DIMACS Technical Report 96-29
July 1996

Constant Ratio Approximations of Feedback Vertex Sets in Weighted Undirected Graphs [1]

by

Vineet Bafna[2,3]
(bafna@dimacs.rutgers.edu)
DIMACS Center, Piscataway, NJ 08855 USA

Piotr Berman
(berman@cse.psu.edu)
Dept. of Computer Science and Engineering
The Pennsylvania State University, University Park, PA 16802 USA

Toshihiro Fujito
(fujito@huis.hiroshima-u.ac.jp)
Dept. of Electrical Engineering, Hiroshima University
1–4–1 Kagamiyama, Higashi–Hiroshima 739 JAPAN

[1] A preliminary version presented at the 6th Annual International Symposium on Algorithms and Computation (Australia, December 1995).
[2] Postdoctoral fellow
[3] Supported by Special Year National Science Foundation grant BIR-9412594.

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

A feedback vertex set of a graph is a subset of vertices that contains at least one vertex from every cycle in the graph. We show that a feedback vertex set approximating a minimum one within a constant factor can be efficiently found in undirected graphs. In fact the derived approximation ratio matches the best constant ratio known today for the vertex cover problem, improving the previous best results for both weighted and unweighted cases. The existence of an approximation preserving reduction of the vertex cover problem to the feedback vertex set problem suggests that further improvement of the obtained ratio will require entirely new ideas.

We extend our approach to handle graphs of bounded vertex degree, and show that it achieves even better performance, $2 - \frac{2}{3\Delta - 2}$, where Δ is the maximum degree of graphs.

These approximation bounds are obtained by using a non–trivial generalization of the classical local ratio principle of Bar–Yehuda and Even.

1 Introduction

For an undirected graph $G = (V, E)$, a set $F \subseteq V$ of vertices is a *feedback vertex set* (FVS for short) if it contains at least one vertex from every cycle in G. In general every vertex u of G is associated with *weight* (or *cost*) $w(u)$, which we can assume to be of a nonnegative value. The *feedback vertex set problem* is to find among all the FVS's in a given graph the one with smallest cardinality if w is some constant function on V and otherwise, the one with minimum sum of weights of vertices in it. This problem is of fundamental importance in combinatorial optimization. A prototypical application appears, as suggested by the name, in the context of combinatorial circuit design. The circuits are represented by graphs in which cycles potentially imply a "racing condition", that is, some circuit element might receive new inputs before it stabilizes. One way of avoiding such a condition is by placing a clocked register at each cycle in the circuit, and in that case, we would like to keep the number of clocked registers as low as possible. The minimum FVS for the graph gives a bound on the number of registers needed. See Bar–Yehuda et al. [3] for other applications in the areas of constraint satisfaction problems and Bayesian inference.

Unfortunately, the feedback vertex set problem is *NP*–hard; for directed graphs Karp showed its NP–completeness even if graphs are unweighted [9], and essentially the same transformation shows that it is equally hard for undirected graphs. Confronted with this (apparent) intractability we are lead to a next best approach: a polynomial time algorithm for computing a near optimal FVS. The quality of such an algorithm is measured by its *performance ratio*: the worst case ratio of weight of an approximate solution found by the algorithm to the optimal solution weight. The first nontrivial approximation ratio of $\log n$ was obtained early by Erdös and Pósa for unweighted graphs with n vertices [5] when they studied the number of (vertex) disjoint cycles in a graph. It was later improved to $\sqrt{\log n}$ by Monien and Schulz in 1981 [12] who considered and compared various approaches to the problem. Only recently, Bar–Yehuda, Geiger, Naor and Roth were able to show that the smallest cardinality FVS can be approximated within constant ratio of 4 [3]. Moreover, they also considered the weighted version and in this case they obtained a performance ratio of $\min\{4 \log n, 2\Delta^2\}$ where Δ is the maximum vertex degree of a graph. In this paper, we will show that constant ratio approximation is possible even for the *weighted* FVS problem, and in fact, better than theirs for the *unweighted* case; namely, the performance guarantee of twice the optimum. Our algorithm is quite efficient and it can be implemented to run in time $O(|E| + |V| \log |V|)$, using the same data structures as Dijkstra's single–source shortest paths algorithm. Recently, Becker and Geiger [4] also show a performance of 2 with a different analysis that does not easily extend to the bounded degree case.

It is also worth noting that achieving a performance ratio better than 2 might be computationally hard. The FVS problem is a typical node-deletion problem for a hereditary property (i.e., acyclic in this case) [10], and a general hardness result for this class of problems is known in the recent framework of complexity of approximating *NP*–hard optimization problems [11]: The node–deletion problem for every nontrivial hereditary property is *MAX SNP*–hard [14]; thus, it does not have a polynomial time approximation scheme unless P

= *NP* [1]. In fact, the Vertex Cover problem (VC for short) can be reduced more directly to the FVS problem in an approximation preserving manner, so that any performance ratio $r < 2$ for the latter problem would imply the same ratio r for the former problem. A better approximation of the VC problem has been a subject of extensive research over the years, yet the best constant approximation ratio has remained same at 2 (and the currently best performance ratio for the VC problem is $2 - \frac{\log\log n}{\log n}$ [2, 13]).

Our approach is based on the "local approximation" principle, which in a most simple form appeared already in the Gavril's maximal matching approximation for the unweighted VC problem (cited in [6, page 134]) and which was explicitly formalized as the local ratio theorem by Bar–Yehuda and Even in [2]. The use of this principle, however, has been usually limited to extraction of subgraphs of "uniform" weights from either unweighted or weighted graphs; e.g., Bar–Yehuda et al. applied the theorem in extracting short cycles in uniform weights to obtain the aforementioned performance ratio of $O(\log n)$ for the weighted FVS problem [3]. Important technical contributions of this paper include a generalization of this theorem and demonstration of the effectiveness resulting from more flexible applications of it; i.e., we show that significantly better approximations can be achieved by "choosing" weights of our own for local structures. It is our hope that this technique is general and can be applied to other weighted optimization problems. It is instructive to note also that our approach is a variant of the linear program based primal–dual methods, and the results of the present paper indicate that a non–standard LP formulation suggested herein could lead to a significant improvement in performance of approximating hard combinatorial problems. See an excellent survey in [7] on the account of recent (very successful) development of this approach to *NP*–hard optimization (and network design in particular) problems.

We also study good approximation of the weighted FVS problem restricted to graphs of bounded vertex degree Δ. This version remains *NP*–hard as well as *MAX SNP*–hard when $\Delta \geq 4$. Unfortunately our algorithm for general graphs does no better in this special version. We elaborate our approach by separating some special cases and introducing general approximation preserving reductions for this problem. It will be then shown that a performance ratio improves to $2 - \frac{2}{3\Delta - 2}$.

Currently the best algorithm approximates the minimum weight VC in bounded degree graphs within a factor of $2 - \frac{3}{2\Delta + 4}$ [8]. It remains an open question whether one can approximate the FVS problem with *exactly* the same ratio as the ones for the VC problem, whether the degree is bounded or not.

The paper is structured as follows. After some basic definitions and notation used throughout the paper are given in the next section we consider the FVS problem for general weighted graphs in Section 3. A key approximation principle is presented in the form of "local ratio theorem", an algorithm based on it is described, and analysis of its correctness and performance ratio follows. In Section 4 the problem is restricted to the one for graphs of bounded degree. We analyze first why the algorithm of Section 3 does no better in this special case and clarify what is needed to overcome its shortcomings. Some general reduction methods are introduced and it will be shown that they preserve a desired approximation ratio. These reductions are in fact used as prerequisites to the local ratio based approximation

method which we develop in Section 4.3 by extending the algorithm of Section 3. Finally all of these are combined to yield a new algorithm, which we show, when applied to a graph of bounded degree, gives a desired performance ratio.

2 Definitions and Notation

Denote a weighted graph as $G = (V, E, \mathrm{w})$, where V is the set of vertices, E is the set of edges and w is a weight function defined on V. We will consider subgraphs of G "induced by" weight functions. Let $\mathbb{R}_+$ be the set of nonnegative reals. Given $w : V \to \mathbb{R}_+$, let $V(w) \stackrel{\text{def}}{=} \{u \in V | w(u) > 0\}$. A subgraph of G induced by a vertex set U is denoted by $G(U)$, and $G(w) \stackrel{\text{def}}{=} G(V(w))$. By $w(U)$ we denote the sum of weights of vertices in U. The degree of a vertex u is denoted by $d(u)$; when it is necessary to specify a subgraph in which the degree is defined we use $d(u, U)$ to denote the degree of u in $G(U)$. For a graph G, $opt(G)$ denotes any optimal FVS in G.

Given below are some basic concepts used in this paper:

- For $G = (V, E)$ and $w : V \to \mathbb{R}_+$ a collection of weight functions $\{w_i, i = 1, \cdots, k\}$ defined on V is a *decomposition* of w if $\sum_i w_i(u) \leq w(u)$ for each $u \in V$.
- A weight function w is *degree–proportional* if for some constant $c > 0$ we have $w(u) = c(d(u, V(w)) - 1)$ for every $u \in V(w)$.
- A graph G is *clean* if it contains no vertex of degree less than 2. A cycle C in a graph $G(V)$ is *semi–disjoint* if for every vertex u of C, with at most one exception, $d(u, V) = 2$. We usually refer to such cycles when graphs are clean.

3 Approximating within Twice the Optimum

In this section we first present our algorithm approximating an optimal FVS in a given weighted graph. The algorithm is designed to construct a solution such that it satisfies certain nice properties. Such properties will then be used to show next that its weight is indeed guaranteed to be within twice that of the optimal solution.

The basis of our algorithm and its analysis is an approximation principle suggested in the following generalization of local ratio theorem of Bar–Yehuda and Even [2]:

Theorem 1 (Local Ratio Theorem) *For $G = (V, E, \mathrm{w})$ let $\{w_i, i = 1, \cdots, k\}$ be a decomposition of* w *and F be any FVS in G s.t.* $\mathrm{w}(F) = \sum_i w_i(F)$. *Then,*

$$\frac{\mathrm{w}(F)}{\mathrm{w}(opt(G(\mathrm{w})))} \leq \max_i \{\frac{w_i(F)}{w_i(opt(G(w_i)))}\}$$

Proof. Since $\{w_i\}$ is a decomposition of w, for any set $X \subseteq V$, $\mathrm{w}(X) \geq \sum_i w_i(X)$. Besides, $G(w_i)$ is a subgraph of $G(\mathrm{w})$ for every i, and if F is an FVS in a graph G so is its restriction to G' in G' for any subgraph G' of G. Therefore, we have $w_i(opt(G(\mathrm{w}))) \geq w_i(opt(G(w_i))$ for all i. It follows now that

$$\frac{\mathrm{w}(F)}{\mathrm{w}(opt(G(\mathrm{w})))} \leq \frac{\sum_i w_i(F)}{\sum_i w_i(opt(G(\mathrm{w}))} \leq \frac{\sum_i w_i(F)}{\sum_i w_i(opt(G(w_i))} \leq \max_i \{\frac{w_i(F)}{w_i(opt(G(w_i))}\}$$

□

In this way the ratio of an FVS weight to the optimal one in an original graph G weighted by w can be reduced to the ones in subgraphs weighted by w_i.

3.1 Algorithm

A basic idea behind our algorithm is to decompose a given graph into a sequence of weighted subgraphs and simultaneously build a solution which, when restricted to any derived subgraph, is a "good" approximation to an optimal one in the subgraph. The local ratio theorem will then ensure that an obtained solution is indeed a "good" approximation to a global optimum.

To be more specific we decompose a given graph into degree–proportionally weighted subgraphs $G(w_i)$; given a current graph $G(w)$, compute w_i, "maximally" degree–proportional under w, to obtain a subgraph $G(w_i)$, and recurse to the remaining graph $G(w-w_i)$. Because w_i is taken maximally some vertex has its weight reduced to zero in this step (the vertex is *saturated*), and hence, $G(w-w_i)$ must be a proper subgraph of $G(w)$ and the process must terminate after at most $|V|$ recursions. Meanwhile we are (implicitly) "half–committed" to those saturated vertices for inclusion into a solution; if we always take them all, however, we might end up with a large solution, and here comes the property we'd like to ensure on our solution: being minimal in the subgraph $G(w_i)$. This is quite easy to accomplish for, given a minimal solution F in the remaining $G(w-w_i)$ we can "minimally" augment F to a solution in $G(w)$ by adding to F only necessary saturated vertices (i.e., those in $V(w)-V(w-w_i)$). Thus, the procedure described above computes an FVS F and a decomposition $\{G(w_i)\}$ of an input $G(\mathrm{w})$ in such a way that, for all i,

1. w_i is degree–proportional, and
2. $F \cap V(w_i)$ is a minimal solution in $G(w_i)$.

It follows then from Theorem 1 that the approximation ratio of F in a given graph is at least as good as that of any minimal solution in an arbitrary degree–proportionally weighted graph, and if we can show that the latter always stays within 2, we're done. Unfortunately it is not the case (and it is easy to construct such an example yielding only a factor of 4). As will be seen later, however, such a bad event occurs only when a degree–proportionally weighted and *clean* graph contains a semi–disjoint cycle in it, and this fact suggests the following modification of our algorithm: whenever the current clean graph $G(w)$ contains a

semi–disjoint cycle C extract it from $G(w)$ by setting $w_i(u)$ to a minimum $w(v)$, over all the vertices v in C, if u is a vertex of C and to zero otherwise. The algorithm ensures that a solution F is minimal in every $G(w_i)$ as before.

These ideas are implemented as displayed in Figure 1. A call of the recursive function Feedback(w) computes an FVS for $G(w)$. At the beginning of the call the graph $G(w)$ is "cleaned up" by recursively removing vertices with degree ≤ 1 (implicitly, by setting their weight to 0 in w'), as they are surely not needed in an FVS. Next, w'' is obtained by calling Reduce(w', w'') which extracts a degree–proportional weight function w_i from w'. If $G(w')$ has a semi-disjoint-cycle C, then $V(w_i)$ is the set of vertices in C. Otherwise, $V(w_i) = V(w')$, and w_i is the largest degree–proportional function bounded by w'. The vertices of $V(w') - V(w'')$ become "candidates" for inclusion in the FVS being computed. The output FVS is computed by joining those vertices to the set returned by Feedback(w'') but by discarding any candidate if it is found to be redundant (see the test in the **for**–loop).

3.2 Performance Guarantee

We establish first that an FVS F_{out} computed by our algorithm satisfies some elementary properties:

1. The algorithm implies a proper decomposition of an input weight s.t. the weight of F_{out} equals to the sum of its weights over the derived subgraphs (Lemma 2), and
2. F_{out} is a minimal FVS in every derived subgraph (Lemma 3).

For an input graph $G = (V, E, \mathrm{w})$ and an output F_{out} of the algorithm, the code of Feedback() easily implies the following, which will be used later to apply Theorem 1.

Lemma 2 *1.* $\mathrm{w(u)} \geq \sum_i w_i(u)$ *for all* u *in* V.

2. $\mathrm{w}(F_{out}) = \sum_i w_i(F_{out})$.

Proof. The initial argument of Feedback() is the input weight function w. Later, the argument weight of the recursive call of Feedback() evolves in two ways. First, in the **while**–loop, vertices of degree 1 or 0 have the weight reduced to 0; these vertices will never be included in F_{out} (in other words a vertex u can be in F_{out} only if it gets saturated). Second, in Reduce(), w_i is subtracted. This implies the claim because the final argument of Feedback() is a weight function $w \equiv 0$. □

Similarly, it is easy to prove the following by induction on depth of the recursion and case analysis.

Lemma 3 $F_{out} \cap V(w_i)$ *is a minimal cardinality FVS in* $G(w_i)$ *for all* i.

Proof. First we show by induction on depth of the recursion that Feedback(w) returns F that is minimal in $G(w)$ (instead of $G(w_i)$). The basis, for $V(w) = \emptyset$, is obvious. Inductively,

Input: a weighted graph $G = (V, E, \mathrm{w})$
Output: an FVS F_{out} for G

main()
$i \leftarrow 0$ /* i is used for analysis only */
return Feedback(w)$\cup(V - V(\mathrm{w}))$

procedure Feedback(w)
begin
$i \leftarrow i + 1$
if $V(w) = \emptyset$ **then**
  return $\emptyset$
else begin
  $w'(u) \leftarrow w(u)$ for all u
  while there exists $u \in V(w')$ s.t. $d(u, V(w')) \leq 1$ **do**
    reset $w'(u)$ to 0 /* clean up $G(w)$ */
  Reduce(w', w'')
  $F \leftarrow (V(w') - V(w'')) \cup$ Feedback(w'')
  for every u in $V(w') - V(w'')$ **do**
    if $V(w') - (F - \{u\})$ induces no cycle
      in $G(V(w'))$**then**
      Remove u from F /* u is redundant */
  return F
  end
end

procedure Reduce(w', w'')
begin
  if there is a semi–disjoint cycle C
    in $G(w')$
  then
    $V_i \leftarrow C$
  else
    $V_i \leftarrow V(w')$
  $\delta \leftarrow \min\{\frac{w'(v)}{(d(v,V_i)-1)} | v \in V_i\}$
  $w_i(v) \leftarrow \begin{cases} \delta(d(v, V_i) - 1) & \text{if } v \in V_i \\ 0 & \text{otherwise} \end{cases}$
  $w'' \leftarrow w' - w_i$
end

Figure 1: A 2–Approximation Algorithm for Feedback Vertex Set Problem

F returned by Feedback(w'') is a minimal FVS in $G(w'')$ that is a subgraph of $G(w')$. After the subsequent **for**–loop F becomes a minimal FVS in $G(w')$.

Next we argue that $F \cap V(w_i)$ is a minimal FVS in $G(w_i)$ for all i. The weight function w_i is computed by Reduce(w', w'') in two ways:

Case 1. $G(w_i)$ is a semi–disjoint cycle C in $G(w')$. Then C is connected with the rest of $G(w')$ via at most one vertex. Since F constructed by Feedback(w) is a minimal FVS in $G(w')$, F can contain at most one vertex from C, and thus, $F \cap V(w_i) = F \cap V_i$ is a minimal FVS in $G(w_i) = C$.

Case 2. $G(w_i)$ is not a cycle. Then, $V(w) = V(w_i)$ and $G(w_i) = G(w')$. So, $F \cap V(w_i)$ is a minimal FVS for $G(w_i)$.

Finally, note that in both cases no vertex of $V(w_i)$ is included in $F_{out} - F$ for F returned by Feedback(w). Hence, $F_{out} \cap V(w_i) = F \cap V(w_i)$ and the claim follows. □

We now bound the weight of our FVS in terms of the optimum weight in every subgraph. Observe that $G(w_i)$ can have two forms:

1. a simple cycle with all the vertices having equal weights, or
2. a graph of degree–proportional weight, with no vertices of degree ≤ 1 and no semi–disjoint cycles.

In the first case, the fact that $F_{out} \cap V(w_i)$ is a minimal FVS for $G(w_i)$ implies that it is also an optimal FVS for $G(w_i)$. The second case is more interesting. We show that for a degree–proportional weighted graph without a semi–disjoint cycle, a minimal cardinality FVS has a weight that is within a factor of 2 from the optimum.

Clearly, it suffices to prove it for the case when for every $u \in V, w(u) = d(u) - 1$, where $d(u)$ denotes the degree of u; we say that such a weight function is *degree–defined*. We will show it using a potential function $p : V \rightarrow \mathbb{R}$, s.t. $p(u) = \frac{d(u)}{2} - 1$ for every $u \in V$. Assume that w is the degree–defined weight function defined on V.

Lemma 4 *For an arbitrary FVS F of G, $w(F) \geq p(V)$.*

Proof. Let $\delta(F)$ denote the set of edges which are incident to some node in F. Since F is an FVS $G(V - F)$ can contain at most $|V - F| - 1$ edges. In other words $|\delta(F)| \geq |E| - (|V - F| - 1)$. Also note that $p(V) = \sum_{u \in V} \frac{d(u)}{2} - 1 = |E| - |V|$. Thus,

$$w(F) = \sum_{u \in F} (d(u) - 1) \geq |\delta(F)| - |F| \geq |E| - (|V - F| - 1) - |F| \geq p(V).$$

□

Lemma 5 *If F is a minimal FVS in a clean graph $G(V)$ without a semi–disjoint cycle then $w(F) \leq 2p(V)$.*

Observe that $G(V-F)$ is a forest and by minimality, each vertex $u \in F$ is *blocked* by a tree in the forest, i.e. each vertex in F has two edges incident on some tree $T \subseteq V-F$. Let e_T be the number of edges with one end in T and another in F.

The next two statements are part of the proof of Lemma 5. We present them independently here because we will also apply them in the next section.

Fact 6 *Each tree T in $G(V-F)$ can block at most $\lfloor \frac{1}{2} e_T \rfloor$ vertices.*

Lemma 7 *Let T be an arbitrary tree in G. Then, $p(T) = \frac{1}{2} e_T - 1$.*

Proof. Assume that T has t vertices. Then $\sum_{u \in T} d(u) = e_T + 2(t-1)$. Consequently,

$$p(T) = \sum_{u \in T} (\frac{d(u)}{2} - 1) = \frac{\sum_{u \in T} d(u)}{2} - t = \frac{e_T + 2(t-1)}{2} - t = \frac{e_T}{2} - 1$$

□

Proof. (of Lemma 5) Each vertex $u \in F$ is a priori allocated a potential of $(d(u)-2)/2$. We show that each $u \in F$ also receives an additional potential of $\frac{1}{2}$ from vertices in $V-F$. Let T be a tree blocking u. Since G contains no semi–disjoint cycles, $e_T \geq 3$. Then, Fact 6 and Lemma 7 imply that each vertex in F can receive an extra potential of $(\frac{1}{2} e_T - 1)/\lfloor \frac{1}{2} e_T \rfloor$, which is at least $\frac{1}{2}$ when $e_T \geq 3$. Therefore, u contributes $d(u)-1$ to $w(F)$ and at least $(d(u)-1)/2$ to $p(V)$. □

Summarizing, for every i we have $w_i(F_{out}) \leq 2w_i(opt(G(w_i)))$, and this allows us to apply Theorem 1, and as a consequence, the desired performance of our algorithm is proven.

Theorem 8 *The algorithm finds an FVS F_{out} in $G = (V, E, \mathrm{w})$ in time $O(|E| + |V| \log |V|)$ s.t. $\mathrm{w}(F_{out}) \leq 2\mathrm{w}(opt(G))$.*

Proof. The running time of algorithm *Feedback*() is dominated by the time required to maintain the vertex with the maximal degree–proportional weight. Therefore its running time is $O(|E| + |V| \log |V|)$. □

4 Better Performance for Bounded Degree Graphs

We here turn our attention to the FVS problem when graphs are restricted to have bounded vertex degree, as not only a general but a degree bounded version of the VC problem has been subjected to extensive research, and show that the performance can be improved in this case from that of the general case. It turns out, however, that the techniques developed in Section 3 is not powerful enough to achieve our goal, and to remedy this situation we elaborate our approach of Section 3 by introducing new approximation preserving reductions and much more carefully applying the techniques developed earlier. Somewhat intricate reasoning on the effects resulting from their interplay will give ratio $2 - \frac{2}{3\Delta - 2}$, strictly less than 2 for any constant Δ. As remarked in Introduction the currently best performance ratio

for the bounded degree (weighted) VC problem is $2 - \frac{3}{2\Delta+4}$ [8]. It should be noted, however, that the approaches taken in approximation of these two problems are quite distinct; ours is an extension of the local approximation for a general problem while the one in [8] cannot be generalized to an unbounded degree version.

We first examine why the algorithm of Section 3 does no better in the present case. The bottleneck for a better performance of our algorithm is the ratio given in Lemma 5. To prove it, we observe that a vertex u of F with weight k has potential $\frac{1}{2}k - \frac{1}{2}$; additionally, it receives at least $\frac{1}{2}$ of the potential from a tree T in $G(V - F)$ that blocks it. We may modify the algorithm to assure that the extra potential received from T is at least $\frac{2}{3}$. This will assure the weight/potential ratio strictly below 2. Assume for now that every vertex of F can receive an extra potential of $\frac{2}{3}$ in a graph of maximal degree Δ. The same argument on the weight/potential ratio as before gives the weight $w(F)$ of a solution F the ratio

$$\frac{\Delta - 1}{\frac{\Delta-2}{2} + \frac{2}{3}} = 2 - \frac{2}{3\Delta - 2}.$$

and our goal of this section is stated as

Goal: find an FVS F whose weight is within a factor $r_\Delta \stackrel{\text{def}}{=} 2 - \frac{2}{3\Delta-2}$ of the optimum.

Recall Fact 6 and Lemma 7. It is clear that as e_T increases, the extra potential that T can supply gets closer to $\frac{1}{2}$ per edge: it is $(e_T - 2)/(2e_T) = \frac{1}{2}(1 - 1/e_T)$. Hence, for $e_T \geq 6$ we have no problem; for $e_T = 5$, T can send $\frac{1}{3}$ on each edge to a vertex blocked by T and $\frac{1}{6}$ along each of the remaining edges (see Fig. 2). What we need to handle in a special manner is the case when $e_T \leq 4$.

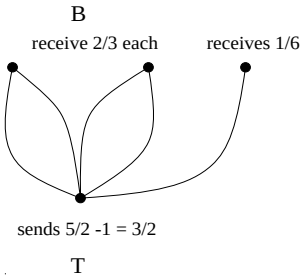

Figure 2: T with $e_T = 5$

For a tree $T \subseteq V - F$ with $e_T \leq 4$ we consider a subgraph induced by T together with the set B of vertices blocked by T. We can assume w.l.o.g. that no vertex of degree 2 is adjacent to another degree–2 vertex (later we will see why). Then, it is easy to see that $|T \cup B| \leq 9$ (the case of $|T \cup B| = 9$ is given in Fig. 3). The new algorithm will attempt to perform some reduction steps on each connected subgraph $G(A)$ with $1 < |A| \leq 9$. Below we refer to such A's as *small–subgraphs*. For each small subgraph A we need to know two FVS's, F_1^A and F_2^A. F_1^A is an optimal FVS for $(G(A), w)$, F_2^A is also an optimal FVS, but with restriction that no path in $G(A - F_2^A)$ may connect two edges that straddle A and $V - A$ (e.g., if two such edges are adjacent to the same vertex $u \in A$, then u forms a path of length 0 connecting them, and therefore u must belong to F_2^A).

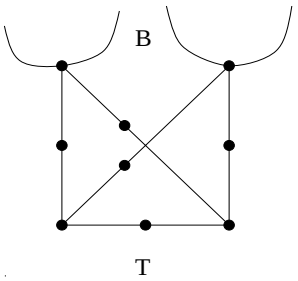

Figure 3: T with $|B \cup T| = 9$

4.1 Ear reduction

Define an *ear* as a small–subgraph A s.t. either all edges straddling A and $V - A$ are incident to the same vertex in A, or there are at most two such edges.

The algorithm proceeds by replacing every ear A found in a graph by a single vertex u_A, designated as an *ear–vertex*, and assigning a modified weight to u_A, as described in the procedure *ReduceEar*(A).

procedure *ReduceEar*(A)
  Compute F_1^A and F_2^A
  Replace A by an ear–vertex u_A, and connect it to edges straddling A and $V - A$
  Set $w(u_A) \leftarrow w(F_2^A) - w(F_1^A)$

After computing an FVS F in the reduced graph, we execute *InsertEar*(u_A) for every ear–vertex u_A to recover an FVS in the original graph.

procedure *InsertEar*(u_A)
  if $u_A \in F$ **then**
    $F \leftarrow F - \{u_A\} \cup F_2^A$
  else
    $F \leftarrow F \cup F_1^A$

Lemma 9 below shows that the ear reduction preserves any performance ratio ≥ 1. It is also useful to observe that the ear reduction strictly generalizes both *branchy reduction* of [3] (the case of $|A| = 2$) and semi–disjoint cycle reduction of Section 3. Another easy observation is that degree–2 vertices form an independent set when the ear reduction is not applicable anymore.

Lemma 9 *Let* (G', w') *be a graph obtained from* (G, w) *by* ReduceEar*(A). Suppose that* InsertEar*(u_A) subsequently changes an FVS* F' *of* G' *into* F. *Then* F *is an FVS in* G *and*

$$\frac{w(F)}{w(opt(G(w)))} \leq \frac{w'(F')}{w'(opt(G'(w')))}.$$

Proof.

It is easy to see that F is an FVS in G if F' is an FVS in G' from the ways *ReduceEar* and *InsertEar* modify G and F', respectively. We shall show separately that

$$w(F) \leq w'(F') + w(F_1^A) \tag{1}$$

and

$$w(\mathrm{opt}(G)) \geq w'(\mathrm{opt}(G')) + w(F_1^A) \tag{2}$$

and then the inequality in question easily follows.

(1) is trivial because in fact $w(F) = w'(F') + w(F_1^A)$ whether u_A is in F' or not ($w(u_A) = w(F_2^A) - w(F_1^A)$, and if $u_A \in F'$ it is subtracted but $w(F_2^A)$ is added to $w'(F')$).

To prove (2) let F be any FVS in G and consider F_A, the *contraction* of F at a set $A \subseteq V$, i.e.,

$$F_A \stackrel{\text{def}}{=} \begin{cases} (F - A) \cup \{u_A\} & \text{if there is no path in } G(A - F) \text{ that connects two edges straddling} \\ & A \text{ and } V - A \\ F - A & \text{otherwise} \end{cases}$$

Since $F \cap A$ must be an FVS in $G(A)$ it easily follows from this definition that

$$\begin{aligned} w(F) &= w(F - A) + w(F \cap A) \\ &\geq \begin{cases} w'(F - A) + w(F_2^A) = w'(F - A) + w'(u_A) + w(F_1^A) = w'(F_A) + w(F_1^A) \\ \quad \text{if there is no path in } G(A - F) \text{ connecting two edges that straddle } A \text{ and } V - A \\ w'(F - A) + w(F_1^A) = w'(F_A) + w(F_1^A) \quad \text{otherwise} \end{cases} \end{aligned}$$

and thus, in either case $w(F) \geq w'(F_A) + w(F_1^A)$.

It now suffices to show that F_A is an FVS in G' for, if true, $w'(F_A) + w(F_1^A) \geq w'(\mathrm{opt}(G')) + w(F_1^A)$. Clearly, this is true if $F_A = F - A \cup \{u_A\}$ since $F \cap (V - A) \subseteq F_A$ and $u_A \in F_A$. So suppose $F_A = F - A$ and that it is not an FVS in G'. Any cycle in $G(((V - A) \cup \{u_A\}) - F_A)$ must contain u_A implying that there is a path in $G((V - A) - F)$ connecting two edges which straddle A and $V - A$. However, there is also a path in $G(A - F)$ connecting these two edges because A is an ear and because of the assumption on the form of F_A. Thus, there must exist a cycle in $G(V - F)$ contradicting the assumption that F is an FVS in G. □

4.2 Local lower bound reduction

Recall that the goal of our algorithm is to provide an FVS of which weight is within a factor r_Δ of the optimum, and define a *good–small–subgraph* as a subgraph A satisfying $w(F_2^A) \leq r_\Delta w(F_1^A)$.

If the algorithm cannot apply the ear reduction, it looks for a good–small–subgraph. For such A, the algorithm computes an FVS F in $(G(V - A), w)$ and returns $F \cup F_2^A$. The definition of F_2^A assures that the latter is an FVS in $G(V)$. By its very nature of this reduction the next lemma is obvious.

Lemma 10 *The local lower bound reduction returns an FVS preserving performance ratio* r_Δ.

4.3 Degree–proportional weight reduction

If none of the previous two reductions is applicable, the algorithm finds w_1, the largest degree–proportional weight s.t. $w_1 \leq w$, and proceeds as in the second case of procedure *Reduce*. However, it later inspects the resulting FVS F to eliminate the situations described at the beginning of this section: a vertex of F receiving only $\frac{1}{2}$ of the extra potential unit from $V - F$.

Before we describe the way we modify F we have to explain exactly how the potential is transferred from $V - F$ to F. Let T be a component tree of $V - F$ and B be the set of vertices blocked by T. Consider an edge going from T to $V - T$. If $e_T \geq 5$, T sends $\frac{2}{3}$ to each vertex in B and $\frac{1}{6}$ along each of the edges going from T to $F - B$. We can do the same when $e_T = 4$ and $|B| \leq 1$ or $e_T = 3$ and $|B| = 0$. The case of $e_T = 2$ does not happen at this stage as it allows for the ear reduction. When $e_T = 4$ and $|B| = 2$ or $e_T = 3$ and $|B| = 1$, T sends only $\frac{1}{2}$ to each element of B. Thus, we can have vertices of F with a deficit: such a vertex is blocked by T that gives it only $\frac{1}{2}$, and it is not adjacent to any edge carrying $\frac{1}{6}$. If $u \in B$ is a vertex with deficit but B contains another vertex that has a "surplus", i.e., receiving $\frac{1}{2} + \frac{2}{6}$ of the potential we shift $\frac{1}{6}$ of the potential to u from its "buddy" in B.

In a summary we need to handle in a special manner such subgraphs $G(A)$, where $A = B \cup T$, when

1. $e_T = 3, |B| = 1$, and $u \in B$ is not incident to an edge with a positive potential, or
2. $e_T = 4, |B| = 2$ and neither vertex of B has a surplus;

such subgraphs are displayed in Figure 4. Let F_1^A be an optimal FVS in $(G(A), w)$ as before.

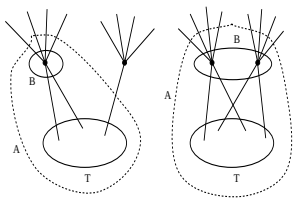

Figure 4: Small subgraphs $G(A)$ with a deficit vertex

The degree–proportional weight reduction can be described as follows:

Given (G, w),

1. Find a maximal degree–proportional weight $w_1 \leq w$
– Compute an FVS F_0 from $(G, w - w_1)$
2. Build an FVS F in (G, w) by minimally augmenting F_0
3. For all small subgraphs $G(A)$ with a deficit vertex set $F \leftarrow (F - A) \cup F_1^A$

Remark: As in the previous algorithm, we make sure that F is a minimal FVS for $G(V)$ before any replacement is executed. But then, even if some B is replaced by F_1^A in F, since F_1^A is optimal in $G(B \cup T)$, F remains a minimal FVS for $G(V)$.

In the remainder of this section we analyze this procedure and justify its effects. As technical preliminaries observe (from Figure 4) that

1. deficit vertices do not occur in graphs of degree ≤ 3, as these subgraphs of Figure 4 are eliminated by the ear reduction if $d(u) \leq 3$ for every vertex u in B,

2. an edge e carries zero potential *only if* it does not straddle F and $V - F$, or it is the unique edge joining T with $F - B$ in a subgraph $G(A)$ of $e_T = 3$ and $|B| = 1$,

3. B is an FVS in $G(A)$ with the same restriction as the one for F_2^A, i.e., no path in $G(A - B)$ joins two edges straddling A and $V - A$. Thus, we know $w(B) \geq w(F_2^A)$;

these facts will be useful in the proofs given below.

If a certain vertex u has a deficit and no buddy with a surplus, we change F and the allocation of the potential; we replace B by F_1^A, an optimal FVS in $(G(A), w)$, in F and send $\frac{1}{6}$ of the potential along each edge emanating from A to $V - A$. We need to show two facts:

1. the new F is still an FVS for (G, w), and

2. F_1^A has so small a weight that $p(A)$ can "justify" the "new" weight allocated to $F \cap A$ and $\frac{1}{6}$'s sent along edges straddling A and $V - A$.

We prove them in the next two lemmas.

Lemma 11 *After replacing B by F_1^A, F remains an FVS in $(G(V), w)$.*

Proof. Assume otherwise, that $G(V - F)$ now contains a cycle C. C must go through A, but it cannot be completely contained in $G(A)$. When $e_T = 4$ and $|B| = 2$, C has to cross B twice, i.e., there exist two edges of C straddling B and $V - A$. In case of $e_T = 3$ and $|B| = 1$, C has to cross B once, implying one edge of C straddling B and $V - A$. In either case, if edges straddling B and $V - A$ carry $\frac{1}{6}$ of the potential each, we would not have done such a replacement, contradiction. Therefore, at least one of them carries zero potential. However, in this case, such an edge either goes from B to $F - B$, or to one of the subgraphs displayed in Figure 5, where every path starting from this edge leads to F, contradicting that it is an edge of C. □

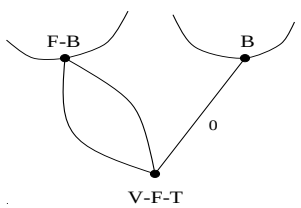

Figure 5: An edge without carrying any potential

The procedure reduces (G, w) to (G, w_0) where $w = w_1 + w_0$ and w_1 is a maximal degree–proportional weight s.t. $w_1 \leq w$. In our analysis, even if some replacement occurs we take

$w_0(F)$ unchanged; i.e., if some replacement takes place changing $F \cap A$ from B to F_1^A we associate $w(F_1^A) - w_0(B)$ with $F \cap A$ as its weight in (G, w_1), instead of $w_1(F_1^A)$, leaving the weight associated with F in (G, w_0) unchanged. The next lemma states that we can pay by $p(A)$ in (G, w_1) for such a weight as well as $\frac{1}{6}$ each along every out–going edge if $w(F_1^A)$ is smaller than $w(B)$ by a factor of $\sqrt{3}$.

Lemma 12 *Assume that w_1 is the degree–defined weight function for G. Let $w_0 \stackrel{\text{def}}{=} w - w_1$ and m be the number of edges straddling A and $V - A$. If $G(A)$ is not a semi–disjoint cycle in (G, w) and if $w(F_1^A) \leq \frac{w(B)}{\sqrt{3}}$, then*

$$p(A) \geq \frac{(w(F_1^A) - w_0(B))}{\sqrt{3}} + \frac{m}{6}.$$

Proof. Let $b = |B|, t = e_T$. Also let $k, l, 2b + x$, and y, respectively, be the numbers of edges going from B to $V - A$, from T to $V - A$, from T to B, and within B (see Fig. 6).

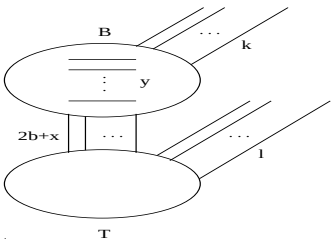

Figure 6: Edges around B and T

By easy calculation,

$$w_1(B) = k + b + x + 2y \quad \text{and} \quad p(A) = p(B) + p(T) = \frac{w_1(B)}{2} - (\frac{b}{2} - \frac{t}{2} + 1).$$

By the assumption $\frac{w(F_1^A) - w_0(B)}{\sqrt{3}} \leq \frac{\frac{w(B)}{\sqrt{3}} - w_0(B)}{\sqrt{3}} \leq \frac{w(B) - w_0(B)}{3}$. So it suffices to show that $p(A) \geq \frac{w_1(B)}{3} + \frac{k+l}{6}$. Since $t = 2b + x + l$,

$$\begin{array}{rrcl} & \frac{w_1(B)}{2} - (\frac{b}{2} - \frac{t}{2} + 1) & \geq & \frac{w_1(B)}{3} + \frac{k+l}{6} \\ \equiv & \frac{k+b+x+2y}{6} - (\frac{b}{2} - \frac{t}{2} + 1) & \geq & \frac{k+l}{6} \\ \equiv & x + 2y + 3t - 2b & \geq & l + 6 \\ \equiv & y + 2b + 2x + l & \geq & 3 \end{array}$$

Since $x, y \geq 0, b \geq 1$ this inequality does not hold only when $b = 1, x = y = l = 0$, which is possible only when $B \cup T$ is a semi–disjoint cycle. □

Using these two lemmas we can prove that the degree–proportional weight reduction preserves performance ratio r_Δ.

Lemma 13 *Let w_1 be the largest degree–proportional weight function s.t. $w_1 \leq w$, and let $w_0 \stackrel{\text{def}}{=} w - w_1$. Suppose F_0 is an FVS in (G, w_0) s.t. $w_0(F_0) \leq r_\Delta w_0(opt(G(w_0)))$. Then, the degree–proportional weight reduction constructs an FVS F in (G, w) from F_0 s.t. $w(F) \leq r_\Delta w(opt(G(w)))$.*

Proof. Let F be an FVS constructed from F_0 by minimally augmenting it for (G, w). When $\Delta \leq 3$, there is no deficit vertices in (G, w_1) and the degree–proportional weight reduction does no replacements in it. Since every vertex in F receives an extra potential of $\frac{2}{3}$ its weight $w_1(F)$ in (G, w_1) is within a factor r_Δ of the optimum. Using Theorem 1 we have the desired ratio on $w(F)$ and the optimum in (G, w).

Assume $\Delta \geq 4$ and that there exists a set B of deficit vertices in (G, w_1) as defined before (otherwise, we're done). The degree–proportional reduction replaces B by F_1^A in F (i.e., $F \leftarrow (F - B) \cup F_1^A$) where F_1^A is an optimal FVS in $(G(A), w)$ and $A = B \cup T$. To see the effect of this replacement on the relative weight of F in (G, w), assign $w^*(F \cap A) \stackrel{\text{def}}{=} w(F_1^A) - w_0(B)$ to $F \cap A$ as its weight in (G, w_1), instead of $w_1(F_1^A)$, for the sake of analysis, for every A in which B and F_1^A are exchanged. This way, $w_0(F)$ remains the same, maintaining the ratio r_Δ in (G, w_0). The remaining portion F' of F has $w_1(F')$ as its weight as before. Clearly the net weight $w(F) - w_0(F)$ of F in (G, w_1) is now estimated as $w_1(F') + \sum_A w^*(F \cap A)$, where the summation is taken over all small subgraphs A with a deficit vertex. Applying Theorem 1 it remains to show that

$$w_1(F') + \sum_A w^*(F \cap A) \leq r_\Delta w_1(\text{opt}(G(w_1))) \tag{3}$$

because $w_0(F_0) \leq r_\Delta w_0(\text{opt}(G(w_0)))$ by the assumption.

Consider first $w^*(F \cap A)$ for any small subgraph A with a deficit vertex. Recall that the local lower bound reduction is not applicable here and $r_\Delta > \sqrt{3}$; moreover $w(B) \geq w(F_2^A)$ as observed before, and hence,

$$w(F_1^A) \leq \frac{w(F_2^A)}{r_\Delta} \leq \frac{w(B)}{r_\Delta} \leq \frac{w(B)}{\sqrt{3}} \tag{4}$$

Assume w.l.o.g. that w_1 is a degree–defined function. Because of (4) and because there is no semi–disjoint cycles (no ear reduction is applicable here), we can apply Lemma 12 to (G, w_1) and A, which assures that a weight $w^*(F \cap A)$ is within a factor $\sqrt{3} \leq r_\Delta$ of the optimum in $(G(A), w_1)$.

On the other hand when such a replacement is no longer applicable there is no deficit vertex in the remainder of F. Therefore, F' has a weight $w_1(F')$ with ratio r_Δ to the optimum in $G(V - \cup A)$. We conclude that (3) must hold. □

4.4 Putting Together

Finally all the reduction procedures of the previous subsections are combined to yield an algorithm as presented below.

Given $G = (V, E)$,
if G is already acyclic, return $\emptyset$
else do the following

1. For all ears A, $G \leftarrow ReduceEar(A)$
2. For all good–small–subgraphs A, $G \leftarrow G(V - A)$
3. Find a maximal degree–proportional weight $w_1 \leq w$ and recursively compute an FVS F_0 from $(G, w - w_1)$
4. Construct a minimal FVS F in (G, w) by minimally augmenting F_0 (i.e., $F \supseteq F_0$)
5. For all small–subgraph A with a deficit vertex, set $F \leftarrow (F - A) \cup F_1^A$
6. For all good–small–subgraphs A, set $F \leftarrow F \cup F_2^A$.
7. For all ear–vertices u_A, set $F \leftarrow InsertEar(u_A)$.
8. Return F

When applied to a graph of maximal degree Δ this algorithm provides an improved performance, i.e.,

Theorem 14 *The algorithm finds an FVS for G with performance ratio $2 - \frac{2}{3\Delta - 2}$ in time $O(|V|^2)$, where Δ is the maximum degree of G.*

Proof. The correctness and the performance ratio of the algorithm easily follow from Lemmas 9, 10, and 13. Observe that once we have applied *ReduceEar()*, each vertex can test if it is part of a small–subgraph in constant time. This implies a running time of $O(|V|^2)$. □

References

[1] S. Arora, C. Lund, R. Motwani, and M. Sudan and M. Szegedy. Proof Verification and Hardness of Approximation Problems. In *33rd Annual Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[2] R. Bar–Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. In *Annals of Discrete Mathematics*, volume 25, pages 27–46. North–Holland, 1985.

[3] R. Bar-Yehuda, D. Geiger, J. Naor and R. M. Roth. Approximation algorithms for the vertex feedback set problem with applications to constraint satisfaction and Bayesian inference. In *Proc. of the 5th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 344–354, 1994.

[4] A. Becker and D. Geiger. Approximation algorithms for the loop cutset problem. In *Proc. of the 10th conference on Uncertainty in Artificial Intelligence*, pages 60–68, 1994.

[5] P. Erdös and L. Pósa. On the maximal number of disjoint circuits of a graph, *Publ. Math. Debrechen*, 9:3–12, 1962.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[7] M.X. Goemans and D.P. Williamson. The Primal–Dual Method for Approximation Algorithms and its Application to Network Design Problems. D. Hochbaum, ed., *Approximation Algorithm for NP–Hard Problems*, chapter 4, PWS Publishing Company, to appear.

[8] M.M. Halldórsson. Approximations via partitioning. Technical report, Japan Advanced Inst. of Sci. and Tech., March 1995.

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

[10] J.M. Lewis and M. Yannakakis. The Node–deletion problem for hereditary properties is NP–complete. *Journal of Computer and System Sciences*, 20:219–230, 1980.

[11] C. Lund and M. Yannakakis. The approximation of maximum subgraph problems. In *Proc. of 20th International Colloquium on Automata, Languages and Programming*, pages 40–51, 1993.

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

[13] B. Monien and E. Speckenmeyer. Ramsey numbers and an approximation algorithm for the vertex cover problem. *Acta Informatica*, 22:115–123, 1985.

[14] C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.