Approximation through local optimality: Designing networks with small degree

R. Ravi[1]*, B. Raghavachari[2] and P. Klein[1]**

[1] Brown University, Providence, RI 02912, USA
[2] Pennsylvania State University, University Park, PA 16802, USA

Abstract. We give quasipolynomial-time approximation algorithms for designing networks with minimum degree. Using our methods, one can design one-connected networks to meet a variety of connectivity requirements. The degree of the output network is guaranteed to be at most $(1+\epsilon)$ times optimal, plus an additive error of $O(\frac{\log n}{\epsilon})$ for any $\epsilon > 0$. We also provide a quasipolynomial-time approximation algorithm for designing a two-edge-connected spanning subgraph of a given two-edge-connected graph of approximately minimum degree. The performance guarantee is identical to that for one-connected networks.
As a consequence of our analysis, we show that the minimum degree in both the problems above is well-estimated by certain polynomially solvable linear programs. This fact suggests that the linear programs we describe could be useful in obtaining optimal solutions via branch-and-bound.

1 Introduction

1.1 On minimum-degree networks

We consider the problem of designing minimum-degree communication networks. The criterion of minimizing the maximum degree reflects decentralization of a communication networks. The advantage of a network with low degree is that the failure of a node does not adversely affect the rest of the network too much. The purpose of the network itself gives rise to certain connectivity requirements in the network. We consider essentially two distinct types of such connectivity requirements on the solution subgraph. In the first type, we consider network design problems where the connectivity requirements can be modeled by a $\{0,1\}$-valued function f on all the cuts in the graph. In the second type, we consider the problem of designing a minimum-degree two edge-connected spanning subgraph

* Research supported by an IBM Graduate Fellowship. Additional support provided by NSF PYI award CCR-9157620 and DARPA contract N00014-91-J-4052 ARPA Order No. 8225.

** Research supported by NSF grant CCR-9012357 and NSF PYI award CCR-9157620, together with PYI matching funds from Thinking Machines Corporation and Xerox Corporation. Additional support provided by DARPA contract N00014-91-J-4052 ARPA Order No. 8225.

of a given graph. For problems of both types, we provide an approximation algorithms to output a network whose degree is guaranteed to be at most $1+\epsilon$ times optimal, plus an additive error of $O(\frac{\log n}{\epsilon})$. Both the algorithms run in quasipolynomial time.

We start by reviewing previously known results on special cases of the problem we consider in this paper. We then describe the framework in which we specify the connectivity requirements for one-connected networks. Then we state our results for general one-connected network design and for two-connected spanning network design.

1.2 Previous work

The minimum-degree spanning tree problem: In this problem, we are required to find a spanning tree of the graph whose maximum degree is minimized. This problem is a generalization of the Hamiltonian Path problem and hence is clearly NP-hard [9]. The first result on approximating the minimum-degree spanning tree was that of Fürer and Raghavachari [6]. They gave a polynomial time approximation algorithm for minimum-degree spanning tree with performance guarantee $O(\log n)$. Their algorithm further generalizes to find rooted spanning trees in directed graphs of approximately minimum indegree. In subsequent work [7], they improved their previous results and provided another polynomial time algorithm to approximate the minimum-degree spanning tree to within one of the optimal. Clearly no better approximation algorithms are possible for this problem.

The minimum-degree Steiner tree problem: This is an extension of the above problem, where given an input graph and a distinguished subset of the vertices, D, we seek to find a Steiner tree (spanning at least the set D) whose maximum degree is minimum. The first polynomial-time approximation algorithm was provided by Agrawal, Klein and Ravi [2]. The performance guarantee is a factor of $O(\log k)$, where k is the size of D. This was improved by Fürer and Raghavachari in [6] and they provide a polynomial-time approximation algorithm for the problem with the performance guarantee essentially the same as that we show here. In fact, our work is a generalization of this algorithm and reduces to their algorithm for this special case. Also, in this special case, their analysis shows that our algorithm is actually polynomial-time. Fürer and Raghavachari [8] later demonstrated a polynomial-time algorithm for the Steiner case that finds a tree whose degree is within one from optimal.

1.3 A framework for specifying connectivity

A framework for specifying connectivity requirements for networks was recently proposed by Goemans and Williamson [10]. The framework captures a wide variety of specifications of requirements, including Steiner and generalized Steiner connectivity, general point-to-point connectivity, and T-joins. Building on the work of Agrawal, Klein, and Ravi on the generalized Steiner tree problem [1],

1.4 Toughness, Weakness and generalizations: A lower bound

Toughness, first defined by Chvátal in [4], is a measure of how tough it is to disconnect a graph into many components by removing few nodes. The toughness of a graph is the minimum ratio of nodes removed per component resulting. That is, it is the minimum ratio of $|X|$ to the number of components in $G - X$ where X ranges over all non-trivial node-subsets of G. Computing the toughness of a graph was recently shown NP-complete by Bauer, Hakimi and Schmeichel [3]. The definition of toughness we have given differs slightly from Chvátal's original definition in [4]. According to our definition, the minimum toughness ratio is never more than 1 (for nonsingleton graphs), since even a singleton X yields a ratio of at most 1. Chvátal defines toughness to be the same minimum ratio, but the minimum is taken only over those node-subsets X for which $G - X$ has at least two components. According to this definition, the minimum ratio can be arbitrarily large.

We generalize the above notion to allow active components defined by proper functions f. The f-toughness of a graph for any given function f is the minimum ratio of the nodes removed to the number of *active* components formed. In other words, it is the minimum ratio of $|X|$ and the number of *active* components in $(G - X)$ where X ranges over all non-trivial node-subsets of G. Call any single node that forms an active set a *terminal*. Note that as long as there is at least one terminal, the f-toughness ratio is at most 1, since this is the ratio achieved on removing a single terminal.

We shall denote the reciprocal of the f-toughness of a graph by its f-*weakness*. The notion of weakness of a graph arises as a dual to the problem of constructing minimum-degree networks.

Lemma 2. *For any function f, the optimum value of (IP) is at least the f-weakness of the graph.*

1.5 An approximate min-max relation and applications

How good a lower bound is the f-weakness of a graph for the minimum degree problem? In the course of proving the performance guarantee for the solutions we construct for (IP), we demonstrate an approximate min-max equality between the optimum value of (IP) and the f-weakness of the graph.

Theorem 3. *The optimum value of (IP) is at most $(1+\epsilon)w^* + O(\frac{\log n}{\epsilon})$ for any $\epsilon > 0$ where w^* is the f-weakness of the graph.*

The proof of Theorem 3 is algorithmic. We provide an algorithm that constructs a f-join whose degree is close to the f-weakness ratio of a set we identify. Theorem 3 and Lemma 2 together prove the performance bounds given in Theorem 1. In addition, we also have the following result about approximating the f-weakness of the graph for any proper function f.

Goemans and Williamson showed how to find a network satisfying given connectivity requirements that has nearly minimum total edge cost. In this paper, we show how to find a network satisfying the requirements that has nearly minimum degree. All we use of [10] is their framework for specifying connectivity requirements; our algorithm and analysis is based on the work of Fürer and Raghavachari [7].

Consider a spanning tree of a graph, and any cut in the graph. At least one edge of the spanning tree must cross this cut. Conversely, if a network crosses every cut, it must span all nodes. More generally, in order to specify connectivity requirements for a network, we designate a subset of the cuts in a graph as *active* cuts, and we require that the network cross every active cut.

A broad class of requirements can be specified using an approach of Goemans and Williamson [10]. They specify which cuts are active using a 0–1 function f on the node-subsets of a graph. For a node-subset S, let $\Gamma(S)$ denote the set of edges each having exactly one endpoint in S. To specify that the cut $\Gamma(S)$ is active, we set $f(S)$ to be 1. Using this formalism, one can formulate an integer program for a minimum-degree network crossing all active cuts:

Min d

subject to constraints:

$$\begin{array}{ll} x(\Gamma(S)) \geq f(S) & \forall\, \emptyset \neq S \subset V \\ \sum_{e \in \Gamma(\{v\})} x_e \leq d & \forall\, v \in V \qquad \text{(IP)} \\ x_e \in \{0,1\} & \forall\, e \in E \end{array}$$

Here $x(F) = \sum_{e \in F} x_e$. We shall call any feasible solution to (IP) an f-join. Minimal f-joins are forests [10].

Goemans and Williamson [10] focus on a class of functions f that can be used to formulate many important connectivity requirements. They called these functions *proper*. A function $f : 2^V \rightarrow \{0,1\}$ is proper if the following properties hold.

- [Null] $f(\emptyset) = 0$;
- [Symmetry] $f(S) = f(V - S)$ for all $S \subseteq V$; and
- [Disjointness] If A and B are disjoint, then $f(A) = f(B) = 0$ implies $f(A \cup B) = 0$.

We are interested in solutions to (IP) for the class of proper functions f. We present algorithms that closely follow the algorithm of Fürer and Raghavachari [7] for approximating the minimum-degree Steiner tree. One of the main results of this paper is the following.

Theorem 1. *Suppose f is a proper function. Let d^* denote the optimum value of (IP). There is an $n^{O(\log n)}$-time algorithm for finding a feasible solution to (IP) whose objective value is at most $(1+\epsilon)d^* + O(\frac{\log n}{\epsilon})$ for any $\epsilon > 0$.*

Some examples of problems that can be solved approximately using the algorithm in the above theorem are the minimum-degree generalized Steiner network problem, the minimum-degree T-join problem, and minimum-degree point-to-point connection problems.

Theorem 4. *Let f be a proper function on the nodes of an n-node graph. Let w^* denote the f-weakness of a graph and let ϵ be a small positive constant. There is an $n^{O(\log n)}$-time algorithm to determine a node subset X for which the f-weakness ratio is at least $(1-\epsilon)w^* - O(\frac{\log n}{\epsilon})$.*

An important application of the approximate min-max equality presented in Theorem 3 arises from the fact that the relaxed version of (IP) is polynomially solvable [11]. It follows from Theorem 3 that the value of the linear program is a good estimate of the minimum degree. While solving the linear program does not give us a solution network with this degree, just knowing the value can be useful, namely, in a branch-and-bound search for an optimum solution [12].

1.6 Minimum-degree two-connected subgraphs

In this paper, we also consider minimum-degree networks of a different type. Given as input a two edge-connected undirected graph, we consider the problem of finding a two edge-connected spanning subgraph of minimum degree. This problem can be easily seen to be NP-complete by a reduction from the Hamiltonian cycle problem. Using a local-improvement algorithm similar to that used in proving Theorem 1, we obtain an approximate solution for this problem as well.

Theorem 5. *Let Δ^* be the minimum degree of a two edge-connected subgraph of a given graph G and let ϵ be a positive constant. There is an $n^{O(\log n)}$-time algorithm for finding a two edge-connected subgraph of G having degree at most $(1+\epsilon)\Delta^* + O(\frac{\log n}{\epsilon})$.*

2 The minimum-degree constrained forest problem

In this section, we describe the algorithm for providing an approximate solution to (IP) and prove Theorem 3. As input we are given an undirected graph $G = (V, E)$, an arbitrary constant $\epsilon > 0$ to determine the performance accuracy, and a proper function f defined on it. We can assume without loss of generality that the graph G is connected, for otherwise we can work on each connected piece independently. We find a forest F that is feasible for the covering constraints of (IP) using an iterative local-improvement algorithm. Note that we can always assume without loss of generality that any feasible solution to (IP) is the incidence vector of a forest [10]. The claim on the running time of the algorithm is proved using a potential function argument.

2.1 Overview

The algorithm starts with a feasible solution to (IP) and iteratively applies improvement steps aimed at reducing the degree of high-degree vertices. Intuitively,

if we find a cycle in the graph that contains a high degree node, and if all edges that must be added to the current feasible solution to form this cycle are incident to low degree nodes, then we can improve the current solution by adding in the cycle and deleting an edge incident to the high degree node. This reduces the degree of the high degree node by one.

2.2 Constructing an initial feasible solution

We start with any spanning tree T of the given connected graph as the initial feasible solution. It is easy to verify that T is feasible.

2.3 An improvement step

Denote the set of nodes in the current forest with degree at least d by S_d. An improvement step with target d tries to reduce the degree of a node in S_d. We use two types of improvement steps. The first and simpler type examines if the forest F remains feasible on deleting an edge incident on a node in S_d. If so, we delete this edge from F to obtain the new forest F' and proceed to the next improvement step. The second type of improvement step is more involved. Starting from G, we delete all the edges in $E - F$ that are incident to nodes in the forest having degree at least $d-2$, i.e., edges that are incident to any node in S_{d-2}. In this graph, we examine if any node in S_d is in a cycle. If so, we add all the edges in $E - F$ in this cycle to the forest F and delete an edge incident to the the node in S_d in this cycle. This gives a new forest F'.

Note that after performing an improvement step with target d, the degree of a node in S_d reduces by one and the resulting degree of all the other affected vertices increases by at most two and is at most $d-1$. We note that the resulting forest F' is a f-join.

Lemma 6. *The forest F' formed at the end of an improvement step is feasible.*

2.4 The algorithm

The algorithm starts with an initial feasible solution. Let the maximum degree of any node in the current feasible solution be k. We apply improvement steps with target d for $d \geq k - \lceil \log_{1+\epsilon} n \rceil$ until no such improvements are possible. The resulting forest is our *locally optimal* approximate solution.

Definition 7. Define an edge e of an feasible forest F to be *critical* if $F - e$ is not feasible.

As a result of performing the first type of improvement steps, note that all the edges incident on nodes of degree at least $k - \lceil \log_{1+\epsilon} n \rceil$ are critical.

2.5 Termination

Each improvement step is polynomial-time implementable. We adapt a potential-function argument from [7] to bound the number of improvement steps. Define the potential of a vertex v with degree d in the forest F to be $\phi(v) = n^d$ where n is the number of nodes in the graph. The potential of the forest F is defined as $\sum_v \phi(v)$. Initially $\phi(F) \leq n^n$. Each improvement step with target d reduces the potential by at least n^{d-2}, since the degree a node of degree d is reduced by 1 and the degree of all the other nodes may increase to $d-1$. Since $d \geq k - \lceil \log_{1+\epsilon} n \rceil$ where k is the maximum degree of the current forest, this reduction in potential is at least a fraction $n^{-O(\log n)}$ of the current potential of the forest $\phi(F)$. It follows that the number of improvement steps is $n^{O(\log n)}$.

2.6 Performance guarantee

We prove the performance guarantee by identifying a node separator from the solution subgraph whose f-weakness value is very close to the degree of the solution. Let k denote the maximum degree of the solution subgraph. The proposition below can be proved by a simple contradiction argument.

Proposition 8. *There is some $i \in [k - \lceil \log_{1+\epsilon} n \rceil, k]$ with $|S_{i-2}| \leq (1+\epsilon)|S_i|$.*

Let i be as in Proposition 8. We prove Theorem 3 by estimating the f-weakness of S_{i-2}. In the following lemma, we show that the number of active components in $G - S_{i-2}$ is at least $i|S_i| - (2+\epsilon)|S_i|$. Therefore the f-weakness of S_{i-2} and hence of G is at least $\frac{i-(2+\epsilon)}{b}$. Note that the degree of the solution F is at most $i + \log_b n$ and $b = 1 + \epsilon$. Combining these proves Theorem 3.

Lemma 9. *Let F be the locally optimal solution network and let i be the index in Proposition 8. The number of active components in $G - S_{i-2}$ is at least $i|S_i| - (2+\epsilon)|S_i|$.*

We prove the above lemma in the remainder of this section. Call a tree in F *relevant* if it contains a node in S_i. Define the equivalence class $\sim$ on the edges of F by the rule: $e_1 \sim e_2$ if there is a path between e_1 and e_2 in F avoiding S_i. Define an auxiliary tree F_i whose nodes are S_i together with the equivalence classes of $\sim$. F_i is bipartite, and there is an edge between a node $v \in S_i$ and an equivalence class C of $\sim$ if there is an edge in C incident on v. Note that if a node v of S_i has degree j in F, it also has degree j in F_i.

Claim 10. *The number of equivalence classes C of degree one in F_i is at least $(i-2)|S_i| + 2r$ where r is the number of relevant trees in F.*

Proof. Let t be the number of classes of degree one and let p be the number of other classes. Then the number of edges in the forest F_i is exactly $t + p + |S_i| - r$. The sum of the degrees of the nodes of F_i is twice the number of edges. That sum is at least $i \cdot |S_i| + 2p + t$. The claim follows. □

Each class C of degree one defines a set X_C of nodes as follows: a node v is in X_C if C contains all the edges of F that are incident on v. Note that each X_C is a component of $F - S_i$ and thus represents a subset of nodes of a relevant tree. Moreover, we have the following proposition from the criticality of all the edges incident on S_i in F.

Proposition 11. *Each set X_C defined by a class C of degree one in F_i is an active set.*

We now derive a lower bound on the number of active components of $G - S_{i-2}$. Most of the node sets representing degree one classes of F_i are useful in forming these active components. There are two ways in which such a set might fail to be an active component of $G - S_{i-2}$. When nodes of $S_{i-2} - S_i$ are removed, some of these components may break up, but few can break up this way. By choice of i, the number of such components is at most $|S_{i-2}| - |S_i|$, which is at most $\epsilon |S_i|$. We will disregard these broken-up components. Secondly, when nonforest edges are added, some of the remaining components may merge, possibly resulting in fewer components and inactive components. We show that in fact there remain many active components.

Claim 12. *The total number of such sets X_C's that merge with others on adding edges of $G - S_{i-2}$ is at most $r - 1$.*

Proof. Suppose for a contradiction that r or more such X_C's merge on adding the edges of $G - S_{i-2}$. This identifies a cycle in $F \cup (G - S_{i-2})$ containing a node of S_i and so F permits an improvement of the second type. This contradicts the local optimality of F. □

The sets X_C's that remain might merge with inactive components of F on adding the nonforest edges of $G - S_{i-2}$. Using the properties of a proper function, we can prove that the resulting component in $G - S_{i-2}$ containing X_C is also active. Counting the number of such sets X_C that remain proves Lemma 9.

3 Minimum-Degree 2-Edge Connected Subgraphs

3.1 Preliminaries

Let $G = (V, E)$ be an arbitrary k-edge connected graph. Consider the problem of computing a k-edge connected subgraph N of G which spans V such that the maximum degree of N is minimum. The case of $k = 1$ corresponds to the minimum-degree spanning tree problem and is NP-hard [9]. So far there have been no results when $k > 1$. In this section, we consider the special case when $k = 2$ (two-edge-connected graphs are also called bridge-connected). In this section, "connectivity" always stands for edge connectivity (and not vertex connectivity). For a graph G, we use $\Delta^*(G)$ to denote the minimum degree of a two-connected spanning subgraph of G.

We first give some preliminary definitions which show that the term k-connected components is meaningful in the context of edge connectivity. It allows the partition of V into k-connected components.

Definition 13. A pair of vertices u and v are said to be *k-connected* in a graph N if there are k edge disjoint paths between u and v in N. This relation is an equivalence relation. It partitions the vertices into equivalence classes of vertices. Within each class, each pair of vertices is k-connected. Such a class is called a *k-component.*

Definition 14. For a graph H, the *bridge-connected forest (bcf)* of H is obtained by contracting each two-component of H to a supervertex and deleting self-loops. A *leaf two-component* of H is a two-component that has degree one in the bcf of H. An *isolated two-component* of H is one of degree zero in the bcf of H.

An easy lower bound on the minimum degree of a two-connected spanning subgraph is given below.

Lemma 15. *Let N be any 2-connected spanning subgraph of G, and let X be any subset of vertices. Let C be a 2-component C of $G - X$. If C is a leaf 2-component, then there is at least one edge of N between C and X. If C is an isolated 2-component, then there are at least two edge of N between C and X.*

Definition 16. Let H be a graph with ℓ leaf two-components and k isolated two-components. Then the *deficiency* of H is defined as $\ell + 2k$ and is denoted by $\mathrm{def}(H)$.

Corollary 17. *Let X be a subset of the nodes of a graph G. Then*

$$\Delta^*(G) \geq \frac{def(G - X)}{|X|}$$

3.2 Overview of the Algorithm

The local search algorithm proceeds as follows. Fix ϵ to be an arbitrary positive constant. The algorithm starts with an arbitrary 2-connected subgraph, and iteratively applies local improvement steps to reduce the degrees of high degree vertices. When a local minimum is reached, the algorithm outputs the current subgraph.

Now we give definitions pertaining to an improvement step. For an edge $e \in N$, we construct an auxiliary graph N_e from N by contracting each two-component of $N - e$ to a single supervertex. Since N is two-connected, it follows that N_e is a simple path.

Definition 18. The *degree-d candidate graph* for an edge e in N is obtained from G as follows: contract each two-component of $N - e$, and then delete edges of $G - N$ incident to nodes of degree more than d in N.

A vertex w of degree d in N is said to *admit an improvement* if there is an edge e incident to w in N such that the degree-$(d-3)$ candidate graph for e is two-connected.

An improvement step consists in replacing e with edges from the candidate graph so that the resulting network remains two-connected. We require that no vertex's degree is increased beyond $d-1$. The following lemma shows that this can be done.

Lemma 19. *Let Q be a minimal collection of edges whose addition two-connects N_e. Then for every node v of N_e, $\deg_Q(v) \leq 2$.*

Note than an improvement step for a vertex w reduces w's degree by one, and does not increase any vertex's degree to more than w's new degree.

Definition 20. Let N be a two-connected spanning subgraph of G, and let Δ be the degree of N. Then N is called *locally optimal* if no vertex of degree at least $\Delta - \lceil \log_{1+\epsilon} n \rceil$ admits an improvement.

Using a potential function argument as in the previous section, it is easy to show that $n^{O(\log n)}$ improvement steps suffice to yield a locally optimal subgraph. Since each improvement step can be executed in polynomial time, the running time is as claimed in Theorem 5. In the remainder of this section, we show that a locally optimal subgraph has low degree.

Theorem 21. *The degree of any locally optimal subgraph of a graph G with n nodes is at most $(1+\epsilon)\Delta^*(G) + \lceil \log_{1+\epsilon} n \rceil + 4(1+\epsilon)$.*

Let S_i denote the set of nodes whose degree in N is at least i. The proof of Theorem 21 consists in showing that there is an i in the range $[\Delta - \lceil \log_{1+\epsilon} n \rceil, \Delta]$ for which $G - S_i$ has many leaf and isolated two-components. We show this in Lemma 25. The result then follows from Corollary 17.

Lemma 22. *Let G be a 2-connected graph, and let S be a set of vertices of G. Then there is a 2-connected subgraph A of G that contains all vertices of S, such that $\sum_{v \in S} \deg_A(v) \leq 4|S|$, where $\deg_A(v)$ denotes the degree of v in A.*

Proof. The proof is constructive and we provide an algorithm to compute A. For a minor H of G, we say a vertex v of H is *active* if one of the vertices contracted to form v belongs to S. Consider the following algorithm:

- Let H_0 be the graph with vertex set $V(G)$ and empty edge-set.
- Let $j := 0$ and let A be the empty graph.
- While S is not 2-connected via A_j, do
  - Let C_{j+1} be a shortest cycle in H_j containing at least two active vertices.
  - Let $A_{j+1} := A_j \cup C_{j+1}$.
  - Obtain H_{j+1} from H_j by contracting together all vertices of C_{j+1}.
  - $j := j+1$.

Let k be the number of iterations. By the termination condition, S is 2-connected in A_k. For each j, let x_j be the number of active vertices in C_j.

Claim 23. $\sum_{v \in S} deg_{C_j}(v) \leq 2x_j$.

It follows that

$$\sum_{v \in S} \deg_A(v) = \sum_{j=1}^{K} \sum_{v \in S} \deg_{C_j}(v) \leq \sum_{j=1}^{K} 2x_j \tag{1}$$

Next we bound the right-hand side of (1).

Claim 24. *(No. of active vertices in H_{j+1}) = (no. of active vertices in H_j) $-(x_j - 1)$*

Since H_K contains 1 active vertex and H_0 contains $|S|$ active vertices, it follows by Claim 24 that

$$\sum_{j=1}^{K} (x_j - 1) = |S| - 1 \tag{2}$$

In each iteration $x_j - 1 \geq 1$, so the number of iterations K is at most $|S| - 1$. Hence $\sum(x_j - 1) \geq \sum x_j - (|S| - 1)$. Combining this inequality with (2) yields $\sum x_j \leq 2(|S| - 1)$. Combining this with (1) yields $\sum_{v \in S} \deg_A(v) \leq 4(|S| - 1)$. □

Lemma 25. *Let N be a locally optimal subgraph with degree Δ. Then there is an integer i in the range $[\Delta - \lceil \log_{1+\epsilon} n \rceil, \Delta]$ such that the deficiency of S_{i-2} is at least $|S_i|(i - 4(1 + \epsilon))$ and $|S_{i-2}| \leq (1 + \epsilon)|S_i|$.*

Proof. By an argument like that of Proposition 8, there is an i in the given range such that $|S_{i-2}| \leq (1+\epsilon)|S_i|$. By Lemma 22, there exists a 2-connected subgraph A of N such that S_{i-2} is contained in A and $\sum_{v \in S_{i-2}} \deg_A(v) \leq 4|S_{i-2}|$. By choice of i, the value $4|S_{i-2}|$ is at most $4(1 + \epsilon)|S_i|$.

Definition 26. We say an edge e of a 2-connected graph H is *critical in H* if $H - e$ is not 2-connected.

By local optimality, every edge e of N incident on S_i is critical in the degree-$(i - 3)$ candidate graph for e.

Claim 27. *Let e be an edge of N not in A such that one endpoint of e belongs to S_i. Then the other endpoint of e belongs to a leaf 2-component or isolated 2-component of bcf$(G - S_{i-2})$.*

Now we complete the proof of Lemma 25. The sum of the degrees of all the nodes of S_i in N is at least $i \cdot |S_i|$. By choice of the set A, we have $\sum_{v \in S_{i-2}} \deg_A(v) \leq 4(1 + \epsilon)|S_i|$. Hence, there are at least $|S_i|(i - 4(1 + \epsilon))$ edges incident on S_i not in A. By Claim 27, each of these edges goes to a leaf 2-component or isolated 2-component of bcf$(G - S_{i-2})$. By the criticality of these edges in N, it also follows that there is at most one such edge to a leaf-component and at most two such edges to a isolated-component in bcf$(G - S_{i-2})$. Hence each such edge contributes one to the deficiency of $G - S_{i-2}$. This concludes the proof of Lemma 25. □

4 Conclusions

Although we could only show a superpolynomial bound on the running times of our algorithms so far, we believe that our techniques can be used to obtain polynomial-time approximation algorithms with the same performance guarantees. This is the most important open problem resulting from this work.

References

1. A. Agrawal, P. Klein and R. Ravi, "When trees collide: an approximation algorithm for the generalized Steiner tree problem on networks," *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing* (1991), pp. 134-144.
2. A. Agrawal, P. Klein and R. Ravi, "How tough is the minimum-degree Steiner tree? An approximate min-max equality (complete with algorithms)", TR-CS-91-49, Brown University (1991), Submitted to SIAM J. on Disc. Math.
3. D. Bauer, S.L. Hakimi, and E. Schmeichel, "Recognizing tough graphs is NP-hard," *Disc. Appl. Math. 28* (1990), pp. 191-195.
4. V. Chvátal, "Tough graphs and Hamiltonian circuits," *Disc. Math. 5* (1973), pp. 215-228.
5. J. Edmonds, and E. L. Johnson, "Matching, Euler tours and the Chinese postman", *Math. Prog. 5*, (1973), pp. 88-124.
6. M. Fürer and B. Raghavachari, "An $\mathcal{NC}$ approximation algorithm for the minimum-degree spanning tree problem," Proceedings of the 28th Annual Allerton Conference on Communication, Control and Computing (1990), pp. 274-281.
7. M. Fürer and B. Raghavachari, "Approximating the minimum-degree spanning tree to within one from the optimal degree", *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms* (1992), pp. 317-324.
8. M. Fürer and B. Raghavachari, "Approximating the minimum-degree spanning and Steiner trees to within one from the optimal degree", TR CS-92-13, Pennsylvania State University, June 1992.
9. M. R. Garey and D. S. Johnson, *Computers and Intractability: A guide to the theory of NP-completeness*, W. H. Freeman, San Francisco (1979).
10. M. X. Goemans, and D. P. Williamson, "A general approximation technique for constrained forest problems", *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms* (1992), pp. 307-316.
11. P. Klein and R. Ravi, "Approximation through uncrossing: From edge-cuts to node-cuts," submitted to the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms.
12. G. L. Nemhauser, and L. A. Wolsey, *Integer and Combinatorial Optimization*, Wiley Interscience series in Disc. Math. and Optimization (1988).