Approximation Algorithms for the Feedback Vertex Set Problem with Applications to Constraint Satisfaction and Bayesian Inference *

Reuven Bar-Yehuda † Dan Geiger Joseph (Seffi) Naor ‡
Ron M. Roth

Computer Science Department
Technion
Haifa 32000, ISRAEL

Abstract

A *feedback vertex set* of an undirected graph is a subset of vertices that intersects with the vertex set of each cycle in the graph. Given an undirected graph G with n vertices and weights on its vertices, polynomial-time algorithms are provided for approximating the problem of finding a feedback vertex set of G with a smallest weight. When the weights of all vertices in G are equal, the performance ratio attained by these algorithms is $4-(2/n)$. This improves a previous algorithm which achieved an approximation factor of $O(\sqrt{\log n})$ for this case. For general vertex weights, the performance ratio becomes $\min\{2\Delta^2, 4\log_2 n\}$ where Δ denotes the maximum degree in G. For the special case of planar graphs this ratio is reduced to 10. An interesting special case of weighted graphs where a performance ratio of $4-(2/n)$ is achieved is the one where a prescribed subset of the vertices, so called *blackout* vertices, is not allowed to participate in any feedback vertex set.

It is shown how these algorithms can improve the search performance for constraint satisfaction problems. An application in the area of Bayesian inference of graphs with blackout vertices is also presented.

---

*A preliminary version of this paper appeared in the *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms,* Arlington, Virginia, 1994, 344–354.

†Part of this research was done while the author was visiting SUNY at Buffalo. This research was supported by the fund for the promotion of research at the Technion.

‡Research supported in part by Grant No. 92-00225 from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel. Part of this research was done while the author was visiting DIMACS, Rutgers University, NJ.

1 Introduction

Let $G = (V, E)$ be an undirected graph and let $w : V(G) \to \mathbb{R}^+$ be a weight function on the vertices of G. A *cycle* in G is a path whose two terminal vertices coincide. A *feedback vertex set* of G is a subset of vertices $F \subseteq V(G)$ such that each cycle in G passes through at least one vertex in F. In other words, a feedback vertex set F is a set of vertices of G such that by removing F from G, along with all the edges incident with F, a forest is obtained. A *minimum feedback vertex set of a weighted graph* (G, w) is a feedback vertex set of G of minimum weight. The weight of a minimum feedback vertex set will be denoted by $\mu(G, w)$.

The *weighted feedback vertex set (WFVS) problem* is defined as finding a minimum feedback vertex set of a given weighted graph (G, w). The special case where w is the constant function 1 is called the *unweighted feedback vertex set (UFVS) problem.* Given a graph G and an integer k, the problem of deciding whether $\mu(G, 1) \leq k$ is known to be NP-Complete [GJ79, pp. 191–192]. Hence, it is natural to look for efficient approximation algorithms for the feedback vertex set problem, particularly in view of the recent applications of such algorithms in artificial intelligence, as we show in the sequel.

Suppose A is an algorithm that finds a feedback vertex set F_A for any given undirected weighted graph (G, w). We denote the sum of weights of the vertices in F_A by $w(F_A)$. The *performance ratio of A for (G, w)* is defined by $R_A(G, w) = w(F_A)/\mu(G, w)$. When $\mu(G, w) = 0$ we define $R_A(G, w) = 1$ if $w(F_A) = 0$ and $R_A(G, w) = \infty$ if $w(F_A) > 0$. The *performance ratio $r_A(n, w)$ of A for w* is the supremum of $R_A(G, w)$ over all graphs G with n vertices and for the same weight function w. When w is the constant function 1, we call $r_A(n, 1)$ the *unweighted performance ratio of A*. Finally, the *performance ratio $r_A(n)$ of A* is the supremum of $r_A(n, w)$ over all weight functions w defined over graphs with n vertices.

An approximation algorithm for the UFVS problem that achieves an unweighted performance ratio of $2 \log_2 n$ is essentially contained in a lemma due to Erdős and Pósa [EP62]. This result was improved by Monien and Schulz [MS81], where they achieved a performance ratio of $O(\sqrt{\log n})$.

In Section 2, we provide an approximation algorithm for the UFVS problem that achieves an unweighted performance ratio of at most $4 - (2/n)$. Our algorithm draws upon a theorem by Simonovits [Si67] and our analysis uses a result by Voss [Vo68]. Actually, we consider a generalization of the UFVS problem, where a prescribed subset of the vertices, called *blackout vertices*, is not allowed to participate in any feedback vertex set. This problem is a subcase of the WFVS problem wherein each allowed vertex has unit weight and each blackout vertex has infinite weight. Our interest in graphs with blackout vertices is motivated by the *loop cutset* problem and its application to the updating problem in Bayesian

inference which is explored in Section 4.

In Section 3, we present two algorithms for the WFVS problem. We first devise a primal-dual algorithm which is based on formulating the WFVS problem as an instance of the *set cover* problem. The algorithm has a performance ratio of 10 for weighted planar graphs and $4\log_2 n$ for general weighted graphs. This ratio is achieved by extending the Erdős-Pósa Lemma to weighted graphs. The second algorithm presented in Section 3 achieves a performance ratio of $2\Delta^2(G)$ for general weighted graphs, where $\Delta(G)$ is the maximum degree of G. This result is interesting for low degree graphs.

A notable application of approximation algorithms for the UFVS problem in artificial intelligence due to Dechter and Pearl is as follows [DP87, De90]. We are given a set of variables $x_1, x_2, \ldots, x_n$, where each x_i takes its values from a finite domain D_i. Also, for every $i < j$ we are given a constraint subset $R_{i,j} \subseteq D_i \times D_j$ which defines the allowable pairs of values that can be taken by the pair of variables (x_i, x_j). Our task is to find an assignment for all variables such that all the constraints $R_{i,j}$ are satisfied. With each instance of the problem we can associate an undirected graph G whose vertex set is the set of variables, and for each constraint $R_{i,j}$ which is *strictly* contained in $D_i \times D_j$ (i.e., $R_{i,j} \neq D_i \times D_j$) there is an edge in G connecting x_i and x_j. The resulting graph G is called a *constraint network* and it is said to represent a *constraint satisfaction problem*.

A common method for solving a constraint satisfaction problem is by backtracking, that is, by repeatedly assigning values to the variables in a predetermined order and then backtracking whenever reaching a dead end. This approach can be improved as follows. First, find a feedback vertex set of the constraint network. Then, arrange the variables so that variables in the feedback vertex set precede all other variables, and apply the backtracking procedure. Once the values of the variables in the feedback vertex set are determined by the backtracking procedure, the algorithm switches to a polynomial-time procedure SOLVE-TREE that solves the constraint satisfaction problem in the remaining forest. If SOLVE-TREE succeeds, a solution is found; otherwise, another backtracking phase occurs.

The complexity of the above modified backtracking algorithm grows exponentially with the size of the feedback vertex set: If a feedback vertex set contains k variables, each having a domain of size 2, then the procedure SOLVE-TREE might be invoked up to 2^k times. A procedure SOLVE-TREE that runs in polynomial-time was developed by Dechter and Pearl, who also proved the optimality of their tree algorithm [DP88]. Consequently, our approximation algorithm for finding a small feedback vertex set reduces the complexity of solving constraint satisfaction problems through the modified backtracking algorithm. Furthermore, if the domain size of the variables varies, then SOLVE-TREE is called a number of times which is bounded from above by the product of the domain-sizes of the variables

whose corresponding vertices participate in the feedback vertex set. If we take the logarithm of the domain size as the weight of a vertex, then solving the WFVS problem with these weights optimizes the complexity of the modified backtracking algorithm in the case where the domain size is allowed to vary.

2 The Unweighted Feedback Vertex Set Problem

The best approximation algorithm prior to this work for the UFVS problem attained a performance ratio of $O(\sqrt{\log n})$ [MS81]. We now use some results of [Si67] and [Vo68] in order to obtain an approximation algorithm for the UFVS problem which attains a performance ratio ≤ 4. In fact, we actually consider a slight generalization of the UFVS problem where we mark each vertex of a graph as either an *allowed vertex* or a *blackout vertex*. In such graphs, feedback vertex sets cannot contain any blackout vertices. We denote the set of allowed vertices in G by $A(G)$ and the set of blackout vertices by $B(G)$. Note that when $B(G) = \emptyset$, this problem reduces to the UFVS problem. A feedback vertex set can be found in a graph G with blackout vertices if and only if every cycle in G contains at least one allowed vertex. A graph G with this property will be called a *valid graph*. The motivation for dealing with this modified problem is clarified in Section 4 where we use the algorithm developed herein to reduce the computational complexity of Bayesian inference.

Throughout this section, G denotes a valid graph with a nonempty set of vertices $V(G)$ which is partitioned into a nonempty set $A(G)$ of allowed vertices, a possibly empty set $B(G)$ of blackout vertices, and a set of edges $E(G)$ possibly with parallel edges and self-loops. We use $\mu_a(G)$ as a short-hand notation for $\mu(G, w)$ where w assigns unit weight to each allowed vertex and an infinite weight to each blackout vertex. A *neighbor* of v is a vertex $u \in V(G)$ which is connected to v by an edge in $E(G)$. The degree $\Delta_G(v)$ of v in G is the number of edges that are incident with v in G. A self-loop at a vertex v contributes 2 to the degree of v. The degree of G, denoted $\Delta(G)$, is the largest among all degrees of vertices in G. A vertex in G of degree 1 is called an *endpoint*. A vertex of degree 2 is called a *linkpoint* and a vertex of any higher degree is called a *branchpoint*. A graph G is called *rich* if every vertex in $V(G)$ is a branchpoint. The notation $\Delta_a(G)$ will stand for the largest among all degrees of vertices in $A(G)$ (a degree of a vertex in $A(G)$ takes into account all incident edges, including those that lead to neighbors in $B(G)$). In a rich valid graph we have $\Delta_a(G) \geq 3$.

Two cycles in a valid graph G are *independent* if their vertex sets share only blackout vertices. Note that the size of any feedback vertex set of G is bounded from below by the largest number of pairwise independent cycles that can be found in G. A cycle Γ in G is called *simple* if it visits every vertex in $V(G)$ at most once. Clearly, a set F is a feedback

vertex set of G if and only if it intersects with every simple cycle in G. A graph is called a *singleton* if it contains only one vertex. A singleton is called *self-looped* if it contains at least one self loop; for a singleton we have $\mu(G,1) = 1$ if it is self-looped and $\mu(G,1) = 0$ otherwise.

A graph G is connected if for every two vertices there is a connecting path in G. Every graph G can be uniquely decomposed into isolated connected components $G_1, G_2, \ldots, G_k$. Similarly, every feedback vertex set F of G can be partitioned into feedback vertex sets $F_1, F_2, \ldots, F_k$ such that F_i is a feedback vertex set of G_i. Hence, $\mu_a(G) = \sum_{i=1}^{k} \mu_a(G_i)$.

A *2-3-subgraph* of a valid graph G is a subgraph H of G such that the degree in H of every vertex in $A(G)$ is either 2 or 3. The degree of a vertex belonging to $B(G)$ in H is not restricted. A 2-3-subgraph exists in any valid graph which is not a forest. A *maximal 2-3-subgraph* of G is a 2-3-subgraph of G which is not a subgraph of any other 2-3-subgraph of G. A maximal 2-3-subgraph can be easily found by applying depth-first-search (DFS) on G.

A linkpoint v in a 2-3-subgraph H is called a *critical linkpoint* if v is an allowed vertex, and there is a cycle Γ in G such that $V(\Gamma) \cap V(H) \cap A(G) = \{v\}$. We refer to such a cycle Γ in G as a *witness cycle of* v. Note that we can assume a witness cycle to be simple and, so, verifying whether a linkpoint v in H is a critical linkpoint is easy: Remove the set of vertices $(V(H) \cap A(G)) - \{v\}$ from G, with all incident edges, and apply a breadth-first-search (BFS) to check whether there is a cycle through v in the remaining graph.

A cycle in a valid graph G is *branchpoint-free* if it does not pass through any allowed branchpoints; that is, a branchpoint-free cycle passes only through allowed linkpoints and blackout vertices of G.

The rest of this section is devoted to showing that the following algorithm correctly outputs a vertex feedback set and achieves an unweighted performance ratio less than 4.

Algorithm SubG-2-3 (*Input:* **valid graph** G;
Output: **feedback vertex set** F **of** G);

if G **is a forest then**
    $F \leftarrow \emptyset$;
else begin:
    Using DFS, find a maximal 2-3-subgraph H **of** G;
    Using BFS, find the set X **of critical linkpoints in** H;
    Let Y **be the set of allowed branchpoints in** H;
    Find a set W **that covers all branchpoint-free cycles of** H **which**

are not covered by X**;**

$F \leftarrow X \cup Y \cup W$**;**

end.

Note that if $B(G) = \emptyset$, then all branchpoint-free cycles are isolated cycles in H and so W consists of one vertex of each such cycle.

We elaborate on how the set W is computed when $B(G) \neq \emptyset$. Let H' be a graph obtained from H by removing the set X along with its incident edges. Let H_b be the subgraph of H' induced by the allowed linkpoints and blackout vertices of H'. For every isolated cycle in H_b, we arbitrarily choose an allowed linkpoint from that cycle to W. Next, we replace each maximal (with respect to containment) chain of allowed linkpoints in H_b by an edge, resulting in a graph H_b^*. We assign unit cost to all edges corresponding to a chain of allowed linkpoints, and a zero cost to all other edges, and compute a minimum-cost spanning forest T of H_b^*. We now add to W one linkpoint from each chain of allowed linkpoints in H_b that corresponds to an edge in $H_b^* - T$. It is now straightforward to verify that the complexity of SubG-2-3 is linear in $|E(G)|$.

The following two lemmas, which generalize some claims used in the proof of Theorem 1 in [Si67], are used to prove that SubG-2-3 outputs a feedback vertex set of a valid graph G.

Lemma 1 *Let H be a maximal 2-3-subgraph of a valid graph G and let Γ be a simple cycle in G. Then, one of the following holds:*

(a) Γ is a witness cycle of some critical linkpoint of H, or —

(b) Γ passes through some allowed branchpoint of H, or —

(c) Γ is a cycle in H that consists only of blackout vertices or allowed linkpoints of H.

Proof. Let Γ be a simple cycle in G and assume to the contrary that neither of (a)–(c) holds. This implies in particular that Γ cannot be entirely contained in H. We distinguish between two cases: (1) Γ does not intersect with H; and (2) Γ intersects with H only in blackout vertices and allowed linkpoints of H.

Case 1: In this case we could join Γ and H to obtain a 2-3-subgraph H^* of G that contains H as a proper subgraph. This however contradicts the maximality of H.

Case 2: If Γ intersects with H only in blackout vertices, then as in case 1, we can join Γ and H and contradict the maximality of H. Suppose now that Γ intersects with H in some allowed linkpoints of H. Note that in such a case Γ must intersect with H in at least two distinct allowed linkpoints of H, or else Γ would be a witness cycle of the only intersecting (critical) linkpoint. Since Γ is not contained in H by assumption, we can find two allowed

linkpoints v_1 and v_2 in $V(\Gamma) \cap V(H)$ that are connected by a path P along Γ such that $V(P) \cap V(H) \cap A(G) = \{v_1, v_2\}$ and P is not entirely contained in H. Joining P and H, we obtain a 2-3-subgraph of G that contains H as a proper subgraph, thus contradicting the maximality of H. □

Lemma 2 *Let H be a maximal 2-3-subgraph of G and let Γ_1 and Γ_2 be witness cycles in G of two distinct critical linkpoints in H. Then Γ_1 and Γ_2 are independent cycles, namely, $V(\Gamma_1) \cap V(\Gamma_2) \subseteq B(G)$.*

Proof. Let v_1 and v_2 be the critical linkpoints associated with Γ_1 and Γ_2, respectively, and assume to the contrary that $V(\Gamma_1) \cap V(\Gamma_2)$ contains an allowed vertex $u \in A(G)$. Then, there is a path P in G that runs along parts of the cycles Γ_1 and Γ_2, starting from v_1, passing through u, and ending at v_2. Since Γ_1 and Γ_2 are witness cycles, we have $V(P) \cap V(H) \cap A(G) = \{v_1, v_2\}$. And, since v_1 and v_2 are distinct critical linkpoints, the vertex u cannot possibly coincide with either of them. Therefore, the path P is not entirely contained in H. Joining P and H we obtain a 2-3-subgraph of G that contains H as a proper subgraph, thus reaching a contradiction. □

Theorem 3 *For every valid graph G, the set F computed by* SubG-2-3 *is a feedback vertex set of G.*

Proof. Let Γ be a simple cycle in G. We follow the three cases of Lemma 1 to show that $V(\Gamma) \cap F \neq \emptyset$.

(a) Γ is a witness cycle of some critical linkpoint of H. By construction, all critical linkpoints of H are in F.

(b) Γ passes through some allowed branchpoint of H. By construction, all allowed branchpoints of H are in F.

(c) Γ is a cycle in H that consists only of blackout vertices or allowed linkpoints of H. When $V(\Gamma)$ contains a critical linkpoint, then SubG-2-3 selects that linkpoint into the feedback vertex set F. Otherwise, the cycle Γ must be entirely contained in the graph H_b that was used to create W. We now show that W covers all cycles in H_b. Assume the contrary and let Γ be a cycle in H_b that is not covered by W. Recall that in the construction of H_b^*, each chain of allowed linkpoints in Γ was replaced by an edge with a unit cost. Let Γ^* be the resulting cycle in H_b^*. Since W does not cover Γ, all unit-cost edges in Γ^* were necessarily chosen to the minimum-cost spanning forest T. On the other hand, since T does not contain any cycles, there must be at least one zero-cost edge of Γ^* which is not contained in T. Hence, by deleting one of the unit-cost edges of Γ^* from T and inserting instead a particular zero-cost edge of Γ^* into T, we can obtain a new spanning forest T' for H_b^*. However, the cost of T' is smaller than that of T, which contradicts our assumption

that T is a minimum-cost spanning forest. □

A *reduction graph* G' of an undirected graph G is a graph obtained from G by a sequence of the following transformations:

- Delete an endpoint and its incident edge.
- Connect two neighbors of a linkpoint v (other than a self-looped singleton) by a new edge and remove v from the graph with its two incident edges.

A reduction graph of a valid graph G is not necessarily valid, since the reduction process may generate a cycle consisting of blackout vertices only. We will be interested in reduction sequences in which each transformation yields a valid graph.

Lemma 4 *Let G' be a reduction graph of G. If G' is valid, then $\mu_a(G') = \mu_a(G)$.*

Proof. Let $H_1 = G, H_2, \ldots, H_{t-1}, H_t = G'$ be a sequence of reduction graphs where each H_i is obtained by a removal of one linkpoint and possibly some endpoints from H_{i-1}. Since G' is valid, each H_i is a valid graph as well. Let v_i be the linkpoint that is removed from H_i to obtain H_{i+1}.

First we show that $\mu_a(G') \geq \mu_a(G)$. Suppose F is a feedback vertex set of H_{i+1} for some i, $1 \leq i < t$ and let Γ be a cycle in H_i that passes through v_i. A reduction of Γ obtained by replacing the linkpoint v_i on Γ by an edge connecting the neighbors of v_i yields a cycle $\hat{\Gamma}$ in H_{i+1}. The vertex set of $\hat{\Gamma}$ intersects the set F. Hence, F is also a feedback vertex set of H_i which implies that $\mu_a(H_{i+1}) \geq \mu_a(H_i)$.

Now we show that $\mu_a(G') \leq \mu_a(G)$. Suppose F is a minimal feedback vertex set of H_i. If F does not contain v_i, then F is also a feedback vertex set of H_{i+1}. Otherwise, write $F = \{v_i\} \cup F'$. We claim that the set F' cannot fail to cover more than one cycle in H_{i+1}. If it failed, then there would be two distinct cycles Γ_1 and Γ_2 in H_i that contain v_i, in which case the cycle in H_i induced by $(V(\Gamma_1) \cup V(\Gamma_2)) - \{v_i\}$ would not be covered by F, thus contradicting the fact that F is a feedback vertex set of H_i. It follows by this and the minimality of F that the set F' fails to cover *exactly* one cycle in H_{i+1}. This cycle contains at least one allowed vertex u because H_{i+1} is a valid graph. Therefore, the set $F' \cup \{u\}$ is a feedback vertex set of H_{i+1}. Hence, $\mu_a(H_{i+1}) \leq \mu_a(H_i)$. □

A reduction graph G^* of a graph G is *minimal* if and only if G^* is a valid graph and any proper reduction graph G' of G^* is not valid.

Lemma 5 *If G^* is a minimal reduction graph of G, then G^* does not contain blackout linkpoints, and every feedback vertex set of G^* contains all allowed linkpoints of G^*.*

Proof. Recall that G^* is a valid graph. If G^* contains a blackout linkpoint, then its removal creates a valid reduction graph which contradicts the minimality of G^*. Now assume F is a feedback vertex set of G^* and v is an allowed linkpoint which is not in F. If the removal of v yields a graph that is not valid, then v must have been included in F. If the removal of v yields a valid graph, then G^* is not minimal. □

The next lemma is needed in order to establish the performance ratio of SUBG-2-3. It is a variant of Lemma 4 in [Vo68].

Lemma 6 *Let G be a valid graph with no blackout linkpoints and such that no vertex has degree less than 2. Then, for every feedback vertex set F of G which contains all linkpoints of G,*

$$|V(G)| \;\leq\; (\Delta_a(G)+1)\,|F| \;-\; 2\,.$$

Proof.

Suppose $F = V(G)$. In this case we have $|V(G)| \leq 3|V(G)| - 2 \leq (\Delta_a(G)+1)|V(G)| - 2$ and, therefore, the lemma holds trivially. So we assume from now on that $|F| < |V(G)|$.

Let E_F denote the set of edges in $E(G)$ whose terminal vertices are all vertices in F. Define $X = V - F$ and let E_X denote the set of edges in $E(G)$ whose terminal vertices are all vertices in X. Also, let $E_{F,X}$ denote the set of those edges in G that connect vertices in F with vertices in X. Clearly, E_F, E_X, and $E_{F,X}$ form a partition on $E(G)$. Now, the graph obtained by deleting F from G is a nonempty forest on X and, therefore, $|E_X| \leq |X| - 1$. However, each vertex in X is a branchpoint in G because all linkpoints are assumed to be in F and there are no vertices of degree less than 2. Therefore,

$$3\,|X| \;\leq\; \sum_{v\in X} \Delta_G(v) \;=\; |E_{F,X}| \;+\; 2\,|E_X| \;\leq\; |E_{F,X}| \;+\; 2\,(|X|-1)$$

i.e.,

$$|E_{F,X}| \;\geq\; |X| \;+\; 2 \;=\; |V(G)| \;-\; |F| \;+\; 2\,.$$

On the other hand,

$$\Delta_a(G)\,|F| \;\geq\; \sum_{v\in F} \Delta_G(v) \;=\; |E_{F,X}| \;+\; 2\,|E_F|\,.$$

Combining the last two inequalities we obtain

$$|V(G)| \;\leq\; (\Delta_a(G)+1)\,|F| \;-\; 2\,|E_F| \;-\; 2\,. \qquad \square$$

The main claim of this section now follows.

Theorem 7 *The unweighted performance ratio of* SUBG-2-3 *is at most* $4 - (2/|V(G)|)$.

Proof. Let F be the feedback vertex set computed by SubG-2-3 for a valid graph G which is not a forest. We show that $|F| \leq 4\,\mu_a(G) - 2$. The theorem follows immediately from this inequality.

Let H, X, Y, and W be as in SubG-2-3. Suppose $\mu_a(G) = 1$. Then, all cycles in G pass through some allowed vertex v in G and, so, no vertex other than v can be a critical linkpoint in H. Now, if v is a linkpoint in H, then H is a cycle. Otherwise, one can readily verify that H must contain exactly two branchpoints. In either case we have $|F| \leq 2$. We assume from now on that $\mu_a(G) \geq 2$.

For every $v_i \in X$, let Γ_i be some witness cycle of v_i in G. By Lemma 2, the cycles Γ_i are pairwise independent. Therefore, the minimum number of vertices needed to cover such cycles is $|X|$.

Let $\{\Gamma_j^*\}$ be the set of branchpoint-free cycles in H that do not contain any critical linkpoints of H. Note that each cycle Γ_j^* is independent with any witness cycle Γ_i. We now claim that any smallest set W' of vertices of $V(H)$ that intersects with the vertex set of each Γ_j^* must be of size $|W|$. To see this, note that W' contains only allowed linkpoints of H. If we remove from H_b^* all the edges that correspond to linkpoints belonging to W', then we clearly end up with a forest. By construction, the minimum number of edges (or allowed linkpoints), needed to be removed from H_b^* so as to make it into a forest, is $|W|$.

Recalling that every cycle Γ_j^* is independent with any witness cycle Γ_i, the set W' cannot possibly intersect with any of the cycles Γ_i. Hence, in order to cover the cycles $\{\Gamma_i\} \cup \{\Gamma_j^*\}$ in G, we will need at least $|X| + |W'|$ vertices. Therefore,

$$\mu_a(G) \;\geq\; |X| + |W'| \;=\; |X| + |W|\,.$$

On the other hand, we recall that $|F| = |X| + |Y| + |W|$.

We distinguish between the following two cases.

Case 1: $|Y| \leq 2|X| + 2|W|$. Here we have,

$$|F| \;=\; |X| + |Y| + |W| \;\leq\; 3|X| + 3|W| \;\leq\; 3\mu_a(G) \;\leq\; 4\mu_a(G) - 2\,.$$

Case 2: $|Y| > 2|X| + 2|W|$. Let F^* be a feedback vertex set of G of size $\mu_a(G)$ and let W' be a smallest subset of F^* that intersects with the vertex set of each Γ_j^*. Clearly, W' consists of allowed linkpoints of H, and, as we showed earlier in this proof, $|W'| = |W|$. Let H_1 be the subgraph of H obtained by removing all critical linkpoints of H and all linkpoints in W'. With each deleted linkpoint, we also remove recursively all resulting endpoints from H while obtaining H_1. Thus, a deletion of a linkpoint from H can decrease the number of branchpoints by 2 at most. Hence, the number of branchpoints left in H_1 is at least $|Y| - 2|X| - 2|W| > 0$. Furthermore, the graph H_1 does not contain any endpoints.

Let H_1^* be a minimal reduction graph of H_1 and let H_2 be a valid graph obtained by removing all singleton components from H_1^*. Since H_1 does not contain any endpoints, the number of branchpoints of H_1 is preserved in H_1^* and in H_2. Therefore, the graph H_2 contains at least $|Y|-2|X|-2|W|$ branchpoints. On the other hand, since H_1^* is a minimal reduction and due to Lemma 5, there are no blackout linkpoints in H_1^* and every feedback vertex set of H_1^* contains all allowed linkpoints of H_1^*. Furthermore, the graphs H_1^* and H_2 do not contain any endpoints.

It follows that we can apply Lemma 6 to H_2 and any feedback vertex set of H_2, thus obtaining

$$|Y| - 2|X| - 2|W| \;\leq\; 4\mu_a(H_2) - 2 \;\leq\; 4\mu_a(H_1^*) - 2 \;=\; 4\mu_a(H_1) - 2\,,$$

where the equality is due to Lemma 4. Therefore,

$$\begin{aligned} |F| &= |X| + |Y| + |W| \\ &\leq 4|X| + 4|W| + |Y| - 2|X| - 2|W| \\ &\leq 4(|X| + |W| + \mu_a(H_1)) - 2\,. \end{aligned} \tag{1}$$

Recall that W' was chosen as a subset of a smallest feedback vertex set F^* of G. Let X' be a smallest subset of F^* that covers the witness cycles $\{\Gamma_i\}$ and let Z' be a smallest subset of F^* that covers the cycles of H_1. Since H_1 does not contain any of the critical linkpoints of H, each witness cycle Γ_i is independent with any cycle in H_1 and, so, we have $X' \cap Z' = \emptyset$. It also follows from our previous discussion that $X' \cap W' = \emptyset$. In addition, by construction of H_1 we have $W' \cap Z' = \emptyset$. It thus follows that

$$|X| + |W| + \mu_a(H_1) \;\leq\; |X'| + |W'| + |Z'| \;\leq\; |F^*| \;=\; \mu_a(G)\,.$$

Combining with (1), we obtain the desired result. □

3 Weighted Feedback Vertex Set

In this section, we consider the approximation of the WFVS problem described in Section 1. That is, given an undirected graph G and a weight function w on its vertices, find a feedback vertex set of (G, w) with minimum weight. As in the previous section, we assume that G may contain parallel edges and self-loops.

A *weighted reduction graph* G' of an undirected graph G is a graph obtained from G by a sequence of the following transformations:

- Delete an endpoint and its incident edge.

- Let u and v be two adjacent vertices such that $w(u) \leq w(v)$ and v is a linkpoint. Connect u to the other neighbor of v, and remove v from the graph with its two incident edges.

The following lemma can be easily verified. (See, e.g., the proof of Lemma 4).

Lemma 8 *Let (G', w') be a weighted reduction graph of (G, w). Then, $\mu(G', w') = \mu(G, w)$.*

A weighted reduction graph G^* of a graph G is *minimal* if and only if any weighted reduction graph G' of G^* is equal to G^*. A graph is called *branchy* if it has no endpoints and, in addition, its set of linkpoints induces an independent set, i.e., each linkpoint is either an isolated self-looped singleton or connected to two branchpoints. Clearly, any minimal weighted reduction graph must be branchy. We note that the complexity of transforming a graph into a branchy graph is linear in $|E(G)|$.

We are now ready to present our algorithms for finding an approximation for a minimum-weight feedback vertex set of a given weighted graph. In Section 3.1 we give an algorithm that achieves a performance ratio of $4 \log_2 |V(G)|$. In Section 3.2 we present an algorithm that achieves a performance ratio of $2\Delta^2(G)$.

3.1 The primal-dual algorithm

The basis of the first approximation algorithm is the next lemma which generalizes a lemma due to Erdős and Pósa [EP62, Lemma 3]. That lemma was obtained by Erdős and Pósa while estimating the smallest number of edges in a graph which contains a given number of pairwise independent cycles. Later on, in [EP64], they provided bounds on the value of $\mu(G, 1)$ in terms of the largest number of pairwise independent cycles in G. Tighter bounds on $\mu(G, 1)$ were obtained by Simonovits [Si67] and Voss [Vo68].

Lemma 9 *The shortest cycle in any branchy graph G with at least two vertices is of length $\leq 4 \log_2 |V(G)|$.*

Proof. Let t be the smallest even integer such that $2 \cdot 2^{t/2} > |V(G)|$. Apply BFS on G of depth t starting at some vertex v. We now claim that the search hits some vertex twice and so there exists a cycle of length $\leq 2t$ in G. Indeed, if it were not so, then the induced BFS tree would contain at least $2 \cdot 2^{t/2}$ distinct vertices of G, which is a contradiction. □

In each iteration of the proposed algorithm, we first find a minimal weighted reduction graph, and then find a cycle Γ with the smallest number of vertices in the minimal weighted reduction graph. The algorithm sets δ to be the minimum among the weights of the vertices in $V(\Gamma)$. This value of δ is subtracted, in turn, from the weight of each vertex in $V(\Gamma)$.

Vertices whose weight becomes zero are added to the feedback vertex set and deleted from the graph. Each such iteration is repeated until the graph is exhausted.

Algorithm MiniWCycle (*Input:* (G, w)**;** *Output:* **feedback vertex set** F **of** (G, w)**);**

$F \leftarrow \emptyset$; $(H, w_H) \leftarrow (G, w)$;
While H **is not a forest do begin:**
    Find a minimal weighted reduction graph $(H', w_{H'})$ **of** (H, w_H);
    Find a cycle Γ' **in** H' **with the smallest number of vertices;**
    Set $\delta \leftarrow \min_{v \in V(\Gamma')} w_{H'}(v)$;
    Set $w_{H'}(v) \leftarrow w_{H'}(v) - \delta$ **for every** $v \in V(\Gamma')$;
    Let $X = \{v \in V(\Gamma') \ : \ w_{H'}(v) = 0\}$;
    Remove X **(with all incident edges) from** H';
    $(H, w_H) \leftarrow (H', w_{H'})$;
    $F \leftarrow X \cup F$;
end.

Finding a shortest cycle can be done by running BFS from each vertex until a cycle is found and then selecting the smallest. A more efficient approach for finding the shortest cycle is described in [IR78].

It is not hard to see that MiniWCycle computes a feedback vertex set of G. We now analyze the algorithm employing techniques similar to those used in [Ho82], [Ho83], and [KhVY94]. We note that the algorithm can also be analyzed using the Local Ratio Theorem of Bar-Yehuda and Even [BaEv85].

Theorem 10 *The performance ratio of algorithm* MiniWCycle *is at most* $4 \log_2 |V(G)|$.

Proof. We assume that $|V(G)| > 1$. Given a feedback vertex set F of (G, w), let $\mathbf{x} = [x_v]_{v \in V(G)}$ be the indicator vector of F, namely, $x_v = 1$ if $v \in F$ and $x_v = 0$ otherwise. We denote by $\mathcal{C}$ the set of cycles in G. The problem of finding a minimum-weight feedback vertex set of (G, w) can be formulated in terms of $\mathbf{x}$ by an integer programming problem as follows:

$$\begin{array}{c} \text{minimize} \quad \sum_{v \in V(G)} w(v) \cdot x_v \\ \text{ranging over all nonnegative integer vectors } \mathbf{x} = [x_v]_{v \in V(G)} \text{ such that} \\ \sum_{v \in V(\Gamma)} x_v \ \geq \ 1 \quad \text{for every} \quad \Gamma \in \mathcal{C} \ . \end{array} \tag{2}$$

Let $\mathcal{C}_v$ denote the set of cycles passing through vertex v in G and consider the following

integer programming *packing* problem:

$$\begin{gathered}\text{maximize} \quad \textstyle\sum_{\Gamma\in\mathcal{C}} y_\Gamma \\ \text{ranging over all nonnegative integer vectors } \mathbf{y} = [y_\Gamma]_{\Gamma\in\mathcal{C}} \text{ such that} \\ \sum_{\Gamma\in\mathcal{C}_v} y_\Gamma \;\leq\; w(v) \quad \text{for every} \quad v \in V\,.\end{gathered} \tag{3}$$

Clearly, the linear relaxation of (3) is the dual of the linear relaxation of (2), with y_Γ, $\Gamma \in \mathcal{C}$, being the dual variables.

Let $(H', w_{H'})$ be a minimal weighted reduction graph computed at some iteration of algorithm MiniWCycle. Then, for each cycle $\Gamma' \in H'$, we associate a unique cycle $\Gamma \in G$ as follows: If all vertices in $V(\Gamma')$ belong to G, then $\Gamma = \Gamma'$. Otherwise, we "unfold" the transformation steps performed in obtaining H' from H in backward order, i.e., from H' back to H: In each such step we add to Γ' chains of linkpoints (connecting vertices in Γ') that were deleted. When this process finishes, the cycle Γ' of H' transforms into a cycle Γ of G.

We now show that MiniWCycle can be interpreted as a primal–dual algorithm. We first show that it computes a dual feasible solution for (3) with a certain maximality property. The initial dual feasible solution is the one in which all the dual variables y_Γ are zero.

Let Γ'_i be a cycle chosen at iteration i of MiniWCycle and let Γ_i be the associated cycle in G. We may view the computation of iteration i of MiniWCycle as setting the value of the dual variable y_{Γ_i} to the weight δ of a lightest vertex in $V(\Gamma'_i)$. The updated weight $w_{H'}(v)$ of every $v \in V(\Gamma'_i)$ is precisely the slack of the dual constraint

$$\sum_{\Gamma\in\mathcal{C}_v} y_\Gamma \;\leq\; w(v) \tag{4}$$

that corresponds to v.

It is clear that by the choice of δ, the values of the dual variables y_Γ at the end of iteration i of MiniWCycle satisfy the dual constraints (4) corresponding to vertices $v \in V(\Gamma'_i)$. It thus follows that the dual constraints hold for all vertices $v \in V(H')$ at iteration i.

Let v be a vertex that was removed from H to obtain H' in iteration i of MiniWCycle. It remains to show that the dual constraint (4) corresponding to such a vertex holds in each iteration j of the algorithm for every $j \geq i$.

We show this by backward induction on j. By the previous discussion it follows that the constraints corresponding to vertices that exist in the last iteration all hold. Suppose now that the dual constraints corresponding to vertices in $V(H')$ in iteration j are not violated. We show that the dual constraints corresponding to vertices in $V(H) - V(H')$ in that iteration are also not violated. Let c be a chain of linkpoints in H in iteration j,

and let v_1 and v_2 be the two branchpoints adjacent to c. Let u be a vertex of minimum weight among v_1, v_2, and the vertices in c. We note that the weighted reduction procedure deletes all vertices in c except possibly for one representative, depending on whether u is in c or is one of its adjacent branchpoints. We now observe that the set of cycles that pass through a linkpoint in c is the same for all linkpoints in c, and is contained in the set of cycles that pass through v_1, and is also contained in the set of cycles that pass through v_2. This implies that if the dual constraint corresponding to u is not violated, then the dual constraints corresponding to any vertex in c is also not violated.

The algorithm essentially constructs a primal solution $\mathbf{x}$ from the dual solution $\mathbf{y}$: It selects into the feedback vertex set all vertices for which: (i) the corresponding dual constraints are tight; and (ii) in the iteration the constraint first became tight, the corresponding vertex belonged to the graph. As stated earlier, this construction yields a feasible solution.

Let $\mathbf{x}^* = [x_v^*]_{v \in V(G)}$ and $\mathbf{y}^* = [y_\Gamma^*]_{\Gamma \in \mathcal{C}}$ denote the optimal primal and dual fractional solutions, respectively. It follows from the duality Theorem that

$$\sum_{v \in V(G)} w(v) \cdot x_v \;\geq\; \sum_{v \in V(G)} w(v) \cdot x_v^* \;=\; \sum_{\Gamma \in \mathcal{C}} y_\Gamma^* \;\geq\; \sum_{\Gamma \in \mathcal{C}} y_\Gamma \;. \tag{5}$$

Hence, to prove the theorem, it suffices to bound the ratio between the LHS and the RHS of (5). First note that $y_\Gamma \neq 0$ only for cycles Γ in G that are associated with cycles Γ' that were chosen at some iteration of MiniWCycle. By the above construction of $\mathbf{x}$, it is clear that the dual variable y_Γ of each such cycle Γ contributes its value to at most $V(\Gamma')$ vertices. Hence,

$$\sum_{v \in V(G)} w_v \cdot x_v \;\leq\; \sum_{v \in V(G)} \sum_{\Gamma \in \mathcal{C}_v} y_\Gamma \;\leq\; \sum_{\Gamma \in \mathcal{C}} y_\Gamma \cdot |V(\Gamma')| \;.$$

Now, in each iteration, the graph H' is a branchy graph. Therefore, by Lemma 9, we have that $|V(\Gamma')| \leq 4 \log_2 |V(G)|$. Hence the theorem is proved. □

Proposition 11 *For planar graphs, the weighted performance ratio of* MiniWCycle *is at most 10.*

Proof. We first notice that the weighted reduction process preserves planarity and, therefore, at each iteration of algorithm MiniWCycle we remain with a planar graph.

We claim that every rich planar graph G must contain a face of length at most 5. Assume the contrary. By summing up the lengths of all the faces, we get that $2|E| \geq 6|Z|$, where Z denotes the set of faces of G. By Euler's formula,

$$|E| - |V| + 2 = |Z|$$

Hence, $2|E| \leq 3|V| - 6$. However, since the degree of each vertex is at least 3, we get that $2|E| \geq 3|V|$, which is a contradiction. Furthermore, this implies that a branchy planar graph must contain a cycle of length at most 10. □

3.2 Low-degree graphs

The algorithm presented in this section is based on the following variant of Lemma 6.

Lemma 12 *Let G be a branchy graph. Then, for every feedback vertex set F of G,*

$$|V(G)| \ \leq \ 2\Delta^2(G) \cdot |F|$$

Proof. Let F be a feedback vertex set of G. We can assume without loss of generality that F contains only branchpoints, since this assumption can only decrease $|F|$. Let G' be the minimal (unweighted) reduction graph of G, i.e., G' contains only branchpoints or isolated self-looped singletons. Clearly, F is also a feedback vertex set of G'. Thus, G' and F satisfy the conditions of Lemma 6 ($\Delta_a = \Delta$), yielding that,

$$|V(G')| \ \leq \ (\Delta(G') + 1) \cdot \ |F| \ - \ 2 \, .$$

Since G' is a branchy graph, the number of linkpoints in G can be at most $\Delta(G') \cdot |V(G')|/2$. Hence,

$$|V(G)| \ \leq \ \frac{(\Delta(G) + 2) \cdot |V(G')|}{2}$$

and, so,

$$|V(G)| \ \leq \ \frac{(\Delta(G) + 2) \cdot (\Delta(G) + 1) \cdot |F|}{2} \ \leq \ 2\Delta^2(G) \cdot |F|$$

□

We now present a weighted greedy algorithm for finding a feedback vertex set in a graph G.

Algorithm WGreedy (*Input:* (G, w)**;** *Output:* **feedback vertex set** F **of** (G, w)**);**

$F \leftarrow \emptyset$; $i \leftarrow 1$; $(H, w_H) \leftarrow (G, w)$;
while H **is not a forest do begin:**
    Find a minimal weighted reduction graph $(H'_i, w_{H'_i})$ **of** (H, w_H);
    $\alpha_i \leftarrow \min_{v \in V(H'_i)} w_{H'_i}(v)$;
    $U_i \leftarrow \{u \in V(H'_i) \mid w_{H'_i}(u) = \alpha_i\}$;
    $F \leftarrow F \cup U_i$;
    remove U_i **from** H'_i **with its incident edges;**

$(H, w_H) \leftarrow (H'_i, w_{H'_i})$;
$i \leftarrow i + 1$;
end.

For a subset $S \subseteq V$, let $w(S)$ denote the sum of weights of the vertices in S. We now prove the following theorem.

Theorem 13 *Let G be a branchy graph. Denote by F the feedback vertex set computed by algorithm* WGREEDY, *and by F^* a minimum-weight feedback vertex set in G. Then, $w(F) \leq 2\Delta^2(G) \cdot w(F^*)$.*

Proof. Assume that the number of iterations the *while* loop is executed in algorithm WGREEDY is p. We define the following weight functions $w_1, \ldots, w_p$ on $V(G)$. The weight function w_i is defined, for $1 \leq i \leq p$, as follows:

$$\text{For all } v \in V(G): \quad w_i(v) = \begin{cases} \alpha_i - \alpha_{i-1} & \text{if } v \in V(H'_i) \\ 0 & \text{otherwise} \end{cases},$$

where $\alpha_0 = 0$.

For a subset S, let $w_i(S)$ denote the sum of weights of the vertices in S, where the weight function is w_i. Clearly,

$$w(F) \;=\; \sum_{i=1}^{p} w(U_i) \;=\; \sum_{i=1}^{p} w_i(F)$$

Suppose that at one of the weighted reduction steps of algorithm WGREEDY, a chain c of equal weight linkpoints was reduced to a single vertex, say, v, which either belongs to c or is one of the two branchpoints adjacent to c. Suppose further that v was added to F. If F^* also contains a vertex from the chain c, then without loss of generality, we can assume that this vertex can be replaced by v.

Let $u \in F^*$. Obviously, $u \in H'_1$. We claim that if $u \notin F$, then $u \in H'_i$ for all $i = 1, 2, \ldots, p$. Assume this is not the case. Then, with respect to the order in which vertices entered F in algorithm WGREEDY, let u be the first vertex such that $u \in F$, $u \notin F^*$, and u was removed from the graph in a weighted reduction step. This means that u was at the time of its removal a linkpoint that had an adjacent vertex u' with smaller weight. But then, by exchanging u for u' in F^*, we obtain a feedback vertex set which has smaller weight, contradicting the optimality of F^*. Hence, for a vertex $u \in F^*$, $w(u) \;\geq\; \sum_{i=1}^{p} w_i(u)$. Therefore,

$$w(F^*) \;\geq\; \sum_{i=1}^{p} w_i(F^*)\,.$$

Notice that in the graph H'_i, the weight function w_i assigns the same weight to all vertices. Hence, by Lemma 12, we have that $w_i(F) \leq 2\Delta^2(H'_i) \cdot w_i(F^*)$ for all $i = 1, 2, \ldots, p$. Since $\Delta(H'_i) \leq \Delta(G)$ for all i, the theorem follows. □

It follows from Lemma 8 that the performance ratio of algorithm WGREEDY for (G, w) is at most $2\Delta^2(G)$ for *any* graph G.

4 The Loop Cutset Problem and its Application

In section 4.1 we consider a variant of the WFVS problem for directed graphs and in section 4.2 we describe its application to Bayesian inference.

4.1 The loop cutset problem

The underlying graph of a directed graph D is the undirected graph formed by ignoring the directions of the edges in D. A *loop* in D is a subgraph of D whose underlying graph is a cycle. A vertex v is a *sink* with respect to a loop Γ if the two edges adjacent to v in Γ are directed into v. Every loop must contain at least one vertex that is not a sink with respect to that loop. Each vertex that is not a sink with respect to a loop Γ is called an *allowed vertex with respect to* Γ. A *loop cutset* of a directed graph D is a set of vertices that contains at least one allowed vertex with respect to each loop in D. Our problem is to find a minimum-weight loop cutset of a given directed graph D and a weight function w. We denote by $\mu(D, w)$ the sum of weights of the vertices in such a loop cutset. Greedy approaches to the loop cutset problem have been suggested by [SuC90] and [St90]. Both methods can be shown to have a performance ratio as bad as $\Omega(n/4)$ in certain planar graphs [St90]. An application of our approximation algorithms to the loop cutset problem in the area of Bayesian inference is described later in this section.

The approach we take is to reduce the weighted loop cutset problem to the weighted feedback vertex set problem solved in the previous section. Given a weighted directed graph (D, w), we define the *splitting* weighted undirected graph (D_s, w_s) as follows. Split each vertex v in D into two vertices v_{in} and v_{out} in D_s such that all incoming edges to v become undirected incident edges with v_{in}, and all outgoing edges from v become undirected incident edges with v_{out}. In addition, we connect v_{in} and v_{out} by an undirected edge. Set $w_s(v_{\text{in}}) = \infty$ and $w_s(v_{\text{out}}) = w(v)$. For a set of vertices X in D_s, we define $\psi(X)$ as the set obtained by replacing each vertex v_{in} or v_{out} in X by the respective vertex v in D from which these vertices originated.

Our algorithm can now be easily stated.

Algorithm LoopCutset (*Input:* (D, w)**;** *Output:* **loop cutset** F **of** (D, w)**);**

Construct (D_s, w_s)**;**

Apply MiniWCycle **on** (D_s, w_s) **to obtain a feedback vertex set** X**;**
$F \leftarrow \psi(X)$**.**

Note that each loop in D is associated with a unique cycle in D_s, and vice-versa, in a straightforward manner. Let $I(\Gamma)$ denote the loop image of a cycle Γ in D_s, and $I^{-1}(K)$ denote the cycle image of a loop K in D. It is clear that the mapping I is $1-1$ and onto.

The next lemma shows that algorithm LoopCutset outputs a loop cutset of (D, w).

Lemma 14 *Let (D, w) be a directed weighted graph and (D_s, w_s) be its splitting graph. Then: (i) If F is a feedback vertex set of (D_s, w_s) having finite weight, then $\psi(F)$ is a loop cutset of (D, w), and $w_s(F) = w(\psi(F))$. (ii) If U is a loop cutset of D, then the set U_s obtained from U by replacing each vertex $v \in U$ by vertex $v_{\text{out}} \in D_s$ is a feedback vertex set of D_s, and $w(U) = w_s(U_s)$.*

Proof. We prove (i). The proof of (ii) is similar. Let Γ be a loop in D. To prove the lemma we show that an allowed vertex with respect to Γ belongs to $\psi(F)$. Let $I^{-1}(\Gamma)$ be the unique cycle image of Γ in D_s. Since F is a cycle cover of D_s having finite weight, there must be a vertex $v_{\text{out}} \in F$ in $I^{-1}(\Gamma)$. Now, it is clear that vertex $v \in \Gamma$ from which v_{out} originated is an allowed vertex with respect to Γ as needed. To complete the proof, by the finiteness of $w_s(F)$, we must have $w_s(F) = w(\psi(F))$, since $w_s(v_{\text{out}}) = w(v)$ for each vertex in F. □

It follows from Lemma 14 that $\mu(D, w) = \mu(D_s, w_s)$. In addition, due to Theorem 10 applied to the graph D_s, and since the number of vertices in D_s is twice the number of vertices in D, we get the following bound on the performance ratio of algorithm LoopCutset.

Theorem 15 *The performance ratio of* LoopCutset *is at most* $4\log_2(2|V(D)|)$.

We now show that in the unweighted loop cutset problem, we can achieve a performance ratio better than 4. In this case, for each vertex $v \in D$, the weight of $v_{\text{in}} \in D_s$ is one unit, and the weight of $v_{\text{out}} \in D_s$ is ∞. This falls within the framework considered in Section 2, since vertices with infinite weight in D_s can be treated as blackout vertices. We can therefore apply SubG-2-3 in the LoopCutset algorithm instead of applying MiniWCycle and obtain the following improved performance ratio.

Theorem 16 *When using* SubG-2-3*, the unweighted performance ratio of* LoopCutset *is at most* $4 - (2/|V(D)|)$.

Proof. We have,

$$w(\psi(F)) = w_s(F) \leq 4\mu(D_s, w_s) - 2$$

where the equality is due to Lemma 14, and the inequality is due to Theorem 7. Since

$\mu(D_s, w_s) = \mu(D, w) \leq |V(D)|$, the claim is proved. □

4.2 An application

We conclude this section with an application of approximation algorithms for the loop cutset problem.

Let $P(u_1, \ldots, u_n)$ be a probability distribution where each u_i draws values from a finite set called the *domain* of u_i. A directed graph D with no directed cycles is called a *Bayesian network of* P if there is a 1–1 mapping between $\{u_1, \ldots, u_n\}$ and vertices in D, such that u_i is associated with vertex i and P can be written as follows:

$$P(u_1, \ldots, u_n) = \prod_{i=1}^{n} P(u_i \mid u_{i_1}, \ldots, u_{i_{j(i)}}) \,, \tag{6}$$

where $i_1, \ldots, i_{j(i)}$ are the source vertices of the incoming edges to vertex i in D.

It is worth noting that Bayesian networks are useful knowledge representation schemes for many artificial intelligence tasks. Bayesian networks allow a wide spectrum of independence assumptions to be considered by a model builder so that a practical balance can be established between computational needs and adequacy of conclusions. For a complete exploration of this subject see [Pe88].

Suppose now that some variables $\{v_1, \ldots, v_l\}$ among $\{u_1, \ldots, u_n\}$ are assigned specific values $\{\mathbf{v}_1, \ldots, \mathbf{v}_l\}$ respectively. The *updating problem* is to compute the probability $P(u_i \mid v_1 = \mathbf{v}_1, \ldots, v_l = \mathbf{v}_l)$ for $i = 1, \ldots, n$. In principle, such computations are straightforward because each Bayesian network defines the joint probability distribution $P(u_1, \ldots, u_n)$ from which all conditional probabilities can be computed by dividing the appropriate sums. However, such computations are inefficient both in time and space unless they use conditional independence assumptions defined by Eq. (6). We shall see next how our approximation algorithms for the loop cutset problem reduce the computations needed for solving the updating problem.

A *trail* in a Bayesian network is a subgraph whose underlying graph is a simple path. A vertex b is called a *sink* with respect to a trail t if there exist two consecutive edges $a \rightarrow b$ and $b \leftarrow c$ on t. A trail t is *active by a set of vertices* Z if (1) every sink with respect to t either is in Z or has a descendant in Z and (2) every other vertex along t is outside Z. Otherwise, the trail is said to be *blocked* by Z.

Verma and Pearl [VePe88] have proved that if D is a Bayesian network of $P(u_1, \ldots, u_n)$ and all trails between a vertex in $\{r_1, \ldots, r_l\}$ and a vertex in $\{s_1, \ldots, s_k\}$ are blocked by $\{t_1, \ldots, t_m\}$, then the corresponding sets of variables $\{u_{r_1}, \ldots, u_{r_l}\}$ and $\{u_{s_1}, \ldots, u_{s_k}\}$ are independent conditioned on $\{u_{t_1}, \ldots, u_{t_m}\}$. Furthermore, Geiger and Pearl [GP90] proved

a converse to this theorem. Both results are presented and extended in [GVP90].

Using the close relationship between blocked trails and conditional independence, Kim and Pearl [KiP83] developed an algorithm UPDATE-TREE that solves the updating problem on Bayesian networks in which every two vertices are connected with at most one trail. UPDATE-TREE views each vertex as a processor that repeatedly sends messages to each of its neighboring vertices. When equilibrium is reached, each vertex i contains the conditional probability distribution $P(u_i \mid v_1 = \mathbf{v}_1, \ldots, v_l = \mathbf{v}_l)$. The computations reach equilibrium regardless of the order of execution in time proportional to the length of the longest trail in the network.

Pearl [Pe86] solved the updating problem on any Bayesian network as follows. First, a set of vertices S is selected, such that any two vertices in the network are connected by at most one *active* trail in $S \cup Z$, where Z is any subset of vertices. Then, UPDATE-TREE is applied once for each combination of value assignments to the variables corresponding to S, and, finally, the results are combined. This algorithm is called the method of *conditioning* and its complexity grows exponentially with the size of S. Note that according to the definition of active trails, the set S in Pearl's algorithm is a loop cutset of the Bayesian network. In this paper we have developed approximation algorithms for finding S.

When the domain size of the variables varies, then UPDATE-TREE is called a number of times which is bounded from above by the product of the domain sizes of the variables whose corresponding vertices participate in the loop cutset. If we take the logarithm of the domain size as the weight of a vertex, then solving the weighted loop cutset problem with these weights optimizes Pearl's updating algorithm in the case where the domain sizes are allowed to vary.

5 Discussion

It is useful to relate the feedback vertex set problem with the vertex cover problem in order to establish lower bounds on the performance ratios attainable for the feedback vertex set problem. A vertex cover of an undirected graph is a subset of the vertex set that intersects with each edge in the graph. The vertex cover problem is to find a minimum weight vertex cover of a given graph. There is a simple polynomial reduction from the vertex cover problem to the feedback vertex set problem: Given a graph G, we extend G to a graph H by adding a vertex v_e for each edge $e \in E(G)$, and connecting v_e with the vertices in G with which e is incident in G. It is easy to verify that there always exists a minimum feedback vertex set in H whose vertices are all in $V(G)$ and this feedback vertex set is also a minimum vertex cover of G. In essence, this reduction replaces each edge in G with a cycle in H, thus transforming any vertex cover of G to a feedback vertex set of H.

Due to this reduction, it follows that the performance ratio obtainable for the feedback vertex set problem cannot be better than the one obtainable for the vertex cover problem. The latter problem has attracted a lot of attention over the years but has so far resisted any approximation algorithm that achieves in general graphs a constant performance ratio less than 2. We note that the above reduction retains planarity. However, for planar graphs, Baker [Bak94] provided a Polynomial Approximation Scheme (PAS) for the vertex cover problem. For the UFVS problem, there are examples showing that 4 is the tightest constant performance ratio of algorithm SubG-2-3.

Another consequence of the above reduction is a lower bound on the unweighted performance ratio of the following greedy algorithm, GreedyCyc, for the feedback vertex set problem. In each iteration, GreedyCyc removes a vertex of maximal degree from the graph, adds it to the feedback vertex set, and removes all endpoints in the graph. A similar greedy algorithm for the vertex cover problem is presented in [Jo74] and in [Lo75]. The latter algorithm was shown to have an unweighted performance ratio no better than $\Omega(\log |V(G)|)$ [Jo74]. Due to the reduction to the cycle cover problem, the same lower bound holds also for GreedyCyc, as demonstrated by the graphs of [Jo74]. A tight upper bound on the worst-case performance ratio of GreedyCyc is unknown.

Finally, one should notice that the following heuristics may improve the performance ratios of our algorithms. For example, in each iteration MiniWCycle chooses to place in the cover all zero-weight vertices found on the smallest cycle. This choice might be rather poor especially if many weights are equal. It may be useful in this case to perturb the weights of the vertices before running the algorithm. Similarly, in algorithm SubG-2-3, there is no point in taking blindly all branchpoints of H. An appropriate heuristic here may be to pick the branchpoints one by one in decreasing order of residual degrees. Furthermore, the subgraph H itself should be constructed such that it contains as many high degree vertices as possible.

Remark

In a preliminary version of this paper, presented in [BaGNR94], we conjectured that a constant performance ratio is attainable by a polynomial time algorithm for the WFVS problem. This has been recently verified in [BeG94, BaBF94] where a performance ratio of 2 has been obtained.

Acknowledgment

We would like to thank David Johnson for bringing [EP62] to our attention, and Samir Khuller for helpful discussions.

References

[BaBF94] Bafna V., Berman P., and Fujito T., *Constant ratio approximations of the weighted feedback vertex set problem for undirected graphs,* to appear in *Proceedings of ISAAC 96.*

[Bak94] Baker B.S., *Approximation algorithms for NP-complete problems on planar graphs, Journal of the ACM*, 41 (1994), 153–180.

[BaEv85] Bar-Yehuda R. and Even S., *A local-ratio theorem for approximating the weighted vertex cover problem, Annals of Discrete Mathematics,* 25 (1985), 27–46.

[BaGNR94] Bar-Yehuda R., Geiger D., Naor J., and Roth R.M., *Approximation Algorithms for the Feedback Vertex Set Problem with Applications to Constraint Satisfaction and Bayesian Inference, Proceedings 5th Annual ACM-SIAM Symposium on Discrete Algorithms,* Arlington, Virginia, 1994, 344–354.

[BeG94] Becker A. and Geiger D., *Approximation algorithms for the loop cutset problem, Proceedings of the tenth conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, Seattle, 1994, 60–68.

[DP87] Dechter R. and Pearl J., *The cycle cutset method for improving search performance in AI, Proceedings 3rd IEEE on AI Applications,* Orlando, 1987.

[DP88] Dechter R. and Pearl J., *Network-based heuristics for constraint satisfaction problems, Artificial Intelligence,* 34 (1988), 1–38.

[De90] Dechter R., *Enhancement schemes for constraint processing: backjumping, learning, and cutset decomposition, Artificial Intelligence,* 41 (1990), 273–312.

[EP62] Erdős P. and Pósa L., *On the maximal number of disjoint circuits of a graph, Publ. Math Debrecen*, 9 (1962), 3–12.

[EP64] Erdős P. and Pósa L., *On the independent circuits contained in a graph, Canad. J. Math,* 17 (1964), 347–352.

[GJ79] Garey M.R. and Johnson D.S., *Computers and Intractability: A Guide to the Theory of NP-completeness,* W. H. Freeman, San Francisco, California, 1979.

[GP90] Geiger, D. and Pearl, J., *On the logic of causal models,* In *Uncertainty in Artificial Intelligence 4,* Eds. Shachter R.D., Levitt T.S., Kanal L.N., and Lemmer J.F., North-Holland, New York, 1990, 3–14.

[GVP90] Geiger, D., Verma, T.S., and Pearl, J., *Identifying independence in Bayesian networks, Networks,* 20 (1990), 507–534.

[Ho82] Hochbaum D.S., *Approximation algorithms for set covering and vertex covering problems, SIAM J. Computing,* 11 (1982), 555–556.

[Ho83] Hochbaum D.S., *Efficient bounds for the stable set, vertex cover, and set packing problems, Discrete Applied Math,* 6 (1983), 243–254.

[IR78] Itai A. and Rodeh M., *Finding a minimum circuit in a graph, SIAM J. Computing,* 7 (1978), 413–423.

[Jo74] Johnson D.S., *Approximation algorithms for combinatorial problems, J. Comput. Sys. Sciences,* 9 (1974), 256–278.

[KhVY94] Khuller S., Vishkin U., and Young, N., *A primal-dual parallel approximation technique applied to weighted set and vertex cover, Journal of Algorithms*, 17 (1994) 280–289.

[KiP83] Kim H. and Pearl J., *A computational model for combined causal and diagnostic reasoning in inference systems,* In *Proceedings of the Eighth IJCAI*, Morgan-Kaufmann, San Mateo, California, 1983, 190–193.

[Lo75] Lovász L., *On the ratio of optimal integral and fractional covers, Discrete Math.,* 13 (1975), 383–390.

[MS81] Monien B.and Schulz R., *Four approximation algorithms for the feedback vertex set problem,* In *Proceedings of the 7th Conference on Graph Theoretic Concepts of Computer Science*, 1981, 315-326.

[Pe86] Pearl, J., *Fusion, propagation and structuring in belief networks, Artificial Intelligence,* 29:3 (1986), 241–288.

[Pe88] Pearl, J., *Probabilistic reasoning in intelligent systems: Networks of plausible inference.* Morgan Kaufmann, San Mateo, California, 1988.

[Si67] Simonovits M., *A new proof and generalizations of a theorem by Erdős and Pósa on graphs without $k+1$ independent circuits, Acta Mathematica Academiae Hungaricae Tomus,* 18 (1967), 191–206.

[St90] Stillman, J., *On heuristics for finding loop cutsets in multiply connected belief networks*, In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, Cambridge, Massachusetts, 1990, 265–272.

[SuC90] Suermondt H.J. and Cooper G.F., *Probabilistic inference in multiply connected belief networks using loop cutsets, Int. J. Approx. Reasoning*, 4 (1990), 283–306.

[VePe88] Verma, T. and Pearl, J., *Causal networks: Semantics and expressiveness,* In *Proceedings of Fourth Workshop on Uncertainty in Artificial Intelligence,* Minneapolis, Minnesota (published by the Association for Uncertainty in Artificial Intelligence, Mountain View, California), 1988, 352–359.

[Vo68] Voss H.J., *Some properties of graphs containing k independent circuits, Proc. Colloq. Tihany*, Academic Press, New York, 1968, 321–334.