap X = \emptyset$. We claim that $v \in X$. Assume otherwise; then v has at most d neighbours in X, so by Proposition 4 after greedily incorporating it to X we would still have $G[X]$ being a d-degenerate graph. This contradicts maximality of X. Hence, we infer that the decision which assigns all the neighbours of v from Q to Z and v itself to A leads to a correct application, it assigns exactly $r+1$ vertices to $A \cup Z$ and has probability γ^{-r-1}. □

We now handle vertices with more than d neighbours in A. Intuitively, there can be at most $d|A|$ such vertices assigned to A, as otherwise A would have an induced subgraph with too high average degree. Hence, if there is significantly more than $2d|A|$ such vertices in total, then picking one of them at random with probability higher than $1/2$ gives a vertex that needs to be assigned to Z. Let us introduce a constant $c > 2$.

Rule 3. If there are at least $cd|A|$ vertices in Q that have more than d neighbours in A, choose one such vertex uniformly at random and assign it to Z.

Lemma 8. *Assume that $A \subseteq X$ and $Z \cap X = \emptyset$ before Rule 3 is applied. Then the application of Rule 3 is correct with probability at least $1 - 1/c$.*

Proof. Let $P = \{v \in Q : |N_G(v) \cap A| > d\}$. As $|P| \geq cd|A|$, to prove the lemma it suffices to show that $|P \cap X| < d|A|$. Assume otherwise, and consider the set $((P \cap X) \cup A) \subseteq X$. The number of edges of the subgraph of $G[X]$ induced by $(P \cap X) \cup A$ is at least

$$(d+1)|P \cap X| = d|P \cap X| + |P \cap X| \geq d(|P \cap X| + |A|) = d|(P \cap X) \cup A|.$$

This contradicts the assumption that $G[X]$ is d-degenerate, due to Proposition 5. □

Note that $1 - 1/c > 1/2$ for $c > 2$.

We now show that if Rules 1, 2 and 3 are not applicable, then $|A \cup Z|$ is large, which means that the algorithm has already made decisions about a significant fraction of the vertices of the graph.

Lemma 9. *If Rules 1, 2 and 3 are not applicable, then $|A \cup Z| > \alpha n$ for some constant $\alpha > 0$ that depends only on the constants d, λ, κ and c.*

Proof. As Rule 1 is not applicable, Q contains at most $\frac{2\lambda}{\kappa}|Q|$ vertices of degree at least κd in $G[Q]$. As Rule 2 is not applicable, the remaining vertices have more than d neighbours in A. As Rule 3 is not applicable, we have that

$$\frac{\kappa - 2\lambda}{\kappa}|Q| < cd|A| \leq cd|A \cup Z|.$$

As $Q = V(G) \setminus (A \cup Z)$, simple computations show that this is equivalent to

$$\frac{|A \cup Z|}{|V(G)|} > \left(\frac{cd\kappa}{\kappa - 2\lambda} + 1\right)^{-1},$$

and the proof is finished. □

Lemma 9 ensures that at this point the algorithm has already performed enough steps to achieve the desired success probability. Therefore, we may finish by brute-force.

Rule 4. If $|A \cup Z| > \alpha n$ for the constant α given by Lemma 9, for each $v \in Q$ independently, assign v to A or Z with probability $1/2$ each, and finish the algorithm by outputting the set A if it induces a d-degenerate graph, or reporting an error otherwise.

We now summarize the bound on the success probability.

Lemma 10. *The algorithm outputs the set X with probability at least*

$$\max\left(\sqrt{\frac{3\lambda}{\lambda - 1}}, \gamma(\lceil \kappa d \rceil - 1), \frac{c}{c-1}\right)^{-\alpha n} 2^{-(1-\alpha)n},$$

which is equal to $(2 - \varepsilon_d)^n$ for some $\varepsilon_d > 0$.

Proof. Recall that $\frac{3\lambda}{\lambda-1} < 4$, $\gamma(\lceil \kappa d \rceil - 1) < 2$, $\frac{c}{c-1} < 2$ and $\alpha > 0$, by the choice of the constants and by Lemma 9. Therefore, it suffices to prove that, before Rule 4 is applied, the probability that $A \subseteq X$ and $Z \cap X = \emptyset$ is at least

$$\max\left(\sqrt{\frac{3\lambda}{\lambda-1}}, \gamma(\lceil \kappa d \rceil - 1), \frac{c}{c-1}\right)^{-|A \cup Z|}.$$

However, this is a straightforward corollary of Lemmata 6, 7 and 8. □

This concludes the proof of Theorem 1. In Table 1 we provide a choice of values of the constants for small values of d, together with corresponding value of $2-\varepsilon_d$.

Table 1. Example values of the constants together with the corresponding success probability

d	1
λ	4.0238224
κ	9
c	2.00197442
α	0.050203
$2-\varepsilon_d$	1.99991

d	2
λ	4.00009156
κ	17/2
c	2.00000763
α	0.01449
$2-\varepsilon_d$	1.9999999

d	3
λ	4.000000357628
κ	25/3
c	2.0000000298
α	0.0066225
$2-\varepsilon_d$	1.9999999999

d	4
λ	4.000000001397
κ	33/4
c	2.0000000001164
α	0.0037736
$2-\varepsilon_d$	1.9999999999996

d	5
λ	4.000000000005457
κ	41/5
c	2.0000000000004548
α	0.0024331
$2-\varepsilon_d$	1.999999999999999

d	6
λ	4.000000000000021316
κ	49/6
c	2.0000000000000017833
α	0.0016978
$2-\varepsilon_d$	1.999999999999999997

4 Conclusions

We have shown that the Maximum d-degenerate Induced Subgraph problem can be solved in time $(2-\varepsilon_d)^n n^{\mathcal{O}(1)}$ for any fixed $d \geq 1$. There are two natural questions arising from our work. First, can the algorithm be derandomized? Rules 2 and 3 can be easily transformed into appropriate branching rules, but we do not know how to handle Rule 1 without randomization.

Second, our constants ε_d are really tiny even for small values of d. This is mainly caused by two facts: the gain over a straightforward brute-force algorithm in Rule 2 is very small (i.e., $\gamma(\lfloor \kappa d \rfloor)$ is very close to 2) and the algorithm falls back to Rule 4 after processing only a tiny fraction α of the entire graph. Can the running time of the algorithm be significantly improved? Another interesting question would be to investigate, whether the Maximum d-degenerate

Induced Subgraph problem can be solved in time $(2-\varepsilon)^n n^{\mathcal{O}(1)}$ for some universal constant ε that is independent of d.

Apart from the above questions, we would like to state here a significantly more challenging goal. Let $\mathcal{G}$ be a polynomially recognizable graph class of bounded degeneracy (i.e., there exists a constant d such that each $G \in \mathcal{G}$ is d-degenerate). Can the corresponding Maximum Induced $\mathcal{G}$-Subgraph problem be solved in $(2-\varepsilon_{\mathcal{G}})^n$ time for some constant $\varepsilon_{\mathcal{G}} > 0$ that depends only on the class $\mathcal{G}$? Can we prove some meta-result for such type of problems?

Our Rules 1 and 3 are valid for any such class $\mathcal{G}$; however, this is not true for the greedy step in Rule 2. In particular, we do not know how to handle the Maximum Induced $\mathcal{G}$-Subgraph problem faster than 2^n even if the input is assumed to be d-degenerate.

Acknowledgements. We would like to thank Marek Cygan, Fedor V. Fomin and Pim van 't Hof for helpful discussions.

References

1. Fomin, F.V., Grandoni, F., Kratsch, D.: A measure & conquer approach for the analysis of exact algorithms. J. ACM 56(5), 1–32 (2009)
2. van Rooij, J.M.M., Nederlof, J., van Dijk, T.C.: Inclusion/Exclusion Meets Measure and Conquer. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 554–565. Springer, Heidelberg (2009)
3. Cygan, M., Pilipczuk, M.: Exact and approximate bandwidth. Theor. Comput. Sci. 411(40-42), 3701–3713 (2010)
4. Björklund, A.: Determinant sums for undirected hamiltonicity. In: 51th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 173–182. IEEE Computer Society (2010)
5. Fomin, F.V., Todinca, I., Villanger, Y.: Exact Algorithm for the Maximum Induced Planar Subgraph Problem. In: Demetrescu, C., Halldórsson, M.M. (eds.) ESA 2011. LNCS, vol. 6942, pp. 287–298. Springer, Heidelberg (2011)
6. Cygan, M., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: Scheduling Partially Ordered Jobs Faster Than 2^n. In: Demetrescu, C., Halldórsson, M.M. (eds.) ESA 2011. LNCS, vol. 6942, pp. 299–310. Springer, Heidelberg (2011)
7. Cygan, M., Pilipczuk, M., Wojtaszczyk, J.O.: Capacitated Domination Faster Than $O(2^n)$. In: Kaplan, H. (ed.) SWAT 2010. LNCS, vol. 6139, pp. 74–80. Springer, Heidelberg (2010)
8. Binkele-Raible, D., Brankovic, L., Cygan, M., Fernau, H., Kneis, J., Kratsch, D., Langer, A., Liedloff, M., Pilipczuk, M., Rossmanith, P., Wojtaszczyk, J.O.: Breaking the 2^n-barrier for irredundance: Two lines of attack. J. Discrete Algorithms 9(3), 214–230 (2011)
9. Cygan, M., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: Solving the 2-Disjoint Connected Subgraphs Problem Faster Than 2^n. In: Fernández-Baca, D. (ed.) LATIN 2012. LNCS, vol. 7256, pp. 195–206. Springer, Heidelberg (2012)
10. Fomin, F.V., Heggernes, P., Kratsch, D., Papadopoulos, C., Villanger, Y.: Enumerating Minimal Subset Feedback Vertex Sets. In: Dehne, F., Iacono, J., Sack, J.-R. (eds.) WADS 2011. LNCS, vol. 6844, pp. 399–410. Springer, Heidelberg (2011)

11. Razgon, I.: Exact Computation of Maximum Induced Forest. In: Arge, L., Freivalds, R. (eds.) SWAT 2006. LNCS, vol. 4059, pp. 160–171. Springer, Heidelberg (2006)
12. Fomin, F.V., Gaspers, S., Pyatkin, A.V., Razgon, I.: On the minimum feedback vertex set problem: Exact and enumeration algorithms. Algorithmica 52(2), 293–307 (2008)
13. Impagliazzo, R., Paturi, R.: On the complexity of k-SAT. J. Comput. Syst. Sci. 62(2), 367–375 (2001)
14. Calabro, C., Impagliazzo, R., Paturi, R.: The Complexity of Satisfiability of Small Depth Circuits. In: Chen, J., Fomin, F.V. (eds.) IWPEC 2009. LNCS, vol. 5917, pp. 75–85. Springer, Heidelberg (2009)
15. Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., van Rooij, J.M.M., Wojtaszczyk, J.O.: Solving connectivity problems parameterized by treewidth in single exponential time. In: Ostrovsky, R. (ed.) FOCS, pp. 150–159. IEEE (2011)
16. Lokshtanov, D., Marx, D., Saurabh, S.: Known Algorithms on Graphs of Bounded Treewidth are Probably Optimal. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 777–789 (2011)
17. Pătraşcu, M., Williams, R.: On the possibility of faster SAT algorithms. In: Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1065–1075 (2010)
18. Cygan, M., Dell, H., Lokshtanov, D., Marx, D., Nederlof, J., Okamoto, Y., Paturi, R., Saurabh, S., Wahlström, M.: On problems as hard as CNFSAT. CoRR abs/1112.2275 (2011)
19. Gupta, S., Raman, V., Saurabh, S.: Fast Exponential Algorithms for Maximum r-Regular Induced Subgraph Problems. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 139–151. Springer, Heidelberg (2006)
20. Fomin, F.V., Villanger, Y.: Finding induced subgraphs via minimal triangulations. In: Marion, J.Y., Schwentick, T. (eds.) STACS. LIPIcs, vol. 5, pp. 383–394. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2010)
21. Angelsmark, O., Thapper, J.: Partitioning Based Algorithms for Some Colouring Problems. In: Hnich, B., Carlsson, M., Fages, F., Rossi, F. (eds.) CSCLP 2005. LNCS (LNAI), vol. 3978, pp. 44–58. Springer, Heidelberg (2006)
22. Gaspers, S., Kratsch, D., Liedloff, M.: On Independent Sets and Bicliques in Graphs. In: Broersma, H., Erlebach, T., Friedetzky, T., Paulusma, D. (eds.) WG 2008. LNCS, vol. 5344, pp. 171–182. Springer, Heidelberg (2008)
23. Gaspers, S.: Exponential Time Algorithms: Structures, Measures, and Bounds. PhD Thesis, University of Bergen (2008)
24. Cygan, M., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: Kernelization Hardness of Connectivity Problems in d-Degenerate Graphs. In: Thilikos, D.M. (ed.) WG 2010. LNCS, vol. 6410, pp. 147–158. Springer, Heidelberg (2010)
25. Kostochka, A.V.: Lower bound of the hadwiger number of graphs by their average degree. Combinatorica 4(4), 307–316 (1984)
26. Thomason, A.: An extremal function for contractions of graphs. Math. Proc. Cambridge Philos. Soc. 95(2), 261–265 (1984)
27. Thomason, A.: The extremal function for complete minors. J. Comb. Theory, Ser. B 81(2), 318–338 (2001)