Planar k-Path in Subexponential Time and Polynomial Space

Daniel Lokshtanov[1], Matthias Mnich[2], and Saket Saurabh[3]

[1] University of California, San Diego, USA, daniello@ii.uib.no
[2] International Computer Science Institute, Berkeley, USA, mmnich@icsi.berkeley.edu
[3] The Institute of Mathematical Sciences, Chennai, India, saket@imsc.res.in

Abstract. In the k-Path problem we are given an n-vertex graph G together with an integer k and asked whether G contains a path of length k as a subgraph. We give the first subexponential time, polynomial space parameterized algorithm for k-Path on planar graphs, and more generally, on H-minor-free graphs. The running time of our algorithm is $O(2^{O(\sqrt{k}\log^2 k)}n^{O(1)})$.

1 Introduction

In the k-Path problem we are given a n-vertex graph G and integer k and asked whether G contains a path of length k as a subgraph. The problem is a generalization of the classical Hamiltonian Path problem, which is known to be NP-complete [16] even when restricted to planar graphs [15]. On the other hand k-Path is known to admit a subexponential time parameterized algorithm when the input is restricted to planar, or more generally H-minor free graphs [4]. For the case of k-Path a subexponential time parameterized algorithm means an algorithm with running time $2^{o(k)}n^{O(1)}$. More generally, in parameterized complexity problem instances come equipped with a *parameter* k and a problem is said to be *fixed parameter tractable* (FPT) if there is an algorithm for the problem that runs in time $f(k)n^{O(1)}$. The algorithm is said to be a subexponential time parameterized algorithm if $f(k) \leq 2^{o(k)}$. For an introduction to parameterized algorithms and complexity see the textbooks [10, 11, 22].

In this paper we pose the following question. Does k-Path on planar graphs admit a subexponential time parameterized algorithm which only uses space polynomial in n? We give a positive answer to this question by presenting a polynomial space, $2^{O(\sqrt{k}\log^2 k)}n^{O(1)}$ time algorithm for k-Path restricted to planar graphs. Our algorithm easily generalizes to any family of graphs which exclude a fixed graph H as a minor.

The fastest parameterized algorithm for k-Path on planar graphs runs in $2^{O(\sqrt{k})}n^{O(1)}$ time and uses $2^{O(\sqrt{k})}\log n + n^{O(1)}$ space [9]. The algorithm we present uses polynomial space, but is slower by a factor of $O(\log^2 k)$ in the exponent of 2. Is the trade-off worth it? In general, does it make sense to settle for slightly slower algorithms if used space is reduced drastically? We believe

that such a trade-off is reasonable because algorithms that use exponential time and space tend to run out of space long before they run out of time. In the survey paper on exponential time algorithms, Woeginger [24] states that "algorithms with exponential space complexities are absolutely useless for real life applications". This line of reasoning has opened up an interesting research direction; for which problems can we obtain space-efficient algorithms that are (almost) as time-efficient as the fastest ones? Some progress has been made – Fomin, Grandoni and Kratsch [12] gave a $6^k n^{O(\log k)}$ time, polynomial space algorithm for the Steiner Tree problem and showed how to use it to obtain a $O(1.60^n)$ time polynomial space algorithm. In a breakthrough paper, Nederlof [21] gave a $2^k n^{O(1)}$ time polynomial space algorithm for Steiner Tree. Subsequently, Lokshtanov and Nederlof [20] devised general sufficient conditions for turning exponential space dynamic programming algorithms into polynomial space algorithms based on algebraic transforms.

It is natural to ask which problems admit polynomial space FPT algorithms. While this might look like a whole new research program, the question has a surprising answer; *any* problem for which a polynomial space FPT algorithm is remotely feasible has one. In particular, a necessary condition for a problem to have a polynomial space FPT algorithm is to have an FPT algorithm. Another necessary condition is that the problem is solvable in polynomial space and time $n^{g(k)}$ for some function g (that is, a polynomial space XP algorithm for the problem). A well-known trick from parameterized complexity shows that these two conditions are not only necessary, but also sufficient.

Theorem 1. *If a parameterized problem Π has an algorithm $\mathcal{A}$ which uses $f(k)n^c$ time and space, and an algorithm $\mathcal{B}$ which uses $n^{g(k)}$ time and polynomial space, then Π can be solved in $n^{c+1} + f(k)^{g(k)}$ time and polynomial space.*

Proof. If $n \geq f(k)$ then run algorithm $\mathcal{A}$, this takes polynomial time and space. If $n \leq f(k)$ then run algorithm $\mathcal{B}$, this takes polynomial space and $f(k)^{g(k)}$ time. □

Because of Theorem 1 the right question to ask is not which problems admit polynomial space FPT algorithms, but rather which problems admit *fast* polynomial space FPT algorithms. Obvious candidates for scrutiny are problems for which the fastest parameterized algorithms require exponential space. Graph problems restricted to planar and H-minor free graphs have this property – the Bidimensionality theory of Demaine et al. [4] gives $2^{O(\sqrt{k})}n^{O(1)}$ or $2^{O(\sqrt{k}\log k)}n^{O(1)}$ time algorithms for a multitude of graph problems on H-minor free graphs. However, these algorithms crucially depend on exponential space dynamic programming algorithms on graphs of bounded treewidth. In particular, the crux of all bidimiensionality based algorithms is to first bound the treewidth of the input graph by $t = O(\sqrt{k})$, and then solve the problems in $2^{O(t)}n$ (or $t^{O(t)}n$) time and $2^{O(t)}\log n$ (or $t^{O(t)}\log n$) space. We refer to following surveys for further details on the the Bidimensionality theory and its several applications [5, 8].

Interestingly, using another simple trick we can make most of the algorithms for bidimensional problems on H-minor free graphs run in polynomial space, at the cost of a $O(\log k)$ factor in the exponent of the running time. The trick has two components, the first is that (almost) all of the $2^{O(t)}n$ time and $2^{O(t \log t)}n$ time dynamic programming algorithms on graphs of treewidth t can be turned into polynomial space divide and conquer algorithms with running time $n^{O(t)}$ and $n^{O(t \log t)}$. The second component is that most problems for which the Bidimensionality theory of Demaine et al. [4] gives fast algorithms admit *linear kernels* on planar and H-minor-free graphs [3, 14]. A linear kernel for a parameterized graph problem is a polynomial time pre-processing algorithm that takes instances (G, k) and transforms them into equivalent instances (G', k') of the same problem such that $k' = O(k)$ and $V(G') = O(k)$. Now the subexponential time algorithms of Demaine et al. [4] can be made to run in polynomial space as follows. Run the pre-processing algorithm first, to ensure that the number of vertices in the input graph is at most $O(k)$. Bound the treewidth by $t = O(\sqrt{k})$ as before, but replace the $2^{O(t)}n$ (or $t^{O(t)}n$) time algorithms by $n^{O(t)}$ (or $n^{O(t \log t)}$) time polynomial space algorithms. Since $n = O(k)$ this second step uses only $2^{O(\sqrt{k} \log k)}$ (or $2^{O(\sqrt{k} \log^2 k)}$) time respectively. We remark that for many of these problems one can even remove the $\log k$ overhead in the exponent by using a Lipton-Tarjan separator approach [18, 19] instead of the polynomial space algorithms for graphs of bounded treewidth after obtaining the linear kernel for the problem, as done in [1].

In the above argument it was crucial that the problem considered admits a linear kernel. What about the problems that do not? Notably, the k-Path problem does have a subexponential time parameterized algorithm on H-minor-free graphs. At the same time k-Path does not admit a linear (or polynomial size) kernel unless the polynomial hierarchy collapses to the third level [2], even when the input is restricted to planar graphs. Thus, with respect to parameterized polynomial space, subexponential time algorithms for planar graph problems, k-Path stands out as a blank spot in an almost chartered map. Our polynomial space, $2^{O(\sqrt{k} \log^2 k)}n^{O(1)}$ time algorithm for k-Path on H-minor free graphs fills this gap. Our approach for obtaining polynomial space subexponential time algorithm for k-Path seems applicable to other problems also that do not admit polynomial kernels or not known to admit a polynomial kernel.

2 Definitions and Notations

In this section we give various definitions which we make use of in the paper. Let G be a graph then we use $V(G)$ and $E(G)$ to denote its vertex set and the edge set respectively. A graph G' is a *subgraph* of G if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. The subgraph G' is called an *induced subgraph* of G if $E(G') = \{uv \in E(G) \mid u, v \in V(G')\}$, in this case, G' is also called the subgraph *induced by* V' and denoted with $G[V']$. By $N(u)$ we denote the (open) neighborhood of u, that is, the set of all vertices adjacent to u. The closed neighbourhood of u is

$N[u] = N(u) \cup \{u\}$. Similarly, for a subset $D \subseteq V$, we define $N[D] = \bigcup_{v \in D} N[v]$ and $N(D) = N[D] \setminus D$.

Parameterized algorithms and Treewidth. A parameterized problem Π is a subset of $\Gamma^* \times \mathbb{N}$ for some finite alphabet Γ. An instance of a parameterized problem consists of (x, k), where k is called the parameter. A central notion in parameterized complexity is *fixed parameter tractability (FPT)* which means, for a given instance (x, k), solvability in time $f(k) \cdot p(|x|)$, where f is an arbitrary function of k and p is a polynomial in the input size.

A *tree decomposition* of a graph G is a pair $(T, \mathcal{B})$ where T is a tree and $\mathcal{B} = \{X_i \mid i \in V(T)\}$ is a collection of subsets of $V(G)$ such that

1. $\bigcup_{i \in V(T)} X_i = V(G)$,
2. for each edge $xy \in E$, $\{x, y\} \subseteq X_i$ for some $i \in V(T)$;
3. for each $x \in V(G)$ the set $\{i \mid x \in X_i\}$ induces a connected subtree of T.

The *width* of the tree decomposition is $\max_{i \in V_T} |X_i| - 1$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G. A tree decomposition $(T, \mathcal{B})$ can be converted in linear time [17] into a *nice* tree decomposition of the same width: here, the tree T is rooted and binary, and its nodes are of four types:

- *Leaf nodes* h are leaves of T and have $|X_h| = 1$.
- *Introduce nodes* h have one child i with $X_h = X_i \cup \{v\}$ for some vertex $v \in V$.
- *Forget nodes* h have one child i with $X_h = X_i \setminus \{v\}$ for some vertex $v \in V$.
- *Join nodes* h have two children i, j with $X_h = X_i = X_j$.

We denote by $\mathbf{tw}(G)$ the treewidth of the graph G.

3 Polynomial Space Algorithm for the k-Path problem

In this section we prove our main result, which is a polynomial space subexponential time algorithm for the k-Path problem. For a set $W \subseteq V(G)$ a set $S \subseteq V(G)$ is a *balanced separator* for W if $V(G)$ can be partitioned into L, S and R such that there is no edge from L to R and $|W \cap L| \leq \frac{2|W \setminus S|}{3}$ and $|W \cap R| \leq \frac{2|W \setminus S|}{3}$. In other words, W is evenly distributed between L and R. It is well-known that in any *tree* T, for every set $W \subseteq V(G)$ there is a balanced separator S for W with $|S| = 1$. This result has been generalized to graphs of bounded treewidth [23][11, Lemma 11.16] - in particular in a graph G of treewidth at most t, for any set W there is a balanced separator S of size at most $t + 1$. Lemma 1 is a subtle strengthening of this fact and states that given any tree-decomposition of G of width at most t the separator S can be chosen as one of the bags of the decomposition. In fact, the proofs given in [23][11, Lemma 11.16] already imply Lemma 1, we include a proof here for completeness.

For a graph G and a nice tree-decomposition $(T, \mathcal{B})$ of G and node $v \in V(T)$ let $X_v \in \mathcal{B}$ be the corresponding bag. Let T_v be the subtree of T rooted at v and let $A(v) = (\bigcup_{u \in V(T_v)} X_u) \setminus X_v$.

Lemma 1. *Let G be a graph, let $(T, \mathcal{B})$ be a nice tree-decomposition of G of width t and let $W \subseteq V$ be a vertex set of size at least 3. Then there exists a vertex v such that X_v is a balanced separator for W and in the corresponding partition $V(G) = L \cup X_v \cup R$, $L = A(v)$.*

Proof. Recall that T is a rooted tree with root r, and that each node of T has at most two children. Observe that $|A(r) \cap W| = |W \setminus X_r| \geq \frac{|W \setminus X_r|}{3}$. Choose a lowermost node v such that $|A(v) \cap W| \geq \frac{|W \setminus X_v|}{3}$. We prove that $|A(v) \cap W| \leq \frac{2|W \setminus X_v|}{3}$. If v is an introduce node or a forget node with child v' then $|A(v') \cap W| \leq \frac{|W \setminus X_{v'}|}{3}$ and hence $|A(v) \cap W| \leq \frac{2|W \setminus X_v|}{3}$. Here we used that $|W| \geq 3$ and that $|X_v| = |X_{v'}| + 1$ if v is an introduce node, and that $|A(v)| = |A(v')| + 1$ if v is a forget node. If v is a join node with children u and w then $X_v = X_u = X_w$ and $A(v) = A(u) \cup A(w)$. Finally $|A(u) \cap W| \leq \frac{|W \setminus X_u|}{3}$ and $|A(w) \cap W| \leq \frac{|W \setminus X_w|}{3}$ and hence $|A(v) \cap W| \leq \frac{2|W \setminus X_v|}{3}$. This concludes the proof. □

A key component of our algorithm is a new divide and conquer algorithm for the k-Path problem on graphs of bounded treewidth.

Lemma 2. *There is an $(nt^t)^{O(\log k)}$ time and polynomial space algorithm for k-Path if a nice tree-decomposition $(T, \mathcal{B})$ of G of width t is given as input.*

Proof. We describe a divide and conquer algorithm for the problem. In order to handle the instances that are generated in the recursive steps of the algorithm we will solve a slightly more general problem. In the *generalized* k-Path problem (k-GP) we are given a graph G, integer k, vertex set V_p and an edge set E_p such that every edge in E_p has both endpoints in V_p. The task is to determine whether there is a path P in G such that $V_p \subseteq V(P)$, $E_p \subseteq E(P)$ and $|V(P) \setminus V_p| = k$. One can think of the sets V_p and E_p as vertices and edges which are pre-determined to be in the path P. We are now ready to give an algorithm for k-GP in graphs of bounded tree-width with running time $O((nk^2(t+1)!3^{t+2})^{\log k + O(1)} \cdot (V_p + t \log k)!)$. Suppose that there is a path P such that $V_p \subseteq V(P)$, $E_p \subseteq E(P)$ and $|V(P) \setminus V_p| = k$. Let $W = V(P) \setminus V_p$, with $|W| = k$. If $k = 0$ the algorithm tries all the $|V_p|!$ possible orderings of V_p and checks whether any of the orderings is a path that contains all edges of E_p. If $k < 3$ the algorithm tries all possible ways to extend V_p by k vertices, and then proceeds to the case when $k = 0$. We now handle the case that $k \geq 3$.

By Lemma 1 there exists a node $v \in V(T)$ such that $X_v \in \mathcal{B}$ is a balanced separator for W. The algorithm guesses the correct vertex v by looping over all the n possible bags in the decomposition. Furthermore, by Lemma 1 there is a partition of $V(G)$ into $L \cup X_v \cup R$ such that $L = A(v)$ and $R = V(G) \setminus (X_v \cup R)$, there is no edge from L to R, $|W \cap L| \leq \frac{2|W \setminus X_v|}{3}$ and $|W \cap R| \leq \frac{2|W \setminus X_v|}{3}$. The algorithm computes L and R from v. Now the algorithm guesses $V(P) \cap X_v = X$ by trying all the 2^{t+1} possible subsets of X_v.

The path P visits the vertices of X in some order, say $x_1, x_2, \ldots x_q$. The algorithm guesses this order by trying all the (at most) $(t+1)!$ possible permutations of X. Now, for each $i < q$ the subpath of P from x_i to x_{i+1} either uses

the edge x_ix_{i+1}, or has all its inner vertices in L, or has all its inner vertices in R. By trying each of the three possibilities for each $i < q$ the algorithm correctly guesses which of the possibilities it is. The algorithm also guesses whether the subpath of P attached to x_1 lies in its entirety in $L \cap X$ or in $R \cap X$. The same guess is performed for the subpath of P attached to x_q. Finally the algorithm guesses $k_L = |(V(P) \setminus V_p) \cap L|$ and $k_R = |(V(P) \setminus V_p) \cap R|$.

Using all the guesses the algorithm constructs two instances G_L, V_p^L, E_p^L, k_L and $G_R, V_p^R, E_p^R, L, k_R$ as follows. We set $V_p^L = (V_p \cap L) \cup X$ and $E_p^L = E_p \cap E(G[L \cup X])$. To construct G_L we start with $G[L \cup X]$ and remove all the edges with both endpoints in X. For every pair x_i, x_{i+1} such that we have guessed that the subpath of P from x_i to x_{i+1} uses the edge x_ix_{i+1} *or* has all its internal vertices in R, we add the edge x_ix_{i+1} to G_L and to E_p^L. Finally, if we have guessed that the subpath of P from the start point until x_1 lies entirely in $X \cup R$ we add a vertex p_{start} to G_L and to V_p^L, make p_{start} adjacent to x_1 and add p_1p_{start} to E_p^L. Similarly if we have guessed that the subpath of P from x_q to the end point of P lies entirely in $X \cup R$ we add a vertex p_{end} to G_L and to V_p^L, and add the edge x_qp_{end} to G_L and to E_p^L. The graph G_R and set V_p^R is constructed symmetrically.

If the two instances G_L, V_p^L, E_p^L, k_L and G_R, V_p^R, E_p^R, k_R are both "yes" instances one can glue their solution paths together to form a solution path for G, V_p, E_p, k. In particular every edge x_ix_{i+1} in G_L will correspond either to an edge x_ix_{i+1} in the solution paths of both G_L, V_p^L, E_p^L, k_L and G_R, V_p^R, E_p^R, k_R, or it can be replaced by a subpath of the solution path of G_R, V_p^R, E_p^R, k_R. Edges x_ix_{i+1} in G_R are handled symmetrically. In the reverse direction for the correct set of guesses the path P breaks up into solution paths to G_L, V_p^L, E_p^L, k_L and G_R, V_p^R, E_p^R, k_R respectively. On the side that contains p_{start} we add the edge $p_{start}x_1$ to the solution path, and on the side that contains p_{end} we add the edge x_qp_{end}.

Now we bound the running time of the algorithm. Observe that since we only add edges between vertices in $X \subseteq X_v$ and pendant vertices p_{start} and p_{end} of degree 1 attached to X, the treewidth of G_L and G_R is at most t. Let $r = |V_p|$, and let $T(k, r, n, t)$ be a function that upper bounds the running time of the algorithm. The function T is bounded by the following recurrence.

$$T(k, r, n, t) \leq n \cdot (t+1)! \cdot 3^t \cdot 2^{t+3} \cdot k^2 \cdot 2T(k/2, r+t, n, t) \text{ when } k \geq 3$$

Here the factors in the recurrence reflect the number of possibilities for each guess, except for the factor 2 in $2T(k/2, n, t)$ which reflects that G_L and G_R are handled independently. Observe that n and t never increase throughout the recurrence. When $k < 3$ we have that $T(k, r, n, t) \leq O(n^3 r!)$. Hence $T(k, r, n, t)$ can be bounded from above by

$$T(k, r, n, t) \leq (nk^2(t+1)!6^{t+3})^{\log k} \cdot n^3(t \log k)! \leq (nt^t)^{O(\log k)}.$$

The space requirement of the algorithm is clearly polynomial. This concludes the proof. □

We are now in position to give the main result of this section.

Theorem 2. *For every fixed graph H, there is an algorithm for k-Path on H-minor-free graphs running in time $2^{O(\sqrt{k}\log^2 k)}n^{O(1)}$ and using polynomial space.*

Proof. We use the fact that for any H there exists a constant h such that in any H-minor-free graph of treewidth at least hk there is a k-path of length at least k^2 [4]. Set $t = h\sqrt{k}$, then if $\mathbf{tw}(G) \geq t$ then G contains a k-path. We use the approximation algorithm of Diestel et al [6] to either compute a tree-decomposition of G of width at most $3t/2$, or to conclude that the treewidth of G is at least t, which implies that G has a k-path. Diestel et al's algorithm uses polynomial space and $2^{O(t)}n^{O(1)}$ time. If we obtain a tree-decomposition we proceed as follows. If $n \leq 2^{\sqrt{k}}$ apply the algorithm from Lemma 2 to solve the problem in time $2^{O(\sqrt{k}\log^2 k)}n^{O(1)}$ and using polynomial space. If, on the other hand $n \geq 2^{\sqrt{k}}$ the standard dynamic programming algorithm on graphs of bounded treewidth that uses at most $2^{O(\sqrt{k})}n^{O(1)}$ time and space [7], runs in polynomial time and space. This concludes the proof. □

We remark that the algorithm presented in Theorem 2 for k-Path can be made to run in time $2^{O(\sqrt{k}\log k)}n^{O(1)}$ and space polynomial in n on planar graphs using sphere cut decompositions introduced by Dorn et al. [9].

4 Conclusion and Discussion

We gave a subexponential time, polynomial space parameterized algorithm for the k-Path problem on H-minor-free graphs. A key component of our algorithm is a new $(n \cdot t^t)^{O(\log k)}$ time and polynomial space algorithm for k-Path in graphs of treewidth at most t. In general, it is possible to design similar $(n \cdot 2^t)^{O(\log k)}$ or $(n \cdot t^t)^{O(\log k)}$ time and polynomial space algorithms for many problems where one is looking for a specific vertex set of size k in a graph of treewidth t. A concrete example where this is useful is the k-Partial Vertex Cover problem. Here we are given a graph G, positive integers k and t and we look for a subset $S \subseteq V(G)$ such that $|S| \leq k$ and the number of edges incident to S is at least t. This problem is not known to admit a polynomial kernel, even on planar graphs. However using the approach described in this paper for k-Path and combining it with an algorithm of Fomin et al. [13] that in polynomial time either finds a solution for an instance (G, k, t) or obtains an equivalent instance (G', k, t) such that $\mathbf{tw}(G') \leq O(\sqrt{k})$, one can give a subexponential time, polynomial space parameterized algorithm for k-Partial Vertex Cover on apex-minor-free graphs.

We conclude with two open problems. First, is there a polynomial space parameterized algorithm for the k-Path problem on planar graphs with running time $2^{O(\sqrt{k})}n^{O(1)}$? Second, by combining the well known $2^t n^{O(1)}$ time and space algorithm for Independent Set in graphs of treewidth t, and the folklore $n^{O(t)}$ time, polynomial space algorithm, Theorem 1 yields a $2^{O(t^2)} + n^2$ time and

polynomial space algorithm for Independent Set. Is there a polynomial space algorithm for Independent Set on graphs of treewidth t with running time $2^{t^{2-\epsilon}}n^{O(1)}$ for some $\epsilon > 0$?.

References

1. J. Alber, H. Fernau, and R. Niedermeier. Graph separators: a parameterized view. *J. Comput. System Sci.*, 67(4):808–832, 2003. Special issue on parameterized computation and complexity.
2. H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *J. Comput. System Sci.*, 75(8):423–434, 2009.
3. H. L. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos. (Meta) Kernelization. In *Proc. 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 629–638. IEEE Computer Society, 2009.
4. E. D. Demaine, F. V. Fomin, M. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, 52(6):866–893 (electronic), 2005.
5. E. D. Demaine and M. Hajiaghayi. The bidimensionality theory and its algorithmic applications. *Comput. J.*, 51(3):292–302, 2008.
6. R. Diestel, T. R. Jensen, K. Y. Gorbunov, and C. Thomassen. Highly connected sets and the excluded grid theorem. *J. Comb. Theory, Ser. B*, 75(1):61–73, 1999.
7. F. Dorn, F. V. Fomin, and D. M. Thilikos. Catalan structures and dynamic programming in H-minor-free graphs. In *Proc. 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 631–640. ACM.
8. F. Dorn, F. V. Fomin, and D. M. Thilikos. Subexponential parameterized algorithms. *Computer Science Review*, 2(1):29–39, 2008.
9. F. Dorn, E. Penninkx, H. L. Bodlaender, and F. V. Fomin. Efficient exact algorithms on planar graphs: exploiting sphere cut branch decompositions. In *Algorithms*, volume 3669 of *Lecture Notes in Comput. Sci.*, pages 95–106. Springer.
10. R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Monographs in Computer Science. Springer-Verlag.
11. J. Flum and M. Grohe. *Parameterized Complexity Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag.
12. F. V. Fomin, F. Grandoni, and D. Kratsch. Faster steiner tree computation in polynomial-space. In *ESA*, volume 5193 of *Lecture Notes in Computer Science*, pages 430–441, 2008.
13. F. V. Fomin, D. Lokshtanov, V. Raman, and S. Saurabh. Subexponential algorithms for partial cover problems. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 4 of *Leibniz International Proc. Informatics*, pages 193–201. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2009.
14. F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Bidimensionality and kernels. In *Proc. 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 503–510. Society for Industrial and Applied Mathematics, 2010.
15. M. R. Garey, D. S. Johnson, and R. E. Tarjan. The planar hamiltonian circuit problem is np-complete. *SIAM J. Comput.*, 5(4):704–714, 1976.
16. R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

17. T. Kloks. *Treewidth. Computations and Approximations.*, volume 842 of *Lecture Notes in Comput. Sci.* Springer, 1994.
18. R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36(2):177–189, 1979.
19. R. J. Lipton and R. E. Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980.
20. D. Lokshtanov and J. Nederlof. Saving space by algebraization. In *STOC*, pages 321–330, 2010.
21. J. Nederlof. Fast polynomial-space algorithms using möbius inversion: Improving on steiner tree and related problems. In *ICALP (1)*, volume 5555 of *Lecture Notes in Computer Science*, pages 713–725, 2009.
22. R. Niedermeier. *Invitation to Fixed-parameter Algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, 2006.
23. B. A. Reed. Tree width and tangles: a new connectivity measure and some applications. In *Surveys in Combinatorics, 1997 (London)*, volume 241 of *London Math. Soc. Lecture Note Ser.*, pages 87–162. Cambridge Univ. Press, 1997.
24. G. J. Woeginger. Space and time complexity of exact algorithms: Some open problems. In *Parameterized and Exact Computation*, volume 3162 of *Lecture Notes in Comput. Sci.*, pages 281–290. Springer, 2004.