On Local Search and Placement of Meters in Networks

Samir Khuller* Randeep Bhatia† Robert Pless‡

Abstract

This work is motivated by the problem of placing pressure meters in fluid networks. The problem is formally defined in graph-theoretic terms as follows. Given a graph, find a cotree (complement of a tree) incident upon the minimum number of vertices. We show that this problem is NP-hard and MAX SNP-hard. We design an algorithm with an approximation factor of $2+\epsilon$ for this problem. This approximation bound comes from the analysis of a local search heuristic, a common practical optimization technique that does not often allow formal worst case analysis. The algorithm is made very efficient by finding restrictive definitions of the local neighborhoods to be searched. We also exhibit a PTAS for this problem when the input is restricted to planar graphs. Although we are motivated by water distribution networks, the algorithms apply to any system that can be formulated as a network, in which Kirchoff's laws are valid and a bijective relation exists between the flow variable and the effort variable, like circuits and electrical networks.

1 Introduction

To measure flow in pipes, one can install flow-meters on edges. However, *you do not need to install flow-meters on all the edges in the network.* If you install flow meters on edges of a cotree[1]then we can *infer* the flow on the edges of the spanning tree due to flow conservation (and the flow into and out of terminals). This requires that we install exactly $|E|-(|V|-1)$ flow meters where $|E|$ is the number of edges and $|V|$ is the number of vertices in a connected network, since every cotree has $|E|-(|V|-1)$ edges. Since "the cost of flow meters is several times the cost of pressure meters" [21], another option is to install pressure meters. Pressure meters can be installed at nodes of the network, and one can *compute* the flow on an edge by measuring the pressure reading on both the incident nodes (see [19, 20]). One simple solution would be to place a pressure meter at each node of the network. However, this is not necessary. Since we only need to compute the flow on edges of a cotree, we are looking for a cotree that is incident on as few vertices as possible (to minimize cost). We place pressure meters at these nodes; directly computing the flow on all cotree edges and inferring the flow for the remaining edges of the network. Although we are motivated by water distribution networks, the algorithms apply to any system that can be formulated as a network, in which Kirchoff's laws are valid and a bijective relation exists between the flow variable and the effort variable, like circuits and electrical networks [21].

We can also express this problem in "standard" optimization nomenclature as follows: A feedback edge set (FES) is a subset of edges in a graph, whose deletion from the graph makes the graph acyclic. In other words, each cycle in the graph is required to have at least one edge from the FES. In graph theoretic terms, minimizing the number of pressure meters leads to the following optimization problem. The minimum vertex feedback edge set (VFES) problem is defined as follows: Given a graph, find a feedback edge set incident upon the minimum number of vertices. We show that this problem is NP-hard and MAX-SNP hard and develop an approximation algorithm for it that is fast, practical, and delivers solutions that are guaranteed to have cost at most $2+\epsilon$ times optimal (for any fixed $\epsilon > 0$).

The complement problem to minimum VFES is the *full degree spanning tree* (FDST) problem [18, 21, 11, 10] – namely that of computing a spanning tree T in a connected graph $G=(V,E)$ so as to *maximize* the number of vertices of *full* degree. These are vertices whose degree in the tree T is the same as their degree in the graph G. In other words, the vertices that are incident to the edges in $E \setminus T$ are exactly the vertices that do not have full degree in T. Thus if K is the maximum number of full degree vertices in some spanning tree T in G, then $|V|-K$ is the cost of the minimum VFES. The *savings* of this solution, compared to installing pressure meters at every vertex, is exactly the number of full degree vertices in a spanning tree, since these vertices have zero degree in the corresponding FES and we do not have to install

*Computer Science Dept., University of Maryland, College Park, MD 20742. E-mail:`samir@cs.umd.edu`. Research supported by NSF Award CCR-9820965 and an NSF CAREER Award CCR-9501355.

†Bell Labs, 700 Mountain Avenue, Murray Hill, NJ 07974. E-mail:`randeep@research.bell-labs.com`.

‡Computer Science Dept., University of Maryland, College Park, MD 20742. E-mail:`pless@cs.umd.edu`.

[1]A cotree in a connected graph is a subset of edges that is the complement of a spanning tree.

pressure meters at these vertices. Hence maximizing the number of vertices of full degree maximizes savings. One pressure meter is installed at each vertex that does not have full degree; by using these meters, we can compute the flow on each edge in the cotree, and then infer the flow on each edge in the tree due to flow conservation (and the flow into and out of terminals).

The full degree spanning tree problem was recently studied by Pothof and Schut [21], Broersma et. al. [11] and Bhatia, Khuller, Pless and Sussmann [10]. In [10] it was shown that for the case of general graphs an approximation factor of $\Theta(\sqrt{n})$ can be obtained using a very simple algorithm called the **Greedy Star-Insertion Algorithm**. Using Håstad's result on the hardness of approximating clique, it was also shown that if there is a polynomial time approximation algorithm, for the FDST problem, with a factor of $O(n^{\frac{1}{2}-\epsilon})$ then $NP = coR$. For the case of planar graphs, a polynomial time approximation scheme was presented [10]. Independently Broersma et al. [11] develop a PTAS for planar graphs, and show that for special classes of graphs such as bounded treewidth graphs and cocomparability graphs the problem can be solved optimally in polynomial time. On the other hand, Pothof and Schut [21] suggest a heuristic for full degree spanning trees based on the following idea – let the weight of an edge be the sum of the degrees of the end vertices. Use this weight function to find an MST and use this as the full degree tree. This heuristic may do extremely badly, and for certain graphs finds no vertices of full degree (even in graphs where a large number of full degree vertices exist).

Local Search is widely used as a general approach to tackle hard optimization problems. Typically, each instance of the optimization problem is associated with a set of feasible solutions, together with a cost function and the objective is to find a solution of minimum cost. To derive a local search algorithm, one superimposes a "local search neighborhood", i.e., for each solution we define a neighborhood of solutions that are "close" to it. We start with some initial solution. The algorithm iteratively checks to see whether or not we can "move" from the current solution to another solution in its neighborhood with lower cost, and if so, we move to the lower cost solution until we reach a locally optimal solution; a solution whose cost is as low as any other in its local search neighborhood. The two-exchange heuristic for the traveling saleman problem illustrates a local search; an initial solution is iteratively improved by attempting to find a better tour which differs by only two edges.

Even though local search is a powerful practical tool for optimization, there are not many problems for which local search leads to good worst case approximation factors. In fact, often when it does lend itself to worst case analysis, the bounds that we obtain are significantly worse than the worst case approximation bounds that we can obtain by other methods. However, since local search is a powerful optimization method and relatively simple to implement, it is interesting to analyze local search algorithms as they enhance our understanding of the problems we are studying. For example, for the traveling salesman problem (with triangle inequality), an analysis of the local search method done by Chandra, Karloff and Tovey [12] shows an approximation factor of $4\sqrt{n}$ for 2-exchange, with improvements for Euclidean instances. Recently, for various facility location problems, Korupolu, Plaxton and Rajaraman [17] showed that a local search method gives constant approximation factors. Local search has also been used for set packing problems by Arkin and Hassin[1].

A feedback edge set (FES) is a subset of edges in a graph, whose deletion from the graph makes the graph acyclic. In other words, each cycle in the graph is required to have at least one edge from the FES. A feedback vertex set (FVS) is a subset of vertices in the graph, whose deletion from the graph makes the graph acyclic. In other words, each cycle in the graph is required to have at least one vertex from the FVS.

Minimum Vertex Feedback Set problems are NP-hard and approximation algorithms for them have been widely studied. The first constant approximation for unweighted feedback vertex sets was given by Bar-Yehuda, Geiger, Naor and Roth [5]. They also gave a constant approximation for the weighted case for planar graphs, and an $O(\log n)$ approximation for general graphs. For planar graphs, there is a constant factor approximation due to Goemans and Williamson [15] and recently a PTAS due to Kleinberg and Kumar [16]. For general graphs (weighted case) a 2 approximation was developed by Becker and Geiger [6] and Bafna, Berman and Fujito [3] (see also Chudak, Goemans, Hochbaum and Williamson [13]).

On the other hand, for undirected graphs the FES problem has largely been ignored since a FES is the complement of a spanning tree (cotree) and both the maximization and minimization versions can be solved in polynomial time. However, in this case we need to find a feedback edge set (FES) so as to *minimize the number of vertices incident on the edges* in the FES. We show that this problem is NP-hard as well as Max SNP-hard. Approximation algorithms for feedback arc problems on directed graphs were given by Even, Naor, Rao and Schieber [14] (see the survey by Shmoys [22] as well).

We study the minimum VFES problem, and show that the local search method applied to this problem yields a good approximation algorithm. The local search neighborhood for a given FES is defined by a constant k, that allows us to "move" to any other FES that differs on fewer than k edges. The main difficulty with making this approach practical is that we need to examine all possible changes, and this is not practical even for small values of k. The naive upper bound on the size of the neighborhood we have to search for an improvement is $O(m^k n^k)$, where m is the number of edges in the graph and n is the number of vertices. However, quite surprisingly we are able to show that we can refine the search so that we do not need to search the entire neighborhood for a better solution. We can implement each step of the local search in time $O(n^2 + nf(k))$ where $f(k)$ is a function only of k and not of the size of the graph. This makes the local search practical. (In fact we found that even for $k = 2$ this led to a very fast algorithm that found the optimal solution in each case for several random planar and non-planar graphs generated using Stanford Graphbase and LEDA [9].)

Main Results

We show that the minimum VFES problem is NP-hard and MAX SNP-hard (proofs omitted). We show that taking *any* minimal feedback edge set, already yields a 3 approximation to the VFES problem. We then show that a local search method gives an approximation factor of $2+\epsilon$ for any fixed $\epsilon > 0$, for the minimum VFES problem. However, the running time of this algorithm is $O(n^{O(1/\epsilon)})$. Since this algorithm is not practical, we develop a restricted local search algorithm that restricts the local search neighborhood, which yields a significantly improved running time of $O(n^3 + n^2 f(\frac{1}{\epsilon}))$, and actually yields a practical algorithm with the same approximation guarantee. (Here f is an exponential function, but has no dependence on n.) We also show that for planar graphs we can design a polynomial time approximation scheme (PTAS). The proof of the following theorem is omitted.

THEOREM 1.1. *The Minimum Vertex Feedback Edge Set problem is MAX-SNP hard and NP-hard.*

2 The Minimum Vertex Feedback Edge Set Problem

In this section we consider the minimum vertex feedback edge set (VFES) problem, namely to find a feedback edge set incident on the minimum number of vertices. Any minimal feedback edge set is also a cotree.

We show that iteratively deleting any edge from a cycle in the graph $G = (V, E)$ yields an approximation algorithm with a factor of 3. We then improve the solution by applying a "local-improvement" strategy, and prove that this is a $2 + \epsilon$ approximation, for any fixed $\epsilon > 0$. We will also assume that our input graph is connected.

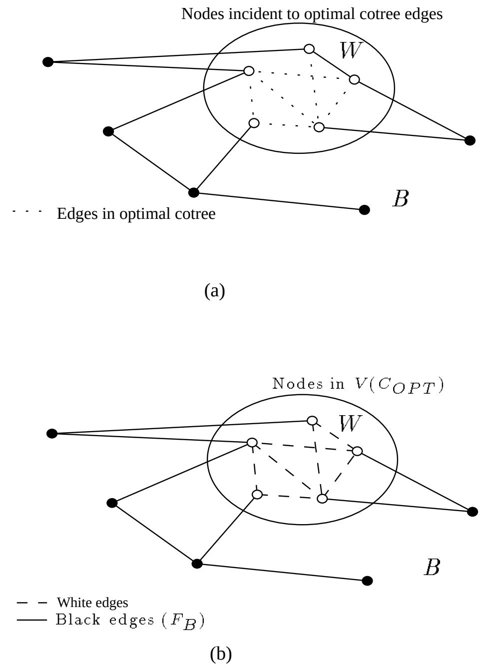


Figure 1: Example showing black and white vertices.

Let S be any subset of edges. Let $V(S)$ be the subset of vertices in G incident on edges in S. Let C_{OPT} be the edges of a minimum VFES. Let W (white vertices) be $V(C_{OPT})$. Let B (black vertices) be the set of remaining vertices, $V \setminus W$. Edges of G incident to at least one black vertex are colored black, all remaining edges are colored white. (See Fig. 1(a) for a minimum cotree and Fig. 1(b) for the colors of edges.) The set of black edges induce a forest F_B in G. We refer to the set of white edges as E_W. Note that $|W| = |V(C_{OPT})|$.

We first prove the following theorem about minimal feedback edge sets (which are easy to find).

THEOREM 2.1. *Any minimal feedback edge set gives a 3 approximation to the vertex feedback edge set problem.*

Proof. Let S be a minimal feedback edge set. Let $S_w = S \cap E_W$ and $S_b = S \setminus S_w$. We prove that $|V(S)| \leq 3|V(C_{OPT})| - 2$ as follows. Since $|V(E_W)| = |W| = |V(C_{OPT})|$ we have $|V(S_w)| \leq |V(C_{OPT})|$. The proof of the following inequality is given in Lemma 2.1.

$$|S_b| \leq |V(C_{OPT})| - 1.$$

Since the number of vertices that can be incident on any set of edges is at most twice the number of edges, we conclude:

$$|V(S_b)| \leq 2(|V(C_{OPT})| - 1).$$

Thus $|V(S)| \leq |V(S_w)| + |V(S_b)| \leq 3|V(C_{OPT})| - 2$. $\square$

LEMMA 2.1. *If we have a minimal feedback edge set S, and S_b is the set of black edges in this set then*

$$|S_b| \leq |V(C_{OPT})| - 1.$$

Proof. Recall that the black edges form a forest F_B. Any black edge $e \in S_b$ is essential to a minimal FES. In other words, introduction of edge e into $E \setminus S$ would create a cycle C_e. C_e contains only edges in $E \setminus S$, except for edge e. Since every cycle in G has at least one white edge C_e must also have two white vertices incident to this white edge. Let $e = (u, v)$. Walking around the cycle from e in both directions, we stop when we first encounter a white vertex. (We will stop at two distinct white vertices.) Consider $F_B \setminus S_b$; edge e connects two components, each of which must contain at least one white vertex. If $|S_b| \geq |V(C_{OPT})|$, then there are at least $|V(C_{OPT})| + 1$ distinct components in $F_B \setminus S_b$. In the forest F_B there are exactly $|V(C_{OPT})|$ white vertices; thus one edge in S_b connects two components, such that one component contains no white vertex, a contradiction. $\square$

It is possible to make examples that show that this bound is tight. If there are k vertices in the optimal cotree, then a minimal solution may have up to $k + 2(k-1)$ vertices.

2.1 Local Improvement

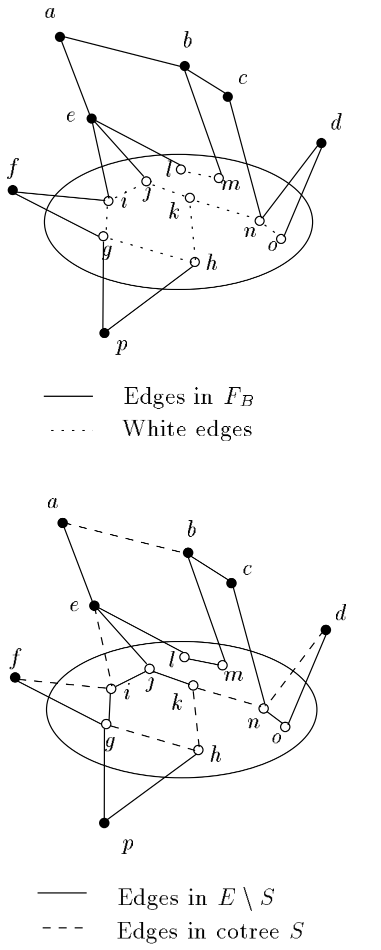
We can apply the following local-improvement strategy to our minimal FES S. Replace a set of at most $k-1$ edges from S, to get another FES with lower cost (cost is the number of vertices incident to the edges in the FES). Keep doing this until no reduction in cost is possible. This new solution will be referred to as S^ℓ. This can be done for any fixed $k > 0$. (For $k = 1$ this is simply any minimal feedback edge set.)

We call this the k-Local-Improvement algorithm.

k-Local Improvement(G) —

```
Pick an arbitrary cotree S (of size |E| − (|V| − 1)).
repeat
        found ← false.
        if there exists a cotree S' such that
        V(S') < V(S) and |S ∩ S'| > |S| − k then
            found ← true.
            S ← S'.
until not found
```

We can partition S^ℓ into two sets as before – white edges S^ℓ_w and black edges S^ℓ_b. The edges in S^ℓ_b form a collection of connected components. Let $\mathcal{C} = \{C_i | C_i$ is a component of black edges from $S^\ell_b\}$. For each black component C_i that shares a white vertex with S^ℓ_w, $|V_b(C_i)| \leq |E(C_i)|$, where $V_b(S)$ denotes the black vertices of C_i. A black component that does not share any vertices with S^ℓ_w is called a *full black component*. In Fig. 2 we show three black components: $\{a, b\}$, $\{e, f, i\}$, and $\{d, n\}$. The first two components are full black components. The third component shares a white vertex (n) with an edge in S^ℓ_w.



$$S^\ell_b = \{(n,d), (a,b), (e,i), (i,f)\}$$
$$S^\ell_w = \{(g,h), (h,k), (k,n)\}$$

Figure 2: Figure to show full and non-full black components

A full black component may have all black vertices, or may have white vertices in $W - V(S^\ell_w)$. We call a full black component C_i *small* if it has at most $k-1$ edges. The other full black components in $\mathcal{C}$ are referred to as *large*.

The proof of the following theorem relies on a lemma bounding the number of small full black components $|\mathcal{C}_{\text{small}}|$ by $|W| - |V(S_w^\ell)|$ (Lemma 2.4). We will prove this lemma later.

THEOREM 2.2.
The k-Local-Improvement algorithm obtains a solution that is at most $2 + \frac{1}{k}$ times the optimal solution. This algorithm runs in polynomial time for any fixed $k > 0$.

Proof. Let $V(S_w^\ell)$ be the vertices that the edges in set S_w^ℓ are incident on. We can bound the number of small full black components $|\mathcal{C}_{\text{small}}|$ by $|W| - |V(S_w^\ell)|$ (Lemma 2.4). For each large full black component C_i, the number of vertices $|V(C_i)|$ is at most $\frac{k+1}{k}|E(C_i)|$. For each black component with at least one white vertex also in $V(S_w^\ell)$ we have $|V_b(C_i)| \leq |E(C_i)|$.

$|V(S_b^\ell) \setminus V(S_w^\ell)| =$

$$\sum_{C_i \in \mathcal{C}_{\text{small}}} |V(C_i)| + \sum_{C_i \in \mathcal{C}_{\text{large}}} |V(C_i)| + \sum_{C_i \in \mathcal{C}_{\text{non-full}}} |V_b(C_i)|$$

$$\leq |\mathcal{C}_{\text{small}}| + \sum_{C_i \in \mathcal{C}_{\text{small}}} |E(C_i)| + \sum_{C_i \in \mathcal{C}_{\text{large}}} \frac{k+1}{k}|E(C_i)|$$

$$+ \sum_{C_i \in \mathcal{C}_{\text{non-full}}} |E(C_i)|$$

By Lemma 2.4 we get

$$|V(S_b^\ell) \setminus V(S_w^\ell)| \leq |W| - |V(S_w^\ell)| + \sum_{C_i \in \mathcal{C}} \frac{k+1}{k}|E(C_i)|.$$

Note that

$$|V(S^\ell)| = |V(S_w^\ell)| + |V(S_b^\ell) \setminus V(S_w^\ell)|.$$

Using the bound on $|V(S_b^\ell) \setminus V(S_w^\ell)|$ obtained above, we get

$$|V(S^\ell)| \leq |V(S_w^\ell)| + |W| - |V(S_w^\ell)| + \frac{k+1}{k} \sum_{C_i \in \mathcal{C}} |E(C_i)|.$$

By Lemma 2.1, since S_b^ℓ is also a minimal set of edges we can bound $\sum_{C_i \in \mathcal{C}} |E(C_i)|$ by $|V(C_{OPT})| - 1$.

$$\begin{aligned} |V(S^\ell)| &\leq |W| + \frac{k+1}{k}(|V(C_{OPT})| - 1) \\ &\leq |V(C_{OPT})| + \frac{k+1}{k}(|V(C_{OPT})| - 1). \end{aligned}$$

This gives us a $2 + \frac{1}{k}$ approximation to VFES; by picking $k = \frac{1}{\epsilon}$ we get the desired result.

The naive bound on the running time of the algorithm can be computed as follows. There are at most n iterations, since in each iteration our cost decreases by at least one. To implement an iteration, we have to try subsets of at most $k-1$ edges from our cotree, and try to replace them by another subset of the same size, such that we obtain a cotree with lower cost. An easy bound on the running time for a single iteration is $O(m^{k-1}n^k)$. □

We now prove Lemma 2.4, which states that the number of small (full) black components in S^ℓ is at most $|W| - |V(S_w^\ell)|$.

Root each black tree in F_B at an arbitrary black vertex chosen as the root and orient all edges away from the root (see Fig. 3). Let C_i be a small (full) black component in S^ℓ rooted at r_i.

Number the vertices in C_i as $u_1^i, \ldots u_{p_i+1}^i$. Let e be the (oriented) edge (u_j^i, u_k^i). Let Shadow(i,e) be the set of white vertices reachable from u_k^i by following a directed path of zero or more black edges that are in $F_B \setminus S^\ell$. The vertices in Shadow(i,e) are descendants of u_k^i in the rooted black forest F_B.

Let Shadow$(C_i) = \cup_{e \in C_i}$Shadow(i,e). Note that Shadow(C_i) is the set of white vertices that are reachable from vertices in $C_i - \{r_i\}$ by following a directed path of black edges that are in $F_B \setminus S^\ell$.

LEMMA 2.2. *For any full black component C_i and $e \in C_i$, we have Shadow$(i,e) \neq \emptyset$.*

Proof. Consider an (oriented) edge $e = (u_j^i, u_k^i) \in C_i$. If $u_k^i \in W$ then clearly Shadow$(i,e) \neq \emptyset$. Otherwise $u_k^i \in B$. If we add edge e to $E \setminus S^\ell$ we get a cycle C_e with a white edge on it, connecting two white vertices. Note that all the edges in F_B incident on u_k^i except for the edge from the parent u_j^i are oriented away from u_k^i. Consider the first white vertex on the path in C_e from node u_k^i to u_j^i that avoids e - we reach this vertex by taking zero or more edges from $F_B \setminus S^\ell$ before reaching this white vertex. Thus there is a directed path of zero or more black edges from $F_B \setminus S^\ell$ to some white vertex. □

LEMMA 2.3. *For each pair of small black components C_i and C_j, Shadow(C_i) and Shadow(C_j) are disjoint.*

Proof. Suppose (for contradiction) that $w \in$ *Shadow*$(C_i) \bigcap$ *Shadow*(C_j). By the definition of Shadow, $\exists u \in C_i - \{r_i\}$, (and $\exists v \in C_j - \{r_j\}$), such that there is a directed path of black edges in $F_B \setminus S^\ell$, from u (and v) to w. Since w is a descendant of both u and v in the black tree F_B, one of $\{u, v\}$ is a descendant of the other. Assume that u is an ancestor of v. The unique path from u to w in F_B includes the tree path from u to v and the path from v to w. Since the edge joining v to its parent is in C_j and in S^ℓ, at least one edge along this path is not in the set $F_B \setminus S^\ell$, and thus w is not in the shadow of u. □

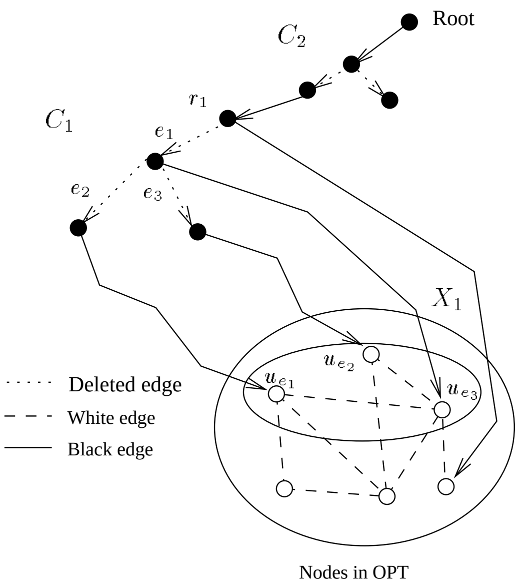


Figure 3: Example showing black components.

LEMMA 2.4. *The number of small (full) black components in S^ℓ is at most $|W| - |V(S^\ell_w)|$.*

Proof. Consider a small (full) black component C_i with $p_i(< k)$ edges. We will show that we can "charge" each component C_i to a node in $W - V(S^\ell_w)$, such that each node in $W - V(S^\ell_w)$ is charged at most once. This would prove that the number of small black components can be upper bounded by $|W| - |V(S^\ell_w)|$.

LEMMA 2.5. *Each small (full) black component C_i, Shadow(C_i) contains at least one vertex in $W - V(S^\ell_w)$.*

Proof. Assume for contradiction that all the vertices in Shadow(C_i) are in $V(S^\ell_w)$. This means that these nodes are incident to at least one white edge in our current FES.

Adding a new edge to the cotree that is incident on a vertex in Shadow(C_i), can only increase the cost of the solution by at most one, since one end point of such an edge is already incident to some white edge in the FES. Let T_{S^ℓ} be the tree corresponding to $E \setminus S^\ell$. We call R_i a *replacement set* for C_i if $T_{S^\ell} \cup C_i - R_i$ is a spanning tree in G. We will now show that we can find a valid replacement set R_i for C_i of cost at most p_i. By doing this replacement, we will obtain a better solution since C_i is incident to $p_i + 1$ vertices, and R_i has cost p_i. This exchange involves a set of at most $k-1$ edges, contradicting the assumption that we have a solution that cannot be locally improved.

We now show how to find the set R_i. First note that the solution S^ℓ is minimal, and the introduction of any edge from S^ℓ into $E - S^\ell$ will create a cycle.

Initially, $R_i = \emptyset$. Let $T = E \setminus S^\ell$. As we add edges from C_i to T, we will delete edges from T to maintain a tree. These deleted edges will be added to R_i to form the replacement set for C_i. We process the edges of C_i as we encounter them in a preorder traversal of C_i from r_i. Assume that we are processing the edge $e = (u^i_j, u^i_k) \in C_i$ (oriented downwards (away from the root)). If we add the edge e to T, we get a cycle C_e. We follow the path in C_e starting from u^i_k and going away from u^i_k and stop at the first white edge (u_e, v_e). If $u_e \in C_i$ then this cannot be a full black component since $u_e \in V(S^\ell_w)$, and we obtain a contradiction. Since $u_e \in$ Shadow(i, e) it is in $V(S^\ell_w)$ and does not cost anything. This is the replacement edge for e, and costs at most one. We add e to T and remove (u_e, v_e) from T, and add (u_e, v_e) to R_i. We repeat this until we find all the replacement edges for C_i. $\square$

We will charge each small black component C_i to a vertex in $W - V(S^\ell_w)$ that also belongs to Shadow(C_i). Since the shadows are disjoint, each node in $W - V(S^\ell_w)$ is charged at most once, and the lemma follows. $\square$

Since this algorithm is not practical, we show how to modify the algorithm to obtain an efficient implementation.

2.2 Restricted Local-Improvement

We now show that we can obtain the $2 + \frac{1}{k}$ bound by modifying the algorithm to only improve "small trees". In other words, once we have a cotree S, we can consider its connected components and try to reduce $V(S)$ by only eliminating small trees (of size $< k$) from S. We check to see if we can delete the edges of a small tree T_s from S (this will create cycles in $T \cup T_s$) and then find a set of edges to add to S so that we again get a cotree. If this replacement reduces the cost of the cotree, we perform this step. Keep doing this until either no small trees are left, or we cannot get a reduction in the number of vertices that are incident to the edges in the cotree. This method can be implemented in $O(n^{k+2})$ time as follows: We show that each iteration can be implemented in $O(n^{k+1})$ time, and there are at most n iterations.

There are only $O(n)$ possible "small trees" to eliminate. For each small tree, we have to find a valid replacement set - this takes $O(n^k)$ time, since there are $O(n^{k-1})$ possible replacement steps, and checking each one takes $O(n)$ time. Thus this gives a bound of $O(n^{k+1})$ for each iteration. The initial cost of the cotree is at most n, the number of vertices, and since the cost is measured as the number of vertices incident to the set of edges in our subset of edges, in each iteration this cost decreases by at least one, we have at most n iterations (could be a lot fewer in practice) giving the overall bound of $O(n^{k+2})$.

The key difficulty is in implementing the replacement step efficiently. The naive implementation takes $O(n^{k+1})$ time as we just observed. We now give an algorithm whose complexity (per iteration) is $O(n^2+nf(k))$ where $f(k)$ is an exponential function in k. The key point is that this algorithm is practical, whereas the previous one is not practical. This improvement is obtained by showing that we only need to "search" for an improvement by restricting our search space to a very small number of possibilities. This gives an overall bound of $O(n^3 + n^2 f(k))$ for the entire algorithm.

We now focus on the implementation of a single iteration of the algorithm. Suppose we have a current cotree S and let T be the tree $E \setminus S$. Let T_s be a tree with $\ell \leq k-1$ edges in the graph (V, S). Our goal is to try to find a "replacement" set of ℓ edges R, such that $(S \setminus T_s) \cup R$ is a cotree, or equivalently $(T \cup T_s) \setminus R$ is a tree. If the cost of $(S \setminus T_s) \cup R$ is lower than the cost of S, we obtain a local improvement (reduction in cost).

Each edge $e \in T_s$ induces a path $P(T, e)$ in T. Let $\mathcal{T}$ be the subtree of T defined by taking the union of the ℓ paths $P(T, e)$ for each $e \in T_s$. Clearly, $R \subseteq \mathcal{T}$. Fig. 4 shows the tree T along with a tree $T_s \subseteq S$ (of three edges in this case) and the corresponding paths $P(T, e)$. Our goal is to find a replacement set R of lower cost. Observe that since we need to "destroy" the ℓ cycles that are created when we add the ℓ edges of T_s to T, we are restricted to picking edges from $\mathcal{T}$ to obtain the set R. We now show that even though $\mathcal{T}$ can be large, we can find a subtree $\mathcal{T}'$ of size $O(\ell)$ such that if a replacement set R of lower cost exists, then a replacement set of edges chosen from $\mathcal{T}'$ can be found.

We now define tree $\mathcal{T}'$. Consider $T_s \cup \mathcal{T}$. Mark the vertices of T_s as "special" and all the vertices in $\mathcal{T} \setminus T_s$ with degree three or more as "special" (special vertices are shown as the circled vertices in Fig. 4). We can now think of $\mathcal{T}$ as a "tree-like" structure on "special" vertices, where we have paths connecting the "special" vertices. If T_s has ℓ edges, then there are at most 2ℓ special vertices, and thus there are at most $2\ell - 1$ "paths" in $\mathcal{T}$ connecting the special vertices (see Theorem 2.4). If any "path" between two special vertices has five or more edges then we will "shrink" this to a path of length at most four (see Theorem 2.3). This will ensure that each path has a constant number of edges, and thus the size of the entire tree $\mathcal{T}'$ is $O(\ell)$. Each "path" between two special vertices is also referred to as a "link".

The cost of the ℓ edges in T_s is exactly $\ell + 1$. The vertices in $V(S \setminus T_s)$ are called *marked*. All other vertices are *unmarked*. Our goal is to pick a replacement set R of cost at most ℓ. Our replacement set includes marked and unmarked vertices, but only the unmarked vertices cost 1, while the marked vertices cost 0 since these nodes are already included in $S \setminus T_s$. To do this we will enumerate over subsets of size ℓ from $\mathcal{T}'$. For each candidate set, we will check if it is a valid replacement set and compute the cost of the new cotree. If the cost of the new cotree is lower, we perform the replacement. It should now be clear that the function f is the number of ways of choosing ℓ edges from the $2\ell - 1$ links, so that they form a valid replacement set.

THEOREM 2.3. *If a replacement set R of lower cost exists, then a replacement set of edges chosen from $\mathcal{T}'$ of the same cost exists.*

Proof. We first make some observations about the replacement set R. At most one edge can be chosen from each link in the replacement set R. Let the path $P = [v_0, \ldots, v_k]$ (see Fig. 5) be the path corresponding to a link. If $k \leq 4$, we do not change the link (this link has at most four edges). If $k > 4$, we define a new compressed link for $\mathcal{T}'$ based on the subset of marked vertices on path P. If there is replacement set R in $\mathcal{T}$ then picking the corresponding edge from $\mathcal{T}'$ yields a replacement set of the same cost.

If there is no marked vertex on P, and R chooses an edge from P, then w.l.o.g, R picks an edge incident to v_0 or v_k. We can compress this path to a path with two edges.

If there exists a pair of adjacent marked vertices on P, and R picks an edge from P, then w.l.o.g, we may assume that R picks the adjacent pair of marked vertices. This can be shown by arguing that if R does not pick an adjacent pair of marked vertices, then we can change R to another valid replacement set which uses the adjacent marked pair, without increasing its cost. See Fig. 6 to see the "compression" of a long path to a short path. In Case (A) the marked pair is an internal pair on the path, and in Case (B) it is an end pair on the path. In either case we perform the transformation as shown in Fig. 6. We leave it to the reader to see that a replacement set of edges in $\mathcal{T}$ can translated to a replacement set of edges in $\mathcal{T}'$ with no increase in cost.

If there exists at least one marked vertex on P, but no adjacent pair of marked vertices, and R chooses an edge from P, then w.l.o.g, R picks an edge that is incident on a marked vertex. In each case it can be verified that we preserve the cost of the set R in $\mathcal{T}$ when we consider the corresponding set R' in $\mathcal{T}'$. (Details will be provided in the full version.) □

THEOREM 2.4. *The size of $\mathcal{T}'$ is $O(\ell)$.*

Proof. In $\mathcal{T}'$ there are at most 2ℓ special vertices. This can easily be proven by induction on the number of edges in T_s. If T_s has one edge then it creates exactly two marked vertices. Consider T_s, delete an edge incident to a leaf vertex of T_s. This has $\ell - 1$ edges, and thus $2(\ell - 1)$ special vertices. When we add the leaf edge, we create one more special vertex (the leaf vertex) and possibly one more vertex whose degree becomes three.

Thus in $\mathcal{T}$ there are at most $2\ell - 1$ "links", where a "link" is a maximal sequence of degree two vertices. Each link is a sequence of at most four edges, hence the number of edges in $\mathcal{T}'$ is at most $4(2\ell - 1)$. $\square$

3 Planar Graphs

To exhibit of a polynomial time approximation scheme (PTAS) for the VFES problem requires two steps. First, we pre-process the graph to eliminate adjacent pairs of degree two vertices, then prove that in the resulting graph (modulo a special case), the optimal solution has size $\Omega(n)$, where n is the number of vertices. Second, we explain the necessary modifications to the problem to be able to use Baker's schema for generating a PTAS for problems solvable optimally on k-outerplanar graphs [4]. Baker's schema requires an optimal solution to a version of the VFES problem when restricted to k-Outerplanar graphs; this is assumed to be available as a subroutine here, and is presented explicitly in section 4.

We first show that the optimal solution has size $\Omega(n)$ where n is the number of vertices. This will be achieved by preprocessing the input graph to decompose it into its 2-edge connected components. We can work with each 2-edge connected component separately, since the edges that are cut-edges (bridges) cannot be on any cycle. However, it may still be the case that the optimal solution is very small. We then observe that a maximal sequence of degree two vertices can be contracted into a sequence containing only one degree two vertex, without changing the cost of the optimal solution.

Once we have this property, *we prove that the number of edges in the graph is at least $\frac{6}{5}n$ (modulo a special case)* and hence any feedback edge set must have at least $\frac{n}{5}$ edges. We use this property later to bound the difference between our approximate solution and the optimal solution.

THEOREM 3.1. *If G is a planar graph with n vertices that is 2-edge connected and has no adjacent vertices of degree 2 then the number of edges is at least $\frac{6}{5}n$.*

Proof. Let n_i be the number of vertices of degree i. Since we are working with the 2-edge connected components of a graph, we can assume that $n_1 = 0$ (each vertex has degree at least 2). Since we do not have an edge between two vertices of degree 2 we have that $|E| \geq 2n_2$. Note that $2|E| = \sum_{i=2}^{n-1} in_i$. Hence $|E| = \sum_{i=2}^{n-1} \frac{i}{2}n_i$. If $n_2 \geq \frac{3}{5}n$ then we immediately obtain $|E| \geq \frac{6}{5}n$. Hence we assume that $n_2 \leq \frac{3}{5}n$. Since $\sum i = 3^{n-1}n_i = n - n_2$, $|E| \geq n_2 + \sum_{i=3}^{n-1} \frac{3}{2}n_i \geq n_2 + \frac{3}{2}(n - n_2)$. We obtain $|E| \geq \frac{3}{2}n - \frac{1}{2}n_2 \geq \frac{6}{5}n$. $\square$

Baker gives a general framework which constructs a PTAS for any problem which can be solved optimally for k-outerplanar graphs – planar graphs where all nodes have a path of length $\leq k$ to a node on the outermost face [4].

Let $d(v)$ = shortest path length from v to any node on the outer face of G. This scheme creates a collection of decompositions of the planar graph G into a set of k-outerplanar graphs. For each value of $i = 0 \ldots (k-1)$ we generate a decomposition as follows: delete edges that connect nodes with label $d(v) - 1$ and $d(v)$, where $d(v)$ is congruent to $i(\text{mod} k)$. For example, for $i = 0$, we delete edges connecting nodes with $d(v)$ value $k - 1$ and k, $2k - 1$ and $2k$ etc. After we delete these edges, we are left with a graph G_i, which is a collection of connected components that are each k-outerplanar. We obtain the optimal solution for G_i by running a linear time algorithm for each connected component (see Section 4). This optimal vertex Feedback Edge Set for the graph G_i can be extended to become a Feedback Edge Set for the input graph G by including a subset of the edges earlier deleted, with an additional cost less than or equal to the number of nodes with label $d(v)-1$ and $d(v)$, where $d(v)$ is congruent to $i(\text{mod} k)$. Define the set of nodes with these labels to be R_i, then the cost of our solution is at most $|OPT| + |R_i|$.

A given partition of G into a collection of k-outerplanar graphs G_i, will have a solution of size at most $|OPT| + |R_i|$, where R_i is the set of "boundary" nodes in G_i. If for all i, $|R_i| > \frac{2|V|}{k}$ then $\sum_{i \text{ even}} |R_i| > |V|$, a contradiction. Therefore for some i, we obtain a solution of size at most $|OPT| + \frac{2|V|}{k}$.

By Theorem 3.1 the optimal feedback edge set must have at least $\frac{n}{5}$ edges. Since the graph is planar, these edges must be incident on $\Omega(n)$ vertices; $|OPT| \geq cn$ for some constant c. Our solution cost is at most $OPT + \frac{2OPT}{kc} \leq OPT(1 + \frac{2}{kc})$. By setting $k = \frac{2}{c\epsilon}$, we get a bound of $(1 + \epsilon)OPT$.

4 k-Outerplanar, and Treewidth Bounded Graphs

In this section we give a linear-time algorithm for treewidth-bounded graphs (if the graph has treewidth k, then the time required for the algorithm to run will be exponential in k but linear in the size of the graph). Bodlaender [7] proves that any k-outerplanar graph has tree-width at most $3k - 1$. Since our graph G_i consists of a collection of connected components, each of tree-width $3k - 1$, we can run this algorithm on each component separately, and take the union of the solutions we obtain. These algorithms rely on a tree-decomposition of a graph:

DEFINITION 4.1. *Let $G = (V, E)$ be a graph. A* **tree-decomposition** *of G is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, where $\{X_i \mid i \in I\}$ is a family of subsets of V and $T = (I, F)$ is a tree with the following properties:*

1. $\bigcup_{i \in I} X_i = V$.
2. *For every edge $e = (v, w) \in E$, there is a subset X_i, $i \in I$, with $v \in X_i$ and $w \in X_i$.*
3. *For all $i, j, k \in I$, if j lies on the path from i to k in T, then $X_i \bigcap X_k \subseteq X_j$.*

The **tree-width** *of a tree-decomposition $(\{X_i \mid i \in I\}, T)$ is $\max_{i \in I}\{|X_i| - 1\}$. The tree-width of a graph is the smallest value k such that the graph has a tree-decomposition with tree-width k.*

Many problems are known to have linear time algorithms on graphs with constant tree-width, and there are frameworks for automatically generating a linear time algorithm, given a problem specification in a particular format [2, 8]. The VFES problem can be expressed in the formalism of Borie *et. al.* [8] as: given input graph $G = (V, E)$, find: $\min |V_1| \, [Forest(V, E - E'), Inc(E', V_1)]$, which states that we want to minimize the set of nodes incident upon a set of edges whose removal makes the graph a forest.

Acknowledgments: We thank Sudipto Guha, Yoram Sussmann and An Zhu for useful discussions.

References

[1] E.M. Arkin, R. Hassin, "On Local Search for Weighted k-set Packing", *Mathematics of Operations Research*, Vol. 23, No. 3, 1998, pp. 640-648.

[2] S. Arnborg, J. Lagergren, D. Seese, "Easy Problems for Tree-Decomposable Graphs", *Journal of Algorithms* Vol 12(2), (1991), pp. 308–340.

[3] V. Bafna, P. Berman and T. Fujito, "Constant factor approximation of the weighted feedback vertex set problem", *Proc. ISAAC*, (1995).

[4] B. Baker, "Approximation Algorithms for NP-Complete Problems on Planar Graphs", *JACM*, Vol 41 (1), (1994), pp. 153–190.

[5] R. Bar-Yehuda, D. Geiger, J. Naor and R. Roth, "Approximation Algorithms for the Vertex Feedback Set Problem with Applications to Constraint Satisfaction and Bayesian Inference", *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 344-354, (1994).

[6] A. Becker and D. Geiger, "Approximation algorithms for the loop cutset problem", *Proc. of 10th Conference on Uncertainty in AI*, (1994).

[7] H. L. Bodlaender, "Some classes of graphs with bounded tree width", *Bulletin of the European Association for Theoretical Computer Science*, (1988), pp. 116–126.

[8] R. B. Borie, R. G. Parker, and C. A. Tovey, "Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families" *Algorithmica*, Vol 7, (1992), pp. 555–581.

[9] A. Bhatia, "Full degree spanning tree problem", MS paper, Univ of Maryland, Dec 1998.

[10] R. Bhatia, S. Khuller, R. Pless and Y. Sussmann, "The full degree spanning tree problem", *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 864-865, (1999). Full version in Tech. Report CS-TR-3931, Univ. of Maryland, (1998).

[11] H.J. Broersma, A. Huck, T. Kloks, O. Koppius, D. Kratsch, H. Müller, H. Tuinstra, "Degree-preserving forests", *Mathematical Foundations of Computer Science*, (1998) LNCS 1450, pp. 713–721

[12] B. Chandra, H. Karloff and C. Tovey, "New Results on the Old k-Opt Algorithm for the TSP", *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 150–159, (1994).

[13] F. Chudak, M. Goemans, D. Hochbaum and D. Williamson, "Primal-Dual Interpretation of Two 2-Approximation Algorithms for the Feedback Vertex Set Problem in Undirected Graphs", *Operations Research Letters*, 22:111-118, (1998).

[14] G. Even, J. Naor, S. Rao and B. Schieber, "Divide-and-conquer approximation algorithms via spreading metrics", *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 62–71, (1995).

[15] M. Goemans, D. Williamson, "Primal-Dual Approximation Algorithms for Feedback Problems in Planar Graphs", *Combinatorica*, 18:37-59, (1998).

[16] J. Kleinberg and A. Kumar, "Wavelength conversion in optical networks", *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 566-575, (1999).

[17] M. R. Korupolu, C. G. Plaxton and R. Rajaraman, "Analysis of a local search heuristic for facility location problems", *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1-10, (1998).

[18] M. Lewinter, "Interpolation theorem for the number of degree-preserving vertices of spanning trees", *IEEE Trans. Circ. Syst.*, CAS-34, (1987), page 205.

[19] L. E. Ormsbee, "Implicit network calibration", *Journal of Water Resources, Planning and Management*, 115(2), (1989), pp. 243–257.

[20] L. E. Ormsbee and D. J. Wood, "Explicit pipe network calibration", *Journal of Water Resources, Planning and Management*, 112(2), (1986), pp. 116–182.

[21] I. W. M. Pothof and J. Schut, "Graph-theoretic approach to identifiability in a water distribution network", *Memorandum No 1283*, Universiteit Twente (1995). Submitted to *Elsevier Science*.

[22] D. Shmoys, "Cut problems and their application to divide-and-conquer", *Approximation Algorithms*, (ed. Dorit Hochbaum), PWS Publishing (1996).

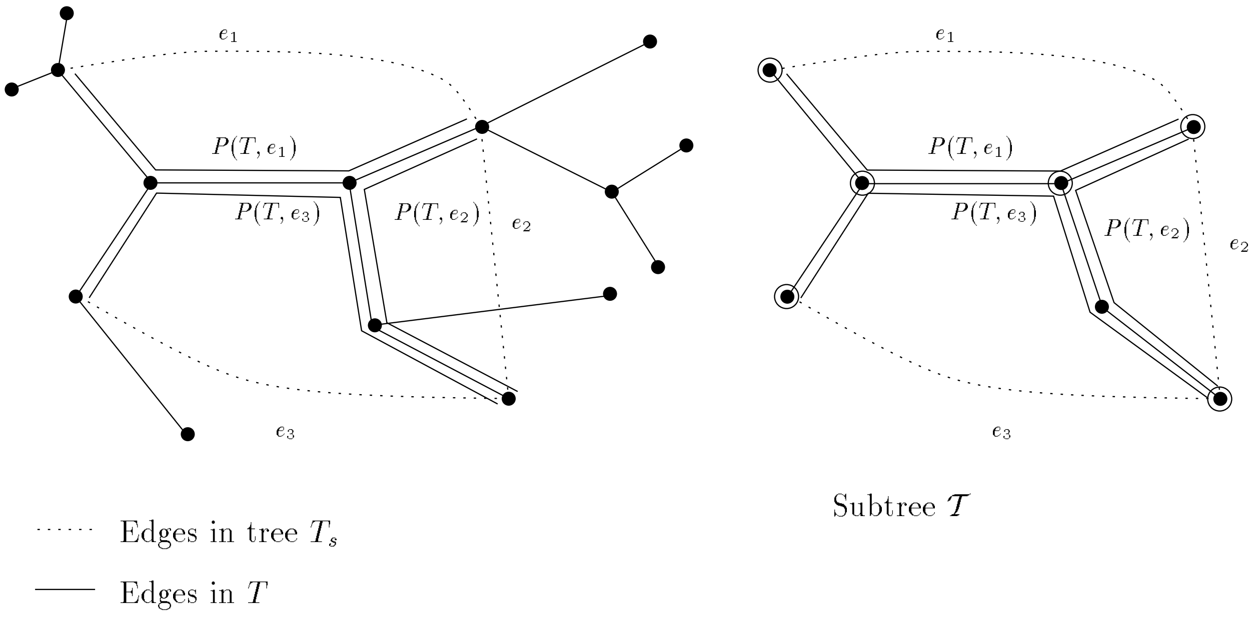


Figure 4: Figure to show $\mathcal{T}$

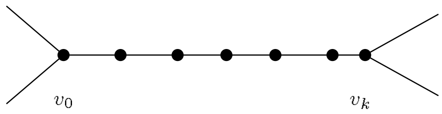


Figure 5: A long path that will be compressed.

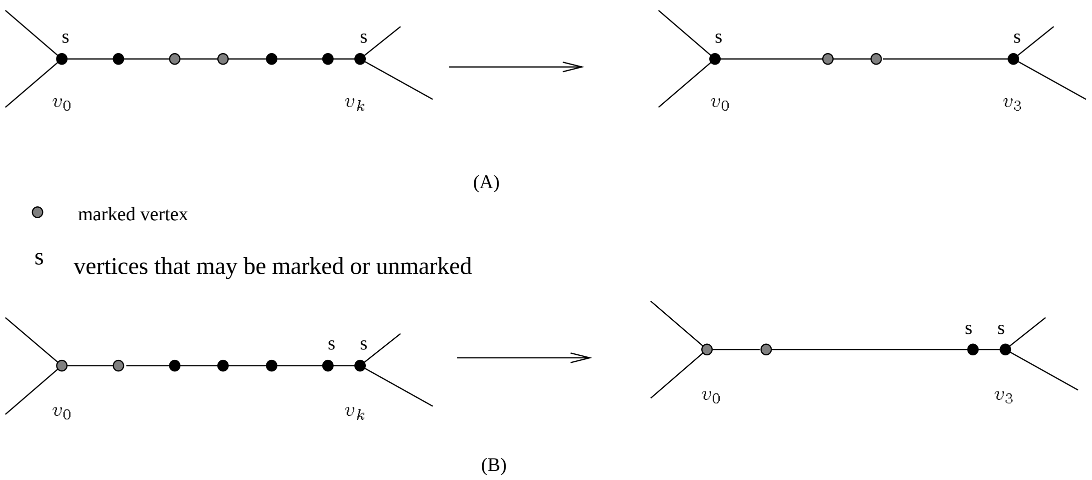


Figure 6: Compressing a link when there is an adjacent pair of marked vertices.