Iterative Methods in Combinatorial Optimization

R. Ravi[1*]
Tepper School of Business
Carnegie Mellon University
Pittsburgh, USA
ravi@cmu.edu

ABSTRACT. We describe a simple iterative method for proving a variety of results in combinatorial optimization. It is inspired by Jain's iterative rounding method (FOCS 1998) for designing approximation algorithms for survivable network design problems, and augmented with a relaxation idea in the work of Lau, Naor, Salvatipour and Singh (STOC 2007) on designing an approximation algorithm for its degree bounded version. At the heart of the method is a counting argument that redistributes tokens from the columns to the rows of an LP extreme point. This token argument was further refined to fractional assignment and redistribution in work of Bansal, Khandekar and Nagarajan on degree-bounded directed network design (STOC 2008).
In this presentation, we introduce the method using the assignment problem, describe its application to showing the integrality of Edmond's characterization (1971) of the spanning tree polyhedron, and then extend the argument to show a simple proof of the Singh and Lau's approximation algorithm (STOC 2007) for its degree constrained version, due to Bansal, Khandekar and Nagarajan. We conclude by showing how Jain's original proof can also be simplified by using a fractional token argument (joint work with Nagarajan and Singh).
This presentation is extracted from an upcoming monograph on this topic co-authored with Lau and Singh.

1 Introduction

Iterative methods are an important tool in the growing toolkit available for designing approximation algorithms based on linear programming relaxations. First we motivate our method via the assignment problem. Through this problem we highlight the basic ingredients and ideas of the method and provide an outline of how a typical result proved using this method is structured. In the following sections, we apply this method to the classical minimum spanning tree problem, and extend it to derive an approximation algorithm for the degree-bounded version. In the last section, we present an application to re-derive an old result of Jain on LP extreme points for survivable network design problems.

The Assignment Problem: Consider the classical assignment problem: Given a bipartite graph $G = (U \cup V, E)$ with $|U| = |V|$ and weight function $w : E \to \mathbb{R}_+$, the objective is to match every vertex in U with a distinct vertex in V to minimize the total weight (cost) of the matching. This is also called the minimum weight bipartite perfect matching problem in the literature, and is a fundamental problem in combinatorial optimization.

One approach to the assignment problem is to model it as a linear programming problem. A linear program is a mathematical formulation of the problem with a system of linear

*Supported in part by NSF grant CCF-0728841.

constraints which can contain both equalities and inequalities, and also a linear objective function that is to be maximized or minimized. In the assignment problem, we associate a *variable* x_{uv} for every $(u, v) \in E$. Ideally, we would like the variables to take one of two values, zero or one (hence in the ideal case, they are binary variables). When x_{uv} is set to one, we intend the model to signal that this pair is matched; when x_{uv} is set to zero, we intend the model to signal that this pair is not matched. The following is a linear programming formulation of the assignment problem.

$$
\begin{array}{lrcll}
\text{minimize} & \displaystyle\sum_{u,v} w_{uv}\, x_{uv} & & & \\
\text{subject to} & \displaystyle\sum_{v:\{u,v\}\in E} x_{uv} & = & 1 & \forall\, u \in U \\
& \displaystyle\sum_{u:\{u,v\}\in E} x_{uv} & = & 1 & \forall\, v \in V \\
& x_{uv} & \geq & 0 & \forall\, \{u, v\} \in E
\end{array}
$$

The objective function is to minimize the total weight of the matching, while the two sets of linear equalities ensure that every vertex in U is matched to exactly one vertex in V in the assignment and vice-versa.

A fundamental result in the Operations Research literature [8] is the polynomial time solvability (as well as the practical tractability) of linear programming problems. There is also a rich theory of optimality (and certificates for it) that has been developed (see e.g., the text by Chvatal [3]). Using these results, we can solve the problem we have formulated above quite effectively for even very large problem sizes.

Returning to the formulation however, our goal is to find a "binary" assignment of vertices in U to vertices in V, but in the solution returned, the x-variables may take fractional values. Nevertheless, for the assignment problem, a celebrated result that is a cornerstone of combinatorial optimization [2] states that for any set of weights that permit a finite optimal solution, there is always an optimal solution to the above LP (linear program) that takes binary values in all the x-variables.

Such *integrality* results of LPs are few and far between, but reveal rich underlying structure for efficient optimization over the large combinatorial solution space [13]. They have been shown using special properties of the constraint matrix of the problem (such as total unimodularity), or of the whole linear system including the right hand side (such as total dual integrality). This article is about a simple and fairly intuitive method that is able to re-prove many (but not all) of the results obtained by these powerful methods. One advantage of our approach is that it can be used to incorporate additional constraints that make the problem computationally hard, and allow us to derive good approximation algorithms with provable performance guarantee for the constrained versions.

2 Iterative Algorithm

Our method is iterative. Using the following two steps, it works inductively to show that the LP has an integral optimal solution.

- If any x_{uv} is set to 1 in an optimal solution to the LP, then we take this pair as matched in our solution, and delete them both to get a smaller problem, and proceed to the next iteration.
- If any variable x_{uv} is set to 0 in the optimal solution, we remove the edge (u, v) to again get a smaller problem (since the number of edges reduces by 1) and proceed to the next iteration.

We continue the above iterations till all variables have been fixed to either 0 or 1. Given the above iterative algorithm, there are two claims that need to be proven. Firstly, that the algorithm works correctly, i.e., it can always find a variable with value 0 or 1 in each iteration and secondly, the matching selected is an optimal (minimum weight) matching. Assuming the first claim, the second claim can be proved by a simple inductive argument. The crux of the argument is that in each iteration our solution pays exactly what the fractional optimal solution pays. Moreover, the fractional optimal solution when restricted to the residual problem remains feasible for the residual problem. This allows us to apply an inductive argument to show that the matching we construct has the same weight as the fractional optimal solution, and is thus optimal. For the first claim, it is not clear a-priori that one can always find a variable with value 1 or 0 at every step. However, we use the important concept of the extreme point (or vertex) solutions of linear program to show that the above iterative algorithm works correctly.

DEFINITION 1. *Let $P = \{x : Ax = b, x \geq 0\} \subseteq \mathbb{R}^n$. Then $x \in \mathbb{R}^n$ is an* **extreme point solution** *of P if there does not exist a non-zero vector $y \in \mathbb{R}^n$ such that $x + y, x - y \in P$.*

Extreme point solutions are also known as vertex solutions and are equivalent to basic feasible solutions [3]. The following basic result shows that there is always an optimal extreme point solution to bounded linear programs.

LEMMA 2. *Let $P = \{x : Ax = b, x \geq 0\}$ and assume that the optimum value $\min\{c^T x : x \in P\}$ is finite. Then for any feasible solution $x \in P$, there exists an extreme point solution $x' \in P$ with $c^T x' \leq c^T x$.*

The following **"Rank lemma"** is an important ingredient in the correctness proofs of all iterative algorithms.

LEMMA 3. *Let $P = \{x : Ax = b, x \geq 0\}$ and let x be an extreme point solution of P such that $x_i > 0$ for each i. Then the number of variables is equal to the number of linearly independent constraints of A, i.e. the rank of A.*

2.1 Contradiction Proof Idea: Lower Bound > Upper Bound

We give an outline of the proof that at each iteration there exists a variable with value 0 or 1. Suppose for contradiction that $0 < x_e < 1$ for every edge e. We use this assumption to derive a lower bound on the number of variables of the linear program. Let n be the remaining vertices in U (or V, they have the same cardinality) at the current iteration. Then each vertex in U must have two edges incident on it, since $\sum_{v \in V:(u,v) \in E} x_{uv} = 1$ and $x_{uv} < 1$ for each $(u, v) \in E$. Thus the total number of edges is at least $2n$. This is a lower bound on the number of variables of the linear program, since we have one variable for each edge.

On the other hand, using the Rank Lemma, we derive an upper bound on the number of variables of the linear program. In the linear program for bipartite matching, we have only $2n$ constraints (one for each vertex in $U \cup V$). Moreover, these $2n$ constraints are dependent since the sum of the constraints for vertices in U equals the sum of the constraints for vertices in V. Hence, the number of linearly independent constraints is at most $2n - 1$. By the Rank Lemma, the number of variables is at most $2n - 1$. This provides us an upper bound on the number of variables. Since our upper bound is strictly smaller than the lower bound, we obtain the desired contradiction. Therefore, in an extreme point solution of the linear program for bipartite matching, there must exist a variable with value 0 or 1, and thus the iterative algorithm works. The number of iterations can be simply bounded by the number of edges in the bipartite graph.

3 Outline of the Approach

We now give a brief outline of the approach to designing algorithms with this approach. The method can be used to prove the integrality of the LP relaxation of a well-studied problem, and once this is well understood, the iterative proof of integrality can be extended to design approximation algorithms for NP-hard variants of the basic problems. Both components follow the natural outline described below.

1. **Linear Programming Formulation:** We start by giving a linear programming relaxation for the optimization problem we study. If the problem is polynomially solvable, this relaxation will be one with integral extreme points and that is what we will set out to show. If the problem is NP-hard, we state an approximation algorithmic result which we then set out to prove.
   (a) **Solvability:** Sometimes the linear programming relaxation we start with will be exponential in size. We then show that the linear program is solvable in polynomial time. Usually, this would entail providing a polynomial time *separation oracle* for the program using the formalism of the ellipsoid method [7]. Informally, the separation oracle is a procedure that certifies that any given candidate solution for the program is either feasible or not and in the latter case provides a separating hyperplane which is a violated inequality of the formulation. In programs with an exponential number of such inequalities that are implicity described, the design of the separation oracle is itself a combinatorial optimization problem, and we sketch the reduction to one.
2. **Characterization of Extreme Point Solution:** We then give a characterization result for the optimal extreme point solutions of the linear program based on the Rank Lemma 3. This part aims to show that any maximal set of independent tight constraints at this extreme point solution can be captured by a sparse structure. Sometimes the proof of this requires the use of the *uncrossing* technique [2] in combinatorial optimization.
3. **Iterative Algorithm:** We present an iterative algorithm for constructing an integral solution to the problem from the vertex solution. The algorithm has two simple steps.
   (a) If there is a variable in the optimal vertex solution that is set to a value of 1, then include the element in the integral solution.
   (b) If there is a variable in the optimal vertex solution that is set to a value of 0, then

remove the corresponding element.

In each of the above cases, at each iteration, we reduce the problem and arrive at a *residual* version and iterate until all variables have been set this way. In designing approximation algorithms we also use the rounding and relaxation steps as stated earlier.

4. **Analysis:** We then analyze the algorithm. This involves arguing the following two facts. First, we establish that the algorithm runs correctly and second, that it returns an optimal solution.
   (a) **Correctness:** We show that the iterative algorithm is correct by arguing that there is always a 1-element or a 0-element to pick in every iteration. This crucially uses the characterization of tight constraints at this optimal extreme point solution. The argument here also follows the same contradiction proof idea (lower bound $>$ upper bound): We assume for a contradiction that there is no 1-element or 0-element and get a large lower bound on the number of nonzero variables in the optimal extreme point solution. On the other side, we use the sparsity of the independent tight constraints to show an upper bound on the number of such constraints. This then contradicts the rank lemma that insists that both these numbers are equal, and proves that there is always a 1- or 0-element.
   (b) **Optimality:** We finally show that the iterative algorithm indeed returns an optimal solution using a simple inductive argument. The crux of this argument is to show that the extreme point solution induced on the residual problem remains a feasible solution to this residual problem.

3.1 Approximation Algorithms for NP-hard Problems

The above framework can be naturally adapted to provide an approximation algorithm via the iterative method. In particular, for this, the iterative algorithm above typically has one or both of two additional steps: *Rounding* and *Relaxation*.

1. **Rounding:** Fix a threshold $\alpha \geq 1$. If there is a variable x_i which in the optimal extreme point solution has a value of at least $\frac{1}{\alpha}$ then include the corresponding element in the solution.
   Adding this rounding step does not allow us to obtain optimal integral solution but only near-optimal solutions. Using the above step, typically one obtains an approximation ratio of $\frac{1}{\alpha}$ for covering problems addressed using this framework.
2. **Relaxation:** Fix a threshold β. If there is a constraint $\sum_i a_i x_i \leq b$ such that $\sum_i a_i \leq b + \beta$ then remove the constraint in the residual formulation.
   The iterative relaxation step removes a constraint and hence this constraint can be violated in later iterations. But the condition on the removal of the constraints ensures that the constraint is only violated by an additive amount of β. This step enables us to obtain *additive* approximation algorithms for a variety of problems.

To summarize, for designing approximation algorithms, we first study the exact optimization problem in the above framework. We then use the above two steps in various combinations to derive strong approximation algorithms for constrained versions of these exact problems.

4 Minimum Spanning Trees

In an instance of the Minimum Spanning Tree (MST) problem we are given an undirected graph $G = (V, E)$, edge costs given as $c : E \to \mathbb{R}$, and the task is to find a spanning tree of minimum total edge cost.

4.1 Linear Programming Relaxation

An exact linear formulation for the convex hull of integral spannign trees is the subtour elimination LP which is related to the study of the Traveling Salesman Problem. For $S \subseteq V$, define $E(S)$ to be the set of edges with both endpoints in S. For a spanning tree, there are at most $|S| - 1$ edges in $E(S)$, where $|S|$ denotes the number of vertices in S. Insisting on this for every set by using the constraint (2) eliminates all the potential subtours that can be formed in the LP solution: this is how the formulation gets its name.

$$\begin{aligned}
&\text{minimize} && \sum_{e \in E} c_e x_e && && (1)\\
&\text{subject to} && x(E(S)) \leq |S| - 1 && \forall\, \emptyset \neq S \subset V && (2)\\
& && x(E(V)) = |V| - 1 && && (3)\\
& && x_e \geq 0 && \forall\, e \in E && (4)
\end{aligned}$$

We will present an iterative algorithm which will prove that the subtour LP is integral.

THEOREM 4. *Every extreme point solution to the subtour LP is integral and corresponds to the characteristic vector of a spanning tree.*

Before we give the iterative algorithm and proof of Theorem 4, we show that one can optimize over the subtour LP in polynomial time. We show this giving a polynomial time separation oracle for the constraints in subtour LP. Polynomial time solvability now follows from results on the equivalence of separation and optimization [7].

THEOREM 5. *There is a polynomial time separation oracle for the subtour LP.*

PROOF. The separation oracle, given a fractional solution x, needs to find a set $S \subseteq V$ such that $x(E(S)) > |S| - 1$ if such a set exists. It is easy to check the equality $x(E(V)) = |V| - 1$. Thus, checking the inequality for S is equivalent to checking if $\min_S \{|S| - 1 - x(E(S))\} < 0$. Using $x(E(V)) = |V| - 1$ we obtain that it is enough to check $\min_S \{|S| - 1 + x(E(V)) - x(E(S)\} < |V| - 1\}$ or equivalently if $\min_S \{|S| + x(E(V)) - x(E(S)\} < |V|\}$.

We set up a min-cut problem in a new digraph D with a new source s and new sink t. We also have one node in the digraph per edge e in the support (i.e., with $x_e > 0$) and a node per vertex of G. The source s has an arc to every edge e with capacity x_e. For every edge $e = i, j$ in G, its corresponding node in D has two arcs of infinite capacity, one to each of the vertices i and j. Finally, every vertex i has an arc of unit capacity to t. To find a violated cut, we need to check if the min $s - t$ cut is smaller than $|V|$.

Suppose there is a violated set S with $x(E(S)) > |S| - 1$. Then consider the cut formed by including on the side of s, all the nodes of D corresponding to edges in $E(S)$ as well as the

vertices of S. The set of arcs coming out of this cut are those coming out of the vertices of S and hence have capacity $|S|$. All edges not in $E(S)$, namely in $E(V) - E(S)$, must now have their incoming arc from s in the cut for a total capacity contribution of $x(E(V)) - x(E(S))$. Thus a violated cut will have cut value less than $|V|$.

Conversely, suppose the min-cut solution returns one of value less than $|V|$: we show how to extract a violated set from it. Since every node in D corresponding to an edge e of G has both its outgoing arcs with infinite capacity, if such a node (say $e = i, j$) is in the s-side of the min cut, then both its successors (i.e. both i and j) must also be in the s-side of the minimum cut. Similarly, if we take all the vertices of G in the s-side of the min-cut, all edges of G which do not have both their endpoints in this set will have to lie on the t-side of this cut. If the min-cut found has the set S' in the s-side of the cut, the capacity of the cut is precisely $|S|$ (from the unit arcs going from these nodes to t) plus the x-value of all edges that do not have both end points in S, namely $x(E(V)) - x(E(S'))$ as required. If this min-cut value is less than $|V|$, we can see that S' is a violating set. ∎

4.2 Characterizations of Extreme Point Solutions via the Uncrossing Technique

In this subsection, we analyze the extreme point solution to the subtour LP. Recall that an extreme point solution is the unique solution defined by n linearly independent tight inequalities, where n is the number of variables in the linear program. There are exponentially many inequalities in the subtour LP, and an extreme point solution may satisfy many inequalities as equalities. To analyze an extreme point solution, an important step is to find a "good" set of tight inequalities defining it. If there is an edge e with $x_e = 0$, this edge can be removed from the graph without affecting the feasibility and the objective value. So henceforth assume every edge e has $x_e > 0$.

The uncrossing technique is a powerful technique and we shall use it to find a *good* set of tight inequalities for an extreme point solution in the subtour LP. Let $E(X, Y)$ denotes the set of edges with one endpoint in X and the other endpoint in Y, and let $E(X) = E(X, X)$ denote the set of edges of G induced in $X \subseteq V(G)$. For a set $F \subseteq E$, let $\chi(F)$ denote the vector in $\mathbb{R}^{|E|}$ that has an 1 corresponding to each edge $e \in F$, and 0 otherwise. This vector is called the *characteristic vector* of F. The following proposition is straightforward.

PROPOSITION 6. *For $X, Y \subseteq V$,*

$$\chi(E(X)) + \chi(E(Y)) \leq \chi(E(X \cup Y)) + \chi(E(X \cap Y)),$$

and equality holds if and only if $E(X \setminus Y, Y \setminus X) = \emptyset$.

PROOF. Observe that

$$\chi(E(X)) + \chi(E(Y)) = \chi(E(X \cup Y)) + \chi(E(X \cap Y)) - \chi(E(X \setminus Y, Y \setminus X))$$

and proof follows immediately. ∎

Given an extreme point solution x to the subtour LP, let $\mathcal{F} = \{S \mid x(E(S)) = |S| - 1\}$ be the family of tight inequalities for an extreme point solution x in the subtour LP. The following lemma shows that this family is closed under intersection and union.

LEMMA 7. *If $S, T \in \mathcal{F}$ and $S \cap T \neq \emptyset$, then both $S \cap T$ and $S \cup T$ are in $\mathcal{F}$. Furthermore, $\chi(E(S)) + \chi(E(T)) = \chi(E(S \cap T)) + \chi(E(S \cup T))$.*

PROOF. Observe that

$$\begin{aligned} |S| - 1 + |T| - 1 &= x(E(S)) + x(E(T)) \\ &\leq x(E(S \cap T)) + x(E(S \cup T))) \\ &\leq |S \cap T| - 1 + |S \cup T| - 1 \\ &= |S| - 1 + |T| - 1. \end{aligned}$$

The first equality follows from the fact that $S, T \in \mathcal{F}$. The second inequality follows from Proposition 6. The third inequality follows from the constraints for $S \cap T$ and $S \cup T$ in the subtour LP. The last equality is because $|S| + |T| = |S \cap T| + |S \cup T|$ for any two sets S, T. Equality must hold everywhere and we have $x(E(S \cap T)) + x(E(S \cup T)) = |S \cap T| - 1 + |S \cup T| - 1$. Thus, we must have equality for constraints for $S \cap T$ and $S \cup T$, i.e., $x(E(S \cap T)) = |S \cap T| - 1$ and $x(E(S \cup T)) = |S \cup T| - 1$, which implies that $S \cap T$ and $S \cup T$ are also in $\mathcal{F}$. Moreover, equality holds for Proposition 6 and thus $\chi(E(S \setminus T, T \setminus S)) = \emptyset$ and $\chi(E(S)) + \chi(E(T)) = \chi(E(S \cap T)) + \chi(E(S \cup T))$. ∎

Denote by $span(\mathcal{F})$ the vector space generated by the set of vectors $\{\chi(E(S)) \mid S \in \mathcal{F}\}$. Call two sets X, Y *intersecting* if $X \cap Y$, $X - Y$ and $Y - X$ are nonempty. A family of sets is *laminar* if no two sets are intersecting. The following lemma says that an extreme point solution is characterized by tight inequalities whose corresponding sets form a laminar family. This is a crucial structure theorem on the extreme point solutions for the subtour LP.

LEMMA 8. *If $\mathcal{L}$ is a maximal laminar subfamily of $\mathcal{F}$, then $span(\mathcal{L}) = span(\mathcal{F})$.*

PROOF. Suppose, by way of contradiction, that $\mathcal{L}$ is a maximal laminar subfamily of $\mathcal{F}$ but $span(\mathcal{L}) \subset span(\mathcal{F})$. For any $S \notin \mathcal{L}$, define $intersect(S, \mathcal{L})$ to be the number of sets in $\mathcal{L}$ which intersect S, i.e. $intersect(S, \mathcal{L}) = |\{T \in \mathcal{L} \mid S \text{ and } T \text{ are intersecting}\}|$. Since $span(\mathcal{L}) \subset span(\mathcal{F})$, there exists a set S with $\chi(E(S)) \notin span(\mathcal{L})$. Choose such a set S with minimum $intersect(S, \mathcal{L})$. Clearly, $intersect(S, \mathcal{L}) \geq 1$; otherwise $\mathcal{L} \cup \{S\}$ is also a laminar subfamily, contradicting the maximality of $\mathcal{L}$. Let T be a set in $\mathcal{L}$ which intersects S. Since $S, T \in F$, by Lemma 7, both $S \cap T$ and $S \cup T$ are in $\mathcal{F}$. Also, both $intersect(S \cap T, \mathcal{L})$ and $intersect(S \cup T, \mathcal{L})$ are smaller than $intersect(S, \mathcal{L})$, which will be proved next in Proposition 9. Hence, by the minimality of $intersect(S, \mathcal{L})$, both $S \cap T$ and $S \cup T$ are in $span(\mathcal{L})$. By Lemma 7, $\chi(E(S)) + \chi(E(T)) = \chi(E(S \cap T)) + \chi(E(S \cup T))$. Since $\chi(E(S \cap T))$ and $\chi(E(S \cup T))$ are in $span(\mathcal{L})$ and $T \in \mathcal{L}$, the above equation implies that $\chi(E(S)) \in span(\mathcal{L})$, a contradiction. It remains to prove Proposition 9.

PROPOSITION 9. *Let S be a set that intersects $T \in \mathcal{L}$. Then $intersect(S \cap T, \mathcal{L})$ and $intersect(S \cup T, \mathcal{L})$ are smaller than $intersect(S, \mathcal{L})$.*

PROOF. Since $\mathcal{L}$ is a laminar family, for a set $R \in \mathcal{L}$ with $R \neq T$, R does not intersect T (either $R \subset T$, $T \subset R$ or $T \cap R = \emptyset$). So, whenever R intersects $S \cap T$ or $S \cup T$, R also intersects S. Also, T intersects S but not $S \cap T$ or $S \cup T$. Therefore, $intersect(S \cap T, \mathcal{L})$ and $intersect(S \cup T, \mathcal{L})$ are smaller than $intersect(S, \mathcal{L})$ ∎

This completes the proof of Lemma 8. ∎

4.3 Iterative 1-edge-finding Algorithm

In this section, we give an iterative procedure to find a minimum spanning tree from an optimal extreme point solution of the subtour LP. The algorithm is shown in Figure 1. To create the residual problem, the chosen edge e is contracted from G to identify its endpoints to result in the graph G/e.

Iterative 1-edge-finding MST Algorithm

1. Initialization $F \leftarrow \emptyset$.
2. While $V(G) \neq \emptyset$ do
   (a) Find an optimal extreme point solution x of the subtour LP and remove every edge e with $x_e = 0$ from G.
   (b) Find an edge $e = \{u, v\}$ such that $x_e = 1$ and update $F \leftarrow F \cup \{e\}$, $G \leftarrow G/e$.
3. Return F.

Figure 1: Iterative 1-edge-finding MST Algorithm

4.4 Correctness and Optimality

LEMMA 10. *For any extreme point solution x of the subtour LP with $x_e \geq 0$ for each edge e there exists an edge f such that $x_f = 1$.*

PROOF. We assign one token for each edge e in the support E, for a total of $|E|$ tokens. We will redistribute the tokens so that each set in $\mathcal{L}$ will receive one token and there are some extra tokens left. This implies that $|E| > |\mathcal{L}|$, giving us the contradiction to Lemma 8 and the Rank Lemma that together imply that $|E| = |\mathcal{L}|$.

For each edge e, we redistribute x_e to the smallest set containing both the endpoints. Now, we show that each set in $\mathcal{L}$ can collect at least one token, and demonstrate some extra leftover fractional edge tokens giving us the contradiction.

Let S be any set in $\mathcal{L}$ with children $R_1, \ldots, R_k$. We have

$$x(E(S)) = |S| - 1$$

and for each i,

$$x(E(R_i) = |R_i| - 1$$

Subtracting, we obtain

$$x(E(S)) - \sum_i x(E(R_i)) = |S| - \sum_i |R_i| + k - 1.$$

This implies that

$$x(A) = |S| - \sum_i |R_i| + k - 1$$

where $A = E(S) \setminus (\cup_i E(R_i))$. Now S obtains exactly x_e fractional token for each edge e in A. If $A = \emptyset$, then $\chi(E(S)) = \sum_i \chi(E(R_i))$ which contradicts the independence of these sets of

constraints in $\mathcal{L}$. Moreover, $x(A)$ is an integer and hence it is at least one, giving S the unit token it needs.

Since every edge is not integral, we have the extra fractional token values of $(1 - x_e)$ for every edge as unused tokens giving the contradiction. ∎

THEOREM 11. *The Iterative MST Algorithm returns a minimum spanning tree in polynomial time.*

PROOF. This is proved by induction on the number of iterations of the algorithm. Note that if the algorithm finds a 1-edge e, for any spanning tree T' of $G' = G/e$, we can construct a spanning tree $T = T' \cup \{e\}$ of G. Hence, the residual problem is to find a minimum spanning tree on G/e, and the same procedure is applied to solve the residual problem recursively.

Since $x_e = 1$, the restriction of x to $E(G')$, denoted by x_{res}, is a feasible solution to the subtour LP for G'. Inductively, the algorithm will return a spanning tree F' of G' of cost at most the optimal value of the subtour LP for G', and hence $c(F') \leq c \cdot x_{res}$. Therefore,

$$c(F) = c(F') + c_e \text{ and } c(F') \leq c \cdot x_{res}$$

which imply that

$$c(F) \leq c \cdot x_{res} + c_e = c \cdot x$$

as $x_e = 1$. Hence, the spanning tree returned by the algorithm is of cost no more than the cost of an optimal LP solution x, which is a lower bound on the cost of a minimum spanning tree. This shows that the algorithm returns a minimum spanning tree of the graph. ∎

5 Minimum Bounded-Degree Spanning Trees

We next turn to the study of the MINIMUM BOUNDED-DEGREE SPANNING TREE (MBDST) problem. In an instance of the MBDST problem we are given a graph $G = (V, E)$, edge cost given by $c : E \to \mathbb{R}$, a degree upper bound B_v for each $v \in V$ and the task is to find a spanning tree of minimum cost which satisfies the degree bounds. We prove the following theorem originally due to Singh and Lau.

THEOREM 12. *There exists a polynomial time algorithm which given an instance of the MBDST problem returns a spanning tree T such that $deg_T(v) \leq B_v + 1$ and cost of the tree T is smaller than the cost of any tree which satisfies the degree bounds.*

We prove Theorem 12 using the iterative relaxation technique.

5.1 Linear Programming Relaxation

We use the following standard linear programming relaxation for the MBDST problem, which we denote by $LP_{mbdst}(G, \mathcal{B}, W)$. In the following we assume that degree bounds are given for vertices only in a subset $W \subseteq V$. Let $\mathcal{B}$ denote the vector of all degree bounds B_v, one for each vertex $v \in W$.

$$\begin{aligned}
\text{minimize} \quad & \sum_{e \in E} c_e x_e & & & & (5)\\
\text{subject to} \quad & x(E(V)) &=& |V| - 1 & & (6)\\
& x(E(S)) &\leq& |S| - 1 & \forall \emptyset \neq S \subset V & (7)\\
& x(\delta(v)) &\leq& B_v & \forall v \in W & (8)\\
& x_e &\geq& 0 & \forall e \in E & (9)
\end{aligned}$$

Separation over the inequalities in the above linear program can be carried out in polynomial time and follows from Theorem 5. An alternative is to write a compact reformulation of the above linear program which has polynomially many variables and constraints.

5.2 Characterization of Extreme Point Solutions

We first give a characterization of an extreme point solution of $LP_{mbdst}(G, \mathcal{B}, W)$. We remove all edges with $x_e = 0$ and focus only on the support of the extreme point solution and the tight constraints from (6)-(8). Let $\mathcal{F} = \{S \subseteq V : x(E(S)) = |S| - 1\}$ be the set of tight constraints from (6)-(7). From an application of Rank Lemma 3 and the characterization of extreme point solution to the spanning tree polyhedron (Lemma 8), we have the following characterization.

LEMMA 13. *Let x be any extreme point solution of $LP_{mbdst}(G, \mathcal{B}, W)$ 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 vectors $\{\chi(E(S)) : S \in \mathcal{L}\} \cup \{\chi(\delta(v)) : v \in T\}$ are linearly independent.*
3. *$|\mathcal{L}| + |T| = |E|$.*

5.3 An Additive One Approximation Algorithm

We now present an iterative algorithm which returns a tree of optimal cost and violates the degree bound within an additive error of one. This algorithm removes degree constraints one by one, and eventually reduces the problem to a minimum spanning tree problem. This can be thought of as a simple extension of the 1-edge-finding iterative MST algorithm presented earlier. The algorithm is given in Figure 2.

MBDST Algorithm

1. While $W \neq \emptyset$ do
   (a) Find an optimal extreme point solution x of $LP_{mbdst}(G, \mathcal{B}, W)$ and remove every edge e with $x_e = 0$ from G. Let the support of x be E.
   (b) **(Relaxation)** If there exists a vertex $v \in W$ with $deg_E(v) \leq B_v + 1$, then update $W \leftarrow W \setminus \{v\}$.
2. Return E.

Figure 2: Additive One MBDST Algorithm

5.4 Correctness and Performance Guarantee

In the next lemma we prove that in each iteration, the algorithm can find some vertex for which the degree constraint can be removed. Observe that once all the degree constraints are removed we obtain the linear program for the minimum spanning tree problem which we showed in Section 4 to be integral. Hence, the algorithm returns a tree. Moreover, at each step we only relax the linear program. Hence, the cost of the final solution is at most the cost of the initial linear programming solution. 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. The degree bound is violated only when we remove the degree constraint and then $deg_E(v) \leq B_v + 1$. Thus, in the worst case, if we include all the edges incident at v in T, the degree bound of v is violated by at most an additive one.

It remains to show that the iterative relaxation algorithm finds a degree constraint to remove at each step. From Lemma 13 we have that there exists a laminar family $\mathcal{L} \subseteq \mathcal{F}$ and $T \subseteq W$ such that $|\mathcal{L}| + |T| = |E|$ and constraints for sets in $\mathcal{L}$ are linearly independent. Observe that if $T = \emptyset$ then only the spanning tree inequalities define the solution x. Hence, x must be integral. In the other case, we show that there must be a vertex in W whose degree constraint can be removed.

LEMMA 14. *Let x be an extreme point solution to $LP_{mbdst}(G, \mathcal{B}, W)$ such that $x_e > 0$. Let $\mathcal{L}$ and $T \subseteq W$ correspond to the tight set constraints and tight degree constraints defining x as given by Lemma 13. If $T \neq \emptyset$ then there exists some vertex $v \in W$ with $deg_E(v) \leq B_v + 1$.*

PROOF. We use the fractional token argument as in the integrality proof of the 1-edge-finding iterative MST algorithm we presented earlier.

Suppose for the sake of contradiction, we have $T \neq \emptyset$ and $deg_E(v) \geq B_v + 2$ for each $v \in W$. We now show a contradiction by a token argument. 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 we still have extra tokens left. This will contradict $|E| = |T| + |\mathcal{L}|$. The token redistribution is as follows. Each edge $e \in E$ gives as before x_e tokens to the smallest set in $\mathcal{L}$ containing both endpoints of e, and $(1 - x_e)/2$ to each of its endpoints for the degree constraints.

We have already argued earlier that the x_e assignment suffices to obtain one token per member in the laminar family (see the proof of Lemma 10).

Thus it suffices to show that each vertex with a tight degree constraint gets one token. Let $v \in T$ be such a vertex. Then v receives $(1 - x_e)/2$ tokens for each edge incident at v for a total of

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

where the first equality holds since $\sum_{e \in \delta(v)} x_e = B_v$ and the inequality holds since $deg_E(v) \geq B_v + 2$ by Step 1b of the algorithm.

To finish the proof, we argue 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 the contradiction. Similarly, if there is a vertex $v \in W \setminus T$ then v also collects one token which it does not need and we get the desired contradiction. Moreover, if

there is a vertex $v \in V \setminus T$ then each edge e incident at v must have $x_e = 1$ else $(1 - x_e)/2 > 0$ tokens are extra. Note that $e \in span(\mathcal{L})$ 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 $e \in span(\mathcal{L})$ for each edge $e \in \delta(v)$ for $v \in V - T$. Since $T \neq \emptyset$, this implies the linear independence of the tight constraints in T and those in $\mathcal{L}$, giving us the contradiction. ∎

5.5 Historical Notes

Edmonds [4] gave the integral linear programming relaxation for minimum spanning tree problem that we presented. There is a long line of work of successively improving the performance guarantees for the degree-bounded minimum-cost spanning tree problem. The algorithm with additive guarantee of one for the unweighted case was first given by Fürer and Raghavachari [5]. The additive algorithm with violation 2 (with both upper and lower degree bounds) was presented by Goemans [6]. The algorithm with additive violation of 1 was first presented by Singh and Lau [14], also for the case with upper and lower bounds on the degree. The fractional token proof which we used for the additive one proof was first presented by Bansal et al. [1].

6 Survivable Network Design Problem

The survivable network design problem generalizes the minimum Steiner tree problem, the minimum Steiner forest problem, and the minimum k-edge-connected subgraph problem, etc. Hence the result in this section also applies to these problems.

6.1 Linear Programming Relaxation

To formulate the problem as a linear program, we represent the connectivity requirements by a *skew supermodular* function. A function $f : 2^V \rightarrow \mathbb{Z}$ is called skew supermodular if at least one of the two following conditions hold for any two subsets $S, T \subseteq V$.

$$\begin{aligned} f(S) + f(T) &\leq f(S \cup T) + f(S \cap T) \\ f(S) + f(T) &\leq f(S \backslash T) + f(T \backslash S) \end{aligned}$$

It can be verified (with some simple case analysis) 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. Hence, one can write the following linear programming relaxation for the survivable network design problem, denoted by LP_{sndp}.

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

This linear program for the case of minimum Steiner networks can be solved in polynomial time by using a minimum cut algorithm as a separation oracle. Designing a separation oracle for more general skew submodular functions as right hand sides needs more work - details can be found in the original paper of Jain [9].

6.2 Characterization of Extreme Point Solutions

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 $\chi(\delta(S))$. Recall that two sets X, Y are intersecting if $X \cap Y$, $X - Y$ and $Y - X$ are nonempty, and that a family of sets is laminar if no two sets are intersecting. It is not hard to verify the two inequalities below using the submodularity of the cut function.

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

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

For any two subsets X and Y, when f is skew supermodular, it follows from standard uncrossing arguments, as in the case of spanning trees, that an extreme point solution to LP_{sndp} is characterized by a laminar family of tight constraints. The Lemma below then follows from these uncrossing arguments and the Rank Lemma (Lemma 3).

LEMMA 15. *Let the requirement function f of LP_{sndp} be skew supermodular, and let x be an extreme point solution to LP_{sndp} with $0 < x_e < 1$ for every edge $e \in E$. Then, there exists a laminar family $\mathcal{L}$ such that:*

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

6.3 Iterative Algorithm

Jain's iterative rounding algorithm is in Figure 3.

Iterative Algorithm for Minimum Steiner Network

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

Figure 3: Minimum Steiner Network Algorithm

6.4 Correctness and Performance Guarantee

Jain proved an important theorem about the extreme point solutions of LP_{sndp}.

THEOREM 16. *[Jain] Suppose f is an integral skew submodular function and x is an extreme point solution to LP_{sndp}. Then there exists an edge $e \in E$ with $x_e \geq \frac{1}{2}$.*

Assuming Theorem 16, then the iterative algorithm will terminate successfully, and it can be shown by a straightforward inductive argument that the returned solution is a 2-approximate solution.

THEOREM 17. *Algorithm 3 is a 2-approximation algorithm for the* SURVIVABLE NETWORK DESIGN *problem.*

PROOF. The proof is by induction on the number of iterations executed by the algorithm. For the base case that requires only one iteration, the theorem follows since it rounds up a single 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 16. Let f' be the residual requirement function after this iteration and let H' be the set of edges picked in subsequent iterations for satisfying f'. The key observation is 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 $H := H' \cup e'$ which satisfies f (by the definition of f'). The cost of H is:

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

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 of the optimal cost, and thus the theorem follows. ∎

We now give a simple proof of Jain's theorem above using the fractional token idea from the previous sections. This proof is due to Nagarajan et al. [12].

PROOF. We first prove that $x_e \geq \frac{1}{2}$ for some edge $e \in E$ in any extreme point solution x to LP_{SNDP}. Suppose that $0 < x_e < \frac{1}{2}$ for each $e \in E$. Then we will show that $|E| > |\mathcal{L}|$, contradicting Lemma 15. The proof is by a fractional token counting argument. We give one token to each edge in E, and then we will reassign the tokens such that we can collect one token for each member in $\mathcal{L}$ and still have extra tokens left, giving us the contradiction that $|E| > |\mathcal{L}|$. Each edge $e = uv$ is given one token which is reassigned as follows.

1. **(Rule 1)** Let $S \in \mathcal{L}$ be the smallest set containing u and $R \in \mathcal{L}$ be the smallest set containing v. Then e gives x_e tokens each to S and R.
2. **(Rule 2)** Let T be the smallest set containing both u and v. Then e gives $1 - 2x_e$ tokens to T.

We now show that each set S in $\mathcal{L}$ receives at least one token. Let S be any set with children $R_1, \ldots, R_k$ where $k \geq 0$ (if S does not have any children then $k = 0$). We have the following equalities.

$$\begin{aligned} x(\delta(S)) &= f(S) \\ x(\delta(R_i)) &= f(R_i) \;\; \forall\, 1 \leq i \leq k \end{aligned}$$

Subtracting we obtain,

$$x(\delta(S)) - \sum_i x(\delta(R_i)) = f(S) - \sum_{i=1}^{k} f(R_i). \tag{10}$$

We divide the edges involved into three types, where

$$\begin{aligned} A &= \{e : |e \cap (\cup_i R_i)| = 0, |e \cap S| = 1\} \\ B &= \{e : |e \cap (\cup_i R_i)| = 1, |e \cap S| = 2\} \\ C &= \{e : |e \cap (\cup_i R_i)| = 2, |e \cap S| = 2\}. \end{aligned}$$

Then (10) can be rewritten as:

$$x(A) - x(B) - 2x(C) = f(S) - \sum_{i=1}^{k} f(R_i). \tag{11}$$

Observe that $A \cup B \cup C \neq \emptyset$; otherwise the characteristic vectors $\chi(\delta(S)), \chi(\delta(R_1)), \ldots, \chi(\delta(R_k))$ are linearly dependent. For each edge $e \in A$, S receives x_e tokens from e by Rule 1. For each edge $e \in B$, S receives $1 - x_e$ tokens from e by Rule 1 and Rule 2. For each edge $e \in C$, S receives $1 - 2x_e$ tokens from e by Rule 2. Hence, the total tokens received by S are exactly,

$$\begin{aligned} 0 &< \sum_{e \in A} x_e + \sum_{e \in B} (1 - x_e) + \sum_{e \in C} (1 - 2x_e) \\ &= x(A) + |B| - x(B) + |C| - 2x(C) \\ &= |B| + |C| + f(S) - \sum_{i=1}^{k} f(R_i), \end{aligned}$$

where the last equality follows from (11). Since f is integral, the right hand side is at least one, and thus every set $S \in \mathcal{L}$ receives at least one token in the reassignment.

It remains to show that there are some unassigned tokens, which would imply the contradiction that $|E| > |\mathcal{L}|$. Let R be any maximal set in $\mathcal{L}$. Consider any edge $e \in \delta(R)$. The fraction of the token by Rule 2 for edge e is unassigned, as there is no set with $|T \cap e| = 2$, and gives us the desired contradiction. ∎

References

[1] N. Bansal, R. Khandekar and V. Nagarajan, *Additive guarantees for degree bounded directed network design*, in Proceedings of the Fourtieth Annual ACM Symposium on Theory of Computing (STOC), 2008.

[2] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, and A. Schrijver, *Combinatorial Optimization*, John Wiley and Sons, New York (1998).

[3] V. Chvatal, *Linear Programming*, Freeman, 1983.

[4] J. Edmonds, *Matroids and the Greedy Algorithm.* Mathematical Programming, 1:125–136, 1971.

[5] M. Fürer and B. Raghavachari, *Approximating the minimum-degree Steiner tree to within one of optimal*, J. of Algorithms 17(3):409-423, 1994.
[6] M.X. Goemans, *Minimum Bounded-Degree Spanning Trees,* Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, 2006, 273–282.
[7] M. Grotschel, L. Lovasz, A. Schrijver *The Ellipsoid Method and its Consequences in Combinatorial Optimization,* Combinatorica 1 (1981), 169-197.
[8] F.S. Hillier and G.J. Lieberman, *Introduction to Operations Research* (6th Ed.), Mcgraw-Hill, 1995.
[9] K. Jain, *A factor* 2 *approximation algorithm for the generalized Steiner network problem*, Combinatorica, **21**, pp.39-60, 2001. Preliminary version in *Proc. 39th IEEE FOCS*, 1998.
[10] L.C. Lau, S. Naor, M. Salavatipour and M. Singh, *Survivable network design with degree or order constraints*, Proceedings of the 40th ACM Symposium on Theory of Computing, 651-660, 2007.
[11] L.C. Lau, R. Ravi and M. Singh, *Iterative Methods in Combinatorial Optimization*, In Preparation, 2009.
[12] V. Nagarajan, R. Ravi and M. Singh, *Unified Analysis of LP Extreme Points for Steiner Network and Traveling Salesman*, submitted (2009).
[13] A. Schrijver, *Combinatorial Optimization - Polyhedra and Efficiency*, Springer-Verlag, New York, 2005.
[14] Mohit Singh and Lap Chi Lau, *Approximating Minimum Bounded Degree Spanning Tress to within One of Optimal*, Proceedings of 39th ACM Symposium on Theory of Computing, 661-670, 2007.