A Better Approximation Ratio for the Minimum Size k-Edge-Connected Spanning Subgraph Problem *

Cristina G. Fernandes †

January 20, 1998

Abstract

Consider the minimum size k-edge-connected spanning subgraph problem: given a positive integer k and a k-edge-connected graph G, find a k-edge-connected spanning subgraph of G with the minimum number of edges. This problem is known to be NP-complete. Khuller and Raghavachari presented the first algorithm which, for all k, achieves a performance ratio smaller than a constant which is less than two. They proved an upper bound of 1.85 for the performance ratio of their algorithm. Currently, the best known performance ratio for the problem is $1+2/(k+1)$, achieved by a slower algorithm of Cheriyan and Thurimella. In this paper, we improve Khuller and Raghavachari's analysis, proving that the performance ratio of their algorithm is smaller than 1.7 for large enough k, and that it is at most 1.75 for all k. Second, we show that the minimum size 2-edge-connected spanning subgraph problem is MAX SNP-hard.

*A preliminary version of this paper appeared in the *Proceedings of the* 8^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1997.*

†E-mail: cris@ime.usp.br. Department of Computer Science, Universidade de Sao Paulo, Rua do Matao, 1010, 05508-900 Sao Paulo - Brazil. Research supported in part by the CNPq (Brazil), under contract 200975/92-7.

1 Introduction

The study of connectivity in graph theory has important applications in the areas of network reliability and network design. Several approximation algorithms have been developed for the problem of finding subgraphs satisfying certain connectivity requirements [2, 4, 6, 7, 8, 9, 12, 13, 16, 18]. In this paper, we concentrate on the minimum size k-edge-connected spanning subgraph problem: given a positive integer k and a k-edge-connected graph G, find a k-edge-connected spanning subgraph of G with the minimum number of edges.

This problem is known to be NP-complete [5], even for $k = 2$: if the graph G is Hamiltonian, a 2-edge-connected spanning subgraph of G with the minimum number of edges must be a Hamiltonian cycle. So the goal is to look for good polynomial-time approximation algorithms for the problem.

A quality of an approximation algorithm is frequently measured by its *approximation* or *performance ratio*. For the minimum size k-edge-connected spanning subgraph problem, the approximation ratio of an algorithm is the infimum, over all possible inputs, of the ratio between the number of edges in the output of the algorithm and the number of edges in an optimal solution.

For a long time, two was the best approximation ratio achieved for all k. This comes from the fact that every minimal k-edge-connected spanning subgraph has at most kn edges [14, 15] and every k-edge-connected graph has minimum degree at least k.

Khuller and Raghavachari [10] presented the first algorithm which, for all k, achieves a performance ratio smaller than a constant which is less than two. They proved an upper bound of 1.85 for the performance ratio of their algorithm. In this paper, we improve their analysis, proving that the performance ratio of their algorithm is smaller than 1.7 for large enough k, and that it is at most 1.75 for all k.

Karger [9] presented an algorithm with performance ratio $1 + O(\sqrt{(\log n)/k})$. This is smaller than two only when $k \gg \log n$. Also, there were algorithms with approximation ratio smaller than two for some particular values of k. An algorithm for $k = 2$ with a ratio of 1.5 was presented by Khuller and Vishkin [13]. As observed in [10], by combining the biconnectivity algorithm in [13] and the sparse certificate algorithm in [2], one can easily obtain a ratio of $2 - 1/k$.

The currently best known performance ratio for the minimum size k-edge-connected spanning subgraph problem is $1 + 2/(k + 1)$. It is achieved by a new algorithm developed by Cheriyan

and Thurimella [3] concurrently with the research presented in this paper. Let $G = (V, E)$ be the input graph. In a first step, Cheriyan and Thurimella's algorithm finds an edge set $M \subseteq E$ of minimum cardinality such that every vertex in V is incident to at least k edges of M. In a second step, their algorithm finds a minimal edge set $F \subseteq E \setminus M$ such that $(V, M \cup F)$, the output of their algorithm, is k-edge-connected. The running time of Cheriyan and Thurimella's algorithm is $O(k^3|V|^2 + |E|^{1.5}(\log |V|)^2)$, while the running time of Khuller and Raghavachari's algorithm is $O(k^2|V|^2)$, which is better. The reader is refered to [3] for more details.

The bound on the approximation ratio of Khuller and Raghavachari's algorithm given in [10] is actually better than 1.85 for small values of k: it is 1.5 for $k = 2$, 1.666... for $k = 3$, 1.75 for $k = 4$, 1.733... for $k = 5$, etc. And these bounds are tight for $k = 2$ and 3. Our analysis improves Khuller and Raghavachari's bound for any fixed $k \geq 4$. In particular, we get 1.65 for $k = 4$, and 1.68 for $k = 5$.

Khuller and Vishkin [13] have introduced the following concept: a *tree-carving in a graph* $G = (V, E)$ is a partition of the vertex set V into subsets $V_1, \ldots, V_a$ with the following properties. Each subset constitutes a node of a tree Γ. For every vertex v in V_j, all the neighbors of v in G belong either to V_j itself or to V_i, where V_i is adjacent to V_j in the tree Γ. They have used this to prove the 1.5 bound on the ratio of their algorithm for $k = 2$. We generalize the concept of tree-carving and, from this generalization, prove that the performance ratio of Khuller and Raghavachari's algorithm is smaller than 1.7 for large enough k.

Finally, we show that the minimum size 2-edge-connected spanning subgraph problem is MAX SNP-hard. This implies that there is a constant $\epsilon > 0$ such that the existence of a polynomial-time approximation algorithm with performance ratio at most $1 + \epsilon$ would imply that $P = NP$ [1].

In the next section, we review Khuller and Raghavachari's algorithm. In Section 3, we present a better analysis and prove a new upper bound of 1.75 for all k on the performance ratio of the algorithm. Section 4 generalizes the concept of a tree-carving, and shows some properties which are then used to prove that the performance ratio of the algorithm is less than 1.7 for large enough k. The proof of MAX SNP-hardness appears in Section 5.

Figure 1: A DFS forest F, plus an edge e which is a back edge of the edge t in F.

2 Khuller and Raghavachari's algorithm

Let us start with some notation. Consider a graph G, a depth-first search (DFS) forest F in G, and the corresponding DFS numbering of the vertices of G. For each edge e in G, call *top (bottom)* the endpoint of e with smaller (bigger) DFS number. It is known that any edge of G which is not in F is a back edge. In fact, this means that its top endpoint is an ancestor in F of its bottom endpoint. For each edge t in F, we call an edge e of G which is not in F a *back edge of* t if t lies on the unique path in (V, F) between the two endpoints of e. (See Figure 1.)

The input for Khuller and Raghavachari's algorithm is a positive integer k and a k-edge-connected graph $G = (V, E)$. Their algorithm works in phases. It consists of $\lfloor k/2 \rfloor$ phases, plus an extra final phase when k is odd. For now, let us assume that k is even. Later we comment on the final phase for odd k.

After phase i, the algorithm will have selected a set S_i of edges of G which induces a $2i$-edge-connected spanning subgraph of G. Hence at the end of $k/2$ iterations, the current set $S_{k/2}$ induces a k-edge-connected spanning subgraph of G which is the output of their algorithm.

How does each phase work? Let $S_0 := \emptyset$. In phase $i \geq 1$, the algorithm chooses two disjoint edge sets F_i and B_i, both disjoint from S_{i-1}. The set F_i is simply the edge set of a DFS forest in the graph $(V, E - S_{i-1})$. The edge set B_i is built as follows. Initially $B_i = \emptyset$. Then the edges in F_i are scanned in the reverse of the DFS order (i.e., in the reverse of the order they were included

in F_i). When edge $t \in F_i$ is scanned, the algorithm finds a minimum cut between the endpoints of edge t in the graph $(V, S_{i-1} \cup F_i \cup B_i)$. Denoting by X one of the sides of this minimum cut, let $(X, \overline{X})$ be the corresponding cut in G, that is, $(X, \overline{X}) = \{h \in E : h$ has exactly one endpoint in $X\}$. If the cut $(X, \overline{X})$ in G contains exactly $2(i-1)+1 = 2i-1$ edges in $S_{i-1} \cup F_i \cup B_i$, then let $A := (E - (S_{i-1} \cup F_i \cup B_i)) \cap (X, \overline{X})$. The algorithm adds to B_i an edge $e \in A$ which has the smallest DFS-numbered top endpoint and sets $t_e := t$. Set $S_i := S_{i-1} \cup F_i \cup B_i$ at the end of phase i. This concludes the definition of the algorithm for even k. For odd k, the algorithm runs $\lfloor k/2 \rfloor$ phases, and in the end, it runs a "half" phase, computing $F_{\lceil k/2 \rceil}$ only (not $B_{\lceil k/2 \rceil}$). The final $S_{\lceil k/2 \rceil}$ is simply $S_{\lfloor k/2 \rfloor} \cup F_{\lceil k/2 \rceil}$.

The reader is refered to [10], where Khuller and Raghavachari give the proof of the correctness of this algorithm, as well as the upper bound of 1.85 on its performance ratio. In fact, for fixed k, Khuller and Raghavachari's analysis gives bounds smaller than 1.85 [10]. More specifically, it gives 1.5 for $k = 2$, 1.666... for $k = 3$, 1.75 for $k = 4$ and 1.733... for $k = 5$. These bounds are tight for $k = 2$ and 3.

In Figure 2, we present our tight examples for $k = 2$ and 3. Our results originated from attempts at generalizing these examples for $k > 3$, and we think they can give some intuition. We give two essentially different types of examples for $k = 2$: G_1 and G_2. In both cases, $opt = n$, where n is the number of vertices in the graph. Also, if the algorithm chooses as F_1 the dark edges, then the output of the algorithm will have $n + (n-1)/2$ edges (in both cases). As n grows, the ratio between the size of the output and opt approaches 1.5. Note that both these examples can be extended for large values of n. Graph G_3 is our tight example for $k = 3$ (only for even n). In this case, OPT can be the outer cycle and the chords from each vertex in the outer cycle to the opposite vertex in the outer cycle. We have that $opt = 3n/2$. Also, assume that the algorithm chooses as F_1 the dark edges. In general, F_1 is built as in G_2: it is a path starting at any vertex in the cycle, and passing by each other vertex in the cycle until it has visited half of the vertices. Then the remaining vertices are leaves of F_1. Note that F_1 is in fact a DFS tree. Consider that all edges from the leaves in F_1 to the root are present in G_3, so that these edges are chosen as B_1. In our example, the dashed edges would be B_1. Besides that, the remaining edges of G_3 should contain a tree. All the remaining edges would be F_2. The output of the algorithm will have $2(n-1) + n/2$ edges. As n grows, the ratio between the size of the output and opt approaches $5/3 = 1.666...$ This example

Figure 2: Graphs G_1, G_2 and G_3, with the depth first search numbering given by F_1.

can be extended, as suggested, for any large value of n such that $n = 2(2t + 1)$ for some positive integer t.

The intuition that we would like to get from these examples comes from the following reasoning. For Khuller and Raghavachari's bound to be tight, opt must be approximately $kn/2$, and the first $\lceil k/4 \rceil$ B_i must have approximately $opt/(k - 2(i - 1))$ edges. As we will see, for large k, the only way to expect that $|B_i| \approx opt/(k - 2(i - 1))$ for the initial B_i's is if F_i has many leaves. However, we will see that this cannot happen if $opt \approx kn/2$.

3 The 1.75 upper bound on the performance ratio

In this section, we improve the analysis of Khuller and Raghavachari's algorithm by presenting a better upper bound on $|B_i|$.

The following definition is critical to understanding the improved analysis. For each e in B_i, recall that t_e is the edge in F_i which made e to be included in B_i.

Definition 3.1 *b_e is the bottom endpoint of t_e.*

For each i, $1 \leq i \leq \lfloor k/2 \rfloor$, let l_i be the number of b_e's which are leaves of F_i.

Our improvement is first based on the following stronger version of Lemma 3.6 in [10]:

Lemma 3.2 $(k - 2(i - 1))|B_i| + (i - 1)l_i \leq opt$, *for all* i, $1 \leq i \leq \lfloor k/2 \rfloor$.

Proof. The number of pairs (u, h), where u is a vertex of G and h is an edge of OPT incident to u, is $2opt$. Next, we compute a lower bound for this number.

For each edge e in B_i, OPT has at least $k - 2(i-1)$ edges in P_e ([10, page 442]). So there are at least $2(k - 2(i-1))$ pairs associated to e: two for each of the $k - 2(i-1)$ edges of OPT in P_e. If b_e is a leaf of F_i, e can be associated to $(k - 2(i-1)) + k$ pairs. The first $k - 2(i-1)$ pairs would be the ones mentioned above whose first coordinate is not b_e. The remaining k pairs have b_e as first coordinate: since OPT is k-edge-connected, there are k edges of OPT incident to b_e.

Let us show that no two edges in B_i have an associated pair in common. Khuller and Raghavachari proved that the sets P_e's are pairwise disjoint ([10, page 442]), so the pairs coming from P_e's are certainly distinct. The k pairs corresponding to b_e cannot be repeated also, since the only edges in P_f's with b_e as endpoint are in P_e.

Therefore, we must have $l_i((k - 2(i-1)) + k) + (|B_i| - l_i)2(k - 2(i-1)) \leq 2opt$, which is equivalent to the statement of the lemma, concluding the proof. ■

We will present a lower bound on l_i, for $1 \leq i \leq \lfloor k/2 \rfloor$, which will imply an improved upper bound on $|B_i|$. Let us fix i such that $1 \leq i \leq \lfloor k/2 \rfloor$. For such an i,

$$|P_e| \geq k - 2(i-1) \geq 2 \tag{1}$$

for any e in B_i. Denote by B the set $\{b_e : e \in B_i\}$. The following fact will be used in the next lemma.

Fact 3.3 *For any $e, f \in B_i$ such that b_f is a child of b_e, there is a child of b_f not in B.*

Proof. Consider $e, f \in B_i$ such that b_f is a child of b_e. Because $P_e \cap P_f = \emptyset$, all the edges in P_f must have b_e as their top endpoint. By (1), $|P_f| \geq 2$. Thus there is at least one edge g in $P_f - \{t_f\}$. The bottom endpoint of g is not b_f (otherwise g would have the same two endpoints of f). Because b_f is a child of b_e, we must have that f was inserted in B_i before e. At the moment f was included in B_i, there could not be other edge h in B_i such that t_h is on the path in F_i between the two endpoints of g. This is true because if, by contradiction, we assume there is an h in B_i such that t_h is on the path in F_i between the two endpoints of g, we must have that $g \in P_h$. But $g \in P_f$ too, contradicting $P_f \cap P_h = \emptyset$. So, the child of b_f on the path from b_f to the bottom endpoint of g is not in B. ■

The lower bound on l_i is given by the following lemma.

Lemma 3.4 *$l_i \geq 2|B_i| - n$, for any i, $1 \leq i < k/2$, and, for even k, $l_{k/2} \geq 3|B_{k/2}| - 2n$.*

Proof. Fix i such that $1 \leq i < k/2$. Denote by N the set of b_e's in B which are not leaves in F_i. We will associate to each b_e in N a distinct vertex z_e in $V - B$. The association is made in two phases:

Phase 1. For each b_e in N which has a child in $V - B$.

Let z_e be a child of b_e in $V - B$.

Phase 2. For each b_e in N whose children are all in B.

Let $h \in B_i$ be such that b_h is one of b_e's children. By Fact 3.3, b_h has a child not in B.

Case 1: b_h has two or more children not in B.

Let z_e be any child of b_h not in B and distinct of z_h.

Case 2: b_h has exactly one child not in B.

Note that b_h satisfies the statement of Phase 1, which means that z_h was assigned in Phase 1. So this child is in fact z_h. But $i \leq (k-1)/2$, which implies that $|P_h| \geq k - 2(i-1) \geq 3$. This means that there is at least one edge g in P_h which has a descendent of z_h as bottom endpoint. Let z_e be the child of z_h on the path from z_h to the bottom endpoint of g. Observe that z_e cannot be in B, otherwise the edge inserted in B_i by the edge whose bottom endpoint is z_e would forbid b_h to be in B. (See Figure 3.)

Let us verify that distinct b_e's are assigned distinct z_e's. In the first phase, clearly no two distinct b_e's are assigned the same z_e. Also no two z_e's assigned in Phase 2, Case 1 coincide. The same for two z_e's assigned in Phase 2, Case 2. We explicitly make sure that a z_e assigned in Phase 2, Case 1 does not coincide with a z_e assigned in Phase 1. A z_e assigned in Phase 2, Case 2 cannot coincide with a z_e assigned in Phase 1, because $z_h \notin B$. Finally, note that, in Phase 2, Case 1, z_e is a grandchild of b_e, while in Phase 2, Case 2 z_e is a great-grandchild of b_e. Thus, the only way of Phase 2, Case 1 and Phase 2, Case 2 select the same z_e is if b_h in Case 2 had its z_h assigned in Phase 2, Case 1. But, as we have said, b_h had its z_h assigned in Phase 1. This completes the proof that we can assign a different vertex in $V - B$ to each vertex in N.

Figure 3: (a) Situation in Phase 1. (b) Situation in Phase 2, Case 1. (c) Situation in Phase 2, Case 2. (d) Situation in the new Phase 2, Case 2 (used later).

Let $Z = \{z_e : e \in B_i \text{ and } b_e \in N\}$. We have that $|Z| = |N|$, and $|B| = |B_i|$. Moreover, $N \subseteq B$ and $Z \subseteq V - B$. Thus $l_i = |B - N| = |B| - |N| = |B| - |Z| \geq |B| - |V - B| = |B| - (n - |B|) = 2|B| - n = 2|B_i| - n$, completing the proof of the first part of Lemma 3.4.

The proof of the second part differs from the above proof only at Phase 2, Case 2. For $i = k/2$ (even k), we just have $|P_h| \geq 2$. Therefore, z_h might not have a child to be selected as z_e. So, instead of associating to each b_e in N a *distinct* z_e in $V - B$, we associate a z_e which might have already been selected, but at most once. In other words, each $z \in Z = \{z_e : e \in B_i \text{ and } b_e \in N\}$ will correspond to at most two b_e's in N. More specifically, in Phase 2, Case 2, we select $z_e = z_h$. This implies $|N| \leq 2|Z|$, and we have $l_{k/2} = |B| - |N| \geq |B| - 2|Z| \geq |B| - 2(n - |B|) = 3|B| - 2n = 3|B_{k/2}| - 2n$, completing the proof of the lemma. ■

Now, using Lemma 3.4 and $n \leq 2opt/k$ in Lemma 3.2, a new upper bound on $|B_i|$ follows.

Corollary 3.5

$$\begin{aligned} |B_i| &\leq \left(\frac{1}{k} + \frac{2(i-1)}{k^2}\right) opt, \quad \textit{for any } i \textit{ such that } \ 1 \leq i < k/2. \\ |B_{k/2}| &\leq \frac{2(3k-4)}{k(3k-2)}\ opt. \end{aligned}$$

This bound is not the best we can get using this technique, as we will show in the next section. But it implies the better upper bound of 1.75 on the performance ratio of Khuller and Raghavachari's algorithm, as stated below.

Theorem 3.6 *The performance ratio of Khuller and Raghavachari's algorithm is at most 1.75 for*

all k. More specifically, it is at most

$$1.75 - \frac{3k^2 - 6k + 8}{2k^2(3k-2)},\ \textit{for even } k,\ \textit{and}\ \ 1.75 - \frac{2k-3}{4k^2},\ \textit{for}\ \ \textit{odd } k > 1.$$

Proof. For $k = 1$, the algorithm is optimum, i.e., the ratio is 1. For odd $k > 1$, we have (the inequality holding because $|F_i| < n$ and by Corollary 3.5):

$$\begin{aligned}
\frac{|S_{\frac{k+1}{2}}|}{opt} &= \frac{\sum_{i=1}^{(k+1)/2}|F_i| + \sum_{i=1}^{(k-1)/2}|B_i|}{opt} \\
&\leq \frac{\frac{k+1}{2}n}{\frac{kn}{2}} + \sum_{i=1}^{(k-1)/2}\left(\frac{1}{k} + \frac{2(i-1)}{k^2}\right) = \left(1 + \frac{1}{k}\right) + \frac{k-1}{2k} + \frac{2}{k^2}\sum_{i=1}^{(k-1)/2}(i-1) \\
&= \frac{3}{2} + \frac{1}{2k} + \frac{(k-1)(k-3)}{4k^2} = \frac{3}{2} + \frac{k^2 - 2k + 3}{4k^2} = \frac{7}{4} - \frac{2k-3}{4k^2}.
\end{aligned}$$

Observe that $2k - 3 > 0$ for all $k \geq 3$ and $4k^2 > 0$ for all positive k.

The accounting is similar for even k.

$$\begin{aligned}
\frac{|S_{k/2}|}{opt} &= \frac{\sum_{i=1}^{k/2}|F_i| + \sum_{i=1}^{k/2}|B_i|}{opt} \leq \frac{\frac{k}{2}n}{\frac{kn}{2}} + \sum_{i=1}^{k/2-1}\left(\frac{1}{k} + \frac{2(i-1)}{k^2}\right) + \frac{2(3k-4)}{k(3k-2)} \\
&= 1 + \left(\frac{1}{2} - \frac{1}{k}\right) + \frac{2}{k^2}\sum_{i=1}^{k/2-1}(i-1) + \frac{2(3k-4)}{k(3k-2)} \\
&= \frac{3}{2} + \frac{k^2 - 10k + 8}{4k^2} + \frac{2(3k-4)}{k(3k-2)} \\
&= \frac{3}{2} + \frac{1}{4} - \frac{5k-4}{2k^2} + \frac{2(3k-4)}{k(3k-2)} = \frac{7}{4} - \frac{3k^2 - 6k + 8}{2k^2(3k-2)}.
\end{aligned}$$

In this case, $3k^2 - 6k + 8 > 0$ always, and $2k^2(3k-2) > 0$ for all positive k. Hence the ratio is in fact always smaller than 1.75 for all values of k. ■

4 Generalized tree-carvings

In this section, we generalize some of the results from the previous section. In particular, we present stronger versions of both Lemmas 3.2 and 3.4. From these, we get an even better upper bound on the $|B_i|$'s, and, consequently, on the performance ratio of Khuller and Raghavachari's algorithm for large values of k.

We start by introducing a generalization of the tree-carving concept and of some results given in [13].

Consider a positive integer k, a k-edge-connected graph $G = (V, E)$, a non-negative integer $c < k$, and a subset S of E such that (V, S) is a c-edge-connected spanning subgraph of G.

A *c-tree-carving in G with respect to S* is a partition of the vertex set V into subsets $V_1, V_2, \ldots, V_a$ with the following properties.

1) Each subset consists of a node of a *rooted forest* Γ.

2) For every edge $e = uv$ of $(V, E - S)$, the unique x and y such that $u \in V_x$ and $v \in V_y$ satisfy either $x = y$ or V_x is adjacent to V_y in Γ.

3) For each arc e in Γ, there is a cut $(Y_e, \overline{Y}_e) = \{e\}$ in Γ such that, for $X_e = \cup\{V_j : V_j \in Y_e\}$, the cut $(X_e, \overline{X}_e)$ in G contains exactly c edges of S. Denote by P_e the set of edges in $(X_e, \overline{X}_e)$ not in S.

When $c = 0$, $S = \emptyset$ and Γ is a tree, this definition is the same as the tree-carving definition.

Khuller and Raghavachari's algorithm can be modified so that, at each phase, a $2(i-1)$-tree-carving in G with respect to S_{i-1} is computed. This is done as follows. At the end of phase i, let $T_i = \{t_e : e \in B_i\}$. Note that $T_i \subseteq F_i$ and $|T_i| = |B_i|$. Let Γ_i be the rooted forest obtained from F_i by contracting all the edges of $F_i - T_i$. Clearly, Γ_i has exactly $|T_i| = |B_i|$ arcs. Each node of Γ_i corresponds to a subset of the original graph $G = (V, E)$ that was collapsed into this node. So, the nodes of Γ_i are subsets $V_1, V_2, \ldots, V_a$ of V which define a partition of V for some $a \leq n$, where n is the number of vertices of G. Observe that this in fact defines a $2(i-1)$-tree-carving with exactly $|B_i|$ arcs.

Consider a c-tree-carving $V_1, \ldots, V_a$ of G with respect to S, and let Γ, $(X_e, \overline{X}_e)$ and P_e be as in the definition above.

Fact 4.1 *Any k-edge-connected spanning subgraph of G has at least $k - c$ edges in each P_e.*

Proof. Let H be a k-edge-connected spanning subgraph of G. For each arc e in Γ, there must be at least k edges of H in $(X_e, \overline{X}_e)$. Exactly c edges in $(X_e, \overline{X}_e)$ are in S. Hence, there must be at least $k - c$ edges of H in $(X_e, \overline{X}_e)$ not in S, that is, in P_e. ■

Fact 4.2 *For any distinct arcs e and f in Γ, the edge sets $P_e \subseteq E(G)$ and $P_f \subseteq E(G)$ are disjoint.*

Proof. Let $e = (V_i, V_j)$ and $f = (V_x, V_y)$ be two arcs in Γ. Suppose that $P_e \cap P_f \neq \emptyset$. Let us prove

that $e = f$. Let $h \in P_e \cap P_f$. By the definition of P_e in 3), h has one endpoint in X_e and the other in $\overline{X}_e$ and $(Y_e, \overline{Y}_e) = \{e\} = \{(V_i, V_j)\}$. So, by 2), h has one endpoint in V_i and the other in V_j. Also, for the same reason, h has one endpoint in V_x and the other in V_y. But, by the unicity mentioned in 2), we must have ($V_i = V_x$ and $V_j = V_y$) or ($V_i = V_y$ and $V_j = V_x$). In both cases, $e = f$. ■

Denote by t the number of non-root nodes of Γ and by l the number of non-root leaves of Γ. (Note that $t = |E(\Gamma)|$.) The following result is an extension of Lemma 3.2.

Lemma 4.3 *Any k-edge-connected spanning subgraph of G has at least $(k-c)t + cl/2$ edges.*

Proof. Let H be a k-edge-connected spanning subgraph of G. Let us count the number of pairs (V_y, h), where h is an edge of H in $(V_y, \overline{V}_y)$.

For each non-root V_j of Γ, let V_i be its parent in Γ, and e be the arc (V_i, V_j). Recall that $P_e \subseteq E - S$. By Fact 4.1, H has at least $k - c$ edges in P_e. So there are at least $2(k-c)$ pairs associated to e: two for each of the $k - c$ edges of H in P_e (one with V_i as first coordinate, the other with V_j).

In fact, if V_j is a leaf of Γ, arc e can be associated with $(k-c) + k = 2k - c$ pairs. The $k - c$ first pairs would be the ones mentioned above whose first coordinate is V_i. The remaining k pairs have V_j as first coordinate: since H is k-edge-connected, there are k edges of H in $(V_j, \overline{V}_j)$.

Let us show that no two arcs have an associated pair in common. By Fact 4.2, the sets P_e's are pairwise disjoint, so the pairs coming from P_e's are certainly distinct. The k pairs corresponding to a leaf V_j cannot be repeated also since V_j is the endpoint of only one arc in Γ.

Denoting by m the number of edges in H, we can have at most $2m$ pairs. Therefore, $2(k-c)(t-l) + (2k-c)l \leq 2m$, which implies that $m \geq (k-c)t + cl/2$, completing the proof of the lemma. ■

The generalization of Lemma 3.4 we want to present gives a lower bound on l. But before stating the generalization, we need to prove one more fact.

Fact 4.4 *For each arc $e = (V_i, V_j)$ in Γ, $|V_i| + |V_j| \geq 2\sqrt{k-c}$.*

Proof. Consider the cut $(X_e, \overline{X}_e)$ in G corresponding to e. Since G is k-edge-connected, and $(X_e, \overline{X}_e)$ contains exactly c edges of S, there must be at least $k - c$ edges of $E - S$ in $(X_e, \overline{X}_e)$. These edges must have one endpoint in V_i and other in V_j. The maximum number of edges with one endpoint in V_i and the other in V_j is $|V_i| \cdot |V_j|$. Thus $|V_i| \cdot |V_j| \geq k - c$. This implies that

$|V_i| + |V_j| \geq 2\sqrt{k-c}$. ■

Now we can prove the following lower bound on l, the number of leaves of the forest Γ.

Lemma 4.5 $\quad l \geq \left(1 + \frac{1}{\lfloor\sqrt{k-c}\rfloor - 1}\right) t - \frac{1}{(\lfloor\sqrt{k-c}\rfloor - 1)}\ n.$

Proof. The proof of this lemma uses basically the same technique used in Lemma 3.4. We will associate a set Z_i of vertices of V with each non-root V_i, where $Z_i \subseteq V_i \cup V_j$ for some V_j which is adjacent to V_i in Γ. The sets Z_i will be pairwise disjoint and will contain exactly $\lfloor\sqrt{k-c}\rfloor$ vertices if V_i is not a leaf of Γ, and exactly one vertex if V_i is a leaf of Γ. The association is done in two phases. (Phase 1 is executed before Phase 2.)

Phase 1. For each non-root V_i which is a leaf of Γ or such that $|V_i| \geq \lfloor\sqrt{k-c}\rfloor$:

Let Z_i be a subset of V_i with exactly $\lfloor\sqrt{k-c}\rfloor$ vertices if V_i is not a leaf of Γ, and with one vertex if V_i is a leaf. (Note that $|V_i| \geq 1$ always.)

Phase 2. For each non-root V_i which is not a leaf of Γ and such that $|V_i| < \lfloor\sqrt{k-c}\rfloor$:

Let $e = (V_i, V_j)$ be an arc of Γ joining V_i to one of its children V_j. By Fact 4.4, $|V_i|+|V_j| \geq 2\sqrt{k-c}$. Thus we must have $|V_j| \geq 2\sqrt{k-c} - |V_i| \geq \sqrt{k-c}$, and so $|V_j| \geq \lfloor\sqrt{k-c}\rfloor$, which implies that V_j had its set Z_j assigned previously in Phase 1. Choose Z_i to be a subset of $V_i \cup (V_j - Z_j)$ with exactly $\lfloor\sqrt{k-c}\rfloor$ vertices. (There is such a subset since $|V_i \cup (V_j - Z_j)| = |V_i| + |V_j| - \lfloor\sqrt{k-c}\rfloor \geq \lfloor\sqrt{k-c}\rfloor$.)

In Phase 1, clearly the assigned sets are pairwise disjoint. It is also clear that a set assigned in Phase 2 does not intersect a set assigned in Phase 1. By Fact 4.4, two sets assigned in Phase 2 cannot be associated with adjacent sets V_y. So they are disjoint.

Since the union of the sets Z_i is a subset of V, we must have $(t-l)\lfloor\sqrt{k-c}\rfloor + l \leq n$, where, as before, t is the number of non-root nodes of Γ. This means

$$l \geq \frac{\lfloor\sqrt{k-c}\rfloor}{\lfloor\sqrt{k-c}\rfloor - 1} t - \frac{1}{\lfloor\sqrt{k-c}\rfloor - 1}\ n = \left(1 + \frac{1}{\lfloor\sqrt{k-c}\rfloor - 1}\right) t - \frac{1}{(\lfloor\sqrt{k-c}\rfloor - 1)}\ n,$$

concluding the proof of the lemma. ■

We showed how to modify the algorithm so that it outputs at each phase a $2(i-1)$-tree-carving, denoted simply by Γ_i, of G with respect to S_{i-1} and having $|B_i|$ arcs. Recall that l_i is the number of b_e's which are leaves of F_i. Recall that b_e is the bottom endpoint of $t_e \in T_i$. If b_e is a leaf of F_i

then t_e is the only edge of F_i incident to b_e, which means that $\{b_e\}$ is the vertex set of a connected component of $(V, F_i - T_i)$. In other words, there is an x such that $V_x = \{b_e\}$. Finally, note that V_x is a non-root leaf of Γ_i. All this implies that Γ_i has at least l_i non-root leaves.

Applying Lemma 4.5 to Γ_i, we get the following inequality:

$$l_i \geq \left(1 + \frac{1}{\lfloor\sqrt{k-2(i-1)}\rfloor - 1}\right)|B_i| - \frac{1}{(\lfloor\sqrt{k-2(i-1)}\rfloor - 1)}\, n, \tag{2}$$

since $t = |E(\Gamma_i)| = |B_i|$.

We also have, for $1 \leq i \leq j$,

$$l_i \geq \left(1 + \frac{1}{\lfloor\sqrt{k-2(j-1)}\rfloor - 1}\right)|B_i| - \frac{1}{(\lfloor\sqrt{k-2(j-1)}\rfloor - 1)}\, n. \tag{3}$$

Using (2), (3), and Lemma 4.3 for OPT, we conclude the new upper bounds on $|B_i|$.

Corollary 4.6

$$|B_i| \leq \frac{k(\lfloor\sqrt{k-2(i-1)}\rfloor - 1) + 2(i-1)}{k((k-(i-1))(\lfloor\sqrt{k-2(i-1)}\rfloor - 1) + (i-1))}\, opt, \tag{4}$$

and, for any $1 \leq i \leq j$,

$$|B_i| \leq \frac{k(\lfloor\sqrt{k-2(j-1)}\rfloor - 1) + 2(i-1)}{k((k-(i-1))(\lfloor\sqrt{k-2(j-1)}\rfloor - 1) + (i-1))}\, opt. \tag{5}$$

Using these new upper bounds, we derive our final result.

Theorem 4.7 *The performance ratio of Khuller and Raghavachari's algorithm, for large enough* k*, is at most 1.7.*

Proof. We will give the proof for even k. The proof for odd k is similar.

We start by using, for some fixed $r \geq 2$, bound (4) for $B_{\frac{k}{2}-r+1}, \ldots, B_{\frac{k}{2}}$. For the remaining B_i's, we use bound (5) with $j = \frac{k}{2} - r$. From this, we infer, for even k, that

$$\begin{aligned}
\frac{|S_{k/2}|}{opt} &= \frac{\sum_{i=1}^{k/2}(|F_i| + |B_i|)}{opt} \leq 1 + \sum_{i=1}^{j}\frac{|B_i|}{opt} + \sum_{i=j+1}^{k/2}\frac{|B_i|}{opt} \\
&\leq 1 + \sum_{i=j+1}^{k/2}\frac{k(\lfloor\sqrt{k-2(i-1)}\rfloor - 1) + 2(i-1)}{k((k-(i-1))(\lfloor\sqrt{k-2(i-1)}\rfloor - 1) + (i-1))} \\
&\quad + \sum_{i=1}^{j}\frac{k(\lfloor\sqrt{k-2(j-1)}\rfloor - 1) + 2(i-1)}{k((k-(i-1))(\lfloor\sqrt{k-2(j-1)}\rfloor - 1) + (i-1))}
\end{aligned}$$

$$
\begin{aligned}
&\leq 1+\sum_{i=j+1}^{k/2}\frac{k\lfloor\sqrt{2r}\rfloor-2}{k((k-(i-1))(\lfloor\sqrt{k-2(i-1)}\rfloor-1)+(i-1))}\\
&\quad+\sum_{i=1}^{j}\frac{k(\lfloor\sqrt{2(r+1)}\rfloor-1)+2(i-1)}{k((k-(i-1))(\lfloor\sqrt{2(r+1)}\rfloor-1)+(i-1))}\\
&\leq 1+\sum_{i=j+1}^{k/2}\frac{2(k\lfloor\sqrt{2r}\rfloor-2)}{k(k-2r)}\\
&\quad+\sum_{i=1}^{j}\frac{k(\lfloor\sqrt{2(r+1)}\rfloor-1)+2(i-1)}{k((k-(i-1))(\lfloor\sqrt{2(r+1)}\rfloor-1)+(i-1))}.
\end{aligned}
$$

Denoting $\lfloor\sqrt{2(r+1)}\rfloor$ by C_r, just to simplify the formulas, we have

$$
\begin{aligned}
\frac{|S_{k/2}|}{opt} &\leq 1+r\frac{2(k\lfloor\sqrt{2r}\rfloor-2)}{k(k-2r)}+\sum_{i=1}^{j}\frac{k(C_r-1)+2(i-1)}{k((k-(i-1))(C_r-1)+(i-1))}\\
&\leq 1+\frac{2r(k\lfloor\sqrt{2r}\rfloor-2)}{k(k-2r)}-\frac{2j}{k(C_r-2)}\\
&\quad+\sum_{i=1}^{j}\left(\frac{2}{k(C_r-2)}+\frac{k(C_r-1)+2(i-1)}{k((k-(i-1))(C_r-1)+(i-1))}\right)\\
&\leq 1+\frac{2r(k\lfloor\sqrt{2r}\rfloor-2)}{k(k-2r)}-\frac{2j}{k(C_r-2)}+\frac{C_r(C_r-1)}{(C_r-2)^2}\left(\ln\frac{q'-1}{q'-1-j}+\epsilon\right)\\
&= 1+\frac{2r(k\lfloor\sqrt{2r}\rfloor-2)}{k(k-2r)}-\frac{k-2r}{k(C_r-2)}+\frac{C_r(C_r-1)}{(C_r-2)^2}\left(\ln\frac{q'-1}{q'-1-\frac{k}{2}+r}+\epsilon\right),
\end{aligned}
$$

where $q'=\left(\frac{C_r-1}{C_r-2}\right)k$. (For the last equality, we used that $j=\frac{k}{2}-r$.) Note that C_r is function solely of r so, as k goes to infinity, the bound on the ratio converges to

$$
1-\frac{1}{C_r-2}+\frac{C_r(C_r-1)}{(C_r-2)^2}\ln\frac{\frac{C_r-1}{C_r-2}}{\frac{C_r-1}{C_r-2}-\frac{1}{2}}=1-\frac{1}{C_r-2}+\frac{C_r(C_r-1)}{(C_r-2)^2}\ln\frac{2(C_r-1)}{C_r}.
$$

Now, let us make r goes to infinity, which implies that C_r goes to infinity as well. The above expression converges to $1+\ln 2<1.7$. From this, we finally conclude that, for large enough k, the performance ratio of the algorithm is at most 1.7. ■

5 The complexity of the problem

In this section, we show that the minimum size 2-edge-connected spanning subgraph problem is MAX SNP-hard. This means that there is a constant $\epsilon>0$ such that the existence of a polynomial-

time approximation algorithm with performance ratio at most $1+\epsilon$ implies that $P = NP$, by results of Arora et al. [1].

As in [17], we use the concept of L-reduction, which is a special kind of reduction that preserves approximability. Let A and B be two optimization problems. We say *A L-reduces to B* if there are two polynomial-time algorithms f and g, and positive constants α and β, such that for each instance I of A,

1. Algorithm f produces an instance $I' = f(I)$ of B, such that the optima of I and I', of costs denoted $opt_A(I)$ and $opt_B(I')$ respectively, satisfy $opt_B(I') \leq \alpha \cdot opt_A(I)$, and
2. Given any feasible solution of I' with cost c', algorithm g produces a solution of I with cost c such that $|c - opt_A(I)| \leq \beta \cdot |c' - opt_B(I')|$.

Theorem 5.1 *The minimum size 2-edge-connected spanning subgraph problem is MAX SNP-hard.*

Proof. Denote by VC_7 the vertex cover problem restricted to graphs with maximum degree seven. Papadimitriou and Yannakakis [17] showed that VC_7 is MAX SNP-hard. We prove that VC_7 L-reduces to the minimum size 2-edge-connected spanning subgraph problem, here denoted by 2-MECSS. The reduction comes from [11], where a directed version of the 2-MECSS is proved to be MAX SNP-hard.

The first part of the L-reduction is a polynomial-time algorithm f and a constant α. Given any instance G of VC_7, f produces an instance H of the 2-MECSS such that the minimum number of edges in a 2-edge-connected spanning subgraph of H, denoted by $opt_{2-MECSS}(H)$, is at most α times the minimum size of a vertex cover in G, denoted by $opt_{VC_7}(G)$. In other words, $opt_{2-MECSS}(H) \leq \alpha \cdot opt_{VC_7}(G)$.

Let us describe algorithm f. Consider an instance G of VC_7. G is a graph with maximum degree seven. Here is a procedure to construct an instance H of the 2-MECSS. Similarly to [11], start with a special vertex, the root. Each vertex in G will have a "current vertex", initially the root. For each edge uv, add a "cover-testing gadget" to H, as illustrated in Figure 4(a). Specifically, add six new vertices $x_1, x_2, x_3, y_1, y_2, y_3$. Vertex x_2 is adjacent only to vertices x_1 and x_3. Analogously, vertex y_2 is adjacent only to vertices y_1 and y_3. There is an edge labeled u^- between x_1 and y_3, and an edge labeled v^- between y_1 and x_3. There is one more edge incident to x_1: an edge labeled u^+ between x_1 and the current vertex of u; and one more edge incident to y_1: an edge labeled v^+

Figure 4: (a) The gadget for edge uv. (b) The modification of the gadget for edge uv.

between y_1 and the current vertex of v. Make y_3 the new current vertex of u, and x_3 the new current vertex of v. Finally, after all edges of G have been considered, for each vertex v in G, add an edge labeled v^+ between its final current vertex and the root. The gadgets are implicitly numbered in the order we have added them. Clearly H can be obtained in polynomial time in the size of G. This completes the description of f.

Next fact will be used in the proof of the existence of a constant α. Let m be the number of edges in G, and s be a positive integer.

Fact 5.2 *If G has a vertex cover with at most s vertices, then H has a 2-edge-connected spanning subgraph with at most $6m + s$ edges.*

Proof. Suppose G has a vertex cover S with at most s vertices. Consider the spanning subgraph H' of H with all edges incident to vertices of type x_2 and y_2, all edges labeled u^+ for $u \in S$, and all edges labeled u^- for $u \notin S$. Observe that all vertices in H', but the root, have degree two. Besides, the only possible cycles in H' are of two types: either they are like $x_1x_2x_3y_1y_2y_3$, or they include the root. Because S is a vertex cover, no cycles of type $x_1x_2x_3y_1y_2y_3$ can occur. This implies that H is a collection of cycles intersecting only at the root, and spanning all vertices. Therefore, H' is a 2-edge-connected spanning subgraph of H. To count how many edges there are in H', just notice that H' has six edges per gadget plus one extra edge per vertex in S. Therefore, in total, H' has at most $6m + s$ edges. ■

Lemma 5.3 $opt_{2-MECSS}(H) \leq 43 \cdot opt_{VC_7}(G)$.

Proof. By Fact 5.2, $opt_{2-MECSS}(H) \leq 6m + opt_{VC_7}(G)$. Also, because G has maximum degree

seven, $opt_{VC_7}(G) \geq m/7$. Putting these together, we get that

$$opt_{2-MECSS}(H) \leq 6m + opt_{VC_7}(G) \leq (6 \cdot 7 + 1) opt_{VC_7}(G) = 43 \cdot opt_{VC_7}(G). \blacksquare$$

So, we can consider $\alpha = 43$.

The second part of the L-reduction is the constant β and algorithm g. We may assume that $m \geq 1$. Note that H has $6m + 1$ vertices, therefore any 2-edge-connected spanning subgraph of H must have at least $6m + 1$ edges. Consider a 2-edge-connected spanning subgraph H' of H with $6m+s$ edges, where s is some positive integer. Algorithm g will produce in polynomial time a vertex cover of size at most s, from H'. From this, and from Fact 5.2, we will have that $opt_{2-MECSS}(H) = 6m+opt_{VC_7}(G)$. And then $|6m+s-opt_{2-MECSS}(H)| = |6m+s-6m-opt_{VC_7}(G)| = |s-opt_{VC_7}(G)|$, meaning that $\beta = 1$ suffices.

So let us see how algorithm g works. In a first phase, g produces another 2-edge-connected subgraph H'' of H with at most as many edges as H', and such that in H'' all vertices, but the root, have degree two. To get H'', each gadget is checked for vertices of degree three (all vertices in H, but the root, have degree at most three). If a vertex x has degree three in H', then the edges labeled u^+ and u^- incident to x appear in H'. Remove the edge labeled u^- incident to x and add (if not already there) the edge labeled u^+ incident to the vertex adjacent to x through the edge labeled u^-. See Figure 4(b).

H'' is the graph obtained after applying this modification while there are vertices, other than the root, of degree three. We can make sure the modification is applied to the lowest numbered gadget, and inside the gadget, to x_1, y_1, x_3, y_3, in this order. This procedure takes polynomial time: the modification can be done in polynomial time in the size of G, and after each modification the number of edges labeled u^- decreases. Clearly H'' has at most as many edges as H'.

The following fact can be verified by the reader.

Fact 5.4 *H'' is a 2-edge-connected spanning subgraph of H.*

Since all vertices in H'', but the root, have degree two, for each vertex in G, either all edges labeled u^+ or all edges labeled u^- appear in H''. In a second phase, a vertex cover of size at most s is computed: let S be the set of all vertices in H'' whose all edges labeled u^+ appear in H''. In any gadget, there must be edges labeled u^+ appearing in H'', otherwise H'' is not connected. Thus S is a vertex cover in G. Recall that H' had $6m + s$ edges. Hence H'' has at most $6m + s$ edges. But

since all vertices but the root have degree two, H'' has $6m + t$ edges, for $t \leq s$, and t is the number of vertices in S. This finishes the description of algorithm g, and the proof of Theorem 5.1. ■

6 Acknowledgments

I would like to thank Vijay Vazirani for suggesting the problem, and S. Khuller for the main reference used on the MAX SNP-hardness proof. I am also grateful to Gruia Calinescu, Naveen Garg, Ion Mandoiu, and Robin Thomas for helpful discussions. I thank the anonymous referees and Howard Karloff for their comments and suggestions.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy, "Proof Verification and Hardness of Approximation Problems," *Proc.* 33^{rd} *IEEE Symposium on Foundations of Computer Science*, 14–23, 1992.

[2] J. Cheriyan, M. -Y. Kao and R. Thurimella, "Algorithms for Parallel k-vertex Connectivity and Sparse Certificates," *SIAM Journal of Computing*, 22 (1), 157–174, 1993.

[3] J. Cheriyan and R. Thurimella, "Approximating Minimum-size k-Connected Spanning Subgraph via Matching," *Proc.* 37^{rd} *IEEE Symposium on Foundations of Computer Science*, October 1996.

[4] H. N. Gabow, M. X. Goemans and D. P. Williamson, "An Efficient Approximation Algorithm for the Survivable Network Design Problem," *Proc.* 3^{rd} *Integer Programming and Combinatorial Optimization Conference* 57–74, 1993.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability*, W. H. Freeman and Co., 1979.

[6] N. Garg, V. Santosh and A. Singla, "Improved Approximation Algorithms for Biconnected Subgraphs via Better Lower Bounding Techniques," *Proc.* 4^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, 103–111, 1993.

[7] M. X. Goemans, A. V. Goldberg, S. Plotkin, D. Shmoys, E. Tardos and D. P. Williamson, "Improved Approximation Algorithms for Network Design Problems," *Proc.* 5^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, 223–232, 1994.

[8] M. X. Goemans and D. P. Williamson, "A General Approximation Technique for Constrained Forest Problems," *SIAM Journal of Computing*, 24 (2), 296–317, 1995.

[9] D. Karger, "Random Sampling in Cut, Flow, and Network Design Problems," *Proc.* 26^{th} *Annual ACM Symposium on the Theory of Computing*, 648–657, 1994.

[10] S. Khuller and B. Raghavachari, "Improved Approximation Algorithms for Uniform Connectivity Problems," *Journal of Algorithms*, 21, 434–450, 1996.

[11] S. Khuller, B. Raghavachari and N. Young, "Approximating the Minimum Equivalent Digraph," *SIAM Journal of Computing*, 24 (4), 859–872, 1995.

[12] S. Khuller and R. Thurimella, "Approximation Algorithms for Graph Augmentation," *Journal of Algorithms*, 14 (2), 214–225, 1993.

[13] S. Khuller and U. Vishkin, "Biconnectivity Approximations and Graph Carvings," *J. Assoc. Comput. Mach.*, 41 (2), 214–235, 1994.

[14] W. Mader, "Minimale n-Fach Kantenzusammenhängende Graphen," *Math. Ann.*, 21–28, 1971.

[15] W. Mader, "Ecken vom Grad n in Minimalen n-Fach Zusammenhängenden Graphen," *Archive der Mathematik*, 23:219–224, 1972.

[16] H. Nagamochi and T. Ibaraki, "Linear-time Algorithm for Finding a Sparse k-connected Spanning Subgraph of a k-connected Graph," *Algorithmica*, 7 (5-6), 583–596, 1992.

[17] C. H. Papadimitriou and M. Yannakakis. "Optimization, Approximation, and Complexity Classes," *Journal of Computer and System Sciences*, 43:425–440, 1991.

[18] R. Ravi and D. Williamson, "An Approximation Algorithm for Minimum-Cost Vertex-Connectivity Problems," *Proc.* 6^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, 332–341, 1995.