RIMS Kôkyûroku Bessatsu
B23 (2010), 171–192

Iterative Rounding and Relaxation

By

Lap Chi LAU[*] and Mohit SINGH[**]

§ 1. Introduction

In this survey paper we present an iterative method to analyze linear programming formulations for combinatorial optimization problems. This method is introduced by Jain to give a 2-approximation algorithm for the survivable network design problem. First we will present Jain's method and the necessary background including the uncrossing technique in Section 2. Then we extend the iterative method by a new relaxation step to tackle degree-bounded network design problems, and obtain approximation algorithms with only additive constant errors on the degrees in Section 3. For the minimum bounded degree spanning tree problem, this gives a very simple approximation algorithm with error at most one on the degrees, proving a conjecture of Goemans in Section 4. This method can also be applied to directed graphs, and some recent results on the minimum bounded degree arborescence problem will be highlighted in Section 5. Finally, we discuss how this method provides new proofs of exact linear programming formulations for classical combinatorial optimization problems, and present some new results for the degree bounded matroid problem and the degree bounded submodular flow problem in Section 6.

Remark. Most of the material in this survey is extracted from a longer survey written with R. Ravi [9].

§ 2. Survivable Network Design

In this section we introduce the survivable network design problem, and Jain's iterative rounding method which gives a 2-approximation algorithm for the problem. The key of this method is a counting technique to analyze the basic solutions of a linear

Received September 15, 2008. Revised July 10, 2009.
2000 Mathematics Subject Classification(s): 90C27, 68W25
[*]The Chinese University of Hong Kong, Hong Kong, China.
[**]Microsoft Research, New England, United States.

program. This shows the use of the uncrossing technique in combinatorial optimization to the design of approximation algorithms.

Given an undirected graph $G = (V, E)$ and connectivity requirements r_{uv} for all pairs of vertices, a *Steiner network* is a subgraph of G in which there are at least r_{uv} edge-disjoint paths between u and v for every pair u, v. In the survivable network design problem, we are given an edge weighted graph $G = (V, E)$ and connectivity requirements r_{uv} for each pair $u, v \in V$, and the task is to find a Steiner network with minimum total weight. This problem generalizes a number of problems in network design; for example, the minimum Steiner tree problem, the minimum Steiner forest problem, and the minimum k-edge-connected subgraph problem. This basic problem has been studied by researchers in algorithmic design, computer networks, graph theory, and operations research.

§ 2.1. Linear Programming Relaxation

One general strategy to design approximation algorithm is to first formulate the problem as an integer linear program. Then we relax the integrality constraints and compute an optimal fractional solution of the linear programming relaxation. Finally we design a "rounding" procedure to turn the fractional solution into an integral solution with cost within a small factor of the fractional solution.

We consider the cut formulation of the survivable network design problem. For each subset $S \subset V$, let $\delta(S)$ be the set of edges with one endpoint in S and one endpoint in $V - S$. To satisfy the connectivity requirements, for each pair u, v so that $u \in S$ and $v \notin S$, a Steiner network must have at least r_{uv} edges in $\delta(S)$. Therefore, if we write $f(S) := \max_{u \in S, v \notin S}\{r_{uv}\}$, then any Steiner network must have at least $f(S)$ edges in $\delta(S)$ for all $S \subset V$. For a subset of edges F, we write $x(F)$ as a shorthand for $\sum_{e \in F} x(e)$. The following is a linear programming relaxation for the survivable network design problem.

$$\begin{array}{lll} \text{(LP1)} \quad \text{minimize} & \displaystyle\sum_{e \in E} c_e\, x_e & \\ \text{subject to} & x(\delta(S)) \geq f(S) & \forall\, S \subseteq V \\ & 0 \leq x_e \leq 1 & \forall\, e \in E \end{array}$$

It can be verified that the function f defined by $f(S) = max_{u \in S, v \notin S}\{r_{uv}\}$ for each subset $S \subseteq V$ is a *skew supermodular* function [6], that is, for any two subsets $S, T \subseteq V$, at least one of the following inequality holds:

$$f(S) + f(T) \leq f(S \cup T) + f(S \cap T)$$
$$f(S) + f(T) \leq f(S - T) + f(T - S)$$

This linear program has exponentially many constraints, but it can be solved in polynomial time by the ellipsoid method if there is a polynomial time separation oracle to decide whether a given solution is a feasible solution to the linear program. In general there is no known separation oracle for an arbitrary skew supermodular function, but for the skew supermodular functions that come from the survivable network design problem, one can use a maximum flow algorithm as a separation oracle [7]. Alternatively, one can write an equivalent linear program with polynomial number of constraints and variables for the linear program (LP1). In short, there is a polynomial time algorithm that computes an optimal basic solution to the linear program (LP1).

§ 2.2. Key Observation

Consider the instance that we are given a Peterson graph in which each edge has the same cost and the connectivity requirement is one for each pair of vertices. The fractional solution $x_e = \frac{1}{3}$ for all e is an optimal fractional solution to the linear program. More generally, consider an instance where we are given a k-regular k-edge-connected graph (e.g. a hypercube) in which each edge has the same cost and the connectivity requirement is one for each pair of vertices. The fractional solution $x_e = \frac{1}{k}$ for all e is an optimal solution. In such an fractional solution with all edges having the same value, it is not clear how to use the fractional solution to construct a good integral solution. Jain's observation is that these fractional solutions are not *basic solutions* of the linear program. In a basic solution of the Peterson graph instance, there are some edges with value $\frac{1}{2}$ and some edges with value $\frac{1}{4}$ (see [7] or [14]). This observation leads him to study basic solutions of (LP1) and prove the following key theorem.

Theorem 2.1 (Jain [7]). *For an integer-valued skew-supermodular function f, any basic feasible solution to the linear programming relaxation (LP1), there exists an edge $e \in E$ with $x_e \geq \frac{1}{2}$.*

§ 2.3. Iterative Algorithm

Using Theorem 2.1, Jain introduced an *iterative rounding* method to construct an integral solution for the survivable network design problem. The algorithm, as shown in Figure 1, recomputes a basic optimal solution after each edge is added to the solution.

Note that the function f' is a skew-supermodular function at every iteration, since f is a skew-supermodular function and $|\delta_F|$ is a submodular function. Therefore, by Theorem 2.1, there exists an edge e with $x_e \geq \frac{1}{2}$ at every iteration. This implies the following theorem that the iterative rounding algorithm is a 2-approximation algorithm for the survivable network design problem.

Theorem 2.2. *Algorithm 1 is a 2-approximation algorithm for the survivable network design problem.*

```
Iterative Rounding Algorithm for Survivable Network Design

1. Initialization $F \leftarrow \emptyset$, $f' \leftarrow f$;
2. While $f' \neq \emptyset$ do
    (a) Find a basic feasible solution $x$ with cut requirement $f'$ and remove every edge $e$ with $x_e = 0$.
    (b) If there exists an edge $e$ with $x_e \geq \frac{1}{2}$, then add $e$ to $F$ and remove $x_e$.
    (c) For every $S \subseteq V$: update $f'(S) \leftarrow f(S) - |\delta_F(S)|$.
3. Return $H = (V, F)$.
```

Figure 1. Iterative Rounding Algorithm for Survivable Network Design

Proof. The proof is by induction on the number of iterations executed by the algorithm. For the base case, that it requires only one iteration, the theorem follows since it rounds up an edge e with $x_e \geq \frac{1}{2}$. For the induction step, let e' be the edge with $x_{e'} \geq \frac{1}{2}$ in the current iteration, which is guaranteed to exist by Theorem 2.1. Let f' be the residual requirement function after the first iteration and let H' be the set of edges picked in subsequent iterations for satisfying f'. Observe that the current solution x restricted to $E - e'$ is a feasible solution for satisfying f', and thus by the induction hypothesis, the cost of H' is at most $2\sum_{e \in E - e'} c_e x_e$. Consider $H := H' + e'$, which clearly satisfies cut requirement f. The cost of H is:

$$\mathit{cost}(H) = \mathit{cost}(H') + c_{e'} \leq 2 \sum_{e \in E - e'} c_e x_e + c_{e'} \leq 2 \sum_{e \in E} c_e x_e,$$

where the last inequality follows because $x_{e'} \geq \frac{1}{2}$. This implies that the cost of H is at most twice the cost of an optimal fractional solution, which is a lower bound on the optimal cost, and thus the theorem follows. □

§ 2.4. Basic Solutions

To prove Theorem 2.1, we need a characterization of the basic solutions of the linear programming relaxation (LP1). A basic solution is defined to be the unique solution of m linearly independent *tight* constraints (constraints which achieve equality), where m denotes the number of variables in the linear program. For a subset $S \subseteq V$, the corresponding constraint $x(\delta(S)) \geq f(S)$ defines a vector in $\mathbb{R}^{|E|}$: the vector has an 1 corresponding to each edge $e \in \delta(S)$, and a 0 otherwise. We call this vector the characteristic vector of $\delta(S)$, and denote it by $\chi_{\delta(S)}$. Two sets X, Y are *intersecting* if

$X \cap Y$, $X - Y$ and $Y - X$ are nonempty. A family of sets is *laminar* if no two sets are intersecting. For any two intersecting subsets X and Y, since

$$x(\delta(X)) + x(\delta(Y)) \geq x(\delta(X \cap Y)) + x(\delta(X \cup Y)) \text{ and}$$

$$x(\delta(X)) + x(\delta(Y)) \geq x(\delta(X - Y)) + x(\delta(Y - X)),$$

and f is a skew supermodular function, it follows from standard uncrossing arguments (see e.g. [2, 7]) that a basic solution to the above linear program is characterized by a laminar family of tight constraints. The following lemma is proved in [7].

Lemma 2.3 ([7]). *Let the requirement function f of the linear programming relaxation (LP1) be skew supermodular, and let x be a basic solution to (LP1) such that $0 < x_e < 1$ for all edges $e \in E$. Then there exists a laminar family $\mathcal{L}$ such that:*

1. *$x(\delta(S)) = f(S)$ for $S \in \mathcal{L}$.*
2. *The characteristic vectors $\chi_{\delta(S)}$ for $S \in \mathcal{L}$ are linearly independent.*
3. *$|E| = |\mathcal{L}|$.*

§ 2.5. A Counting Argument

Instead of proving Theorem 2.1, we prove a weaker version that every basic solution has an edge with value at least $\frac{1}{3}$ (this weaker version is also proved in [7]). This proof is much simpler and contains the main ideas of the proof of Theorem 2.1.

The proof is by a counting argument. Suppose, by way of contradiction, that $0 < x_e < \frac{1}{3}$ for every edge e. Let x be the current basic solution. By Lemma 2.3, there is a laminar family $\mathcal{L}$ of tight constraints that defines x. We assign two tokens to each edge, one to each endpoint, for a total of $2|E|$ tokens. Then we will redistribute the tokens so that each member in $\mathcal{L}$ receives at least 2 tokens and there are some tokens left. This would imply that $|E| > |\mathcal{L}|$ and contradicts Lemma 2.3.

A laminar family $\mathcal{L}$ defines naturally a forest as follows: Each node of the forest corresponds to a set in $\mathcal{L}$, and there is an edge from set R to set S if R is the smallest set containing S. R is called the *parent* of S, and S is called the *child* of R. A node with no parent is called a *root*, and a node with no children is called a *leaf.* Given a node R, the *subtree rooted at R* consists of R and all its descendants.

We say an endpoint v is *owned* by a set S if S is the smallest set in $\mathcal{L}$ that contains v. Initially each vertex v gives its tokens to the set $S \in \mathcal{L}$ that owns v. The redistribution of tokens is by an inductive argument using the forest structure of the laminar family $\mathcal{L}$. We will prove the following lemma, which would yield the contradiction that $|E| > |\mathcal{L}|$.

Lemma 2.4 ([7]). *For any rooted subtree of the forest $\mathcal{L}$ with root S, the tokens assigned to vertices in S can be redistributed such that every node in the subtree gets at least two tokens, and the root S gets at least four tokens.*

Proof. The proof is by induction. In the base case, consider a leaf node S in the laminar family. Since $f(S) \geq 1$ and $x_e < \frac{1}{3}$ for all e, this implies that $|\delta(S)| \geq 4$ and thus S can collect four tokens. This verifies the base case.

For the induction step, consider a non-leaf node S. Note that by induction each child has at least two extra tokens. If S has at least two children, then S can collect four tokens by taking two extra tokens from each child. The only case left is when S has only one child R. Since $\chi_{\delta(S)}$ and $\chi_{\delta(R)}$ are linearly independent, S must owns at least one endpoint. As both $f(S)$ and $f(R)$ are integers and there is no edge of integral value, this actually implies that S cannot own exactly one endpoint, and thus S owns at least two endpoints. Therefore, S can collect four tokens by taking two extra tokens from R and two tokens from the endpoints that it owns. This completes the proof of the induction step. $\square$

Lemma 2.4 implies that there are extra tokens at the roots of the laminar family, and thus $|E| > |\mathcal{L}|$, contradicting that x is a basic solution. Therefore, in a basic solution, there exists an edge with value at least $\frac{1}{3}$, completing the proof of the weaker version of Theorem 2.1.

To prove Theorem 2.1, one requires a more careful counting argument; the interested reader is referred to [7] or [14] for details. Also, we mention that Nagarajan, Ravi and Singh (see [9]) have a simple proof of Theorem 2.1 using the idea of *fractional* token assignment by Bansal, Khandekar and Nagarajan in [1]. We will also see this fractional token idea in Section 4 to give a simple proof of an iterative relaxation algorithm for the minimum bounded degree spanning tree problem.

§ 3. Degree Bounded Network Design

In this section we introduce the minimum bounded-degree Steiner network problem, and see how to extend Jain's method to tackle this problem. The key is a new relaxation step in the iterative method.

In the minimum bounded degree Steiner network problem, we are given an undirected graph $G = (V, E)$, a cost function $c : E \to \mathbb{R}$, a connectivity requirement function $r : V \times V \to \mathbb{Z}$, and a degree upper bound B_v for each vertex $v \in V$. The task is to find a Steiner network H of G with minimum total cost satisfying the additional constraints that $d_H(v) \leq B_v$ for all $v \in G$.

Note that finding a feasible solution to this problem is already NP-hard, as the Hamiltonian path problem is a special case. Therefore, the minimum bounded degree

Steiner network problem has two optimization objectives: to minimize the total cost and to minimize the degree violation. The goal is to design approximation algorithms that optimize both objectives simultaneously. Let OPT be the optimal cost of a solution satisfying all the degree constraints. We say an algorithm is an $(\alpha, f(B_v))$-*bicriteria* approximation algorithm if the returned solution has cost at most $\alpha \cdot \text{OPT}$ and the degree of each vertex v is at most $f(B_v)$.

We will first present a constant factor bicriteria approximation algorithm in Section 3.2, and then mention how to improve it to obtain additive approximation guarantee on the degree violation in Section 3.3. Note that the latter result implies additive approximation algorithms for the minimum maximum-degree Steiner network problem, where the goal is to find a Steiner network with minimum maximum-degree.

§ 3.1. Previous Work

A simpler problem is the minimum maximum-degree Steiner network problem, where the goal is to find a Steiner network with minimum maximum degree. For the minimum maximum-degree spanning tree problem, Fürer and Raghavachari [4] gave an approximation algorithm returning a solution with maximum degree at most one more than the optimal solution. (Their result holds for Steiner trees as well.) This result has generated much interest to degree-bounded network design problems.

Goemans conjectured that there is a $(1, B_v + 1)$-bicriteria approximation algorithm for the minimum bounded degree spanning tree problem. This problem has been studied by several group of researchers, and Goemans made a breakthrough by giving a $(1, B_v + 2)$-bicriteria approximation algorithm [5]. Only some partial results are known for more general connectivity requirements. The interested reader is referred to [5, 12] for previous work on the minimum bounded degree spanning tree problem, and [10, 11] for previous work on the minimum bounded degree Steiner network problem.

§ 3.2. A Constant Factor Approximation Algorithm

The iterative rounding method is extended to prove the following result.

Theorem 3.1 ([10]). *There is a polynomial time $(2, 2B_v + 3)$-approximation algorithm for the minimum bounded degree Steiner network problem.*

As in the survivable network design problem, we define $f(S) := \max_{u \notin S, v \in S}\{r_{u,v}\}$ for every subset $S \subset V$, which is a skew supermodular function. The linear programming relaxation is almost the same as (LP1), with the addition of degree constraints on a subset $W \subseteq V$ of vertices.

$$\begin{array}{llll}
\text{(LP2)} & \text{minimize} & \displaystyle\sum_{e\in E} c_e\, x_e & \\
& \text{subject to} & x(\delta(S)) \geq f(S), & \forall\, S \subseteq V \\
& & x(\delta(v)) \leq B_v, & \forall\, v \in W \\
& & 0 \leq x_e \leq 1 & \forall\, e \in E
\end{array}$$

Note that degree constraints are defined only on single vertices, and so the uncrossing technique as in [7, 5] can be applied to show that a basic optimal solution is characterized by a laminar family of tight constraints (see Lemma 2.3). This immediately implies that, in the first iteration, there exists an edge having value at least $\frac{1}{2}$. Now comes the main difference. Since degree constraints are *packing* constraints, after we have picked some fractional edges in the previous iterations, we need to allow for *non-integral* degree constraints in the residual problem, otherwise the residual problem may be infeasible, or its cost may be significantly higher. By doing so, however, it is not necessarily true that the picked edges in later iterations have value at least $\frac{1}{2}$, as the proof of Theorem 2.1 uses the fact that the function f is integer-valued.

The idea of *iterative relaxation* is introduced in [10] to overcome this difficulty. When there is no edge of value at least $\frac{1}{2}$ in a basic optimal solution, it is proved in [10] that there is a vertex v with degree constraint and it has degree at most 4. The new step is to "relax" the problem by removing the degree constraint on v. After that, a basic optimal solution is recomputed for the residual problem, and this procedure is iterated. So, in each iteration, either an edge of value at least $\frac{1}{2}$ is rounded up or the problem is relaxed by removing the degree constraint of a vertex of degree at most 4. Note that the relaxation step only incurs an extra additive constant 3 in the degree violation. This implies a $(2, 2B_v + 3)$-approximation algorithm for the problem [10], which is described formally in the following.

Similar to the proof of Lemma 2.3, the following lemma about the basic solutions of (LP2) is needed to prove Theorem 3.1.

Lemma 3.2 ([10]). *Let the requirement function f of (LP2) be skew supermodular, and let x be a basic solution of (LP2) such that $0 < x_e < 1$ for all edges $e \in E$. Then, there exists a laminar family $\mathcal{L}$ of tight inequalities, where $\mathcal{L}$ partitions into a set of singletons $\mathcal{L}'$ for the degree constraints, and the remaining sets $\mathcal{L}'' = \mathcal{L} - \mathcal{L}'$ for the connectivity constraints such that:*

1. *$x(\delta(v)) = B_v$ for each $v \in \mathcal{L}'$ and $x(\delta(S)) = f(S)$ for each $S \in \mathcal{L}''$.*

2. *$|\mathcal{L}| = |E|$.*

```
Iterative Algorithm for Minimum Bounded Degree Steiner Network

1. Initialization $F \leftarrow \emptyset$, $f' \leftarrow f$, and $\forall v \in W$: $B'_v = B_v$;
2. While $f' \neq \emptyset$ do
    (a) Find a basic optimal solution $x$ with cut requirement $f'$ and remove every edge $e$ with $x_e = 0$.
    (b) If there exists a vertex $v \in W$ with degree at most 4, remove $v$ from $W$ and goto (a).
    (c) If there exists an edge $e = (u, v)$ with $x_e \geq \frac{1}{2}$, then add $e$ to $F$ and remove $x_e$ and decrease $B'_u$ and $B'_v$ by $\frac{1}{2}$.
    (d) For every $S \subseteq V$: $f'(S) \leftarrow f(S) - |\delta_F(S)|$.
3. Return $H = (V, F)$.
```

Figure 2. Iterative Algorithm for Minimum Bounded Degree Steiner Network

3. The characteristic vectors $\chi_{\delta(S)}$ for $S \in \mathcal{L}$ are linearly independent.

Then a counting argument similar to Jain's counting argument is used to prove that the algorithm in Figure 2 always terminates successfully.

Lemma 3.3 ([10]). *Let x be a basic solution of (LP2), and W be the set of vertices with degree constraints. Then either one of the following is true:*

1. There exists an edge with value at least $\frac{1}{2}$.

2. There exists a vertex $v \in W$ such that $deg(v) \leq 4$.

We prove the weaker version of the lemma by replacing $\frac{1}{2}$ in Lemma 3.3(1) by $\frac{1}{3}$, whose proof is much simpler and contains the main ideas. Suppose, by way of contradiction, that every edge e with $0 < x_e < \frac{1}{3}$ and each vertex $v \in W$ has $deg(v) \geq 5$. We use a counting argument to prove that $|E| > |\mathcal{L}|$. Each edge is assigned two tokens, for a total of $2|E|$ tokens. For each edge e, one token is assigned to each endpoint. We show that the tokens can be redistributed in such a way that each set in $\mathcal{L}$ can collect two tokens, and there are some tokens left. This would imply $|E| > |\mathcal{L}|$, contradicting that x is a basic solution. Initially each vertex v gives its tokens to the set $S \in \mathcal{L}$ that owns v. The redistribution of tokens is by an inductive argument using the forest structure of the laminar family $\mathcal{L}$.

Lemma 3.4. *For any rooted subtree of the forest $\mathcal{L}$ with root S, the tokens assigned to vertices in S can be redistributed such that every member in the subtree gets at least two tokens, and the root S gets at least four tokens.*

Proof. Since each vertex with degree constraint has degree at least five, each degree constraint has at least three extra tokens. For a leaf node S in $\mathcal{L}$ which is not a degree constraint, since $f(S) \geq 1$ and there is no edge with $x_e \geq \frac{1}{3}$, $|\delta(S)| \geq 4$ and so S can collect four tokens. This verifies the base case.

For the induction step, consider a non-leaf node S. Note that by induction each child has at least two extra tokens. If S has at least two children, then S can collect four tokens by taking two extra tokens from each child. The only case left is when S has only one child R. Since $\chi_{\delta(S)}$ and $\chi_{\delta(R)}$ are linearly independent, S must owns at least one endpoint. If R is a degree constraint, then S can collect four tokens by taking one token from the endpoint it owns and three tokens from R. Otherwise, R is a connectivity constraint. Since both $f(S)$ and $f(R)$ are integers and there is no edge of integral value, this implies that S cannot own exactly one endpoint, and thus S owns at least two endpoints. Therefore, S can collect four tokens by taking two extra tokens from R and two tokens from the endpoints that it owns. This completes the proof of the induction step. □

Therefore, by Lemma 3.4, there are extra tokens left in the roots of the laminar family, and this gives us the contradiction that $|E| > |\mathcal{L}|$. This completes the proof of the weaker version of Lemma 3.3. The proof of Lemma 3.3 is by a more careful counting argument, which we refer to the reader to [10] for details. Theorem 3.1 follows immediately from Lemma 3.3.

§ 3.3. An Additive Approximation Algorithm

Motivated by the results on the minimum bounded degree spanning tree problem, it is natural to ask whether there is a bicriteria approximation algorithm with degree violation bounded by an additive constant. However, the following example shows that the integrality gap for the worst case degree violation is at least a multiplicative $\frac{3}{2}$ or an additive $\frac{n}{4}$.

We note that in the integrality gap example there is a pair of vertices with high connectivity requirement. In the minimum bounded degree Steiner tree problem, the maximum connectivity requirement is one, and there is an approximation algorithm by Fürer and Raghavachari with degree violation at most one in the unweighted case [4]. This leads to the question whether there is an additive approximation algorithm when the maximum connectivity requirement is small, and the following result provides a positive answer.

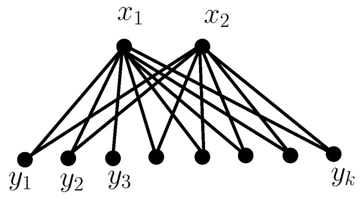

Figure 3. In this example, we have a complete bipartite graph $B = (X, Y, E)$ where $X = \{x_1, x_2\}$ and $Y = \{y_1, \ldots, y_n\}$. We set the connectivity requirements between y_i and y_j to be 1 for all i, j, between x_1 and x_2 to be $\frac{n}{2}$, and 0 otherwise. The fractional solution where all edges have fractional value $\frac{1}{2}$ is the optimal solution, in which the degree of x_1 and x_2 is equal to $\frac{n}{2} = \Delta_f^*$. On the other hand, in any integer solution, the degree of x_1 and x_2 must be at least $\frac{3}{4}n = \frac{3}{2}\Delta_f^*$. This example also shows that the integrality gap is at least an additive $\frac{n}{4}$.

Theorem 3.5 ([11]). *There is a polynomial time $(2, B_v{+}6r_{\max}{+}3)$-approximation algorithm for the minimum bounded degree Steiner network problem, where $r_{\max} = \max_{u,v}\{r_{u,v}\}$ is the maximum connectivity requirement.*

In particular, for the minimum bounded degree Steiner forest problem, when the maximum connectivity requirement is one, Theorem 3.5 gives a bicriteria approximation algorithm with degree violation bounded by an additive constant. Similar results hold for the minimum bounded degree k-edge-connected subgraph problem when k is a constant.

To achieve additive approximation on the degree bounds, the algorithm needs to avoid picking many edges with value $\frac{1}{2}$ incident on the same vertex. Interestingly, it turns out that in any basic solution there is an edge with $x_e \geq \frac{1}{2}$ between two "low degree" vertices.

Lemma 3.6 ([11]). *Let x be a basic feasible solution of (LP), W be the set of vertices with degree constraints, and $W_h = \{v \in W \mid \sum_{e\in\delta(v)} x_e \geq 6r_{max}\}$. Then at least one of the following must be true.*

1. *There exists an edge e with $x_e = 1$.*
2. *There exists an edge $e = \{u, v\}$ with $x_e \geq 1/2$ and $u, v \notin W_h$.*
3. *There exists a vertex $v \in W$ such that $deg_E(v) \leq 4$.*

The counting argument of Lemma 3.6 is more involved; we refer the reader to [11] for its proof. This lemma leads to the algorithm in Figure 4 that only picks edges with $x_e \geq \frac{1}{2}$ when both endpoints have low degrees.

It is easy to see that the cost of the solution is at most twice the cost of an optimal fractional solution, since only edges with value at least $\frac{1}{2}$ are picked. We argue that

```
Additive Approximation for Minimum Bounded Degree Steiner Network
  1. Initialization $F \leftarrow \emptyset$, $f'(S) \leftarrow f(S)\ \forall S \subseteq V$.
  2. While $f' \neq 0$ do
     (a) Find a basic optimal solution $x$ satisfying cut requirement $f'$ and remove every edge $e$ with $x_e = 0$. Set $W_h = \{v \in W | \sum_{e\in\delta(v)} x_e \geq 6r_{max}\}$ and $B_v = \sum_{e\in\delta(v)} x_e$ for $v \in W$.
     (b) For every $v \in W$ with degree at most 4, remove $v$ from $W$ and goto (a).
     (c) For each edge $e = (u, v)$ with $x_e = 1$, add $e$ to $F$ and remove $x_e$ and decrease $B_u$ and $B_v$ by 1.
     (d) For each edge $e = (u, v)$ with $x_e \geq 1/2$ and $u, v \notin W_h$, add $e$ to $F$ and remove $x_e$ and and decrease $B_u$ and $B_v$ by 1/2.
     (e) For every $S \subseteq V$: $f'(S) \leftarrow f(S) - |\delta_F(S)|$.
  3. Return $H = (V, F)$.
```

Figure 4. Additive approximation for minimum bounded degree Steiner network.

the degree of any vertex v in solution H is at most $B_v + 6r_{max} + 3$. In Step 2(a), we define the set W_h of vertices with fractional degree at least $6r_{\max}$ as “high” degree vertices. Consider an edge e with v as an endpoint. By Step 2(c) of the algorithm, when $v \in W_h$, e is picked only if $x_e = 1$. Hence, while $v \in W_h$, at most $B_v - 6r_{max}$ edges incident at v are added to H. By Step 2(d) of the algorithm, while $v \in W \setminus W_h$, e is picked only if $x_e \geq \frac{1}{2}$. Hence, while $v \in W \setminus W_h$, strictly less than $12r_{max}$ edges incident at v are added to H. Finally, by Step 2(b) of the algorithm, $v \notin W$ only if v is incident to at most four edges. Therefore, the degree of v in H is strictly less than $(B_v - 6f_{max}) + 12f_{max} + 4 = B_v + 6f_{max} + 4$. As B_v is an integer, the degree of v in H is at most $B_v + 6f_{max} + 3$. This proves Theorem 3.5. A stronger result can be proved for the special cases of Steiner trees and Steiner forests.

Theorem 3.7 ([11]). *There is a polynomial time $(2, B_v + 3)$-bicriteria approximation algorithm for the minimum bounded degree Steiner forest problem.*

§ 4. Degree Bounded Spanning Trees

In this section we consider the minimum bounded degree spanning tree problem, and prove Goemans conjecture by a simple iterative relaxation algorithm.

Theorem 4.1 ([12]). *There is a polynomial time $(1, B_v + 1)$-bicriteria approximation algorithm for the minimum bounded degree spanning tree problem.*

This result is first proved in [12], and the analysis is subsequently simplified in [1]. Here we use the idea in [1] to present an even simpler algorithm for Theorem 4.1.

§ 4.1. Linear Programming Relaxation

The following linear programming relaxation is used for the minimum bounded degree spanning tree problem [5, 12], where the degree bounds are given for vertices in $W \subseteq V$. In the following $E(S)$ denotes the set of edges with both endpoints in S.

$$\begin{array}{llll}
\text{(LP3)} & \text{minimize} & \sum_{e\in E} c_e\, x_e & \\
& \text{subject to} & x(E(V)) = |V| - 1 & \\
& & x(E(S)) \le |S| - 1 & \forall\, S \subset V \\
& & x(\delta(v)) \le B_v & \forall\, v \in W \\
& & x_e \ge 0 & \forall\, e \in E
\end{array}$$

Although this linear program has exponentially many constraints, there is a polynomial time separation oracle to decide whether a given fractional solution is feasible, and thus it can be solved in polynomial time by the ellipsoid method.

§ 4.2. Iterative Relaxation Algorithm

The following algorithm in Figure 5 removes degree constraints one by one, and eventually reduces the problem to a minimum spanning tree problem. Note that there is no rounding step in this algorithm.

Iterative Relaxation for Minimum Bounded Degree Spanning Tree

```
1. While $W \neq \emptyset$ do
    (a) Find a basic optimal solution $x$ of (LP3) and remove every edge $e$ with $x_e = 0$.
    (b) If there exists a vertex $v \in W$ with $deg(v) \le B_v + 1$ then remove $v$ from $W$.
2. Return a basic solution $x$ of (LP3).
```

Figure 5. Iterative Relaxation Algorithm for Minimum Bounded Degree Spanning Tree

In the next subsection we prove that in each iteration the algorithm can always find some vertex to remove the degree constraint. Once all the degree constraints are removed, the problem reduces to the minimum spanning tree problem, and hence a basic solution is integral. Since we only relax the linear program at each step, the cost of the final solution is at most the cost of the initial solution, and thus the tree returned by the algorithm has optimal cost. A simple inductive argument also shows that the degree bound is violated by at most an additive one.

§ 4.3. A Counting Argument

Using standard uncrossing technique, one can obtain the following characterization of the basic solutions of (LP3).

Lemma 4.2 ([5, 12]). *Let x be a basic solution of (LP3) with $x_e > 0$ for each edge $e \in E$. Then there exists a set $T \subseteq W$ and a laminar family $\mathcal{L}$ such that*

1. *$x(\delta(v)) = B_v$ for each $v \in T$ and $x(E(S)) = |S| - 1$ for each $S \in \mathcal{L}$.*
2. *The characteristic vectors $\chi_{E(S)}$ for $S \in \mathcal{L}$ and the characteristic vectors $\chi_{\delta(v)}$ for $v \in T$ are linearly independent.*
3. *$|\mathcal{L}| + |T| = |E|$.*

To prove Theorem 4.1, it remains to prove that the iterative relaxation algorithm can always find a degree constraint to remove at each step. The proof of the following lemma uses the fractional token idea by Bansal, Khandekar and Nagarajan [1].

Lemma 4.3. *If $W \neq \emptyset$, then in any basic solution to (LP3) with $x_e > 0$ for all $e \in E$, there exists some vertex $v \in W$ with $deg(v) \leq B_v + 1$.*

Proof. Suppose, by way of contradiction, that $W \neq \emptyset$ and $deg(v) \geq B_v + 2$ for each $v \in W$. We show by a counting argument that $|E| > |T| + |\mathcal{L}|$, contradicting Lemma 4.2. We give one token for each edge in E. We then redistribute the token such that each vertex in T and each set in $\mathcal{L}$ gets one token and there are some extra tokens left. This will contradict $|E| = |T| + |\mathcal{L}|$. The token redistribution is as follows. Each edge $e \in E$ gives $(1 - x_e)/2$ to each of its endpoints for the degree constraints and x_e token to the smallest set in $\mathcal{L}$ containing both endpoints of e.

We now show that each vertex with a degree constraint gets one token. Let $v \in W$ be such a vertex. Then v receives $(1 - x_e)/2$ tokens for each edge incident at v for a total of

$$\sum_{e \in \delta(v)} \frac{1 - x_e}{2} \geq \frac{deg(v) - B_v}{2} \geq 1$$

token, where the first inequality holds since $\sum_{e \in \delta(v)} x_e \leq B_v$ and the second inequality holds since $deg(v) \geq B_v + 2$ by the relaxation step of the algorithm.

Now we show that each member $S \in \mathcal{L}$ also obtains one token. By the token redistribution rule, S receives x_e token for each edge e such that S is the smallest set containing both endpoints of e. Let $R_1, \ldots, R_k$ be the children of S in the laminar family $\mathcal{L}$ where $k \geq 0$. We have

$$\begin{aligned} x(E(S)) &= |S| - 1 \\ x(E(R_i)) &= |R_i| - 1 \text{ for each } 1 \leq i \leq k \\ \implies x(E(S)) - \sum_{i=1}^{k} x(E(R_i)) &= |S| - 1 - \sum_{i=1}^{k}(|R_i| - 1) \\ \implies x(A) &= |S| - 1 - \sum_{i=1}^{k}(|R_i| - 1), \end{aligned}$$

where $A = E(S) \setminus (\cup_{i=1}^{k} E(R_i))$. By the token redistribution rule, S receives exactly $x(A)$ tokens, which is an integer by the above equation. Note that $x(A) \neq 0$; otherwise $\chi_{E(S)} = \sum_{i=1}^{k} \chi_{E(R_i)}$ which contradicts the linear independence of the characteristic vectors in $\mathcal{L}$. Hence each set also receives at least one token.

It remains to show that there is some extra token left for contradiction. If $V \notin \mathcal{L}$ then there exists an edge e which is not contained in any set of $\mathcal{L}$ and the x_e token for that edge gives us some extra tokens. Similarly, if there is a vertex $v \in W - T$ then v collects one extra token. Moreover, if there is an edge e with $x_e < 1$ incident on a vertex $v \in V - T$, then there are $(1 - x_e)/2 > 0$ extra tokens on v. Note that $e \in \mathit{span}(\mathcal{L})$ for each e with $x_e = 1$, since e is a tight set of size two. We have

$$2\chi_{E(V)} = \sum_{v \in V} \chi_{\delta(v)} = \sum_{v \in T} \chi_{\delta(v)} + \sum_{v \in V-T} \chi_{\delta(v)} = \sum_{v \in T} \chi_{\delta(v)} + \sum_{v \in V-T} \sum_{e \in \delta(v)} \chi_e.$$

We have argued that $V \in \mathcal{L}$ and $e \in \mathit{span}(\mathcal{L})$ for each edge $e \in \delta(v)$ for $v \in V - T$. Since $T = W \neq \emptyset$, this implies the linear dependence of the tight constraints in T and those in $\mathcal{L}$, giving us the contradiction. $\square$

§ 5. Degree Bounded Arborescences

There is a natural analog of the minimum bounded degree spanning tree problem in directed graphs - the minimum bounded degree arborescence problem. In this problem, we are given an edge weighted directed graph and an out-degree bound B_v for each vertex v, and the task is to find an arborescence with minimum total cost satisfying all the out-degree bounds. A $(2, 2B_v + 2)$-bicriteria approximation algorithm is obtained in [10] for the minimum bounded degree arborescence problem using a similar approach as in Section 3. It is a very natural question whether there is a $(1, B_v + 1)$-bicriteria approximation algorithm for the minimum bounded degree arborescence problem. Bansal, Khandekar and Nagarajan [1] proved some surprising results for this problem. On one hand, they give an additive approximation algorithm for the minimum maximum out-degree arborescence problem, where the goal is to find an arborescence with minimum maximum out-degree.

Theorem 5.1 ([1]). *There is an approximation algorithm with error at most an additive constant* 2 *for the minimum maximum out-degree arborescence problem.*

On the other hand, they show that for the linear programming relaxation (LP4), there is a cost-degree tradeoff for approximating the minimum bounded degree arborescence problem.

Theorem 5.2 ([1]). *For (LP4), for any* $0 < \epsilon < 1$*, there are instances in which any arborescence with* $|\delta^{out}(v)| \leq \frac{B_v}{1-\epsilon} + O(1)$ *for all* v *has cost at least* $(\frac{1-o(1)}{\epsilon})$ *times the optimal fractional cost.*

This shows that, unlike the results in undirected graphs, one cannot simultaneously minimize both the cost and the degree violation using the natural linear programming relaxation for the problem. In the following we present the proof of Theorem 5.1, in which the idea of fractional token is first used.

§ 5.1. Linear Programming Relaxation

For the purpose of iterative relaxation, the problem is defined in a more general setting where the connectivity requirement is defined by a 0-1 *intersecting supermodular* function f, where $f(S) = \{0,1\}$ and for two intersecting subsets S and T,

$$f(S) + f(T) \leq f(S \cap T) + f(S \cup T).$$

In the following the out-degree constraints are defined on a subset $W \subseteq V$.

$$\begin{array}{lll} \text{(LP4)} & \text{minimize} & \displaystyle\sum_{e \in E} c_e x_e \\ \text{subject to} & x(\delta^{in}(S)) \geq f(S) & \forall\, S \subseteq V - r \\ & x(\delta^{out}(v)) \leq B_v & \forall\, v \in W \\ & 0 \leq x_e \leq 1 & \forall\, e \in E \end{array}$$

Although this linear program has exponentially many constraints, it can be solved by the ellipsoid method using a minimum cut algorithm as a separation oracle.

§ 5.2. Iterative Relaxation Algorithm

For the minimum maximum out-degree arborescence problem, the iterative algorithm in Figure 6 is based on rounding a basic solution of (LP4) where there is no objective function.

```
Iterative Relaxation for Minimum Maximum Out-Degree Arborescence

1. Initialization $F \leftarrow \emptyset$.
2. While $W \neq \emptyset$ do
   (a) Find a basic solution $x$ of (LP4) where there is no objective function, and
       remove every edge $e$ with $x_e = 0$.
   (b) If there is a vertex $v \in W$ with $|\delta^{out}(v)| \leq B_v + 2$, then add all the edges in
       $\delta^{out}(v)$ to $F$ and remove all the edges in $\delta^{out}(v)$ and remove $v$ from $W$.
3. Return any arborescence in $F$.
```

Figure 6. Iterative Relaxation for Minimum Maximum Out-Degree Arborescence

The degree constraint is violated only in Step 2(b) by at most two. So if the algorithm terminates successfully, then the algorithm is an additive approximation algorithm for the minimum maximum out-degree arborescence problem.

Remark. In Step 2(b) of the algorithm, some edges with very small fractional values may be added, and thus there is no guarantee on the total cost of the returned arborescence if we are also given costs on the edges.

Remark. The algorithm in Figure 6 is a slightly simplified version of the algorithm in [1], in which there is a step of picking an edge of value one. The algorithm in Figure 6 is very similar to the algorithm in Figure 5 for the minimum bounded degree spanning tree problem.

§ 5.3. A Counting Argument

Using standard uncrossing argument, one can obtain the following characterization of the basic solutions of (LP4).

Lemma 5.3 ([10, 1]). *Let x be any basic solution of the linear programming relaxation (LP4). Then there exists a set $T \subseteq W$ and a laminar family $\mathcal{L}$ such that*

1. $x(\delta^{out}(v)) = B_v$ for each $v \in T$ and $x(\delta^{in}(S)) = f(S)$ for each $S \in \mathcal{L}$.

2. *The characteristic vectors* $\{\chi_{\delta^{in}(S)} : S \in \mathcal{L}\} \cup \{\chi_{\delta^{out}(v)} : v \in T\}$ *are linearly independent.*

3. $|E| = |\mathcal{L}| + |T|$.

We are ready to prove that the algorithm in Figure 6 can always terminate successfully, which will then complete the proof of Theorem 5.1.

Lemma 5.4. *If* $W \neq \emptyset$*, then in any basic solution* x *of (LP4) with* $x_e > 0$ *for all* e*, there exists a vertex* v *with out-degree at most* $B_v + 2$*.*

Proof. Suppose, by way of contradiction, that $x_e > 0$ for all e and $|\delta^{out}(v)| \geq B_v + 3$ for each $v \in W$. Each edge is assigned one token, for a total of $|E|$ tokens. For each edge e, $1 - x_e$ token is assigned to its tail, and x_e token is assigned to its head. We will redistribute the tokens so that each set in $\mathcal{L}$ and each degree constraint in T can collect one token, and there are some tokens left. This would imply that $|E| > |\mathcal{L}| + |T|$, which contradicts that x is a basic solution.

For each vertex v with nonzero out-degree, it collects

$$\sum_{e \in \delta^{out}(v)} (1 - x_e) = |\delta^{out}(v)| - \sum_{e \in \delta^{out}(v)} x_e \geq |\delta^{out}(v)| - B_v \geq 3$$

tokens; the first inequality follows from the constraints in (LP4), and the last inequality follows because of the relaxation step in the algorithm. This shows that each vertex with nonzero out-degree can collect at least three tokens, and thus has at least two extra tokens.

For a leaf node $S \in \mathcal{L}$, it collects $\sum_{e \in \delta^{in}(v)} x_e = 1$ token. Furthermore S has at least one extra token if $|\delta^{out}(S)| \geq 1$. We call S with $|\delta^{out}(S)| = 0$ a *sink* node. Hence each non-sink node has at least one extra token. We prove inductively that each sink node has at least one token and each non-sink node has at least two tokens, which holds in the base case when S is a leaf node. Consider a non-leaf node $S \in \mathcal{L}$, and let its children be $R_1, \ldots, R_l$. If S has at least two non-sink children, then S can collect one extra token from each non-sink child by the induction hypothesis, and hence S has at least one extra token, as required. So assume S has at most one non-sink child R_1. Since $x(\delta^{in}(S)) = x(\delta^{in}(R_1)) = 1$ and $\chi_{\delta^{in}(S)} \neq \chi_{\delta^{in}(R_1)}$, there is an edge $f \in \delta^{in}(R_1) - \delta^{in}(S)$. Since other children of S are sink nodes, the tail of f is contained in $S - (R_1 \cup R_2 \cup \ldots \cup R_l)$, and hence can contribute two tokens to S, as required. Therefore, by an inductive argument, there are extra tokens left at the roots of the laminar family. This completes the proof. □

§ 6. Degree Bounded Matroids and Submodular Flows

One can also use an iterative method to prove exact linear programming formulations for classical combinatorial optimization problems. To use this method, it is enough to prove that there is a variable with value one using a counting argument. This approach can be used to give new proofs of exact linear programming formulations for classical combinatorial optimization problems [9], including spanning trees, arborescences, maximum matchings in general graphs, matroid intersections, submodular flows, etc. These new proofs can be used to obtain new results in approximation algorithms. In this section we mention some results in degree bounded matroids and submodular flows.

§ 6.1. Degree Bounded Matroids

The minimum bounded degree matroid basis problem is a generalization of the minimum bounded degree spanning tree problem. In this problem, we are given a matroid $M = (V, \mathcal{I})$, a cost function c on the ground set V, a hypergraph $H = (V, E)$, and an upper bound $g(e)$ for each hyperedge $e \in E(H)$. The task is to find a basis B of minimum cost such that $|B \cap e| \leq g(e)$ for each hyperedge $e \in E(H)$. The following result can be obtained by extending the proof technique in Section 4.

Theorem 6.1 ([8]). *There is a polynomial time algorithm for the minimum bounded degree matroid basis problem which returns a basis B of cost at most* OPT *such that $|B \cap e| \leq g(e) + \Delta - 1$ for each $e \in E(H)$, where $\Delta = \max_{v \in V} |\{e \in E(H) : v \in e\}|$ is the maximum degree of the hypergraph H and* OPT *is the cost of an optimal solution which satisfies all the degree constraints.*

This result has application in the minimum crossing spanning tree problem, in which we are given a graph $G = (V, E)$ with edge cost function c, a collection of cuts (edge subsets) $\mathcal{C} = \{C_1, \ldots, C_m\}$ and an upper bound g_i for each cut C_i. The task is to find a tree T of minimum cost such that T contains at most g_i edges from cut C_i. The minimum bounded degree spanning tree problem is the special case where $\mathcal{C} = \{\delta(v) : v \in V\}$. The following result can be obtained as a corollary of Theorem 6.1. Note that $d = 2$ for the minimum bounded degree spanning tree problem, and so the following result generalizes Theorem 4.1.

Corollary 6.2 ([1]). *There is a polynomial time algorithm for the* Minimum Crossing Spanning Tree *problem that returns a tree T with cost at most* OPT *and such that T contains at most $g_i + d - 1$ edges from cut C_i for each i where $d = \max_{e \in E} |\{C_i : e \in C_i\}|$, where* OPT *is the cost of an optimal solution which satisfies all the cut constraints.*

Theorem 6.1 can also be applied to the minimum bounded degree spanning tree union problem; see [8] for details.

§ 6.2. Degree Bounded Submodular Flows

The minimum bounded degree submodular flow problem is a generalization of the submodular flow problem [3, 13]. In this problem we are given a digraph $D = (V, E)$, a crossing submodular set function $b : 2^V \to \mathbb{Z} \cup \{+\infty\}$, a subset of vertices $W \subseteq V$, and a function $g : W \to \mathbb{Z}_+$. A *degree-constrained 0-1 submodular flow* is a vector $x \in E \to \{0, 1\}$ with the following properties:

$$\begin{aligned} x(\delta^{in}(X)) - x(\delta^{out}(X)) &\leq b(X) && \text{for every } X \subseteq V, \\ x(\delta(v)) &\leq g(v) && \text{for every } v \in W. \end{aligned}$$

If $W = \emptyset$, then this is the well-studied submodular flow problem [3]. There are several efficient algorithms for finding a feasible submodular flow, or even a minimum cost submodular flow for a linear cost function. However, the addition of the degree constraints makes the feasibility problem NP-complete [8]. The following result can be obtained using an iterative relaxation algorithm.

Theorem 6.3. *[8] There is a polynomial time algorithm for the minimum bounded degree submodular flow problem which returns an* 0-1 *submodular flow of cost at most* OPT *that violates each degree constraint by at most one, where* OPT *is the cost of an optimal solution which satisfies all the degree constraints.*

Theorem 6.3 can be applied to the minimum bounded degree graph orientation problem, in which we are given a digraph $D = (V, E)$, a cost function $c : E \to \mathbb{Z}$, and a degree bound $g(v)$ for every $v \in V$. The task is to find an edge set of minimum cost whose reversal makes the digraph k-edge-connected, so that the number of edges reversed at each node v is at most $g(v)$. As graph orientation problems (with crossing supermodular requirements) can be reduced to the submodular flow problem, Theorem 6.3 implies the following result.

Corollary 6.4. *[8] There is a polynomial time algorithm for the minimum bounded degree graph orientation problem which finds an edge set of cost at most* OPT *whose reversal makes the digraph k-edge-connected and such that the number of edges reversed at each node v is at most $g(v) + 1$, where* OPT *is the cost of an optimal solution which satisfies all the degree constraints.*

§ 7. Concluding Remarks

In this survey we present an iterative method to analyze linear programming relaxations of combinatorial optimization problems. We hope that this approach applies to a larger class of combinatorial optimization problems, and provides a more unified method to analyze linear programming relaxations. Let us conclude with some open questions.

§ 7.1. Traveling Salesman Problems

A major open question in approximation algorithm is whether there is a constant factor approximation algorithm for the asymmetric traveling salesman problem, when the cost function satisfies triangle inequalities. The asymmetric traveling salesman problem is an instance of degree bounded network design problem in directed graphs. Is it possible to apply the iterative method in this paper to the asymmetric traveling salesman problem? It is also interesting to see whether this method can provide new insight for the (symmetric) traveling salesman problem.

§ 7.2. Packing Problems

One difficulty for the traveling salesman problems is that there are hard packing constraints, for which the solutions must satisfy exactly. Jain's iterative rounding method is suitable for covering problems, and the iterative relaxation method is suitable for producing solutions with small violation on the packing constraints. One general direction to investigate is whether this iterative method can be applied to packing problems.

§ 7.3. Combinatorial Algorithms

Another direction to investigate is whether there are combinatorial algorithms for the problems discussed in this paper. Jain's algorithm is still the only algorithm that achieves a constant factor approximation ratio for the survivable network design problem. Is there a purely combinatorial algorithm for this problem? Are there purely combinatorial algorithms for the degree bounded network design problems? Note that the algorithm by Fürer and Raghavachari [4] is an elegant purely combinatorial algorithm for the minimum maximum-degree Steiner tree problem.

References

[1] N. Bansal, R. Khandekar and V. Nagarajan, *Additive Guarantees for Degree Bounded Directed Network Design*, in Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC), 769-778, 2008.

[2] S.C. Boyd, W.R. Pulleyblank, *Optimizing over the Subtour Polytope of the Travelling Salesman Problem*, Mathematical Programming 2, 163-187, 1990.

[3] J. Edmonds and R. Giles, *A Min-Max Relation for Submodular Functions on Graphs*, Annals of Discrete Mathematics 1, 185-204, 1977.

[4] M. Fürer and B. Raghavachari, *Approximating the Minimum-Degree Steiner Tree to within One of Optimal*, J. of Algorithms 17(3), 409-423, 1994.

[5] M.X. Goemans, *Minimum Bounded-Degree Spanning Trees,* Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, 273-282, 2006.

[6] M.X. Goemans, A.V. Goldberg, S. Plotkin, D.B. Shmoys, É. Tardos, D.P. Williamson, *Improved Approximation Algorithms for Network Design Problems*, Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms, 223-232, 1994.

[7] K. Jain, *A Factor* 2 *Approximation Algorithm for the Generalized Steiner Network Problem*, Combinatorica 21, 39-60, 2001.

[8] T. Király, L.C. Lau and M. Singh, *Degree Bounded Matroids and Submodular Flows*, Proceedings of the 13th Conference on Integer Programming and Combinatorial Optimization (IPCO), 259-272, 2008.

[9] L.C. Lau, R. Ravi and M. Singh, *Iterative Methods in Combinatorial Optimization.* In Preparation, 2009.

[10] L.C. Lau, S. Naor, M. Salavatipour and M. Singh, *Survivable Network Design with Degree or Order Constraints*, Proceedings of the 39th ACM Annual Symposium on Theory of Computing (STOC), 651-660, 2007.

[11] L.C. Lau, M. Singh, *Additive Approximation for Bounded Degree Survivable Network Design*, Proceedings of the 40th ACM Annual Symposium on Theory of Computing (STOC), 759-768, 2008.

[12] M. Singh, L.C. Lau, *Approximating Minimum Bounded Degree Spanning Trees to within One of Optimal*, Proceedings of the 39th ACM Annual Symposium on Theory of Computing (STOC), 661-670, 2007.

[13] A. Schrijver, Combinatorial Optimization - Polyhedra and Efficiency, Springer-Verlag, New York, 2003.

[14] V. Vazirani, Approximation Algorithms, Springer, 2001.