# Faster deterministic Feedback Vertex Set

Tomasz Kociumaka* Marcin Pilipczuk†

**Abstract**

We present a new deterministic algorithm for the Feedback Vertex Set problem parameterized by the solution size. Our algorithm runs in $\mathcal{O}^*((2+\phi)^k)$ time, where $\phi<1.619$ is the golden ratio, surpassing the previously fastest $\mathcal{O}^*((1+2\sqrt{2})^k)$-time deterministic algorithm due to Cao et al. [SWAT 2010]. In our development we follow the approach of Cao et al.; however, thanks to a new reduction rule, we obtain not only better dependency on the parameter in the running time, but also a solution with simple analysis and only a single branching rule.

## 1 Introduction

The Feedback Vertex Set problem (FVS for short), where we ask to delete as few vertices as possible from a given undirected graph to make it acyclic, is one of the fundamental graph problems, appearing on Karp's list of 21 NP-hard problems [20]. It is also one of the most-studied problems in parameterized complexity, and there has been a long 'race' for the fastest FPT algorithm parameterized by the solution size, denoted by $k$, [2, 15, 16, 23, 19, 14, 18, 7, 6, 1, 12]. The fastest known *deterministic* algorithm prior to this work, due to Cao et al. [6], runs in $\mathcal{O}^*((1+2\sqrt{2})^k)\leq\mathcal{O}^*(3.83^k)$ time[1]; if we allow randomization, the Cut&Count technique yields an $\mathcal{O}^*(3^k)$ time algorithm [12]. Related research investigates kernelization complexity of FVS [5, 4, 25] and other variants, e.g., in directed graphs [8, 13, 9].

In this work we claim the lead in the 'FPT race' for the fastest *deterministic* algorithm for FVS.

**Theorem 1.** Feedback Vertex Set, *parameterized by the solution size $k$, can be solved in $\mathcal{O}^*((2+\phi)^k)\leq\mathcal{O}^*(3.619^k)$ time and polynomial space where $\phi=\frac{1+\sqrt{5}}{2}<1.619$ is the golden ratio.*

In our developments, we closely follow the approach of the previously fastest algorithm due to Cao et al. [6]. That is, we first employ the iterative compression technique [24] in a standard manner to reduce the problem to the disjoint compression variant (Disjoint-FVS), where the vertex set is split into two parts, both inducing forests, and we are allowed to delete vertices only from the second part. Then we develop a set of reduction and branching rules to cope with this structuralized instance. We rely on the core observation of Cao et al. that the problem becomes polynomial-time solvable once the maximum degree of the deletable vertices drops to 3.

The main difference between our algorithm and the one of Cao et al. is the introduction of a new reduction rule that reduces deletable vertices with exactly one deletable neighbour and two undeletable ones. Branching on such vertices is the most costly operation in the $\mathcal{O}^*(5^k)$ time algorithm of Chen et al. [7] and avoiding such branching leads to three branching rules, with associated case analysis, in the algorithm of Cao et al. [6]. As a consequence of the new reduction rule, in our algorithm we only need a single straightforward branching rule. Thus, the new rule not only leads to a better time complexity, but also allows us to simplify the algorithm and analysis.

We also present a new and shorter proof of the main technical contribution of Cao et al., which says that Disjoint-FVS is polynomial-time solvable if all deletable vertices are of degree at most 3. Thus, apart from the better time complexity of our algorithm, we also simplify the arguments of Cao et al. [6].

### 1.1 Preliminaries and notation

All graphs in our work are undirected and, unless explicitly specified, simple. For a graph $G$, by $V(G)$ and $E(G)$ we denote its vertex- and edge-set, respectively. For $v\in V(G)$, the neighbourhood of $v$, denoted $N_G(v)$, is defined

*Institute of Informatics, University of Warsaw, Poland, kociumaka@mimuw.edu.pl
†Institute of Informatics, University of Warsaw, Poland, malcin@mimuw.edu.pl
[1]The $\mathcal{O}^*$-notation suppresses factors polynomial in the input size.

as $N_G(v)=\{u\in V(G):uv\in E(G)\}$. For a set $X\subseteq V(G)$, we denote the subgraph induced by $X$ by $G[X]$, and we use $G\setminus X$ to denote $G[V(G)\setminus X]$. For $X\subseteq V(G)$ and $v\in V(G)$ we define the $X$-degree of $v$, denoted by $\deg_X(v)$, as $|N_G(v)\cap X|$. Moreover, for $v\in X$ we say that $v$ is $X$-isolated if $\deg_X(v)=0$, and an $X$-leaf if $\deg_X(v)=1$.

If $e=uv$ is an edge in a (multi)graph $G$ such that $u\neq v$, by *contracting the edge* $e$ we mean the following operation resulting in a multigraph $G'$: we replace $u$ and $v$ with a new vertex $w$, and define $x'=w$ for $x\in\{u,v\}$ and $x'=x$ for $x\in V(G)\setminus\{u,v\}$. For each edge $xy\in E(G)\setminus\{e\}$, we add an edge $x'y'$ to $E(G')$. In other words, we do not suppress multiple edges and loops in the process of contraction. Moreover, there is a natural bijection between $E(G)\setminus\{e\}$ and $E(G')$. We abuse the notation and identify these edges. Note that, if $G$ is a simple graph and $N_G(u)\cap N_G(v)=\emptyset$, no loop nor multiple edge is introduced when contracting $uv$.

We say that a set $F\subseteq E(G)$ is *acyclic*, if it is the edge set of a forest in $G$. For an acyclic set $F\subseteq E(G)$ by *contracting* $F$ we mean a composition of subsequent contractions of edges in $F$, in arbitrary order. This operation is valid for every *acyclic* set $F\subseteq E(G)$ and the resulting multigraph $H$ does not depend on the order of edge contractions.

**Observation 2.** *Let $F\subseteq E(G)$ be an acyclic in a multigraph $G$, and let a multigraph $H$ be obtained from $G$ by contracting $F$. Then any set $F'\subseteq E(G)\setminus F$ is acyclic in $H$ if and only if $F'\cup F$ is acyclic in $G$.*

# 2 The algorithm

## 2.1 Iterative compression

Following the approach of Cao et al. [6], we employ the iterative compression technique [24] in a standard manner. Consider the following variant of FVS.

DISJOINT-FVS
**Input:** A graph $G$, a partition $V(G)=U\cup D$ such that both $G[U]$ and $G[D]$ are forests, and an integer $k$.
**Question:** Does there exist a set $X\subseteq D$ of size at most $k$ such that $G\setminus X$ is a forest?

A $D$-isolated vertex of degree 3 is called a *tent*. For a DISJOINT-FVS instance $I=(G,U,D,k)$ we define the following functions: $k(I)=k$, $\ell(I)$ is the number of connected components of $G[U]$, $t(I)$ is the number of tents in $I$, and $\mu(I)=k(I)+\ell(I)-t(I)$ is the *measure* of $I$. Note that our measure differs from the one used by Cao et al. For each function we omit the argument if the instance is clear from the context.

In the rest of the paper we focus on solving DISJOINT-FVS, and proving the following theorem.

**Theorem 3.** DISJOINT-FVS *on an instance $I$ can be solved in $\mathcal{O}^*(\phi^{\max(0,\mu(I))})$ time and polynomial space.*

For sake of completeness, we show how Theorem 3 implies Theorem 1.

*Proof of Theorem 1.* Assume we are given an FVS instance $(G,k)$. Let $v_1,\ldots,v_n$ be an arbitrary ordering of $V(G)$. Define $V_i=\{v_1,v_2,\ldots,v_i\}$, and $G_i=G[V_i]$; we iteratively solve FVS instances $(G_i,k)$ for $i=1,2,\ldots,n$. Clearly, if $(G_i,k)$ turns out to be a NO-instance for some $i$, $(G,k)$ is a NO-instance as well. On the other hand, $(G_i,k)$ is a trivial YES-instance for $i\leq k+1$.

To finish the proof we need to show how, given a solution $X_{i-1}$ to $(G_{i-1},k)$, to solve the instance $(G_i,k)$. Let $Z=X_{i-1}\cup\{v_i\}$ and $D=V_i\setminus Z=V_{i-1}\setminus X_{i-1}$. Clearly, $G[D]$ is a forest. We branch into $2^{|Z|}\leq 2^{k+1}$ subcases, guessing the intersection of the solution to $(G_i,k)$ with the set $Z$. In a branch where we guess $Y\subseteq Z$, we delete $Y$ from $G_i$ and disallow deleting vertices of $Z\setminus Y$. More formally, for any $Y\subseteq Z$ such that $G_i[Z\setminus Y]$ is a forest, we define $U=Z\setminus Y$ and apply the algorithm of Theorem 3 to the DISJOINT-FVS instance $I_Y=(G_i\setminus Y,U,D,k-|Y|)$. Clearly, $I_Y$ is a YES-instance of DISJOINT-FVS if and only if $(G_i,k)$ has a solution $X_i$ with $X_i\cap Z=Y$.

As for the running time, note that $\ell(I_Y)\leq|U|=|Z\setminus Y|\leq(k+1)-|Y|$. Hence, $\mu(I_Y)\leq 2(k-|Y|)+1$ and the total running time of solving $(G_i,k)$ is bounded by

$$\mathcal{O}^*\left(\sum_{Y\subseteq Z}\phi^{2(k-|Y|)+1}\right)=\mathcal{O}^*\big((1+\phi^2)^k\big)=\mathcal{O}^*\big((2+\phi)^k\big).$$

This completes the proof of Theorem 1. □

## 2.2 Reduction rules

Assume we are given a DISJOINT-FVS instance $I=(G,U,D,k)$. We first state the following three reduction rules, which in the current or slightly modified version were used in previous algorithms for FVS [6, 7]. At any time, we apply the lowest-numbered applicable rule first.

**Reduction Rule 1.** Remove all vertices of degree at most 1 from $G$.

**Reduction Rule 2.** If a vertex $v\in D$ has at least two neighbours in the same connected component of $G[U]$, delete $v$ and decrease $k$ by one.

**Reduction Rule 3.** If there exists a vertex $v\in D$ of degree 2 in $G$, move it to $U$ if it has a neighbour in $U$, or contract one of its incident edges otherwise.

We say that a rule is *safe* if the output instance is a YES-instance if and only if the input one is. For completeness, we now shortly elaborate on the safeness of Rules 1–3. For Rule 1, observe that no vertex of degree at most 1 in $G$ lies on a cycle in $G$. For Rule 2, note that if Rule 2 triggers on a vertex $v$, there exists a cycle in $G[U\cup\{v\}]$ and thus $v$ needs to be included in any solution to the instance $I$. For Rule 3, it suffices to prove that there exists a minimum solution to the DISJOINT-FVS instance $I$ that does not contain $v$. Let $X$ be a minimum solution to $I$ and assume $v\in X$. As $X$ is a minimum solution to $I$, $G\setminus(X\setminus\{v\})$ contains a cycle. As $v$ is of degree two in $G$, $G\setminus(X\setminus\{v\})$ contains exactly one simple cycle $C$, and $v$ lies on $C$. Since Rule 2 is not applicable, the two neighbours of $v$ in $G$ do not lie in the same connected component of $G[U]$ and, consequently, there exists a vertex $w\neq v$ on $C$ that belongs to $D$. Hence, $(X\setminus\{v\})\cup\{w\}$ is a minimum solution to $I$ that does not contain $v$, and Rule 3 is safe.

We note that we prefer to contract an edge in Reduction 3 in case when $N_G(v)\subseteq D$, instead of moving $v$ to $U$, to avoid an increase of the measure $\mu(I)$. As $G$ is simple and $G[D]$ is a forest, no multiple edge is introduced in such contraction and $G$ remains a simple graph.

It is easy to observe that Rules 1–3 are applicable in polynomial time. Let us now verify the following.

**Lemma 4.** *An application of any of the Rules 1–3 does not increase $\mu(I)$.*

*Proof.* Consider first an application of Rule 1 to a vertex $v$. If $v\in D$, $k(I)$, $\ell(I)$ and remain unchanged, and $t(I)$ does not decrease, as neither $v$ nor its sole neighbour is a tent in $G$. If $v\in U$, $k(I)$ remains unchanged and $\ell(I)$ does not increase. Note that $t(I)$ may decrease if the sole neighbour of $v$ is a tent. However, in this case $\ell(I)$ also drops by one, and $\mu(I)$ remains unchanged.

If Rule 2 is applied to $v$, $k(I)$ drops by one, $\ell(I)$ remains unchanged and $t(I)$ remains the same or drops by one, depending on whether $v$ is a tent or not.

If Rule 3 is applied to $v$, $k(I)$ remains unchanged, $t(I)$ does not decrease and $\ell(I)$ does not increase (thanks to the special case of $N_G(v)\subseteq D$). □

We now prove a lower bound on the measure $\mu(I)$ (cf. [6], Lemma 1):

**Lemma 5.** *Let $I$ be a* DISJOINT-FVS *instance. If $t(I)\geq k(I)+\frac{1}{2}\ell(I)$, then $I$ is a NO-instance.*

*Proof.* Let $I=(G,U,D,k)$ be an instance and let $T\subseteq D$ be its set of tents. Assume $I$ is a YES-instance and let $X$ be a solution. Then $G'=G[U\cup(T\setminus X)]$ is a forest. Consequently

$$|U|+|T\setminus X|=|V(G')|>|E(G')|\geq|E(G[U])|+3|T\setminus X|=|U|-\ell(I)+3|T\setminus X|,$$

and thus $\ell(I)>2|T\setminus X|$. As $|X|\leq k$, we get that $|T|\leq|X|+|T\setminus X|<k+\frac{1}{2}\ell(I)$ and the lemma follows. □

Consequently, we may apply the following rule.

**Reduction Rule 4.** If $\mu(I)\leq\frac{1}{2}\ell(I)$, conclude that $I$ is a NO-instance.

Note that Rule 4 triggers also when $\mu(I)\leq 0$.

We now introduce a new reduction rule, as promised in the introduction.

**Reduction Rule 5.** If $v$ is a $D$-leaf of $U$-degree 2 with $w$ being its only neighbour in $D$, replace the edge $vw$ with a new vertex $x_{vw}$ of degree 2, adjacent to $v$ and $w$ (i.e., subdivide the edge $vw$) and insert the newly created vertex $x_{vw}$ to $U$.

First, we note that Rule 5 is safe: introducing an undeletable degree-2 vertex in a middle of an edge does not change the set of feasible solutions to DISJOINT-FVS. Second, note that Rule 5 does not increase the measure of the instance: although the newly added vertex creates a new connected component in $G[U]$, increasing $\ell(I)$ by one, at the same time $v$ becomes a tent and $t(I)$ increases by at least one ($w$ may become a tent as well). Third, one may be worried that Rule 5 in facts expands $G$, introducing a new vertex. However, it also increases the number of tents; as our algorithm never inserts a vertex to $D$, Rule 5 may be applied at most $|D|$ times.

## 2.3 The polynomial-time solvable case

Before we move to the branching rule, let us recall the polynomial-time solvable case of Cao et al. [6].

**Theorem 6** ([6]). *There exists a polynomial-time algorithm that solves a special case of* DISJOINT-FVS *where each vertex of $D$ is a tent.*

Cao et al. consider a seemingly more general case where each vertex of $D$ is of degree exactly three in $G$. It is easy to see that an exhaustive application of Rule 5 to such an instance results in an instance where each vertex of $D$ is a tent. This way, our new reduction rule lets us focus on more structured instances and significantly simplify the solution.

Theorem 6 allows us to state the last reduction rule.

**Reduction Rule 6.** If each vertex of $D$ is a tent, resolve the instance in polynomial time using the algorithm of Theorem 6.

Cao et al. [6] prove Theorem 6 by a reduction to the matroid parity problem in a cographic matroid. We present here a shorter proof, relying on the matroid parity problem in a graphic matroid of some contraction of $G$.

Given a (multi)graph $H$, the *graphic matroid*[2] of $H$ is a matroid with ground set $E(H)$ where a set $A \subseteq E(H)$ is independent if and only if $A$ is acyclic in $H$.

The *matroid parity problem* on the graphic matroid of a multigraph $H$ is defined as follows. In the input, apart from the multigraph $H$ with an even number of edges, we are given a partition of $E(H)$ into pairs; in other words, $|E(H)|=2m$ and $E(H)=\{e_1^1,e_1^2,e_2^1,e_2^2,\ldots,e_m^1,e_m^2\}$. We seek for a maximum set $J \subseteq \{1,2,\ldots,m\}$ such that $A(J):=\bigcup_{j\in J}\{e_j^1,e_j^2\}$ is independent in the graphic matroid of $H$, that is, acyclic in the multigraph $H$. The matroid parity problem in graphic matroids is polynomial-time solvable [17].

Having introduced the matroid parity problem in graphic matroids, we are ready to present a shorter proof of Theorem 6.

*Proof of Theorem 6.* Let $I=(G,D,U,k)$ be an instance of DISJOINT-FVS where each vertex of $D$ is a tent. For each $v \in D$ we arbitrarily enumerate the edges incident to $v$ as $e_v^0,e_v^1,e_v^2$. Define $F=E(G[U])\cup\{e_v^0 : v\in D\}$. Note that $F$ is acyclic, which lets us define $H$ as the multigraph obtained from $G$ by contracting $F$ (recall that we do not suppress the multiple edges and loops). Clearly, $E(H)=\{e_v^1,e_v^2 : v\in D\}$. Treat $H$, with pairs $\{e_v^1,e_v^2\}_{v\in D}$, as an input to the matroid parity problem in the graphic matroid of $H$.

Let $J \subseteq D$. We claim that $A(J)=\bigcup_{v\in J}\{e_v^1,e_v^2\}$ is independent in the graphic matroid of $H$ if and only if $G\setminus(D\setminus J)$ is a forest. Note this claim completes the proof of Theorem 6 as it implies that a maximum solution $J$ to the matroid parity problem corresponds to a minimum solution to DISJOINT-FVS on $I$.

By Observation 2, $A(J)$ is acyclic in $H$ if and only if $F\cup A(J)$ is acyclic in $G$. Thus, it suffices to prove that the latter is equivalent to $G\setminus(D\setminus J)$ being a forest. If $F\cup A(J)$ is acyclic in $G$, then $G\setminus(D\setminus J)$ is a forest, as $E(G\setminus(D\setminus J)) \subseteq F\cup A(J)$. For a proof in the other direction, note that

$$F\cup A(J)=E(G\setminus(D\setminus J))\uplus\{e_v^0 : v\in D\setminus J\}.$$

In particular, each vertex $v\in D\setminus J$ has only one incident edge that belongs to $F\cup A(J)$. Hence, if $G\setminus(D\setminus J)$ is acyclic, then $F\cup A(J)$ is acyclic in $G$, and the claim is proven. □

## 2.4 The branching rule

Before we proceed, let us consider an instance for which none of our reduction rules can be applied. As Rule 6 is not applicable, there exists a connected component of $G[D]$ that is not a tent, and since $G[D]$ is a forest, there exists a vertex $v$ in this component whose degree in $G[D]$ is at most one. As Rules 1, 3 and 5 are not applicable, $v$ has at least three neighbours in $U$. This lets us use the following branching rule.

[2]For basic definitions and results on matroids and graphic matroids, we refer to the monograph [22].

**Branching Rule 7.** Pick a vertex $v \in D$ that is not a tent and has at least three neighbours in $U$. Branch on $v$: either delete $v$ and decrease $k$ by one or move $v$ to $U$.

First, note that the branching of Rule 7 is exhaustive, since each vertex $v \in D$ is, trivially, either in the (unknown) solution which we seek for, or it is not in this solution.

In the branch where $v$ is removed, the number of tents does not decrease and the number of connected components of $G[U]$ remains the same. Hence, the measure $\mu(I)$ drops by at least one. As Rule 2 is not applicable, in the branch where $v$ is moved to $U$, $G[U]$ remains a forest and the number of its connected components, $\ell(I)$, drops by at least two. Hence, as $t(I)$ does not decrease and $k(I)$ remains unchanged in this branch, the measure $\mu(I)$ drops by at least two.

Consider an implementation of the algorithm as a recursive procedure, where a DISJOINT-FVS instance $I = (G,U,D,k)$ is passed as an argument. The procedure first exhaustively applies the reduction rules and, if the instance remains unresolved, it applies the branching rule, invoking itself recursively on the two subcases considered by Rule 7. Observe that each rule can be implemented in polynomial time, and reduction rules are applied at most polynomially many times in a row. Consequently, each call to the recursive procedure takes polynomial time and space.

By Lemma 4 and the arguments after the statement of Rule 5, no reduction rule increases the measure $\mu(I)$. Rule 4 terminates the algorithm if $\mu(I)$ is not positive. The branching rule, Rule 7, yields drops of measure 1 and 2 in its two branches. We infer that there are in total $\mathcal{O}(\phi^{\mu(I)})$ calls to the recursive procedure, yielding a $\mathcal{O}^*(\phi^{\mu(I)})$ bound on the time complexity of the algorithm. Moreover, as the measure drops by at least one in each recursive call, the depth of the recursion is bounded by $\mu(I)$ and, consequently, a stack of polynomial size suffices to perform the recursion. This completes the proof of Theorem 3.

# 3 Concluding remarks

In our paper we presented a new deterministic FPT algorithm for FEEDBACK VERTEX SET that can be seen as a reinterpretation and simplification of the previously fastest algorithm of Cao et al. [6].

However, we are still far from matching the time complexity of the best *randomized* algorithm, using the Cut&Count technique [12]. In particular, in our work we do not use any insights from the Cut&Count technique or its later derandomizations [11, 3]. Obtaining a *deterministic* algorithm for FVS running in $\mathcal{O}^*(3^k)$ time remains a challenging open problem.

We note that it is not hard to develop a much more elaborate case analysis and subsequent set of branching rules that, if used instead of our single branching rule, leads to a slightly better time complexity: an example set of such rules were provided by the authors in the technical report [21]. However, the case analysis becomes tedious very quickly (the branching vectors in [21] are already of size up to 80) and, in our opinion, such an approach has little technical merit.

We would also like to note that we are not aware of any lower bounds for FPT algorithms of FVS that are stronger than a refutation of a subexponential algorithm, based on the Exponential Time Hypothesis, that follows directly from a similar result for VERTEX COVER[3]. Can we show some limits for FPT algorithms for FVS, assuming the Strong Exponential Time Hypothesis, as it was done for e.g. STEINER TREE or CONNECTED VERTEX COVER [10]?

# Acknowledgements

This work has been partially supported by NCN grant UMO-2012/05/D/ST6/03214 and Foundation for Polish Science.

# References

[1] A. Becker, R. Bar-Yehuda, and D. Geiger. Randomized algorithms for the loop cutset problem. *J. Artif. Intell. Res. (JAIR)*, 12:219–234, 2000.

[2] H. L. Bodlaender. On disjoint cycles. *Int. J. Found. Comput. Sci.*, 5(1):59–68, 1994.

[3] Replace each edge with a short cycle to obtain a reduction from VERTEX COVER to FEEDBACK VERTEX SET.

[3] H. L. Bodlaender, M. Cygan, S. Kratsch, and J. Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. In F. V. Fomin, R. Freivalds, M. Z. Kwiatkowska, and D. Peleg, editors, *ICALP (1)*, volume 7965 of *Lecture Notes in Computer Science*, pages 196–207. Springer, 2013.

[4] H. L. Bodlaender and T. C. van Dijk. A cubic kernel for feedback vertex set and loop cutset. *Theory Comput. Syst.*, 46(3):566–597, 2010.

[5] K. Burrage, V. Estivill-Castro, M. R. Fellows, M. A. Langston, S. Mac, and F. A. Rosamond. The undirected feedback vertex set problem has a $poly(k)$ kernel. In H. L. Bodlaender and M. A. Langston, editors, *IWPEC*, volume 4169 of *Lecture Notes in Computer Science*, pages 192–202. Springer, 2006.

[6] Y. Cao, J. Chen, and Y. Liu. On feedback vertex set new measure and new structures. In H. Kaplan, editor, *SWAT*, volume 6139 of *Lecture Notes in Computer Science*, pages 93–104. Springer, 2010.

[7] J. Chen, F. V. Fomin, Y. Liu, S. Lu, and Y. Villanger. Improved algorithms for feedback vertex set problems. *J. Comput. Syst. Sci.*, 74(7):1188–1198, 2008.

[8] J. Chen, Y. Liu, S. Lu, B. O'Sullivan, and I. Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM*, 55(5), 2008.

[9] R. H. Chitnis, M. Cygan, M. T. Hajiaghayi, and D. Marx. Directed subset feedback vertex set is fixed-parameter tractable. In A. Czumaj, K. Mehlhorn, A. M. Pitts, and R. Wattenhofer, editors, *ICALP (1)*, volume 7391 of *Lecture Notes in Computer Science*, pages 230–241. Springer, 2012.

[10] M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlström. On problems as hard as CNF-SAT. In *IEEE Conference on Computational Complexity*, pages 74–84. IEEE, 2012.

[11] M. Cygan, S. Kratsch, and J. Nederlof. Fast hamiltonicity checking via bases of perfect matchings. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *STOC*, pages 301–310. ACM, 2013.

[12] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. M. M. van Rooij, and J. O. Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In R. Ostrovsky, editor, *FOCS*, pages 150–159. IEEE, 2011.

[13] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. Subset feedback vertex set is fixed-parameter tractable. *SIAM J. Discrete Math.*, 27(1):290–309, 2013.

[14] F. K. H. A. Dehne, M. R. Fellows, M. A. Langston, F. A. Rosamond, and K. Stevens. An $O(2^{O(k)})n^3$ FPT algorithm for the undirected feedback vertex set problem. *Theory Comput. Syst.*, 41(3):479–492, 2007.

[15] R. G. Downey and M. R. Fellows. Fixed parameter tractability and completeness. In *Complexity Theory: Current Research*, pages 191–225, 1992.

[16] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[17] H. N. Gabow and M. F. M. Stallmann. Efficient algorithms for graphic matroid intersection and parity (extended abstract). In W. Brauer, editor, *ICALP*, volume 194 of *Lecture Notes in Computer Science*, pages 210–220. Springer, 1985.

[18] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Comput. Syst. Sci.*, 72(8):1386–1396, 2006.

[19] I. A. Kanj, M. J. Pelsmajer, and M. Schaefer. Parameterized algorithms for feedback vertex set. In R. G. Downey, M. R. Fellows, and F. K. H. A. Dehne, editors, *IWPEC*, volume 3162 of *Lecture Notes in Computer Science*, pages 235–247. Springer, 2004.

[20] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[21] T. Kociumaka and M. Pilipczuk. Faster deterministic feedback vertex set. *CoRR*, abs/1306.3566, 2013.

[22] J. Oxley. *Matroid Theory*. Oxford University Press, 2 edition, 2011.

[23] V. Raman, S. Saurabh, and C. R. Subramanian. Faster fixed parameter tractable algorithms for finding feedback vertex sets. *ACM Transactions on Algorithms*, 2(3):403–415, 2006.

[24] B. A. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004.

[25] S. Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Transactions on Algorithms*, 6(2):32:1–32:8, 2010.