Improved Fixed-Parameter Algorithms for Two Feedback Set Problems

Jiong Guo[1,⋆], Jens Gramm[2,⋆⋆], Falk Hüffner[1,⋆], Rolf Niedermeier[1], and Sebastian Wernicke[1,⋆⋆⋆]

[1] Institut für Informatik, Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany
{guo,hueffner,niedermr,wernicke}@minet.uni-jena.de.
[2] Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Sand 13, D-72076 Tübingen, Germany
gramm@informatik.uni-tuebingen.de.

Abstract. Settling a ten years open question, we show that the NP-complete Feedback Vertex Set problem is deterministically solvable in $O(c^k \cdot m)$ time, where m denotes the number of graph edges, k denotes the size of the feedback vertex set searched for, and c is a constant. As a second result, we present a fixed-parameter algorithm for the NP-complete Edge Bipartization problem with runtime $O(2^k \cdot m^2)$.

1 Introduction

In feedback set problems the task is, given a graph G and a collection C of cycles in G, to find a minimum size set of vertices or edges that meets all cycles in C. We refer to Festa, Pardalos, and Resende [9] for a 1999 survey. In this work we restrict our attention to undirected and unweighted graphs, giving significantly improved exact algorithms for two NP-complete feedback set problems.

- Feedback Vertex Set (FVS): Here, the task is to find a minimum cardinality set of *vertices* that meets *all* cycles in the graph.
- Edge Bipartization: Here, the task is to find a minimum cardinality set of *edges* that meets *all odd-length* cycles in the graph.[1]

Concerning the FVS problem, it is known that an optimal solution can be approximated to a factor of 2 in polynomial time [1]. FVS is MaxSNP-hard [15] (hence, there is no hope for polynomial-time approximation schemes). A question of similar importance as approximability is to ask how fast one can find an

⋆ Supported by the Deutsche Forschungsgemeinschaft, Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4.

⋆⋆ Supported by the Deutsche Forschungsgemeinschaft, project OPAL (optimal solutions for hard problems in computational biology), NI 369/2.

⋆⋆⋆ Supported by the Deutsche Telekom Stiftung and the Studienstiftung des deutschen Volkes. Main work done while the author was with TU München.

[1] That is, the deletion of those edges would make the graph bipartite.

optimal feedback vertex set. There is a very simple randomized algorithm due to Becker et al. [3] which solves the FVS problem in $O(c \cdot 4^k \cdot kn)$ time by finding a feedback vertex set of size k with probability at least $1-(1-4^{-k})^{c4^k}$ for an arbitrary constant c. Note that this means that by choosing an appropriate value for c, one can achieve any constant error probability independent of k. As to deterministic algorithms, Bodlaender [4] and Downey and Fellows [6] were the first to show that the problem is fixed-parameter tractable. An exact algorithm with runtime $O((2k+1)^k \cdot n^2)$ was described by Downey and Fellows [7]. In 2002, Raman, Saurabh, and Subramanian [20] made a significant step forward by proving the upper bound $O(\max\{12^k, (4\log k)^k\} \cdot n^\omega)$ (n^ω denotes the time to multiply two $n \times n$ matrices). Recently, this bound was slightly improved to $O((2\log k + 2\log\log k + 18)^k \cdot n^2)$ by Kanj, Pelsmajer, and Schaefer [14] using results from extremal graph theory. Lastly, Raman et al. [21] published an algorithm running in $O((12\log k/\log\log k + 6)^k \cdot n^\omega)$ time.

The central question left open for more than ten years is whether there is an $O(c^k \cdot n^{O(1)})$ time algorithm for FVS for some constant c. We settle this open problem by giving an $O(c^k \cdot mn)$ time algorithm. Independently, this result was also shown by Dehne et al. [5], proving the constant $c \approx 10.6$. Surprisingly, although both studies were performed completely independent of each other, the developed algorithms turn out to be quite similar. The advantage of the result by Dehne et al. is a better upper bound on the constant c, whereas our advantage seems to be a more compact, easier accessible presentation of the algorithm. Since it seems hard to bring the constant c close to the constant 4 achieved by Becker et al., the described deterministic algorithms for FVS are of more theoretical interest.

Compared with Dehne et al. our algorithm also shows that FVS can be solved deterministically in *linear* time for constant k, a property which also holds for the randomized algorithm. Hence, with our corresponding $O(c^k \cdot m)$ algorithm we can conclude that FVS is "*linear-time* fixed-parameter tractable." Very recently, Fiorini et al. [10] showed, by significant technical expenditure, the analogous result concerning the Graph Bipartization problem (which is basically the same problem as Edge Bipartization, only deleting vertices instead of edges) restricted to planar graphs.

We now turn our attention to the Edge Bipartization problem. This problem is known to be MaxSNP-hard [18] and can be approximated to a factor of $O(\log n)$ in polynomial time [11]. It has applications in genome sequence assembly [19] and VLSI chip design [13]. In a recent breakthrough paper, Reed, Smith, and Vetta [22] proved that the Graph Bipartization problem is solvable in $O(4^k \cdot kmn)$ time, where k denotes the number of vertices to be deleted for making the graph bipartite. (Actually, it is straightforward to observe that the exponential factor 4^k can be lowered to 3^k by a more careful analysis of the algorithm [12].) Since there is a "parameter-preserving" reduction from Edge Bipartization to Graph Bipartization [23], one can use the algorithm by Reed et al. to directly obtain a runtime of $O(3^k \cdot k^3 m^2 n)$ for Edge Bipartization, k denoting the size of the set of edges to be deleted. In this work our main concern is to shrink the combinatorial explosion and the polynomial complexity related with the fixed-parameter tractability of Edge Bipartization. We achieve an algorithm running in $O(2^k \cdot m^2)$ time. This shows that we can save a cubic-time factor k^3 as well as a linear-time factor n, and that we can shrink the combinatorial explosion from 3^k to 2^k.

2 Preliminaries and Previous Work

This work considers undirected graphs $G = (V, E)$ with $n := |V|$ and $m := |E|$. Given a set $E' \subseteq E$ of edges, $V(E')$ denotes the set $\bigcup_{\{u,v\} \in E'} \{u, v\}$ of endpoints. We use $G[X]$ to denote the subgraph of G induced by the vertices $X \subseteq V$. For a set of edges $E' \subseteq E$, we write $G \setminus E'$ for the graph $(V, E \setminus E')$. For $u \in V$, we use $N(u)$ to denote the neighbor set $\{v \in V \mid \{u, v\} \in E\}$. With a *side* of a bipartite graph G, we mean one of the two classes of an arbitrary but fixed two-coloring of G.

The two problems we study are formally defined as follows:

Feedback Vertex Set (FVS)
Given an undirected graph $G = (V, E)$ and a nonnegative integer k, find a subset $V' \subseteq V$ of vertices with $|V'| \leq k$ such that each cycle in G contains at least one vertex from V'. (The removal of all vertices in V' from G therefore results in a forest.)

Edge Bipartization
Given an undirected graph $G = (V, E)$ and a nonnegative integer k, find a subset $E' \subseteq E$ of edges with $|E'| \leq k$ such that each odd-length cycle in G contains at least one edge from E'. (The removal of all edges in E' from G therefore results in a bipartite graph.)

We investigate FVS and Edge Bipartization in the context of parameterized complexity [7, 17] (see [8, 16] for surveys). A parameterized problem is *fixed-parameter tractable* if it can be solved in $f(k) \cdot n^{O(1)}$ time, where f is a computable function solely depending on the parameter k, not on the input size n.

To the best of our knowledge, Reed et al. [22] were the first to make the following simple observation: To show that a minimization problem is fixed-parameter tractable with respect to the size of the solution k, it suffices to give a fixed-parameter algorithm which, given a size-$(k + 1)$ solution, proves that there is no size-k solution or constructs one. Starting with a trivial instance and inductively applying this compression step a linear number of rounds to larger instances, one obtains the fixed-parameter tractability of the problem. This method is called *iterative compression.* The main challenge of applying it lies in showing that there is a "fixed-parameter compression algorithm." It is this hard part where Reed et al. achieved a breakthrough concerning Graph Bipartization. The compression step, however, is highly problem-specific and no universal standard techniques are known.

3 Algorithm for Feedback Vertex Set

In this section we show that FEEDBACK VERTEX SET can be solved in $O(c^k \cdot m)$ time for a constant c by presenting an algorithm based on iterative compression. The following lemma provides the compression step.

Lemma 1. *Given a graph G and a size-$(k+1)$ feedback vertex set (fvs) X for G, we can decide in $O(c^k \cdot m)$ time for some constant c whether there exists a size-k fvs X' for G and if so provide one.*

Proof. Consider the smaller fvs X' as a modification of the larger fvs X. The smaller fvs retains some vertices $Y \subseteq X$ while the other vertices $S := X \setminus Y$ are replaced with $|S| - 1$ new vertices from $V \setminus X$. The idea is to try by brute force all $2^{|X|}$ partitions of X into such sets Y and S. In each case, we then have significant information about a possible smaller fvs X'—it contains Y, but not S—and it turns out that there is only a "small" set V' of candidate vertices to draw from in order to complete Y to X'. More precisely, we later show in Lemma 4 that the size of V' is bounded by $14 \cdot |S|$ and that, given S, we can compute V' in $O(m)$ time. Since $|S| \leq k+1$, $|V'|$ thus only depends on the problem parameter k and not on the input size. We again use brute force and consider each of the at most $\binom{14 \cdot |S|}{|S|-1}$ possible choices of vertices from V' that can be added to Y to form X'. The test whether a choice of vertices from V' together with Y forms an fvs can be easily done in $O(m)$ time. We can now bound the overall runtime T, where the index i corresponds to a partition of X into Y and S with $|Y| = i$ and $|S| = |X| - i$:

$$T = O\left(\sum_{i=0}^{k} \binom{|X|}{i} \cdot \left(O(m) + \binom{14 \cdot (|X| - i)}{|X| - i - 1} \cdot O(m)\right)\right).$$

With Stirling's inequality, we arrive at the lemma's claim with $c \approx 37.7$.[2] ☐

Theorem 2. FEEDBACK VERTEX SET *can be solved in $O(c^k \cdot mn)$ time for a constant c.*

Proof. Given as input a graph G with vertex set $\{v_1, \ldots, v_n\}$, we can apply iterative compression to solve FEEDBACK VERTEX SET for G by iteratively considering the subgraphs $G_i := G[\{v_1, \ldots, v_i\}]$. For $i = 1$, the optimal fvs is empty. For $i > 1$, assume that an optimal fvs X_i for G_i is known. Obviously, $X_i \cup \{v_{i+1}\}$ is an fvs for G_{i+1}. Using Lemma 1, we can in $O(c^k \cdot m)$ time either determine that $X_i \cup \{v_{i+1}\}$ is an optimal fvs for G_{i+1}, or, if not, compute an optimal fvs for G_{i+1}. For $i = n$, we thus have computed an optimal fvs for G in $O(c^k \cdot mn)$ time. ☐

[2] The value of c can be significantly improved by a more careful analysis in Lemma 4. Indeed, Dehne et al. [5] achieve $c \approx 10.6$.

Theorem 2 shows that FVS is fixed-parameter tractable with the combinatorial explosion bounded from above by c^k for some constant c. Next, we show that FVS is also *linear-time* fixed-parameter tractable (with the combinatorial explosion bounded by c^k for a larger constant c). The result of Fiorini et al. [10], accepting a much worse combinatorial explosion compared to [22], is to show the analogous result for GRAPH BIPARTIZATION restricted to planar graphs.

Theorem 3. FEEDBACK VERTEX SET *can be solved in $O(c^k \cdot m)$ time for a constant c.*

Proof. We first calculate in $O(m)$ time a factor-4 approximation as described by Bar-Yehuda et al. [2]. This gives us the precondition for Lemma 1 with $|X| = 4k$ instead of $|X| = k + 1$. Now, we can employ the same techniques as in the proof of Lemma 1 to obtain the desired runtime: we examine 2^{4k} partitions $S \dot\cup Y$ of X, and—by applying the arguments from Lemma 4—for each there is some constant c' such that the number of candidate vertices is bounded from above by $c' \cdot |S|$. In summary, there is some constant c such that the runtime of the compression step is bounded from above by $O(c^k \cdot m)$. Since one of the 2^{4k} partitions must lead to the optimal solution of size k, we need only one compression step to obtain an optimal solution, which proves the claimed runtime bound. □

Note that any improvement of the approximation factor of a linear-time approximation algorithm for FEEDBACK VERTEX SET below 4 will immediately improve the runtime of the exact algorithm described in Theorem 3.

It remains to show the size bound of the "candidate vertices set" V' for fixed partition Y and S of a size-$(k+1)$ fvs X. To this end, we make use of two simple data reduction rules.

Lemma 4. *Given a graph $G = (V, E)$, a size-$(k+1)$ fvs X for G, and a partition of X into two sets Y and S. Let X' denote a size-k fvs for G with $X' \cap X = Y$ and $X' \cap S = \emptyset$. In $O(m)$ time, we can either decide that no such X' exists or compute a subset V' of $V \setminus X$ with $|V'| < 14 \cdot |S|$ such that there exists an X' as desired consisting of $|S| - 1$ vertices from V' and all vertices from Y.*

Proof. The idea of the proof is to use a well-known data reduction technique for FVS to get rid of degree-1 and degree-2 vertices and to show that if the resulting instance is too large as compared to the part S (whose vertices we are not allowed to add to X'), then there exists no set X' as desired.

First, check that S does not induce a cycle; otherwise, no X' with $X' \cap S = \emptyset$ can be an fvs for G. Then, remove in G all vertices from Y as they are determined to be in X'. Finally, apply a standard data reduction to the vertices in $V \setminus X$ (the vertices in S remain unmodified): remove degree-1 vertices and successively bypass any degree-2 vertex by a new edge between its neighbors (thereby removing the bypassed degree-2 vertex). There are two exceptions to note: One exception is that we do not bypass a degree-2 vertex which has two neighbors in S. The other exception is the way to deal with parallel edges. If we create two parallel edges between two vertices during the data reduction process—these two edges

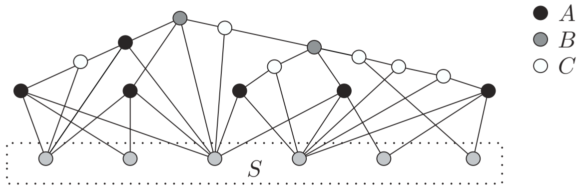

Fig. 1. Partition of the vertices in V' into three disjoint subsets A, B, and C.

form a length-two cycle—, then exactly one of the two endpoints of these edges has to be in S since S is an fvs of $G[V \setminus Y]$ and $G[S]$ contains no cycle. Thus, we have to delete the other endpoint and add it to X' since we are not allowed to add vertices from S to X'. Given an appropriate graph data structure, all of the above steps can be accomplished in $O(m)$ time. Proofs for the correctness and the time bound of the data reduction technique are basically straightforward and omitted here.

In the following we use $G' = (V' \cup S, E')$ with $V' \subseteq V \setminus X$ to denote the graph resulting after exhaustive application of the data reduction described above; note that none of the vertices in S have been removed during the data reduction process. In order to prove that $|V'| < 14 \cdot |S|$, we partition V' into three subsets, each of which will have a provable size bound linearly depending on $|S|$ (the partition is illustrated in Fig. 1):

$$A := \{v \in V' \mid |N(v) \cap S| \geq 2\},$$
$$B := \{v \in V' \setminus A \mid |N(v) \cap V'| \geq 3\},$$
$$C := V' \setminus (A \cup B).$$

To bound the number of vertices in A, consider the bipartite subgraph $G_A = (A \cup S, E_A)$ of $G' = (V' \cup S, E')$ with $E_A := (A \times S) \cap E'$. Observe that if there are more than $|S| - 1$ vertices in A, then there is a cycle in G_A: If G_A is acyclic, then G_A is a forest, and, thus, $|E_A| \leq |S| + |A| - 1$. Moreover, since each vertex in A has at least two incident edges in G_A, $|E_A| \geq 2|A|$, which implies that $|A| \leq |S| - 1$ if G_A is acyclic. It follows directly that if $|A| \geq 2|S|$, it is impossible to delete at most $|S|$ vertices from A such that $G'[A \cup S]$ is acyclic.

To bound the number of vertices in B, observe that $G'[V']$ is a forest. Furthermore, all leaves of the trees in $G'[V']$ are from A since G' is reduced with respect to the above data reduction rules. By the definition of B, each vertex in B has at least three vertices in V' as neighbors. Thus, there cannot be more vertices in B than in A, and therefore $|B| < 2|S|$.

Finally, consider the vertices in C. By the definitions of A and B, and since G is reduced, each vertex in C has degree two in $G'[V']$ and exactly one neighbor in S. Hence, graph $G'[C]$ is a forest consisting of paths and isolated vertices. We now separately bound the number of isolated vertices and those participating in paths.

Each of the isolated vertices in $G'[C]$ connects two vertices from $A \cup B$ in $G'[V']$. Since $G'[V']$ is acyclic, the number of isolated vertices in $G'[C]$ cannot exceed $|A \cup B| - 1 < 4|S|$. The total number of vertices participating in paths in $G'[C]$ can be bounded as follows: Consider the subgraph $G'[C \cup S]$. Each edge in $G'[C]$ creates a path between two vertices in S, that is, if $|E(G'[C])| \geq |S|$, then there exists a cycle in $G'[C \cup S]$. By an analogous argument to the one that bounded the size of A (and considering that removing a vertex from $G'[C]$ destroys at most two edges), the total number of edges in $G'[C]$ may thus not exceed $|S| + 2|S|$, bounding the total number of vertices participating in paths in $G'[C]$ by $6|S|$.

Altogether, $|V'| = |A| + |B| + |C| < 2|S| + 2|S| + (4|S| + 6|S|) = 14|S|$. □

4 Algorithm for Edge Bipartization

In this section we present a new algorithm for Edge Bipartization which is based on iterative compression and runs in $O(2^k \cdot m^2)$ time. The algorithm is structurally similar to the $O(4^k \cdot kmn)$ time algorithm for Graph Bipartization given by Reed et al. [22]: Their compression routine starts by enumerating all partitions of the known solution into two parts, one containing vertices to keep in the solution and one containing the vertices to exchange. This is followed by a second step that tries to find a compressed bipartization set under this constraint. Our algorithm for Edge Bipartization does not need the first step by enforcing that the smaller solution is disjoint from the known one, thereby gaining a factor of $O(2^k)$ in the runtime.

We note that a similar runtime of $O(2^k \cdot |G|^{O(1)})$ for Edge Bipartization can be achieved by first reducing the input instance to Graph Bipartization [23], and exploiting a solution disjointness property analogously to the presented algorithm. This, however, involves several nontrivial modifications to the algorithm of Reed et al., whereas we give a self-contained presentation here. Moreover, our proof reveals details about the structure of Edge Bipartization that might be of independent interest.

The following lemma provides some central insight into the structure of a minimal edge bipartization set. (Note that in this section, we always use the notion of paths in which every vertex is allowed to occur at most once.)

Lemma 5. *Given a graph $G = (V, E)$ with a minimal edge bipartization set X for G, the following two properties hold:*

1. *For every odd-length cycle C in G, $|E(C) \cap X|$ is odd.*
2. *For every even-length cycle C in G, $|E(C) \cap X|$ is even.*

Proof. For each edge $e = \{u, v\} \in X$, note that u and v are on the same side of the bipartite graph $G \setminus X$, since otherwise we do not need e to be in X and X would not be minimal. Consider a cycle C in G. The edges in $E(C) \setminus X$ are all between the two sides of $G \setminus X$, while the edges in $E(C) \cap X$ are between vertices of the same side as argued above. In order for C to be a cycle, however, this implies that $|E(C) \setminus X|$ is even. Since $|E(C)| = |E(C) \setminus X| + |E(C) \cap X|$, we conclude that $|E(C)|$ and $|E(C) \cap X|$ have the same parity. □

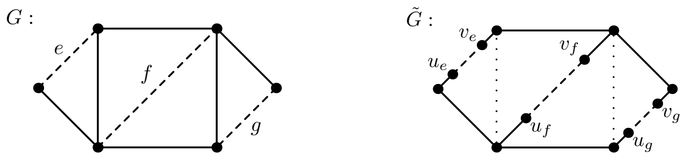

Fig. 2. Left: Graph G with a minimal edge bipartization marked by dashed lines. Right: Edge-extension graph $\tilde{G}$ of G with the corresponding edge bipartization $\tilde{X}$ marked by dashed lines. The mapping Φ which maps $\Phi(u_e) = \Phi(u_f) = \Phi(u_g) = A$, and $\Phi(v_e) = \Phi(v_f) = \Phi(v_g) = B$ is a valid 2-partition of $V(\tilde{X})$. Note that when choosing this valid 2-partition Φ, then the dotted edges are an edge cut between the A-vertices and the B-vertices in $\tilde{G} \setminus \tilde{X}$. Therefore, the dotted edges are an edge bipartization for the graph on the left (Lemma 7).

When subdividing all edges in a graph G that are contained in an edge bipartization set X for G by two vertices, we can assume without loss of generality that an edge bipartization set smaller than X is disjoint from X. This input transformation is formalized in the following definition.

Definition 6. *For a graph $G = (V, E)$ with minimal edge bipartization X, let the corresponding* edge-extension graph $\tilde{G} := (\tilde{V}, \tilde{E})$ *be given by*

$$\tilde{V} := V \cup \{u_e, v_e \mid e \in X\} \text{ and}$$
$$\tilde{E} := (E \setminus X) \cup \{\{u, u_e\}, \{u_e, v_e\}, \{v_e, v\} \mid e = \{u, v\} \in X\}.$$

Let $\tilde{X} := \{\{u_e, v_e\} \mid e \in X\}$. A mapping $\Phi : V(\tilde{X}) \to \{A, B\}$ is called valid 2-partition of $V(\tilde{X})$ *if for each $\{u_e, v_e\} \in \tilde{X}$, either $\Phi(u_e) = A$ and $\Phi(v_e) = B$ or $\Phi(u_e) = B$ and $\Phi(v_e) = A$.*

An illustration of edge-extension graphs is given in Fig. 2. It is easy to see that $\tilde{G}$ has an edge bipartization with k edges if and only if G has an edge bipartization with k edges. Observe that, hence, the set $\tilde{X}$ as defined above constitutes a minimal edge bipartization for $\tilde{G}$.

Lemma 7. *Consider an edge-extension graph $G = (V, E)$ and a minimal edge bipartization X for G. For a set of edges $Y \subseteq E$ with $X \cap Y = \emptyset$, the following are equivalent:*

(1) Y is an edge bipartization for G.
(2) There is a valid 2-partition Φ of $V(X)$ such that Y is an edge cut between $A_\Phi := \Phi^{-1}(A)$ and $B_\Phi := \Phi^{-1}(B)$ in $G \setminus X$ (see Fig. 2).

Proof. (2) $\Rightarrow$ (1): Consider any odd-length cycle C in G. We show that $E(C) \cap Y \neq \emptyset$. Let $s := |E(C) \cap X|$. By Property 1 in Lemma 5, s is odd. Without loss of generality, we assume that $E(C) \cap X = \{\{u_0, v_0\}, \{u_1, v_1\}, \ldots, \{u_{s-1}, v_{s-1}\}\}$ with vertices v_i and $u_{(i+1) \bmod s}$ being connected by a path in $C \setminus X$. Since Φ is a valid 2-partition, we have $\Phi(u_i) \neq \Phi(v_i)$ for all $0 \leq i < s$. With s being odd, this implies that there is a pair $v_i, u_{(i+1) \bmod s}$ such that $\Phi(v_i) \neq \Phi(u_{(i+1) \bmod s})$. Since the removal of Y destroys all paths in $G \setminus X$ between A_Φ and B_Φ, we obtain that $E(C) \cap Y \neq \emptyset$.

(1) $\Rightarrow$ (2): Let $C_X : V \to \{A, B\}$ be a two-coloring of the bipartite graph $G \setminus X$ and $C_Y : V \to \{A, B\}$ a two-coloring of the bipartite graph $G \setminus Y$. Define

$$\Phi : V \to \{A, B\}, \; v \mapsto \begin{cases} A & \text{if } C_X(v) = C_Y(v) \\ B & \text{otherwise.} \end{cases}$$

We show that $\Phi|_{V(X)}$ (that is, Φ with domain restricted to $V(X)$) is a valid 2-partition with the desired property.

First we show that $\Phi|_{V(X)}$ is a valid 2-partition. Consider an edge $\{u, v\} \in X$. There must be at least one even path in $G \setminus X$ from u to v, or $\{u, v\}$ would be redundant; therefore $C_X(u) = C_X(v)$. In $G \setminus Y$, the vertices u and v are connected by an edge, and therefore $C_Y(u) \neq C_Y(v)$. It follows that $\Phi(u) \neq \Phi(v)$.

Since both C_X and C_Y change in value when going from a vertex to its neighbor in $G \setminus (X \cup Y)$, the value of Φ is constant along any path in $G \setminus (X \cup Y)$. Therefore, there can be no path from any $u \in A_\Phi$ to any $v \in B_\Phi$ in $G \setminus (X \cup Y)$, that is, Y is an edge cut between A_Φ and B_Φ in $G \setminus X$. □

Theorem 8. Edge Bipartization *can be solved in $O(2^k \cdot m^2)$ time.*

Proof. Through Lemma 7 we obtain the compression step that, from a given minimal edge bipartization X, computes a smaller edge bipartization Y in $O(2^k \cdot km)$ time or proves that no such Y exists: We enumerate all 2^k valid 2-partitions Φ of $V(X)$ and determine a minimum edge cut between A_Φ and B_Φ until we find an edge cut Y of size $k - 1$ (see Fig. 2). Note that the condition of Lemma 7 that $Y \cap X = \emptyset$ does not restrict generality: Since G is an edge extension graph (Definition 6), we can replace each edge in $Y \cap X$ by one of its two adjacent edges in G. Each of the minimum cut problems can individually be solved in $O(km)$ time with the Ford-Fulkerson method that finds and augments a flow augmenting path k times. By Lemma 7, Y is an edge bipartization; furthermore, if no such Y is found, we know that $|X|$ is minimum.

Given as input a graph G with edge set $\{e_1, \ldots, e_m\}$, we can apply iterative compression to solve Edge Bipartization for G by iteratively considering the graphs G_i containing edges $\{e_1, \ldots, e_i\}$, for $i = 1, \ldots, m$. For $i = 1$, the optimal edge bipartization is empty. For $i > 1$, assume that an optimal edge bipartization X_{i-1} with $|X_{i-1}| \leq k$ for G_{i-1} is known. If X_{i-1} is not an edge bipartization for G_i, then we consider the set $X_{i-1} \cup \{e_i\}$, which obviously is a minimal edge bipartization for G_i. Using Lemma 7, we can in $O(2^{k'} \cdot k'i)$ time (where $k' := |X_{i-1} \cup \{e_i\}| \leq k+1$) either determine that $X_{i-1} \cup \{e_i\}$ is an optimal edge bipartization for G_i or, if not, compute an optimal edge bipartization X_i for G_i. This process can be stopped if $|X_i| > k$.

Summing over all iterations, we have an algorithm that computes an optimal edge bipartization for G in $O(\sum_{i=1}^{m} 2^k \cdot ki) = O(2^k \cdot km^2)$ time.

With the same technique used by Hüffner [12] to improve the runtime of the iterative compression algorithm for Graph Bipartization, the runtime here can also be improved to $O(2^k \cdot m^2)$. For this, one uses a Gray code to enumerate the valid 2-partitions in such a way that consecutive 2-partitions differ in only one element. For each of these (but the first one), one can then solve the flow problem by a constant number of augmentation operations on the previous flow network in $O(m)$ time. $\square$

5 Conclusion

We present significantly improved results on the fixed-parameter tractability of Feedback Vertex Set and Edge Bipartization. To our belief, the iterative compression strategy due to Reed et al. employed in this work will become an important tool in the design of efficient fixed-parameter algorithms.

We succeeded in proving that FVS is even solvable in *linear* time for constant parameter value k. Employing a completely different technique, a similar result could very recently be shown for Graph Bipartization restricted to planar graphs (where the problem remains NP-complete) [10]. For general Graph Bipartization as well as for Edge Bipartization, this remains open.

Finally, it remains a long-standing open problem whether Feedback Vertex Set on *directed* graphs is fixed-parameter tractable. The answer to this question would mean a significant breakthrough in the field.

Acknowledgement. The authors would like to thank the anonymous referees of WADS 2005 for helpful and inspiring comments.

References

1. V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 3(2):289–297, 1999.
2. R. Bar-Yehuda, D. Geiger, J. Naor, and R. M. Roth. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference. *SIAM Journal on Computing*, 27(4):942–959, 1998.
3. A. Becker, R. Bar-Yehuda, and D. Geiger. Randomized algorithms for the Loop Cutset problem. *Journal of Artificial Intelligence Research*, 12:219–234, 2000.
4. H. L. Bodlaender. On disjoint cycles. *International Journal of Foundations of Computer Science*, 5:59–68, 1994.
5. F. Dehne, M. Fellows, M. Langston, F. Rosamond, and K. Stevens. An $\mathcal{O}^*(2^{O(k)})$ FPT algorithm for the undirected feedback vertex set problem. In *Proc. 11th COCOON*, LNCS. Springer, Aug. 2005.
6. R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness. *Congressus Numerantium*, 87:161–187, 1992.
7. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
8. M. R. Fellows. New directions and new challenges in algorithm design and complexity, parameterized. In *Proc. 8th WADS*, volume 2748 of *LNCS*, pages 505–520. Springer, 2003.

9. P. Festa, P. M. Pardalos, and M. G. C. Resende. Feedback set problems. In D. Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization, Vol. A*, pages 209–258. Kluwer, 1999.
10. S. Fiorini, N. Hardy, B. Reed, and A. Vetta. Planar graph bipartization in linear time. In *Proc. 2nd GRACO*, Electronic Notes in Discrete Mathematics, 2005.
11. N. Garg, V. V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing*, 25(2):235–251, 1996.
12. F. Hüffner. Algorithm engineering for optimal graph bipartization. In *Proc. 4th WEA*, volume 3503 of *LNCS*, pages 240–252. Springer, 2005.
13. A. B. Kahng, S. Vaya, and A. Zelikovsky. New graph bipartizations for double-exposure, bright field alternating phase-shift mask layout. In *Proc. Asia and South Pacific Design Automation Conference*, pages 133–138, 2001.
14. I. Kanj, M. Pelsmajer, and M. Schaefer. Parameterized algorithms for feedback vertex set. In *Proc. 1st IWPEC*, volume 3162 of *LNCS*, pages 235–247. Springer, 2004.
15. C. Lund and M. Yannakakis. The approximation of maximum subgraph problems. In *Proc. 20th ICALP*, volume 700 of *LNCS*, pages 40–51. Springer, 1993.
16. R. Niedermeier. Ubiquitous parameterization—invitation to fixed-parameter algorithms. In *Proc. 29th MFCS*, volume 3153 of *LNCS*, pages 84–103. Springer, 2004.
17. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, forthcoming, 2005.
18. C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.
19. M. Pop, D. S. Kosack, and S. L. Salzberg. Hierarchical scaffolding with Bambus. *Genome Research*, 14:149–159, 2004.
20. V. Raman, S. Saurabh, and C. R. Subramanian. Faster fixed parameter tractable algorithms for undirected feedback vertex set. In *Proc. 13th ISAAC*, volume 2518 of *LNCS*, pages 241–248. Springer, 2002.
21. V. Raman, S. Saurabh, and C. R. Subramanian. Faster algorithms for feedback vertex set. In *Proc. 2nd GRACO*, Electronic Notes in Discrete Mathematics, 2005.
22. B. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32:299–301, 2004.
23. S. Wernicke. On the algorithmic tractability of single nucleotide polymorphism (SNP) analysis and related problems. Diplomarbeit, WSI für Informatik, Universität Tübingen, 2003.