Approximation Algorithms for Network Design: A Survey

Anupam Gupta* Jochen Könemann†

In a typical instance of a *network design* problem, we are given a directed or undirected graph $G = (V, E)$, non-negative edge-costs c_e for all $e \in E$, and our goal is to find a minimum-cost subgraph H of G that satisfies some design criteria. For example, we may wish to find a minimum-cost set of edges that induces a connected graph (this is the *minimum-cost spanning tree* problem), or we might want to find a minimum-cost set of arcs in a directed graph such that every vertex can reach every other vertex (this is the *minimum-cost strongly connected subgraph problem*). This abstract model for network design problems has a large number of practical applications; the design process of telecommunication and traffic networks, and VLSI chip design are just two examples.

Many practically relevant instances of network design problems are NP-hard, and thus likely intractable. This survey focuses on *approximation algorithms* as one possible way of circumventing this impasse. Approximation algorithms are efficient (i.e., they run in polynomial-time), and they compute solutions to a given instance of an optimization problem whose objective values are *close* to those of the respective optimum solutions. More concretely, most of the problems discussed in this survey are minimization problems. We then say that an algorithm is an α*-approximation* for a given problem if the ratio of the cost of an approximate solution computed by the algorithm to that of an optimum solution is at most α over all instances. In the following we will also sometimes refer to α as the *performance guarantee* of the respective approximation algorithm.

The last 30 years have seen a tremendous amount of research on approximation algorithms for network design problems. And over this period, several technical themes have emerged, and have been explored and exploited to give algorithms and analyze their performance. Our aim in this survey is to provide an overview over these techniques. Each of the following sections focuses on one technique and has two main parts: first, we present an introductory application to the well-known classical *minimum-spanning tree* problem. The second part of each section demonstrates more sophisticated recent example applications of the respective technique. Throughout we assume that the reader is familiar with fundamental concepts of graph theory, combinatorial optimization, and approximation algorithms. While we may recap certain key definitions, we rely on the reader to be familiar with others. We refer to the excellent text books [45, 160, 163] for background reading.

The minimum spanning tree problem has been studied for at least a century, and it is clearly one of the most prominent network design problems. The input to an instance of this problem consists of an undirected graph $G = (V, E)$ each of whose edges $e \in E$ is endowed by an arbitrary cost c_e, and the goal is to compute a spanning tree of smallest cost. The earliest known algorithm for this problem was developed by Borůvka [21], and since then a vast number of techniques have been developed and subsequently used in order to devise increasingly sophisticated algorithms.

*Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA. Supported in part by NSF awards CCF-0448095 and CCF-0729022, and an Alfred P. Sloan Fellowship.

†Department of Combinatorics & Optimization, University of Waterloo, Waterloo, ON N2L 3G1, Canada. Supported in part by NSERC grant no. 288340.

1 Greedy Algorithms

The first known algorithms for the minimum spanning tree problem use a *tree growing* approach (see [48, 95, 112, 134]); we describe Kruskal's algorithm: the algorithm starts with the empty spanning forest $F = (V, \emptyset)$ and in each step adds a cheapest edge $uv \in E$ whose endpoints lie in different connected components of F. The algorithm terminates as soon as F is a spanning tree.

Kruskal's algorithm is a good example for a *greedy* algorithm, as it constructs the final set of edges iteratively, starting from the empty set, and each of its edge-addition steps is independent of future iterations. Acting in a locally optimal way in each step is maybe the most characteristic feature of a greedy algorithm.

Theorem 1. *For any connected graph G with arbitrary edge costs c, Kruskal's algorithm finds a minimum-cost spanning tree.*

A direct combinatorial proof of this theorem is based on idea by Jarník [95] and shows that each intermediate forest is contained in a some minimum-cost spanning tree of G. We omit details and refer the reader to [151].

1.1 Matroids and greedy algorithms

A slight generalization of Kruskal's algorithm turns out to be optimal for a much more general class of problems.

Definition 2. *Let E be a ground-set of elements and consider a set $\mathscr{F} \subseteq 2^E$ of subsets of E. $(E, \mathscr{F})$ is called an* independence system *if (M1) $\emptyset \in \mathscr{F}$, and (M2) if $X \in \mathscr{F}$ and $Y \subseteq X$, then $Y \in \mathscr{F}$ as well.*

The elements of $\mathscr{F}$ are called independent *and the elements in $2^E \setminus \mathscr{F}$ are* dependent*. For any set S of elements, an inclusion-wise maximal subset T of S is called a* basis *of S. For brevity, we say that T is a basis, if it is a basis of E.*

For example, consider a connected undirected graph $G = (V, E)$ and let $S \subseteq E$ be in $\mathscr{F}$ iff S has no cycles. It is then not hard to see that $(E, \mathscr{F})$ is an independence system whose bases are the spanning trees of G.

Definition 3. *Given an independence system $(E, \mathscr{F})$ and a subset $S \subseteq E$. The* rank $r(S)$ *is defined as the cardinality of the largest basis of S, and the* lower rank $\rho(S)$ *is the cardinality of the smallest basis of S. The* rank-quotient $q(E, \mathscr{F})$ *then defined as*

$$q(E, \mathscr{F}) = \min_{S \subseteq E} \frac{\rho(S)}{r(S)}.$$

Given an independence system $(E, \mathscr{F})$ and a non-negative weight w_e for each of the elements $e \in E$, we now consider the problem of finding a basis of E of maximum total weight. Consider the following greedy algorithm.

Best-In-Greedy Algorithm:

Input: independence system $(E, \mathscr{F})$, weights $w_e \geq 0$ for all $e \in E$
Sort $E = \{e_1, \ldots, e_m\}$ such that $w_{e_1} \geq \cdots \geq w_{e_m}$
$F = \emptyset$
for $1 \leq i \leq m$ **do**
  if $F \cup \{e_i\} \in \mathscr{F}$ **then**
    Add e_i to F
  end if
end for

9: Return F

The following result is due to Jenkyns [96], and Korte and Hausmann [105]. We omit its proof and refer the reader to [108].

Theorem 4. *Given an independence system $(E, \mathscr{F})$, the Best-In-Greedy algorithm finds a set F such that $w(F) \geq q(E, \mathscr{F}) \cdot w(F^*)$, where F^* is a basis of maximum total weight.*

Consider the spanning tree example from before, and let $S \subseteq E$ be a subset of the edges. Clearly, $T \subseteq S$ is a basis of S iff it is an inclusion-wise maximal forest in the graph $G[S]$ induced by the edges in S. Any such forest has $n - \kappa(S)$ edges, where $\kappa(S)$ is the number of connected components of $G[S]$. Hence, the independence system for acyclic subgraphs has rank quotient equal to 1.

Definition 5. *An independence system $(E, \mathscr{F})$ with $q(E, \mathscr{F}) = 1$ is called a* matroid.

The specific matroid whose independent sets correspond to acyclic sub-graphs of G is called the *graphic* or *forest* matroid. It follows that the Best-In-Greedy algorithm is an exact algorithm for finding a spanning tree of maximum weight in a given graph G. Given an instance of the minimum-cost spanning tree problem, we define the weight w_e of each edge $e \in E$ as $c_{max} - c_e$ where c_{max} is the maximum cost among all edges. Observe now that any maximum-weight spanning tree for these weights is, at the same time, a minimum-cost spanning tree under the original costs, and vice versa. Moreover, all weights are non-negative, and we can therefore use the Best-In-Greedy method to compute a minimum-cost spanning tree. In fact, it is not hard to see that the Best-In-Greedy method is equivalent to Kruskal's algorithm in this case.

We remark that Korte, Lovász, and Schrader [106] showed that the class of *greedoids* which contains that of all matroids admits optimal greedy algorithms if a certain *strong exchange property* is satisfied.

It is easily checked that the independence system given by the intersection of k matroids has $q(E, \mathscr{F}) \leq k$, and hence the greedy algorithm is a k-approximation for the max-weight independent set in the intersection of k matroids; this generalizes the observation that the greedy algorithm is a 2-approximation for the max-weight matching in a graph. Also, Theorem 4 can be generalized to show that given any independence system $(E, \mathscr{F})$ and a monotone *submodular* function $f : 2^E \to \mathbb{R}_{\geq 0}$, the greedy algorithm is a $(q(E, \mathscr{F}) + 1)$-approximation for the problem of maximizing $f(S)$ over sets $S \in \mathscr{F}$ [129, 96, 24]; recently other algorithms have been used to give a $\frac{e}{e-1}$-approximation for the case of $q(E, \mathscr{F}) = 1$ [161]. These algorithms for submodular maximization have many applications to approximation algorithms, see, e.g. [24] and the references therein.

2 Primal-Dual Approaches

The original *primal-dual method* was introduced by Dantzig, Ford and Fulkerson [47] as an alternate method for solving linear programs. Its main feature, the reduction of weighted instances of an optimization problem to unweighted ones, is at the heart of many classical algorithms in combinatorial optimization (e.g., [48, 50, 60]).

Later work on the *vertex-cover* problem due to Bar-Yehuda and Even [14] showed that primal-dual techniques can also be employed to obtain good approximation algorithms for NP-hard optimization problems. In [75, 76], Goemans and Williamson provided a rigorous description of a primal-dual framework for the design of approximation algorithms. In the course of the last 20 years, this method has evolved to one of the most powerful techniques in the design optimization problems for network design problems. Central to this approach is the idea that for a minimization problem (say), given a LP relaxation of the problem (the "primal") and its LP dual, the cost of any feasible dual solution is no more than the cost of the optimal primal solution—hence, if we can construct an integral primal solution and a dual solution such that the cost of this primal is no more than α times the cost of the dual solution, we have an α-approximate integral solution.

2.1 A primal-dual interpretation of Kruskal's algorithm

We demonstrate the main features of primal-dual algorithms by reinterpreting Kruskal's algorithm. For this, we first exhibit a linear programming formulation for the minimum-cost spanning tree problem. The LP has a variable x_e for each edge $e \in E$, and its feasible $0,1$-valued solutions correspond to incidence vectors of spanning trees. Let Π denote the set of all partitions of the vertex set V, and let the *rank* $r(\pi)$ of $\pi \in \Pi$ be the number of parts of π. We use E_π to denote the set of edges whose ends lie in different parts of π and also say that an edge e *crosses* partition π if $e \in E_\pi$. The following formulation is due to Fulkerson [64].

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e & (\mathrm{P}^+_{SP}) \\ \text{s.t.} \quad & \sum_{e \in E_\pi} x_e \geq r(\pi) - 1 \quad \forall \pi \in \Pi, \\ & x \geq 0. \end{aligned}$$

The validity of the above formulation is not difficult to see: for a spanning tree T and a partition π of V, T must have at least $r(\pi) - 1$ edges in E_π. Hence, the optimum value of the above LP is at most the cost of an MST. In fact, Fulkerson [64] states the following theorem whose first explicit proof was given by Chopra [41].

Theorem 6. *The feasible region of* (P^+_{SP}) *is the dominant of the convex hull of incidence vectors of spanning trees of* G.

The standard linear programming dual of (P^+_{SP}) has a variable y_π for each partition $\pi \in \Pi$ and a constraint for each edge $e \in E$.

$$\begin{aligned} \max \quad & \sum_{\pi \in \Pi} (r(\pi) - 1) \cdot y_\pi & (\mathrm{D}^+_{SP}) \\ \text{s.t.} \quad & \sum_{\pi : e \in E_\pi} y_\pi \leq c_e \quad \forall e \in E, & (1) \\ & y \geq 0. \end{aligned}$$

Constraint (1) for edge e restricts the total dual value of partitions that e crosses by the cost c_e of the edge. Kruskal's algorithm can now be viewed as a continuous process over *time*: we start with an empty tree at time $\tau = 0$ and add edges as time progresses. The algorithm terminates at time τ^* with a spanning tree of the input graph G. At any time $0 \leq \tau \leq \tau^*$ we keep a pair (x_τ, y_τ), where x_τ is the incidence vector of a spanning forest in G and y_τ is a feasible dual solution for (D^+_{SP}). Initially, we let $x_{e,0} = 0$ for all $e \in E$ and $y_{\pi,0} = 0$ for all $\pi \in \Pi$.

Let F_τ denote the forest corresponding to partial solution x_τ and E_τ denote its edges, i.e., $E_\tau = \{e \in E : x_{e,\tau} = 1\}$. We use π_τ to denote the partition induced by the connected components of F_τ. At time τ, the algorithm increases y_{π_τ} until constraint (1) for some edge $e \in E_{\pi_\tau}$ is satisfied with equality; we say that edge e *becomes tight*. Suppose that this happens at time $\tau' \geq \tau$. We then include e into our solution by setting $x_{e,\tau'} = 1$ and update the dual by letting

$$y_{\pi_\tau, \tau'} = \tau' - \tau.$$

The above view of Kruskal's algorithm first appeared in [41]. Chopra showed that the final primal and dual solutions have the same objective value (and are, by LP duality, optimal). We remark that a similar interpretation for *Prim's* algorithm appears in Lawler's book [118]. We sketch Chopra's proof for completeness.

Proof of Theorem 1. All edges $e \in E_{\tau^*}$ are tight and therefore $c_e = \sum_{\pi: e \in E_\pi} y_{\pi,\tau^*}$. We can use this to express the cost of the final tree as follows:

$$c(F_{\tau^*}) = \sum_{e \in E_{\tau^*}} \sum_{\pi: e \in E_\pi} y_{\pi,\tau^*} = \sum_{\pi \in \Pi} |E_{\tau^*} \cap E_\pi| \cdot y_{\pi,\tau^*}.$$

By construction the set $E_{\tau^*} \cap E_\pi$ has cardinality exactly $r(\pi) - 1$ for all $\pi \in \Pi$ with $y_{\pi,\tau^*} > 0$. We obtain that

$$\sum_{e \in E} c_e x_{e,\tau^*} = \sum_{\pi \in \Pi} (r(\pi) - 1) \cdot y_{\pi,\tau^*}$$

whose right-hand side is the objective value of the dual solution y_{τ^*}. The theorem follows from weak duality. □

2.2 Modeling network design problems: f-connectivity

In an *f-connectivity* problem we are given an undirected graph $G = (V, E)$, non-negative costs c_e for all $e \in E$, and a *cut-requirement* function $f : 2^V \to \{0, 1\}$. Our goal is to find a minimum-cost subgraph H of G such that H has at least $f(S)$ edges crossing each set $S \subseteq V$. We will assume $f(V) = 0$ throughout this section. The following is a natural LP formulation of the problem:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e & (\mathrm{IP}_f) \\ \text{s.t.} \quad & \sum_{e \in \delta(S)} x_e \geq f(S) \quad \forall S \subseteq V \\ & x_e \text{ integer} \quad \forall e \in E \end{aligned}$$

Replacing the integrality requirement by non-negativity constraints yields the canonical LP relaxation (P_f). Its LP dual has a variable y_S for every set $S \subseteq V$.

$$\begin{aligned} \max \quad & \sum_{S \subseteq V} f(S) \cdot y_S & (\mathrm{D}_f) \\ \text{s.t.} \quad & \sum_{S: e \in \delta(S)} y_S \leq c_e \quad \forall e \in E & (2) \\ & y \geq 0 \end{aligned}$$

Note the minimum spanning tree problem can be cast in this framework by using the function $f(S) = 1$ for all $S \notin \{\emptyset, V\}$, $f(\emptyset) = f(V) = 0$.

2.3 Advanced Primal-Dual Applications: Steiner trees and forests

The *Steiner tree* problem is a well-studied classical problem that fits into the framework of Section 2.2. The input in an instance of this problem specifies a set of *terminals* R and the goal is to find a minimum-cost tree in the input graph that connects all terminals. In the following we say that two vertex sets S and T *intersect* if all of $S \cap T$, $S \setminus T$, and $T \setminus S$ are non-empty. The Steiner tree problem is then modeled by letting $f(S) = 1$ if S intersects R, and $f(S) = 0$ otherwise; such a set is sometimes called a *Steiner cut*. The resulting integer program (IP_f) is commonly known as the *undirected cut* formulation for Steiner tree [9].

The Steiner tree problem is a fundamental problem in combinatorial optimization with a rich history and many practical applications; we refer the reader to the two excellent surveys in [92, 135] for a much more detailed account. The Steiner tree problem is well-known to be NP-hard in general (see [98]), and remains

hard even in Euclidean spaces [68] and in other restricted topologies. Moreover, Chlebík and Chlebíková showed in [40] that it is NP-hard even to *approximate* the minimum-cost Steiner tree within any ratio better than $\frac{96}{95}$.

Until very recently, the best known algorithm for the Steiner tree problem was due to Robins and Zelikovsky [149]. Robins and Zelikovsky's algorithm follows a greedy strategy, and achieves a performance ratio of $1+\frac{\ln 3}{2} \approx 1.55$; the algorithm is the final refinement of a long list of greedy algorithms for the problem [18, 91, 99, 168]. Roughly speaking, the main insight in these algorithms is to view the Steiner tree problem as a minimum-cost spanning tree problem in a suitably defined hypergraph. This *hypergraphic* view of the Steiner tree problem inspired a number of linear programming relaxations [101, 133, 162]. Using an elegant randomized rounding approach applied to the directed relaxation of [133], Byrka et al. [22] recently obtained the currently best $\ln 4 < 1.39$-approximation for the problem.

We start this section by presenting a primal-dual approximation algorithm due to Agrawal, Klein and Ravi [4]. This algorithm (henceforth denoted by AKR) is a primal-dual algorithm and as such it constructs a solution to (P_f) and (D_f) simultaneously. Much like the primal-dual algorithm for minimum-cost spanning trees presented earlier, it does this in an incremental fashion.

We can think of an execution of AKR as a process over time. Let x^τ and y^τ, respectively, be the primal incidence vector and feasible dual solution at time τ. We use F^τ to denote the forest corresponding to x^τ. Initially, $x^0_e = 0$ for all $e \in E$ and $y^0_S = 0$ for all $S \subseteq V$. Assume that the forest F^τ at time τ is infeasible. We use $\bar{F}^\tau$ to denote the subgraph of G that is induced by the tight edges for dual y^τ. In the following, we will also use the term *moat* to refer to a connected component S of $\bar{F}^\tau$.

Algorithm AKR raises the dual variables of all moats uniformly as long as F^τ is infeasible. Let S_1 and S_2 be two distinct connected components of the forest maintained by the algorithm. We say that these components *collide* at time τ if τ is the first time during the execution of the algorithm where S_1 and S_2 are contained in a common moat.

If this happens, we add the edges on a shortest S_1, S_2-path to F^τ and continue. The algorithm stops at the earliest time τ^* where F^{τ^*} is a feasible Steiner tree. Suppose

$$\mathscr{P} = \{P_1, \ldots, P_q\}$$

is the set of paths that the algorithm adds in the process of constructing F^{τ^*} and let τ_i be the time where P_i is added for all $1 \leq i \leq q$ (we order the paths such that $\tau_1 \leq \ldots \leq \tau_q$). Also let S^1_i and S^2_i be the colliding components that caused AKR to add path P^i. We obtain the following lemma:

Lemma 7. *For all times $0 \leq \tau \leq \tau^*$ and for all moats $S \in \bar{F}^\tau$ we have*

$$\sum_{e \in E(S)} c_e x^\tau_e \leq 2 \cdot \sum_{S' \subseteq S} y^\tau_S - 2\tau.$$

Proof. In the following let $\tau_0 = 0$ be the start time of the algorithm. We prove the claim using induction over $0 \leq i \leq q$. The claim is trivially true for time $i = 0$. Now assume that the claim holds for $i \geq 0$. We state the following technical fact and refer the reader to [4] for a proof:

Claim 8. *The cost of path P_i is at most $2\tau_i$.*

By induction, we have that

$$\sum_{e \in E(S)} c_e x^{\tau_i}_e \leq 2 \cdot \sum_{S' \subseteq S} y^{\tau_i}_S - 2\tau_i. \tag{3}$$

for all moats S of $\bar{F}^{\tau_i}$. Consider such a moat S; the algorithm grows the dual variable for this moat at all times $\tau \in [\tau_i, \tau_{i+1})$. During this interval, no edges are added to the partial tree and the left-hand side of (3) remains

unchanged. Similarly, it is not hard to see that the right hand side of the equation remains unchanged as well and we have

$$\sum_{e \in E(S^j_{i+1})} c_e x_e^{\tau_i} \leq 2 \cdot \sum_{S' \subseteq S^j_{i+1}} y_S^{\tau_{i+1}} - 2\tau_{i+1}. \tag{4}$$

for $j \in \{1,2\}$ (where we abuse notation slightly and let $E(S^j_{i+1})$ denote the set of edges with both ends in S^j_{i+1} of the partial forest *before* path P_{i+1} has been added). Let S be the new moat created at time τ_{i+1} by merging moats S^1_{i+1} and S^2_{i+1}. We have

$$\begin{aligned}
\sum_{e \in E(S)} c_e x_e^{\tau_{i+1}} &= \sum_{j=1}^{2} \sum_{e \in E(S^j_{i+1})} c_e x_e^{\tau_{i+1}} \\
&\leq \sum_{j=1}^{2} \sum_{S' \subseteq S^j_{i+1}} y_S^{\tau_{i+1}} + c(P_{i+1}) - 4\tau_{i+1} \\
&\leq \sum_{j=1}^{2} \sum_{S' \subseteq S^j_{i+1}} y_S^{\tau_{i+1}} - 2\tau_{i+1}
\end{aligned}$$

where the last inequality uses Claim 8. The lemma follows. □

This immediately implies the following corollary.

Corollary 9 ([4])**.** *Algorithm* AKR *computes a* $(2-2/|R|)$*-approximate Steiner tree.*

Proof. As AKR grows at most one moat per terminal at all times $0 \leq \tau \leq \tau^*$, we have

$$\sum_{S \subseteq V} y_S^{\tau^*} \leq |R| \cdot \tau^* \tag{5}$$

and hence we must have

$$\begin{aligned}
\sum_e c_e x_e^{\tau^*} &= \sum_{P \in \mathscr{P}} c(P) \\
&\leq 2 \sum_{S \subseteq V} y_S^{\tau^*} - 2\tau^* \\
&\leq (2 - 2/|R|) \cdot \sum_{S \subseteq V} y_S^{\tau^*}
\end{aligned}$$

where the first inequality follows from Lemma 7 and the second inequality uses (5). The corollary follows from weak duality as y^{τ^*} is feasible for (D_f). □

The above result is easily seen to be tight as (P_f) has an integrality gap of nearly 2 even for the minimum spanning tree setting (where $R = V$): consider a circle with n terminals where each edge has a cost of 1. The cost of an optimum solution to (P^+_{SP}) is $n-1$ while the optimum solution x^* to the LP relaxation (P_f) assigns a value of $1/2$ to each edge and thus has cost $n/2$.

We note that there is a large body of work on alternate relaxations for the Steiner tree problem (e.g., see [73, 101, 132, 133, 162]) that, until very recently, were not known to be stronger than the undirected cut relaxation. Chakrabarty et al. [26] show that the hypergraphic relaxations of [101, 133, 162] are equivalent, and Byrka et al. [22] prove that their integrality gap in general graphs is at most 1.55.

The downside of using hypergraphic LPs is their size: they have an exponential number of constraints, and no compact alternative formulations are known. On the other hand, the *bidirected cut* relaxation [51,

166] is a compact relaxation for the Steiner tree problem, and it is widely conjectured to have a gap smaller than 2. The worst-known integrality gap example shows a factor of $8/7$ and is due to Goemans (see [2]). It is well known that the bidirected cut relaxation is weaker than the hypergraphic relaxations in general. However, in *quasi-bipartite* instances (where no two Steiner nodes are connected by an edge), these relaxations are equivalent [26], and their gap is bounded by 1.28 [22].

2.4 More Applications of the Primal-Dual Method

Goemans and Williamson [74] showed that the above primal-dual algorithm can be adapted to yield a 2-approximation for (IP_f) if f is a $0,1$-*downwards monotone* function, where a cut-requirement function f is downwards monotone if

$$f(B) \leq f(A)$$

for any pair of sets $A, B \subseteq V$ with $A \subseteq B$.

A more advanced application of the primal-dual method is for the *survivable network design* problem (SNDP). In this problem, we are given non-negative integers r_{ij} for each pair of nodes $i, j \in V$, and the goal is to find a minimum-cost subgraph H of G that contains r_{ij} edge-disjoint paths for each pair $i, j \in V$. The central motivation for SNDP lies in fault tolerance issues in the design of communication networks; nodes i and j stay connected in H even if $r_{ij} - 1$ edges are removed (see also [79]). SNDP clearly generalizes the Steiner tree and forest problems discussed in the previous section. It also falls into the f-connectivity framework by letting

$$f(S) = \max_{i \in S, j \notin S} r_{ij} \tag{6}$$

for all $S \subseteq V$ in integer program (IP_f).

The SNDP cut-requirement function given in (6) is easily seen to be proper, where a function f is *proper* if it is symmetric (i.e., if $f(A) = f(V \setminus A)$ for all $A \subseteq V$) and if it satisfies "maximality":

$$f(A \cup B) \leq \max\{f(A), f(B)\}$$

for all pairs of disjoint sets $A, B \subseteq V$. For this class of cut-requirement functions, Goemans and Williamson [75] showed that the primal-dual method from the previous section can be used to obtain a 2-approximation algorithm.

Klein and Ravi [100] were the first to consider non-$0,1$ proper functions, and gave a primal-dual 3-approximation algorithm for (IP_f) where f is a proper function with range $\{0,2\}$. Subsequently, Williamson, Goemans, Mihail and Vazirani [165] presented a primal-dual $2k$-approximation algorithm for general proper functions, where $k = \max_S f(S)$ is the maximum cut requirement. Goemans, Goldberg, Plotkin, Shmoys, Tardos and Williamson [72] presented a further improvement by giving a primal-dual $2H(k)$-approximation algorithm for the problem, where $H(k)$ is the kth harmonic number $1 + 1/2 + \ldots + 1/k$.

The authors of [72] also showed that the same approximation guarantee can be obtained for the larger class of *weakly* or *skew supermodular* cut-requirement functions. A function $f : 2^V \rightarrow \mathbb{N}$ is skew supermodular if $f(\emptyset) = f(V) = 0$, and

$$\begin{aligned} f(A) + f(B) &\leq f(A \setminus B) + f(B \setminus A), \text{ or} \\ f(A) + f(B) &\leq f(A \cap B) + f(A \cup B) \end{aligned}$$

for all $A, B \subseteq V$. In the next section we will see a direct LP rounding algorithm due to Jain that achieves a performance ratio of 2 for this problem.

Finally, we refer the reader to the surveys in [76, 110, 164] for further applications, and pointers.

3 Iterative Rounding: A Polyhedral Approach

In this section, we discuss the technique of *iterative rounding*, a powerful and elegant scheme which was first introduced by Jain in his work on the generalized Steiner network problem in [93]. Extension of this technique (iterative *relaxation* [116]) has more recently lead to breakthrough results in the area of constrained network design, where a number of linear constraints are added to a classical network design problem. Such constraints arise naturally in a wide variety of practical applications, and model limitations in processing power, bandwidth or budget. The design of powerful techniques to deal with these problems is therefore an important goal.

In this section we first show how iterative rounding can be used to obtain an alternative proof of Theorem 6; subsequently, we discuss Jain's work on the survivable network design problem. We conclude this section by showing an application of the technique for a degree-bounded network design problem.

3.1 Spanning trees via iterative rounding

As the key step in the proof of Theorem 6, Chopra showed that the vertices of the feasible region of (P^{+}_{SP}) are identical to those of the following formulation:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e && (P_{SP}) \\ \text{s.t.} \quad & x(E) = n-1 && (7) \\ & x(E(S)) \leq |S|-1 \quad \forall S \subset V && (8) \\ & x \geq 0. \end{aligned}$$

where $n = |V|$ is the number of vertices in G, $E(S)$ denotes the set of edges in E for which both endpoints lie in S, and where we use $x(A)$ as a short-hand for $\sum_{e \in A} x_e$ for $A \subseteq E$. Edmonds [52] showed that (P_{SP}) is a formulation of the convex hull of incidence vector of spanning trees of G. We now provide an alternate proof that is based on iterative rounding; our exposition follows that of Singh and Lau [152].

Our strategy will be as follows: first solve LP (P_{SP}) and obtain an optimal basic feasible solution x^*. We shall show that there is at least one edge $e \in E$ for which $x^*_e = 1$. We add this edge to a partial spanning forest and subsequently obtain a new graph by contracting it. We continue recursively by computing an MST in the resulting contracted graph, and obtain a spanning tree for the original graph by adding in e again. We will be able to show that the cost of the final tree is at most the objective value of x^* and hence it is optimal.

We begin by stating a fundamental structural lemma but defer its proof to a later point in this section.

Lemma 10. *Let G be a connected graph with at least two vertices and let x^* be a basic solution of (P_{SP}) with support $E^* = \{e \in E : x^*_e > 0\}$. There exists a vertex v that is incident to exactly one edge in E^*.*

Let $uv \in E^*$ be the single support edge incident to vertex v described in Lemma 10. It is easy to see that constraints (7) and (8) imply that $x^*_{uv} = 1$. We obtain the following iterative algorithm:

Iterative MST Algorithm:

1: Input: connected graph $G = (V, E)$.
2: $F = \emptyset$
3: **while** $V(G) \neq \emptyset$ and $|V| > 1$ **do**
4: Compute optimum basic solution x^* of (P_{SP}), and remove all edges with $x^*_e = 0$ from G.
5: Find vertex u as in Lemma 10 and let uv be the single support edge incident to it.
6: Add uv to F.

7: Delete u and all incident edges from G.
8: **end while**
9: Return F

Theorem 11. *Given a connected graph $G = (V,E)$ and edge costs c_e for all $e \in E$, the above algorithm correctly computes an MST of G.*

Proof. We will prove a slightly stronger statement by showing that the tree computed by the algorithm has cost at most the optimal value of (P_{SP}). We will use induction on the number of vertices of G.

In the base case where G has at most one vertex, the above algorithm clearly does the right thing by returning the empty set. Now assume that G has at least two vertices. As G is connected, Lemma 10 guarantees that step 5 of the above algorithm will identify a vertex u that is incident to a single support edge uv with $x^*_{uv} = 1$.

As described in the algorithm, we obtain $G' = (V', E')$ from G by deleting vertex u and its incident edges. Let x' be the *projection* of x^* onto the edge set of G'; i.e., $x' \in \mathbb{R}^{E'}$ and $x'_e = x^*_e$ for all $e \in E'$. Observe that x' is feasible for (P_{SP}) for graph G'. Let F' be the tree computed by the algorithm from iteration 2 on. Inductively, it follows that F' is a minimum-cost spanning tree of G' and, as the projection of x^* onto E' is feasible for (P_{SP}), we also must have

$$c(F') \leq \sum_{e \in E'} c_e x^*_e.$$

Finally, $F = F' \cup uv$ is a spanning tree of G and its cost is equal to $c_{uv} + c(F') \leq \sum_{e \in E} c_e x^*_e$ where the inequality uses the fact that $x^*_{uv} = 1$. □

We remark (see [152]) that the projection x' of x^* onto the edge set of G' is a basic feasible solution for (P_{SP}) for graph G'. This means that the algorithm does not need to resolve LP (P_{SP}) in every iteration of the while loop, and may instead just work with a suitable projection of the initial basic feasible solution. Theorem 11 therefore shows that x^* is the incidence vector of a spanning tree and it provides an alternate argument for Edmonds result in [52].

It now remains to give a proof of Lemma 10. Its fundamental techniques are ubiquitous in LP rounding algorithms for network design problems.

Proof of Lemma 10. Let x^* be a basic feasible solution for (P_{SP}) and assume that $x^*_e > 0$ for all $e \in E$. This assumption is w.l.o.g. as we can always delete 0-value edges; x^* projected onto the space defined by the remaining edges is an optimal basic feasible solution for (P^+_{SP}) in the resulting graph. Now let

$$\mathcal{F} = \{S \subseteq V : x^*(E(S)) = |S| - 1\}$$

be the set of all *tight* constraints among (7) and (8). We use χ^A to denote the *characteristic* vector of a set $A \subseteq R$: χ^A has an element χ^A_e for each edge $e \in E$, where $\chi^A_e = 1$ if $e \in A$ and $\chi^A_e = 0$ otherwise. As x^* is a basic solution, it follows that the characteristic vectors of sets in $\mathcal{F}$ span the set

$$\text{span}(\mathcal{F}) = \{x \in R^m_+ : x_e = 0 \text{ for all } e \notin E^*\}$$

where E^* is the support of x^*. We say that two sets $A, B \subseteq V$ *intersect* if $A \setminus B$, $B \setminus A$, and $A \cap B$ are all non-empty. We obtain the following well-known *uncrossing* argument (e.g., see [46, 93, 121]):

Claim 12 ([71])**.** *Let S, T be two intersecting sets in $\mathcal{F}$. Then, both $S \cap T$ and $S \cup T$ are sets in $\mathcal{F}$ and $\chi^{E(S)} + \chi^{E(T)} = \chi^{E(S \cap T)} + \chi^{E(S \cup T)}$.*

Proof. Since $S \cap T \neq \emptyset$ it follows that

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

and hence all inequalities above hold with equality. Thus, $S \cap T$ and $S \cup T$ are in $\mathscr{F}$ and, furthermore, no edges $uv \in E^*$ exist with $u \in S \setminus T$ and $v \in T \setminus S$. This implies that $\chi^{E(S)} + \chi^{E(T)} = \chi^{E(S\cap T)} + \chi^{E(S\cup T)}$. □

This innocuous looking lemma has important structural consequences for bases of span($\mathscr{F}$). In the following we call a family of subsets of V *laminar* if no two of its sets intersect (i.e., for any two sets in a laminar family, either they are disjoint, or one of them is contained within the other).

Corollary 13 ([93])**.** *Let $\mathscr{L}$ be an inclusion-wise maximal laminar family in $\mathscr{F}$, then span$(\mathscr{L})$ = span$(\mathscr{F})$.*

Proof. Assume that the above statement is false and let $\mathscr{L}$ be a maximal laminar family in $\mathscr{F}$ whose span is different from that of $\mathscr{F}$. Then pick a set $S \in \mathscr{F}$ that intersects a minimum number of sets in $\mathscr{L}$. As $\mathscr{L}$ is maximal, there must exist a set $T \in \mathscr{L}$ that intersects S. Claim 12 implies that $S \cap T$ and $S \cup T$ are also in $\mathscr{F}$ and

$$\chi^{E(S)} + \chi^{E(T)} = \chi^{E(S\cap T)} + \chi^{E(S\cup T)}.$$

Therefore, at least one of $S \cap T$ and $S \cup T$ cannot be in span$(\mathscr{L})$ either. Now note that every set in $\mathscr{L}$ that intersects $S \cap T$ or $S \cup T$ must also intersect S and hence both $S \cap T$ and $S \cup T$ intersect fewer sets of $\mathscr{L}$ than S. This contradicts the choice of S. □

Armed with the above corollary we are now ready to finish the proof of Lemma 10. The fact that x^* is basic, the assumption that $x^*_e > 0$ for all $e \in E$, and Corollary 13 imply that there is a maximal laminar family $\mathscr{L}$ in $\mathscr{F}$ such that x^* is the unique solution of

$$x(E(S)) = |S|-1 \quad \forall S \in \mathscr{L}.$$

As $\mathscr{F}$ does not contain singleton sets, it follows that $\mathscr{L}$ has at most $n-1$ sets and this is an upper bound on the rank of the above linear system. Consequently, E^* can have at most $n-1$ non-zeros and this in turn implies that there must be a vertex in V that is incident to exactly one edge of E^*. □

3.2 Advanced iterative rounding: Survivable network design

In this section we return to the survivable network design problem discussed in Section 2.4. In particular, we will see Jain's 2-approximation algorithm for (IP$_f$) for the case where f is skew-supermodular (see [93]). Jain's algorithm iteratively rounds optimal solutions to LP (P$_f$). The main technical lemma whose proof we defer until later is the following:

Lemma 14. *Let x^* be a basic feasible solution to (P$_f$) where f is a skew-supermodular function, and assume that $x^*_e > 0$ for all $e \in E$. Then there is an edge $e \in E$ for which $x^*_e \geq 1/2$.*

The algorithm resembles the iterative rounding algorithm presented in Section 3.1. For a subgraph H of G and a vertex set S, we use $\deg_H(S)$ for the number of edges in H that have exactly one endpoint in S.

Jain's Algorithm:

1: Input: $G = (V,E)$, cost $c_e \geq 0, \forall e \in E$, skew supermodular function f.

2: $H = \emptyset$, $f' = f$
3: **while** $\exists S \subseteq V$ s.t. $f'(S) > 0$ **do**
4: Compute optimum basic solution x^* of $(\mathrm{P}_{f'})$ and remove all edges with $x^*_e = 0$ from G.
5: Find an edge $e \in E$ with $x^*_e \geq 1/2$ and add $\lceil x^*_e \rceil$ copies of e to H.
6: Update f' by letting $f'(S) = f(S) - \deg_H(S)$ for all $S \subseteq V$.
7: **end while**
8: Return H

At this point we remark that it is not known whether LP (P_f) can be solved in polynomial time for arbitrary skew supermodular cut-requirement functions. However, this is possible – via the Ellipsoid method – for general proper functions (e.g., see Theorem 20.19 of [108] for a proof of this fact). In the following, we assume that step 4 of Jain's algorithm can be implemented to run in polynomial time. Modulo this assumption, Lemma 14 yields the following main theorem of [93]:

Theorem 15. *Jain's algorithm achieves a performance guarantee of* 2 *for the survivable network design problem.*

Proof. We will prove a stronger result and show that the algorithm returns a subgraph H of cost at most twice the optimal value of LP (P_f). We use induction on the number of iterations of the algorithm's main loop.

The theorem is trivially true if $f(S) = 0$ for all $S \subseteq V$. If $f(S) > 0$ for at least one $S \subseteq V$, then Lemma 14 guarantees that there is at least one edge $e \in E$ with $x^*_e \geq 1/2$. Let f' be the residual cut requirement function obtained by the algorithm, i.e.,

$$f'(S) = \begin{cases} f(S) - \lceil x^*_e \rceil : e \in \delta(S) \\ f(S) \qquad\quad : \text{otherwise} \end{cases}$$

for all $S \subseteq V$. The reader verifies that function f' is skew-supermodular if f is. Furthermore, it is not hard to see that the projection x' of x^* onto $E' = E \setminus \{e^*\}$ is feasible for $(\mathrm{P}_{f'})$ with the above residual cut requirement function. Inductively, we can therefore conclude that the algorithm finds a subgraph H' of G that is feasible for f' and has cost at most $2\sum_{e' \in E'} c_e x^*_e$. We obtain H from H' by adding e^*. The resulting subgraph is feasible for f and its cost is

$$c(H) = c_{e^*} + c(H') \leq 2 \cdot c_e x^*_{e^*} + 2 \cdot \sum_{e \in E'} c_e x^*_e \leq 2 \cdot \sum_{e \in E} c_e x^*_e.$$

□

We now prove Lemma 14. Our exposition follows a recent argument due to Nagarajan, Ravi and Singh [127].

Proof of Lemma 14. The strategy pursued in this proof resembles that used in the proof of Lemma 10. Let x^* be the positive basic feasible solution of (P_f), and let

$$\mathscr{F} = \{S \subseteq V \,:\, x^*(\delta(S)) = f(S)\}$$

be the set of tight inequalities for the given basic solution. Using an argument similar to that of Claim 12 and Corollary 13 together with the skew supermodularity of f, Jain [93] showed that there is a laminar family $\mathscr{L} \subseteq \mathscr{F}$ such that x^* is uniquely defined by the system

$$x(\delta(S)) = f(S) \quad \forall S \in \mathscr{L}. \tag{9}$$

In the following we assume that the rank of the above system is exactly $|\mathscr{L}|$.

Let us assume, for the sake of contradiction, that $0 < x^*_e < 1/2$ for all $e \in E$. We will use a counting argument to arrive at a contradiction. To start, we assign one *token* to each edge $e \in E$, and we redistribute these tokens over the sets in $\mathscr{L}$ so that every set obtains at least one token, and at least one set obtains strictly more than a token. This would show that the number of positive variables x^*_e is larger than $\mathscr{L}$, and hence larger than the rank of the system in (9); this is, of course, a contradiction.

The token of edge $uv \in E$ is reassigned as follows. We give x^*_{uv} tokens each to the inclusion-wise smallest sets in $\mathscr{L}$ that contain u and v. Furthermore, we give $1 - 2x^*_{uv}$ tokens to the inclusion-wise smallest set in $\mathscr{L}$ that contains both u and v. Note that, by assumption, $0 < x^*_{uv} < 1/2$, and hence all three sets in the above redistribution receive a positive amount of tokens.

We proceed by showing that each set in $\mathscr{L}$ receives at least one token. Consider one such set S, and let $T_1, \ldots, T_p$ be its children in $\mathscr{L}$; i.e., $T_1, \ldots, T_p$ are the inclusion-wise maximal proper subsets of S in $\mathscr{L}$. The set of edges that contribute tokens to S can be partitioned into sets $A, B, C \subseteq E$.

[A] Edges e with one endpoint in $S \setminus (T_1 \cup \ldots \cup T_p)$, and the other in $V \setminus S$ contribute x^*_e tokens to S.

[B] Edges e with one end in $S \setminus (T_1 \cup \ldots \cup T_p)$, and the other in $T_1 \cup \ldots \cup T_p$ contribute $1 - x^*_e$ tokens to S.

[C] Edges e with one end in T_i and the other in T_j for some $1 \leq i < j \leq p$ contribute $1 - 2x^*_e$ to S.

Thus S receives

$$x^*(A) + (|B| - x^*(B)) + (|C| - 2x^*(C)) = |B| + |C| + x^*(\delta(S)) - \sum_{i=1}^{p} x^*(\delta(T_i))$$

tokens in total. Using the tightness of the constraints for sets $S, T_1, \ldots, T_p$ in (P_f), we can rewrite the number of tokens of S as

$$|B| + |C| + f(S) - \sum_{i=1}^{p} f(T_i). \tag{10}$$

Note that each edge $e \in A \cup B \cup C$ contributes a positive amount of tokens to S, and the expression in (10) is therefore non-negative. Furthermore, it is 0 only if $A = B = C = \emptyset$. But in this case, we observe that

$$\chi^{\delta(S)} = \chi^{\delta(T_1)} + \ldots + \chi^{\delta(T_p)},$$

which contradicts the fact that the system of equalities in (9) has full rank. Thus S obtains a positive amount of tokens, and since (10) is integral it follows that S receives at least one token.

Finally consider an inclusion-wise maximal set $S \in \mathscr{L}$. As the constraint for S in (P_f) is tight, there must be at least one edge $e \in \delta(S)$ with $x^*_e > 0$. Observe that $\mathscr{L}$ has no set containing both endpoints of e, and hence there are $1 - 2x^*_e > 0$ tokens of edge e that have not yet been redistributed. We conclude that the number of edges is bigger than $|\mathscr{L}|$, and this is the desired contradiction. □

3.3 Advanced iterative rounding: Degree-bounded network design

In this section we study degree-bounded variants of the f-connectivity and survivable network design problems introduced so far. Adding degree constraints to network design problems is natural and models communication- as well as hardware constraints in physical networks. Initial work on these problems mostly focused on uniform cost (i.e., unweighted) network design problems. Very recently, several breakthrough results have been obtained for weighted instances as well using beautiful extensions of the iterative rounding paradigm discussed above. Once again, we will use the spanning trees as a canonical example to illustrate the essential ideas.

We begin with the *minimum-degree spanning tree problem* (MDST). In this problem, we are given a connected unweighted graph $G = (V, E)$, and the goal is to find a spanning tree of smallest maximum degree. We

first note that this problem, unlike the minimum-spanning tree problem, is NP-hard; an undirected graph has a Hamiltonian path iff it has a spanning tree with maximum degree at most 2. Fürer and Raghavachari [65] were the first to consider the problem, and gave an $O(\log |V|)$ approximation algorithm for it, as well as for the related problem of finding an arborescence of smallest maximum out-degree in a directed graph. Subsequently, Fürer and Raghavachari improved upon their previous result and showed how to compute a spanning tree that has degree within an *additive one* of the minimum possible degree [66]. The algorithm in [66] and its performance guarantees extend to the minimum-degree Steiner tree problem as well. We refer the reader to [138] for an excellent survey discussing the above and related algorithms.

The *minimum-cost degree-bounded spanning tree* problem (BMST) is a natural generalization of MDST to weighted graphs. In this problem, we are given a non-negative integer parameter B_v for each vertex $v \in V$, and the goal is to find a minimum-cost tree T such that $\deg_T(v) \leq B_v$ for all vertices v. Let T^* be an optimum solution for an instance of this problem. Given Fürer and Raghavachari's algorithm for MDST, it is natural to ask whether one can find an algorithm that computes a spanning tree T of cost at most $c(T^*)$, and $\deg_T(v) \leq B_v + 1$ for all vertices v. After much work on the topic (e.g., see [30, 31, 71, 102, 103, 143, 146]), this question was recently resolved by Singh and Lau [152] via an elegant adaptation of Jain's iterative rounding technique. We give a simplified presentation of their algorithm that is due to Bansal, Khandekar and Nagarajan [13].

Just like in Jain's algorithm, Singh and Lau propose a natural linear programming relaxation for BMST, and their algorithms produces an integral solution via an iterative rounding process. The new ingredient in their algorithm concerns the treatment of degree constraints. Singh and Lau observed that, whenever the number of fractional (*support*) edges incident to a vertex becomes small, then the corresponding degree constraint can be dropped. The reason is simple: no matter what the algorithm does in subsequent iterations, in the worst case, it decides to include all support edges; we drop a constraint if these are few. This refinement of Jain's iterative rounding technique is commonly referred to as *iterative relaxation* and it first appeared in [116].

In a bit more detail, Singh and Lau's algorithm (`SL`) maintains a set F of edges that have already been included, and a set W of remaining degree constraints. Given these two sets, `SL` repeatedly solves the following linear programming relaxation, where we use E' for the set of remaining (non-dropped) edges in $E \setminus F$, and $\deg_F(v)$ denotes the number of edges in F that are incident to v.

$$\begin{aligned}
\min \quad & \sum_{e \in E'} c_e x_e && (\mathrm{P}_{BMST}) \\
\text{s.t.} \quad & x(E') = n - |F(V)| - 1 && (11) \\
& x(E(S)) \leq |S| - |F(S)| - 1 \quad \forall S \subset V && (12) \\
& x(\delta(v)) \leq B_v - \deg_F(v) \quad \forall v \in W && (13) \\
& x \geq 0.
\end{aligned}$$

Constraints (11) and (12) are familiar from the spanning tree application in Section 3.1; (13) captures the new degree constraints. The following lemma states the key structural property needed buy the algorithm.

Lemma 16. *Let x^* be a basic feasible solution for (P_{BMST}). Then either $x^*_e \in \{0,1\}$ for some $e \in E'$, or there is a vertex $v \in W$ with*

$$\deg_{E'}(v) \leq B_v - \deg_F(v) + 1. \tag{14}$$

Before we prove Lemma 16, let us describe the algorithm. Given F and W, algorithm `SL` first solves (P_{BMST}); let x^* be the solution obtained. If there is an edge e with $x^*_e = 0$, then e is simply removed from E', and we iterate. Similarly, if there is an edge e with $x^*_e = 1$, we include e into F, and delete it from E'. Finally, if none of the above two cases holds, Lemma 16 implies that there must be a vertex $v \in W$ with few incident

support edges; we drop this vertex from W and continue. The proof of the following main theorem of [152] is similar to that of Theorem 15, and we omit it here.

Theorem 17. *Let T^* be an optimum solution for the given BMST instance. Algorithm* SL *computes a spanning tree T of cost at most $c(T^*)$, and $\deg_T(v) \leq B_v + 1$ for all $v \in W$.*

In the following sketch of the proof Lemma 16 we follow [13].

Proof-Sketch of Lemma 16. Let us assume that $0 < x^*_e < 1$ for all $e \in E'$; we are done, otherwise. An uncrossing argument, similar to that used in the proofs of Claim 12 and Corollary 13 implies that there is a laminar family $\mathscr{L}$ of vertex sets, and a set $T \subseteq W$ such that x^* is the unique solution to the system

$$x(E'(S)) = |S| - |F(S)| - 1 \quad \forall S \in \mathscr{L} \tag{15}$$

$$x(\delta(v)) = B_v - |F \cap \delta(v)| \quad \forall v \in W \tag{16}$$

Furthermore, the characteristic vectors $\{\chi^{E(S)}\}_{S \in \mathscr{L}}$ and $\{\chi^{\delta(v)}\}_{v \in T}$ are linear independent, and the size of the support is $|E| = |\mathscr{L}| + |T|$. Now define the *spare* of vertex $v \in W$ as

$$\text{spare}(v) = \deg_{E'}(v) - x^*(\delta(v)).$$

The following is the central technical claim needed for this proof.

Claim 18 ([13])**.** $\sum_{v \in W} \text{spare}(v) < 2|W|$

Let us first see how this claim implies the existence of a vertex v with small support. The lemma shows that there is a vertex $v \in W$ with $\text{spare}(v) < 2$. We obtain

$$\deg_{E'}(v) < 2 + x^*(\delta(v)) \leq 2 + B_v - \deg_F(v),$$

where the second inequality uses the feasibility of x^* for (P_{BMST}). Note that the left- and right-hand sides of the latter inequality are integral, and hence (14) is satisfied for v.

In order to see Claim 18, observe that each edge in E' contributes to the spare of at most two vertices of W, and hence

$$\sum_{v \in W} \text{spare}(v) \leq 2 \sum_{e \in E'} (1 - x^*_e) = 2(|E'| - x^*(E')) = 2(|\mathscr{L}| + |T| - x^*(E')),$$

where we have used the assumption that $0 < x^*_e < 1$ for all e, and the fact that the system in (15) and (16) is non-singular. A counting argument shows that $|\mathscr{L}| \leq x^*(E')$ (see [13]), and hence

$$\sum_{v \in W} \text{spare}(v) \leq 2|T| \leq 2|W|.$$

Finally, Bansal et al. show that the non-singularity of the system in (15) and (16) implies that the above inequality holds strictly. □

In [13], Bansal et al. applied the above spare argument to the more general *minimum crossing spanning tree* problem (MCSP). In this problem, we are given edge subsets $E_1, \ldots, E_k$ and integers $B_1, \ldots, B_k$, and the goal is to find a spanning tree T of smallest cost that has at most B_i edges from E_i for all i. For the case where each edge $e \in E$ is in at most r of the given sets E_i, Bansal et al. presented an algorithm that returns a tree of at most optimum cost, that has at most $B_i + r - 1$ edges from E_i for all $1 \leq i \leq k$. An improvement for the structured special case where the sets $E_1, \ldots E_k$ are induced by a laminar system of cuts was given in [12]. Checkuri et al. [37] recently presented a very elegant randomized technique for rounding a fractional point in a matroid polytope to an integral one. The authors derive concentration bounds on linear functions

of variables for the elements of the underlying matroid. In the special case of MCSP, this yields a tree of at most optimum (expected) cost with $(1+\varepsilon)B_i + O(\frac{1}{\varepsilon}\log n)$ edges from set E_i for all i.

Lau et al. [116] were the first to apply iterative rounding ideas to the more general degree-bounded survivable network design problem (BSNDP). The authors showed how to find a subgraph H satisfying all connectivity constraints whose cost is at most twice that of an optimum solution. Moreover, the degree of each vertex v is at most $2B_v + 3$.

The multiplicative degree-guarantees obtained in [116] were recently improved by Lau and Singh [117] who presented algorithms for degree-bounded survivable network design problems with additive degree guarantees. The authors presented an algorithm that computes a 2-approximate solution where the degree of each vertex is violated by at most $6r_{max} + 3$, where r_{max} is the maximum requirement across any cut. In the special case of Steiner forests, this can be improved to 3.

While we have focused on undirected problems in this section, we emphasize that similar progress has been made for the directed counterparts of the problems considered. In [116], Lau et al. consider degree-bounded problems in directed graphs where the connectivity requirements are given by *crossing supermodular* cut requirement functions (see also [62]), and where the in- and out-degree for each vertex v is constrained to be at most a_v and b_v, respectively. For this case, the authors showed how to compute a feasible solution of cost at most 3 times the optimum whose in-degree at v is at most $3a_v + 4$, and whose out-degree at v is at most $3b_v + 4$, for all $v \in V$. Subsequently, Bansal et al. [13] presented several further improvements of the directed results given in [116]. The authors also demonstrated how their techniques can be used to compute arborescences whose out-degree at each vertex v is at most $B_v + 2$.

3.4 Other applications of iterative rounding

The survivable network design problem, the way we introduced it in Section 2.4, is concerned with establishing pre-specified number of edge-disjoint paths between any pair of vertices i and j at a small total cost. It is natural to ask a similar question in the vertex-connectivity setting: given integer parameters r_{ij} for each pair of vertices $i, j \in V$, find a minimum-cost subgraph H of G such that H has at least r_{ij} internally vertex-disjoint paths between any pair of vertices $i, j \in V$. (Here, *internally* vertex-disjoint means the paths only share the endpoints i and j, and no other vertices.)

Ravi and Williamson [147, 148] describe a primal-dual 3-approximation for this problem when $r_{ij} \in \{0,1,2\}$ for all $i, j \in V$. Subsequently, Fleischer [58] presented an IP formulation for the problem and used iterative rounding to obtain a 2-approximation algorithm for this special case. Kortsarz et al. [109] showed that no polynomial-time approximation algorithm with performance guarantee better than $2^{\log^{1-\varepsilon} n}$ exists for any $\varepsilon > 0$ unless NP $\subseteq$ DTIME$(n^{polylog(n)})$. Recently, Chuzhoy and Khanna [44] gave an $O(k^3 \log n)$-approximation for this problem, which we shall discuss in Section 5.2, via a reduction to the element-connectivity problem we talk about next.

The *element-connectivity* problem was introduced by Jain et al. [94] as a problem which is intermediate in difficulty between the edge- and vertex-connectivity versions described above. Here, the vertex set V of graph G is partitioned into terminals and non-terminals. Edges and non-terminals are called elements. For each pair of terminals i, j we are given a parameter r_{ij} and we require r_{ij} element-disjoint paths between them. Jain et al. [94] presented a $2H(k)$ approximation algorithm where k is the largest value of r_{ij} over all i, j. Fleischer et al. [59] then presented a 2-approximation algorithm for the problem, once again using iterative rounding, and matching the approximation ratio for the edge-connectivity case.

4 Approximate Packing Results

The network design problems considered so far aim at selecting a minimum-cost subgraph of a given weighted undirected graph that satisfies certain prescribed connectivity properties. This section focuses

on packing problems which are, in a sense, *dual* to these problems. The arc-version of Menger's classical theorem serves as a canonical example:

Theorem 19 (Menger [123])**.** *A directed graph $G = (V,E)$ has k arc-disjoint s,t-paths for some $s,t \in V$ if and only if any s,t-cut has at least k arcs.*

Menger's theorem characterizes the class of graphs in which we can *pack* k arc-disjoint s,t-paths; the theorem provides a succinct certificate for the existence and non-existence of such paths. We will see that similar *min-max* type statements exist for the case where we pack spanning trees instead of paths, but no such exact characterization exists for the packing variants of the Steiner tree and f-connectivity problems.

In this section we will be interested in the implications of min-max characterizations of packing problems for network design problems. Packing is an important and active area of research that has applications in VLSI circuit design (e.g., see [77, 78, 107]), in computer networking (e.g., see [20, 38, 39]), and in many other domains. Research in this area has been very active throughout the last decade and before, and a comprehensive account is beyond the scope of this survey. We refer the reader to the survey by Frank [61], and to Lau's recent thesis [114] for more information.

4.1 Packing spanning trees

In their ground-breaking work Nash-Williams [128] and Tutte [159] provided the following beautiful generalization of Menger's theorem.

Theorem 20. *There exist k edge-disjoint spanning trees in an undirected graph $G = (V,E)$ if and only if $|E_\pi| \geq k \cdot (r(\pi) - 1)$ for all partitions $\pi \in \Pi$.*

The theorem immediately yields a short alternative proof of integrality of (P^+_{SP}). Let x^* be an optimum basic feasible solution for the LP and let k be smallest such that $\bar{x} = kx^*$ is integral. Note that the existence of an optimal solution implies that all edges costs are non-negative. As x^* is feasible for (P^+_{SP}) it follows that

$$\sum_{e \in E_\pi} \bar{x}_e \geq k(r(\pi) - 1)$$

for all $\pi \in \Pi$. Let H be a graph with vertex set V and $\bar{x}_e$ parallel *copies* of each edge $e \in E$. By Theorem 20, H has k edge-disjoint spanning trees $T_1, \ldots, T_k$. Let x^i be the incidence vector of tree T_i for $1 \leq i \leq k$, then

$$\sum_{i=1}^{k} x^i \leq kx^*, \tag{17}$$

and hence kx^* *dominates* the sum of k integer solutions. Polyhedra whose extreme points satisfy (17) are said to have the *integer decomposition property*. We refer the reader to [151] for more information on this well-studied property.

An immediate consequence of (17) is that there is an $1 \leq i \leq k$ such that $c(T^i) \leq c^T x^*$, and the optimality of x^* implies that the inequality as well as that in (17) must hold with equality. Hence, we showed that x^* is a convex combination of minimum-cost spanning trees.

4.2 Advanced Packing Theorems

Unless NP=co-NP, Nash-Williams' and Tutte's result in Theorem 20 cannot be extended to the Steiner tree and the more general f-connectivity setting. Nevertheless, Kriesell [111] conjectured that a graph G has k edge-disjoint Steiner trees if every Steiner cut is crossed by at least $2k$ edges. While this conjecture remains open, some partial progress has been made. In particular, Frank, Király, and Kriesell [63] showed that

a given quasi-bipartite graph G has k edge-disjoint Steiner trees if the terminal set is $3k$-edge-connected. Kriesell [111] showed that the conjecture in general Eulerian graphs G. For a general undirected graph G, Lau [115] recently proved that G has k edge-disjoint Steiner trees if every Steiner cut has at least $26k$ edges; in his thesis ([114], Theorem 3.1.2) he slightly improves this bound to $24k$.

It is natural to ask whether Kriesell's conjecture extends to more general network design problems. For example, given a Steiner forest instance such that each of the pairs is connected by k edge-disjoint paths, does the graph have k edge-disjoint Steiner forests? Chekuri and Shepherd [36] pose this conjecture for the more general $0,1$-f-connectivity problem.

Conjecture 1 ([36]). *Let G be an undirected graph, and f a $0,1$-skew-supermodular function. If x^* is a feasible solution to (P_f) such that kx^* is integral, there exist f-connected integer vectors $x^1,\ldots x^k$ such that $\sum_{i=1}^k x^i \leq kx$.*

In fact, Chekuri and Shepherd remark that there is no reason why this result should not also be true for general (non-$0,1$) skew-supermodular functions. Using Nash-Williams and Tutte's theorem for spanning trees as well as Mader's well-known *splitting-off* technique [119], the authors arrive at the following main decomposition theorem.

Theorem 21. *Let G be an undirected graph and let f be a $0,1$-valued downwards monotone or proper function. Further let x^* be a feasible solution of (P_f) and let $k \geq 1$ be an integer such that kx^* is integral. Then there are solutions $x_1,\ldots,x_k$ to (IP_f) such that*

$$\sum_{i=1}^{k} x_i \leq 2k \cdot x^*.$$

The theorem provides an alternate 2-approximation algorithms for the f-connectivity problem where f is a $0,1$ proper or downwards monotone function. Extending his work on Steiner trees, Lau [113] recently showed that Conjecture 1 is approximately true for the Steiner forest problem: if each of the given terminal pairs is $32k$-edge-connected, then G has k edge-disjoint Steiner forests.

5 Randomization

Randomization has long been a widely used technique in approximation algorithms: perhaps the most common use has been in randomized rounding of linear programming relaxations, but often randomization is a useful tool in its own right. In this section, we show a randomized rounding for minimum spanning tree, and then give two recent examples of the use of randomization in non-rounding contexts.

5.1 Randomized Rounding for Minimum Spanning Tree

The general idea in randomized rounding is to take an integer programming (IP) formulation of the problem at hand, solve a convex programming relaxation (CP) of this formulation, and then convert the resulting fractional solution to (CP) into an integral solution for (IP) by randomly rounding the variables. The goal is to do this so that the cost of the integral solution thus obtained is not much more than that of the fractional solution to the convex program.

In this section, we show how to randomly round the linear programming relaxation (P_{SP}) of the minimum spanning tree problem. In particular, we give a proof due to Alon [5] that, given *any* feasible solution to the minimum spanning tree LP relaxation with cost $Z^* = \sum_e c_e x_e$, shows that there is a randomized rounding procedure to obtain a tree with expected cost at most $O(Z^* \log n)$. The procedure is simple:

Random-Round-MST

$i \leftarrow 0$
repeat
  $i \leftarrow i+1$
  Let A_i be the set of edges obtained by picking each edge e independently with probability x_e.
until $G_i = (V, \cup_{j \le i} A_j)$ is connected.
return any spanning tree of G_i.

Lemma 22. *Given any (multi)graph $H = (U, E')$ and a feasible solution $\{x_e\}_{e \in E'}$ to (P_{SP}) for this graph, suppose we pick each edge with probability x_e independently. Then the number of components is at most $0.9|U|$ with probability at least $1/2$.*

Proof. We call a vertex u isolated if no edge incident to u is picked by the random process: the probability that u is isolated is $\prod_{e \in \delta(\{u\})}(1-x_e) \le e^{-\sum_{e \in \delta(\{u\})} x_e} \le 1/e$. By linearity of expectation, the number of isolated vertices is at most $|U|/e$, and Markov's inequality implies that we have at most $2|U|/e$ isolated vertices with probability at least $1/2$. Since all the other non-isolated vertices must be in components of size at least 2, the total number of components is at most $|U|(2/e + 1/2(1-2/e)) < 0.9|U|$ with probability at least $1/2$. □

Theorem 23. *The expected cost of the tree returned by MST-Rand-Round is $O(\log n)$ times the cost of the LP solution Z^*.*

Proof. We show that the expected number of rounds before the procedure stops is $O(\log n)$. Let $L = 20 \log n$, and recall that $G_i = (V, \cup_{j \le i} A_j)$. We claim that the graph G_L is disconnected with probability at most $1/2$; conditioned on it being disconnected, an identical argument shows that G_{2L} will be disconnected with probability at most $1/2$ etc., and hence the expected number of rounds will be at most $2L$.

To prove that G_L is connected with probability at least $1/2$, define C_i to be the number of components of the graph G_i, and define index i to be *successful* if either G_{i-1} is connected (i.e., if $C_{i-1} = 1$) or if $C_i \le 0.9 \cdot C_{i-1}$ (if the number of components decreases by a constant fraction due to the edges in A_i). The crucial claim is that the probability of i being successful is at least $1/2$ *regardless of all random choices made in previous rounds*. The proof of this claim is simple: if G_{i-1} is connected then i is always successful. Else, let H_{i-1} be the graph on C_{i-1} vertices obtained by contracting all edges in G_{i-1}. Since each cut in H_{i-1} corresponds to a cut in G_{i-1}, we have that $\sum_{e \in \delta(S)} x_e \ge 1$ holds for H_{i-1}; now Lemma 22 ensures that the number of components after another round of randomly adding edges will cause $C_i \le 0.9 \cdot C_{i-1}$ with probability at least $1/2$.

Given the above claims, the probability that the graph G_L is not connected is bounded above by the probability that a sequence of L independent unbiased coin-flips contains fewer than $\log n / \log 0.9$ heads. Note that while the events in the random rounding process are not independent, we proved a lower bound of $1/2$ on the probability of success *regardless of the history*. Since $L = 20 \log n$, the probability we see fewer than $\log n / \log 0.9 < 10 \log n$ heads (and hence the probability that G_L is not connected) is at most $1/2$, which completes the proof. □

5.1.1 Advanced Applications of Randomized Rounding

Randomized rounding and derandomization techniques have been widely used since the results of Raghavan and Thompson [140, 139] for the low-congestion routing problems, and even surveying the wealth of ideas developed in this area is beyond the scope of this survey (see, e.g., the texts [125, 126] or the surveys [150, 153]). Apart from the idea of independent rounding used above, techniques have been developed to round the fractional solutions in more advanced ways: e.g., in some cases randomized rounding does not give the desired solution, but the blemishes can be fixed by alterations (see, e.g., [7, 23, 25, 155]; in other cases, we might need to round the variables in some dependent fashion (see, e.g., [3, 67, 154]). Moreover, a variety

of probabilistic bounds, such as Chernoff-Hoeffding bounds, the Lovász Local Lemma, Janson's inequality, and the FKG inequality are often used to bound the probability of failure: see, e.g., [7] for definitions, and the books and surveys above for applications.

5.2 Two Other Randomized Algorithms

A recent paper of Chuzhoy and Khanna [44] used a simple randomized argument to approximate the vertex-connectivity problem using algorithms for element-connectivity problem (see Section 3.4 for definitions; recall that $k = \max_{ij} r_{ij}$ is the maximum element-connectivity requirement for the instance). Their argument is the following: let $V_1, \ldots, V_L$ be independent random samples of the vertex set V, where for any ℓ, each vertex in V is added to V_ℓ independently with probability $p = 1/k$. For each $\ell \in [L]$, define an element-connectivity instance $\mathscr{I}_\ell$ on the original graph G where $r^\ell_{ij} = r_{ij}$ if both $i, j \in V_\ell$, and $r^\ell_{ij} = 0$ otherwise. Clearly, the optimal vertex-connected solution is feasible for all these element-connectivity instances; the claim is that the union of the solutions to these instances is a solution to the original vertex-connectivity instance with high probability. The proof is simple: given i, j and any set $S \subseteq V \setminus \{i, j\}$ of size k, the probability that at least one $\ell \in [L]$ has both $i, j \in V_\ell$ and also $S \cap V_\ell = \emptyset$ is $1 - [1 - p^2(1-p)^k]^L = 1/2n^k$ if $L = \Omega(k^3 \log n)$. For that choice of ℓ, the set S is not a vertex cut for i, j in the element-connected solution for $\mathscr{I}_\ell$. By a trivial union bound over all $n^2\binom{n-2}{k}$ choices of i, j, S, we have that the union of these random element-connectivity solutions is solution to the original vertex-connectivity instance with probability at least half. As discussed in Section 3.4, each element-connectivity instance is approximable to within a factor of 2, which implies a $2L = O(k^3 \log n)$ approximation to the vertex-connectivity problem. Note that for the single-source case of vertex-connectivity, the above argument can be improved to $L = O(k^2 \log n)$, which matches the best result known for this case via other involved arguments [43, 130, 131].

A different use of randomization appears in an algorithm for the *connected facility location* problem: here the input is a metric space (V, d) along with a set of clients $D \subseteq V$, and an integer $M \geq 1$. The goal is to designate a set $F \subseteq V$ of the vertices as "facilities", to assign each client $j \in D$ to its closest facility $i(j) \in F$, and to build a Steiner tree $T = (V_T, E_T)$ connecting all the facilities. The objective function is $\sum_{j \in D} d(j, i(j)) + M \sum_{e=uv \in E_T} d(u, v)$. Several algorithms have been proposed for this problem, e.g., based on rounding linear programming relaxations [84, 145], the primal-dual method [156], and reductions to uncapacitated facility location [97]. Subsequently a simple randomized algorithm was proposed [85]: randomly pick each client location in D to also be in F independently with probability $1/M$, assign each client to its closest facility in F, and build an ρ_{ST}-approximate Steiner tree on F (using, say, the $\rho_{ST} = \ln 4$-approximation algorithm of Byrka et al. [22]). This algorithm gives a $2 + \rho_{ST} \approx 3.39$-approximation to the connected facility location problem. This random sampling technique has since been used in approximation algorithms for other problems in network design (see, e.g., [53] and the references therein), and for approximation algorithms for stochastic optimization problems [87, 90].

6 Metric Embedding Techniques

Metric embedding techniques have been very widely used in recent years in network design, and one of the most popular results has been that of embedding a general metric space into distributions over tree metrics. The main idea is simple: given any metric space, one can randomly generate a tree metric such that distances in the original metric space are closely approximated by the expected distances in this random tree; Hence, this allows us to randomly reduce many optimization problems on general metrics to their counterparts on tree metrics, which are often much simpler to solve. In this section, we give the main result on which this technique is based, and illustrate it by giving an approximation algorithm for the buy-at-bulk problem.

6.1 Embeddings into Distributions of Trees

For any graph $G = (V,E)$ with edge-lengths w_e, we define d_G to be the induced shortest-path metric; i.e., $d_G(x,y)$ is the length of the shortest-path between x and y according to these edge lengths w_e, which can be found using, say, an all-pairs-shortest-paths computation. Given a metric space (V,d), consider a tree $T = (V,E')$ with edge-lengths $w'_{e'}$; note that since there is a unique path between any pair of nodes in T, the computation of the shortest-path distances d_T in the tree T is especially simple. Such a tree T is called a *dominating tree* for the metric (V,d) if $d_T(x,y) \geq d(x,y)$ for all $x,y \in V$. We define $\mathsf{DT}(V,d)$ to be the set of all dominating trees defined on the vertex set V.

Definition 24. *A metric space (V,d) α-*probabilistically embeds into dominating tree metrics *if there is a probability distribution $\mathscr{D}$ defined on $\mathsf{DT}(V,d)$ such that*

$$\mathbf{E}_{T \in \mathscr{D}}[d_T(x,y)] \leq \alpha \cdot d(x,y) \tag{18}$$

for all $x,y \in V$. The embedding is efficient *if there is a polynomial-time algorithm that returns a sample from the distribution $\mathscr{D}$.*

Theorem 25. *Any metric space (V,d) with $|V| = n$ points α-probabilistically embeds into dominating tree metrics with $\alpha = O(\log n)$; moreover, this embedding is efficient.*

Theorem 25 was proved by Fakcharoenphol et al. [55], building on a series of important papers [6, 15, 16, 28, 29]. The procedure of [55] outputs random trees on a larger set of vertices $V' \supseteq V$, but it is simple to remove the "Steiner" nodes in $V' \setminus V$ using ideas from [81, 104] whilst changing distances only by a constant factor. For some applications, however, it is more convenient to directly use the trees output by the procedure of [55] due to their "hierarchical well-separated" structure, despite the presence of these Steiner nodes (see, e.g., [17]).

In fact, there is a deterministic polynomial-time algorithm that outputs a set of $O(n \log n)$ trees from $\mathsf{DT}(V,d)$ such that the uniform distribution on this set satisfies Theorem 25 (see [29]). The value $\alpha = O(\log n)$ in Theorem 25 is the best possible for general n-point metrics: lower bounds of $\Omega(\log n)$ can be proved even for "simple" graphs like the square grid [6] and treewidth-2 graphs [86].

6.2 An Application: Approximating Buy-at-Bulk Network Design

Theorem 25 can be used to give a randomized $O(\log n)$-approximation algorithm to the minimum spanning tree problem on any graph: i.e., given any edge-weighed graph $G = (V,E)$, the algorithm outputs a random spanning tree of G whose expected cost is at most $O(\log n)$ the cost of the MST. If G is a complete graph with metric edge-weights, this algorithm simply applies Theorem 25 and returns a random tree drawn from the distribution. If $G = (V,E)$ is not a complete graph, this random tree returned may contain edges not in E, and one needs to replace these edges by paths in the underlying graph. Let us give this algorithm and its proof in the context of a more general problem, the buy-at-bulk network design problem.

The *(uniform) buy-at-bulk* problem is the following: the input consists of a graph $G = (V,E)$ with edge lengths w_e, and a set of k terminal pairs $(s_i,t_i) \in V \times V$ for $i \in \{1,2,\ldots,k\}$, and a concave function $g : \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that the cost of routing t paths on an edge e incurs a cost of $w_e \cdot g(t)$. The goal is to find a path P_i between s_i and t_i for each i: if t_e is the number of paths using edge e, the *(uniform) buy-at-bulk* objective function seeks to minimize the total cost $\sum_{e \in E} w_e \cdot g(t_e)$. (The "uniform" indicates that the cost functions for different edges are all multiples of the same function $g(\cdot)$, and hence proportional to each other—see Section 7.2 for a discussion on the "non-uniform" problem.)

Note that the concave cost function causes us to prefer solutions where many paths use the same edges. The problem is NP-hard (since the convex function $g(t) = \mathbf{1}_{(t \geq 1)}$ captures the Steiner forest problem), and the best approximation guarantee known for the problem is via probabilistic embeddings into trees.

Theorem 26 ([11]). *The (uniform) buy-at-bulk problem admits a randomized algorithm that returns a solution with expected cost at most $O(\log n)$ times the optimum.*

Proof. Most approximation algorithms using probabilistic embeddings into trees have proofs that resemble this one: we first show that given any solution on the graph G of cost Opt, there is a solution on the random tree T of "low" expected cost $\mathsf{Opt} \cdot O(\log n)$. Of course, we then find the best solution on T, which costs no more, and then show that any solution on T can be translated back to the graph G without an increase in cost, thus completing the proof.

The case of the buy-at-bulk problem is trivial if the input graph G is a tree: choosing the unique shortest paths between terminals is the optimal solution in this case. To solve the problem when G is not a tree, apply Theorem 25 to the shortest-path metric (V, d_G) to obtain a random dominating tree $T = (V, E')$ with edge-lengths $w'_{e'}$ such that the expected distance $d_T(x,y)$ in the tree is at most $O(\log n) \cdot d_G(x,y)$ for all $x, y \in V$. For any edge $e' = uv \in E'$, fix a shortest u,v-path $Q_{e'} = Q_{uv}$. Now, for any terminal pair (s_i, t_i), consider the unique path $\langle s_i, u_1, u_2, \ldots, u_l, t_i \rangle$ in the tree T, and define the path P_i to be the concatenation of the paths $Q_{s_i u_1}, Q_{u_1 u_2}, \ldots, Q_{u_{l-1} u_l}, Q_{u_l t_i}$.

We claim the cost of the final solution on G is at most the cost of the solution on the tree. Indeed, let X_e be the set of pairs whose paths use the graph edge $e \in E$, and let $Y_{e'}$ be the set of terminal pairs whose paths use the tree edge $e' \in E'$: the final cost is

$$\sum_{e \in E} w_e \cdot g(|X_e|) \leq \sum_{e \in E} w_e \cdot \sum_{e' \in E' : e \in Q_{e'}} g(|Y_{e'}|) \tag{19}$$

$$= \sum_{e' \in E'} g(|Y_{e'}|) \cdot \sum_{e \in E : e \in Q_{e'}} w_e \tag{20}$$

$$\leq \sum_{e' \in E'} g(|Y_{e'}|) \cdot w'_{e'}, \tag{21}$$

which is the cost of the optimal solution on T. Note that the inequality (19) followed from the concavity of the function $g(\cdot)$, and inequality (21) follows from the fact that the tree T is a *dominating tree* for the metric (V, d_G) and that $Q_{e'}$ is a shortest path in G.

Finally, it remains to show that if there is a solution in G of cost Opt, there is a solution of expected cost at most $\mathsf{Opt} \cdot O(\log n)$ on the random tree T. The argument is similar in spirit to the one above. Suppose the solution in G is given by the paths $P_i^* = \langle s_i, v_1, v_2, \ldots, v_\ell, t_i \rangle$. For each edge $e = uv \in E$, consider the tree path $R_e = R_{uv}$. Now, define the path P'_i on the tree for (s_i, t_i) to be the concatenation of the paths $R_{s_i v_1}, R_{v_1 v_2}, \ldots, R_{v_{l-1} v_l}, R_{v_l t_i}$; note that this path P'_i may be non-simple. If A_e be the set of paths P_i^* using the graph edge e, and $B_{e'}$ be the (multi-)set of paths P'_i using the tree edge e', then the expected tree cost is

$$\mathbf{E}\left[\sum_{e' \in E'} g(|B_{e'}|) \cdot w'_{e'}\right] \leq \mathbf{E}\left[\sum_{e' \in E'} \sum_{e \in E : e' \in R_e} g(|A_e|) \cdot w'_{e'}\right] \tag{22}$$

$$\leq \sum_{e \in E} g(|A_e|) \cdot \mathbf{E}\left[\sum_{e' \in E' : e' \in R_e} w'_{e'}\right] \tag{23}$$

$$\leq \sum_{e \in E} g(|A_e|) \cdot O(\log n) \cdot w_e, \tag{24}$$

where the last inequality follows from the guarantees of the probabilistic tree embedding. This completes the proof of the $O(\log n)$-approximation guarantee. □

Using the deterministic extension mentioned in the discussion after Theorem 25, one can obtain a deterministic algorithm for uniform buy-at-bulk by enumerating the $O(n \log n)$ trees in the support of the distribution $\mathscr{D}$, solving the problem on each, and returning the best solution amongst them.

The paper of Awerbuch and Azar [11] shows how to extend the algorithm to handle a slightly more general buy-at-bulk problem, where the terminal pairs (s_i, t_i) also have demand values $d_i \in \mathbb{Z}_{\geq 0}$, and the cost function for the edges $g : \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$ can be any *subadditive* function (that is, any function satisfying $g(x_1) + g(x_2) \leq g(x_1 + x_2)$). Andrews [8] showed that an $O(\log^{1/4-\varepsilon} n)$ approximation algorithm for the (uniform) buy-at-bulk problem, for any $\varepsilon > 0$, would imply that $NP \subseteq ZTIME(n^{\text{poly}\log n})$. Given the hardness results, the special cases of *single-sink* buy-at-bulk problem (where all the flow goes to a single sink) has been considered, and constant-factor approximation algorithms are known [80, 157, 85]. Recently, variants of the buy-at-bulk problems with higher connectivity requirements (e.g., where each client requires two edge-disjoint paths to the root, and the cost incurred at an edge is a concave function of the number of paths using this edge) have also been studied—e.g., see [10, 35].

A different direction of research considers the case when the function $g(\cdot)$ is not given up-front, and the goal is to find paths P_i that are good for all concave functions $g(\cdot)$, even when the adversary choosing the function is allowed to look at the paths. Goel and Estrin [70] gave an algorithm for single-sink instances that outputs paths which are simultaneously an $O(\log n)$ approximation for all concave functions $g(\cdot)$. Gupta et al. [82] extended this result to give an algorithm that outputs paths for each pair of vertices, such that regardless of the demand values d_{uv} between the vertex pairs and the subadditive cost function $g(\cdot)$, the solution given by this set of paths is an $O(\log^2 n)$ approximation.

6.3 Extensions and Other Applications

The idea of embedding arbitrary metric spaces into distributions over trees has had a major impact on approximation algorithms for problems on metric space. In some cases, the algorithms obtained by randomly reducing to the problem on trees are still the best algorithms known for the problem: this is true for the buy-at-bulk problem, as well as the Group Steiner tree [69], the min-communication spanning tree problem [167]. For other problems (such as the k-median problem [28]), these tree embeddings have given us a general technique to obtain a first-cut approximation algorithm, which have then been improved upon by other problem-specific techniques.

The idea of embedding a metric into a distribution over trees can be extended to embedding the shortest-path metric of a graph G into a distribution over its subtrees. Specifically, given a graph $G = (V, E)$ with edge lengths w_e, one can consider the set $\mathsf{ST}(G)$ of spanning trees of G (with the same edge-lengths w_e): note that this is a subset of the set $\mathsf{DT}(V, d_G)$, where d_G is the shortest-path metric for G—we now consider a variant of Definition 24 where one α-probabilistically embeds the graph metric d_G into dominating tree metrics, but the support of the distribution is restricted to $\mathsf{ST}(G)$. The results of [1, 54] show that any graph metric $O(\log n \,\text{poly}\log\log n)$-probabilistically embeds into its spanning trees, thus almost matching the result in Theorem 25 up to $\text{poly}\log\log n$ factors. (It remains an interesting open question to remove these lower-order factors.) These results have been useful in a number of network design contexts, including a result of Chekuri et al. [33] on non-uniform buy-at-bulk problems to be discussed in the next section, and online algorithms for k-connectivity problems [83].

Other techniques from metric embeddings have been useful in other contexts as well: e.g., *well-padded tree embeddings* [82] have been used for oblivious network design problems, and the similar *maximum-gradient embeddings* [122] have proved useful for the fault-tolerant facility location problems. Arora's techniques for polynomial-time approximation schemes for the traveling salesman problem have been extended to arbitrary metric spaces of low *doubling dimension* via embeddings into distributions of low-treewidth graphs [158].

7 Matching Based Augmentation

A particularly simple technique with wide applicability to connectivity problems is that of matching-based augmentation. Given a problem where some set of terminals desire connectivity to each other, one may be able to efficiently solve a "partial" connectivity problem that reduces the number of connected components by a constant factor. Iterating this process then gives a solution to the original connectivity problem, with an approximation guarantee that is related to the number of rounds of this process. In Section 7.1, we will see how such an idea gives an $O(\log n)$-approximate MST; subsequently, we will see an advanced application of this problem to the (non-uniform) buy-at-bulk problem in Section 7.2.

7.1 Approximate MST via Matching Based Augmentation

Start with a metric space (V,d) and define $V_0 = V$. We first find a min-cost near-perfect matching M_0 (which leaves at most one node unmatched) on the nodes V, where the cost of matching two nodes (u,v) is the distance $d(u,v)$ between them. For each matched pair, pick one of the nodes and add them to the set V_1; also add in the unmatched node, if any. Find a min-cost near-perfect matching M_1 on V_1; pick one of the newly matched vertices (and the unmatched vertex, if any) and form V_2, and continue until $|V_t| = 1$ for some t. Return the set of edges $T = \cup_{j=0}^{t-1} M_j$.

Theorem 27. *The above algorithm returns a tree of cost at most $O(\log n)$-times that of the MST.*

Proof. To show that the set of edges output by the algorithm is a tree, we inductively claim that for any i, all nodes in $V \setminus V_i$ are connected to some node in V_i using the edges in $\cup_{j<i} M_i$. Indeed, this is true when $i = 0$; moreover, any node in $V_i \setminus V_{i-1}$ is dropped because it is connected to the other node using an edge from M_{i-1}. Since we stop when we have a single node, all other nodes are connected to this node, and hence to each other. Moreover, the number of edges added is precisely $n-1$, and hence we have a spanning tree.

Since the number of rounds is at most $O(\log n)$, it suffices to show that the cost of each matching M_i is at most $d(T^*)$, the cost of the MST T^*. To make the argument simpler, assume there are $2k$ nodes in V_i. The Euler tour C of the MST T^* has cost at most $2d(T^*)$, twice the cost of the MST. Each node in V_i may appear at several places: we choose one of these positions arbitrarily. If we rename these nodes to be $\{x_0, x_1, \ldots, x_{2k-1}, x_{2k} = x_0\}$ in the order we encounter them as we go around the tour, then by the triangle inequality, $\sum_{i=0}^{2k-1} d(x_i, x_{i+1}) \leq 2d(T^*)$. Hence, one of the sums $\sum_{i=0}^{k-1} d(x_{2i}, x_{2i+1})$ or $\sum_{i=0}^{k-1} d(x_{2i+1}, x_{2i+2})$ is at most half that value, i.e., $d(T^*)$. But this is a valid matching of the nodes in V_i, and hence the min-cost perfect-matching has cost at most the weight of the MST, which proves the theorem. □

7.2 An Application: A Non-Uniform Buy-at-Bulk Problem

An elegant use of these matching-based ideas appears in a paper by Meyerson, Munagala, and Plotkin [124], who gave an $O(\log n)$ approximation for the following buy-at-bulk problem. The *single-sink non-uniform buy-at-bulk* problem (also known as the *cost-distance* problem) is the following: we are given a graph $G = (V,E)$, where the edges have both a *buying cost* $c_B : E \to \mathbb{R}_{\geq 0}$ and a *renting cost* $c_R : E \to \mathbb{R}_{\geq 0}$. We are given a root vertex $r \in V$, and a subset of terminals $R \subseteq V$: the goal is to define a path P_v from each terminal $v \in R$ to the root r (hence it is a "single-sink" problem), with the cost of the solution being

$$\mathsf{cost}(\{P_v\}_{v \in R}) = \sum_{e \in \cup_{v \in R} P_v} c_B(e) + \sum_{v \in R} \sum_{e \in P_v} c_R(e). \tag{25}$$

One can think of the buying cost as being a fixed charge we pay for using the edge (regardless of the number of paths using it), and the renting cost as being an incremental charge paid for *every* path using the edge. The cost function for each edge is an affine function $f_e(x) = c_B(e) + c_R(e) \cdot x$, and hence is a concave function

in the number of paths x using the edge. Importantly, the cost functions for different edges are allowed to be unrelated to each other—hence the use of "non-uniform" to refer to this problem. (One can consider a variant of the non-uniform buy-at-bulk problem where the concave cost functions on each edge are not just affine functions $a_e + b_e x$, but are allowed to be arbitrary concave function; however, the general concave functions case can be reduced the affine functions case with only a constant factor loss in the approximation, and hence we will focus on the affine functions case.)

The non-uniformity, where different edges have different functions, is in contrast to the "uniform" problem formulation in Section 6.2 where the cost functions were all multiples of some fixed concave function $g(\cdot)$. This non-uniformity in the cost function makes buy-at-bulk harder: for instance, while the single-sink *uniform* buy-at-bulk problem admits constant-factor approximations [80], the single-sink *non-uniform* buy-at-bulk problem is $\Omega(\log\log n)$-hard to approximate unless NP admits $O(n^{\log\log\log n})$-time deterministic algorithms [42]. On the other hand, the best approximation guarantee known for the single-sink non-uniform problem, with a proof based on random matching-based augmentations, is the following.

Theorem 28 ([124])**.** *The single-sink non-uniform buy-at-bulk problem admits a randomized $O(\log|R|)$-approximation algorithm.*

Proof. Let us first give the algorithm: for simplicity, we assume that the number of terminals $k = |R|$ is a power of 2. To begin, all the terminals are marked *active*, and define $A_1 = R$. The algorithm proceeds in rounds $i = 1, 2, \ldots$: at the beginning of each round i, we ensure that the size of the active set A_i is $|A_i| = \frac{k}{2^{i-1}}$.

In round i, if $|A_i| > 1$, we find a perfect matching M_i of the vertices in A_i, and also paths between each of these matched pairs (using a procedure described later). If u and v are matched in M_i and the path between them is $Q^i_{uv} \subseteq E$, we randomly choose one of u or v to mark as *inactive*: say this vertex is u. The path P_u from the (now inactive) terminal u to the root is defined to be Q^i_{uv} concatenated with the path P_v for v that we will construct in future rounds. This concludes round i; the set A_{i+1} is now defined to be the terminals that are still active. It remains to specify how to find the matching M_i: for this, we define edge lengths $\ell_i(e) = c_B(e) + 2^{i-1} \cdot c_R(e)$ for each edge $e \in E$, and define Q^i_{uv} to be a shortest path between $u, v \in A_i$ with edge-lengths $\ell_i(\cdot)$. We then find a matching M_i of the active set A_i that minimizes

$$L_i = \sum_{(u,v)\in M_i} \textit{length of the path } Q^i_{uv}.$$

Finally, if $i = \log_2 k + 1$ and there is only one active vertex v, we define P_v to be a shortest path from v to the root r in (V, E) where the edge lengths are again $\ell_i(e) = c_B(e) + 2^{i-1} \cdot c_R(e) = c_B(e) + k \cdot c_R(e)$. This completes the definition of the paths for every vertex in R.

The proof the theorem follows from two claims: firstly, that the cost of the solution is at most $\sum_i L_i$, the sum of the lengths of all the paths over all the $O(\log k)$ stages, and secondly, that $\mathbf{E}[L_i]$ for each stage is at most Opt. The former claim is not difficult to verify, and we focus on the latter. To begin, note that the probability that a terminal $u \in R$ belongs to A_i is exactly $2^{-(i-1)}$. Consider an optimal solution $\{P^*_v \mid v \in R$ with cost $\mathsf{cost}(\{P^*_v\}) = OPT$, and define $E^* = \cup_{v\in R} P^*_v$ to be the edges used by these paths. Consider the random set of paths $\{P^*_v \mid v \in A_i\}$ used by only the vertices active in stage i: their union $E^*_i = \cup_{v\in A_i} P^*_v$ connects A_i to the root, and has total length $\ell_i(E^*_i) = \sum_{e\in E^*_i}(c_B(e) + 2^{i-1} c_R(e))$. For an edge $e \in E^*$, its expected contribution to this length $\ell_i(E^*_i)$ is

$$\begin{aligned}\Pr[e \in E^*_i] \times (c_B(e) + 2^{i-1} c_R(e)) &\le c_B(e) + \sum_{v\in R: e\in P^*_v} \Pr[v \in A_i] \times 2^{i-1} c_R(e) \\ &= c_B(e) + \sum_{v\in R: e\in P^*_v} c_R(e).\end{aligned}$$

Summing over all edges in E^*, we get that $\ell_i(E^*_i) \le \sum_{e\in E^*} c_B(e) + \sum_{v\in R}\sum_{e\in P^*_v} c_R(e) = \mathsf{Opt}$. Hence, E^*_i is a subgraph of expected length at most Opt that connects all terminals in A_i to the root. In turn, this implies

the existence of a tree T_i spanning $A_i \cup \{r\}$ with expected length at most Opt; the Euler tour of this tree thus has expected weight at most 2Opt. Now, just as in the proof of Theorem 27, we can infer the existence of a perfect matching on A_i of expected total length at most half the cost of this tour, i.e., at most Opt. Similarly, we can infer that the weight of a path connecting any vertex in A_i is at most Opt. These two bounds show that the expected length of the paths in each round i is at most Opt, thus completing the proof. □

The paper of Meyerson et al. [124] considered a slightly more general problem where each terminal i wants to send $d_i \in \mathbb{Z}_{\geq 0}$ units of flow to the sink, and the cost of using an edge e to send $x > 0$ units of flow is then $c_B(e) + x \cdot c_R(e)$; the proof above can be extended to this more general problem. The derandomization for the algorithm above was given by Chekuri et al. [34].

The results for the single-sink problem have only recently been extended to the general (multiple-sink) case: a poly-logarithmic approximation algorithm has been given by Chekuri et al. [33]. Andrews [8] has shown that the general non-uniform buy-at-bulk problem is hard to approximate better than $\Omega(\log^{1/2-\varepsilon} n)$.

7.3 Extensions and Other Applications

The name "matching-based augmentation" comes from the survey by Ravi [142]. Perhaps the first application of the matching-based heuristic was for Christofides' $\frac{3}{2}$-approximation algorithm for the metric traveling salesman problem: we first find a minimum spanning tree (whose cost is at most the cost of the optimal TSP tour), and then we find a matching on the odd-degree nodes (whose cost is at most half the cost of the optimal tour, using an argument akin to those above). Matching-based augmentation ideas were also used in a series of works on finding trees with small degree and diameter [120, 141, 144].

8 Other Ideas and Techniques

Directed Problems. While we have focused on undirected graphs in this chapter, most of the above problems can also be posed for directed graphs. Generally, directed variants of the problems discussed are harder than their undirected counterparts, with the exception of the minimum spanning tree problem, whose directed variant—the min-cost arborescence problem—can be solved using a primal-dual algorithm [51]. Halperin and Krauthgamer [88] proved that an $O(\log^{2-\varepsilon} n)$ approximation algorithm for directed Steiner tree for some constant $\varepsilon > 0$ would imply that NP has randomized quasi-polynomial time algorithms; on the positive side, if there are k terminals to be connected to the root, the best known algorithm for the problem gives an approximation ratio of $\varepsilon^{-3}k^{\varepsilon}$ and runs in time $n^{O(1/\varepsilon)}$ [27]. The directed Steiner forest problem is hard to approximate better than $\Omega(2^{\log^{1-\varepsilon} n})$ for any $\varepsilon > 0$, via a reduction from the Label Cover problem [49]; the current best algorithms gives an approximation guarantee of $O(k^{1/2+\varepsilon})$ [32] and $n^{\varepsilon} \cdot \min\{n^{4/5}, m^{2/3}\}$ [57] (note that these two results are incomparable since k could be as large as $\Theta(n^2)$).

Flow-Equivalent Trees and Oblivious Routing. The existence of the "flow-equivalent" Gomory-Hu trees for single-commodity flows has been long known. Its counterpart for multicommodity flows was proved by Räcke [19, 136]: he showed that for some parameter $\beta = O(\text{poly}\log n)$, given any capacitated graph $G = (V, E, u)$ there exists a capacitated tree $T_G = (V', E', u')$ such that any traffic matrix that can be feasibly routed in G can be feasibly routed in T_G, while any traffic matrix that can be routed in T_G can be routed in the graph G with edge-congestion β (that is, the flow in G violates the edge capacities by at most a β factor). The current best value of $\beta = O(\log^2 n \log\log n)$ is given by Harrelson et al. [89]. At a high level, this result allows us to reduce problems on finding cuts in graphs (which are often challenging) to finding them in trees, where the problems are much simpler. For example,

In a subsequent paper [137], Räcke extended his definition to distributions over trees: instead of constructing a single tree from the graph G, he proved that there existed distributions over trees such that (a)

any traffic matrix feasibly routable in the graph could also be routed in each of the trees in the distribution's support, and (b) if for each tree T belonging to the support of the distribution, the traffic matrix D_T could be feasibly routed in the tree T, then the "average" traffic matrix $E_T[D_T]$ could be routed in the graph with congestion at most $\beta = O(\log n)$. This definition may remind the reader of Definition 24: interestingly enough, Räcke used distance-preserving tree embeddings to prove his result, which shows that this factor β obtained in the congestion result and the distance-preservation factor α obtained in Theorem 25 are exactly the same! Räcke's paper surveys many of the problems whose approximation guarantees can be improved using his results: e.g., the results can be used to get an $O(\log n)$-approximation for min-bisection (improving on the earlier $O(\log^{3/2} n)$ of Feige and Krauthgamer [56].

References

[1] I. Abraham, Y. Bartal, and O. Neiman. Nearly tight low stretch spanning trees. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 781–790, 2008.

[2] A. Agarwal and M. Charikar. On the advantage of network coding for improving network throughput. In *Proceedings, IEEE Information Theory Workshop*, 2004.

[3] A. A. Ageev and M. I. Sviridenko. Pipage rounding: a new method of constructing algorithms with proven performance guarantee. *J. Comb. Optim.*, 8(3):307–328, 2004.

[4] A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem in networks. *SIAM J. Comput.*, 24:440–456, 1995.

[5] N. Alon. A note on network reliability. In *Discrete probability and algorithms (Minneapolis, MN, 1993)*, volume 72 of *IMA Vol. Math. Appl.*, pages 11–14. Springer, New York, 1995.

[6] N. Alon, R. M. Karp, D. Peleg, and D. West. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.*, 24(1):78–100, 1995.

[7] N. Alon and J. H. Spencer. *The probabilistic method*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Inc., Hoboken, NJ, third edition, 2008. With an appendix on the life and work of Paul Erdős.

[8] M. Andrews. Hardness of buy-at-bulk network design. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 115–124, 2004.

[9] Y. P. Aneja. An integer linear programming approach to the Steiner problem in graphs. *Networks*, 10:167–178, 1980.

[10] S. Antonakopoulos, C. Chekuri, F. B. Shepherd, and L. Zhang. Buy-at-bulk network design with protection. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 634–644, 2007.

[11] B. Awerbuch and Y. Azar. Buy-at-bulk network design. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 542–547, 1997.

[12] N. Bansal, R. Khandekar, J. Könemann, B. Peis, and V. Nagarajan. On generalization of network design problems with degree bounds. In *Proceedings, MPS Conference on Integer Programming and Combinatorial Optimization*, 2010.

[13] N. Bansal, R. Khandekar, and V. Nagarajan. Additive guarantees for degree bounded directed network design. In *Proceedings, ACM Symposium on Theory of Computing*, pages 769–778, 2008.

[14] R. Bar-Yehuda and S. Even. A linear time approximation algorithm for the weighted vertex cover problem. *J. Algorithms*, 2:198–203, 1981.

[15] Y. Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 184–193, 1996.

[16] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proceedings, ACM Symposium on Theory of Computing*, pages 161–168, 1998.

[17] Y. Bartal, M. Charikar, and D. Raz. Approximating min-sum k-clustering in metric spaces. In *Proceedings, ACM Symposium on Theory of Computing*, pages 11–20, New York, 2001. ACM.

[18] P. Berman and V. Ramaiyer. Improved approximations for the Steiner tree problem. *J. Algorithms*, 17(3):381–408, 1994.

[19] M. Bienkowski, M. Korzeniowski, and H. Räcke. A practical algorithm for constructing oblivious routing schemes. In *Proceedings, ACM Symposium on Parallelism in Algorithms and Architectures*, pages 24–33, 2003.

[20] D. Bienstock and A. Bley. Capacitated network design with multicast commodities. *Proceedings, 8th International Conference on Telecommunication Systems*, 2000.

[21] O. Borůvka. O jistém problému minimálním. *Práca Moravské Přírodověcké Společnosti*, 3:37–58, 1926. In Czech.

[22] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. An improved LP-based approximation for Steiner tree. To appear in Proc. 42nd STOC, 2010.

[23] G. Calinescu, A. Chakrabarti, H. J. Karloff, and Y. Rabani. Improved approximation algorithms for resource allocation. In *Proceedings, MPS Conference on Integer Programming and Combinatorial Optimization*, pages 401–414, 2002.

[24] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint (extended abstract). In *Proceedings, MPS Conference on Integer Programming and Combinatorial Optimization*, pages 182–196, 2007.

[25] A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar. Approximation algorithms for the unsplittable flow problem. *Algorithmica*, 47(1):53–78, 2007.

[26] D. Chakrabarty, J. Könemann, and D. Pritchard. Hypergraphic LP relaxations for Steiner trees. In *Proceedings, MPS Conference on Integer Programming and Combinatorial Optimization*, 2010.

[27] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed Steiner problems. *J. Algorithms*, 33(1):73–91, 1999.

[28] M. Charikar, C. Chekuri, A. Goel, and S. Guha. Rounding via trees: deterministic approximation algorithms for group Steiner trees and k median. In *Proceedings, ACM Symposium on Theory of Computing*, pages 114–123, 1998.

[29] M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. A. Plotkin. Approximating a finite metric by a small number of tree metrics. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 379–388, 1998.

[30] K. Chaudhuri, S. Rao, S. Riesenfeld, and K. Talwar. What would edmonds do? augmenting paths and witnesses for degree-bounded MSTs. In *Proceedings, International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 26–39, 2005.

[31] K. Chaudhuri, S. Rao, S. Riesenfeld, and K. Talwar. A push-relabel algorithm for approximating degree bounded MSTs. In *Proceedings, International Colloquium on Automata, Languages and Processing*, pages 191–201, 2006.

[32] C. Chekuri, G. Even, A. Gupta, and D. Segev. Set connectivity problems in undirected graphs and the directe d Steiner network problem. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms*, pages 532–541, 2008.

[33] C. Chekuri, M. Hajiaghayi, G. Kortsarz, and M. R. Salavatipour. Approximation algorithms for non-uniform buy-at-bulk network design problems. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 677–686, 2006.

[34] C. Chekuri, S. Khanna, and J. S. Naor. A deterministic algorithm for the cost-distance problem. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 232–233, 2001.

[35] C. Chekuri and N. Korula. Single-sink network design with vertex connectivity requirements. In *Proceedings, Foundations of Software Technology and Theoretical Computer Science.*, 2008.

[36] C. Chekuri and F. B. Shepherd. Approximate integer decompositions for undirected network design problems. *SIAM J. Discrete Math.*, 3(1):163–177, 2008.

[37] C. Chekuri, J. Vondrak, and R. Zenklusen. Dependent randomized rounding for matroid polytopes and applications. Technical Report 0910.4348, arXiv, 2009.

[38] S. Chen, O. Günlük, and B. Yener. Optimal packing of group multi-casting. In *Proceedings, IEEE INFOCOM*, pages 980–987, 1998.

[39] S. Chen, O. Günlük, and B. Yener. The multicast packing problem. *IEEE/ACM Trans. Netw.*, 8(3):311–318, 2000.

[40] M. Chlebík and J. Chlebíková. Approximation hardness of the Steiner tree problem on graphs. In *Proceedings, Scandinavian Workshop on Algorithm Theory*, pages 170–179, 2002.

[41] S. Chopra. On the spanning tree polyhedron. *Operations Research Letters*, 8:25–29, 1989.

[42] J. Chuzhoy, A. Gupta, J. S. Naor, and A. Sinha. On the approximability of network design problems. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 943–951, 2005.

[43] J. Chuzhoy and S. Khanna. Algorithms for single-source vertex connectivity. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 105–114, 2008.

[44] J. Chuzhoy and S. Khanna. An $O(k^3 \log n)$-approximation algorithm for vertex-connectivity survivable network design. *CoRR*, abs/0812.4442, 2008.

[45] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley, New York, 1997.

[46] G. Cornuéjols, J. Fonlupt, and D. Naddef. The traveling salesman problem on a graph and some related integer polyhedra. *Math. Programming*, 33:1–27, 1985.

[47] G. B. Dantzig, L. R. Ford, and D. R. Fulkerson. A primal-dual algorithm for linear programs. In H. W. Kuhn and A. W. Tucker, editors, *Linear inequalities and related systems*, pages 171–181. Princeton University Press, Princeton, NJ, 1956.

[48] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[49] Y. Dodis and S. Khanna. Designing networks with bounded pairwise distance. In *Annual ACM Symposium on Theory of Computing (Atlanta, GA, 1999)*, pages 750–759. ACM, New York, 1999.

[50] J. Edmonds. Maximum matching and a polyhedron with (0,1) vertices. *Journal of Research of the National Bureau of Standards B*, 69B:125–130, 1965.

[51] J. Edmonds. Optimum branchings. *J. Res. Nat. Bur. Standards*, B71:233–240, 1967.

[52] J. Edmonds. Matroids and the greedy algorithm. *Math. Programming*, 1:127–136, 1971.

[53] F. Eisenbrand, F. Grandoni, T. Rothvoß, and G. Schäfer. Approximating connected facility location problems via random facili ty sampling and core detouring. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 1174–1183, 2008.

[54] M. Elkin, Y. Emek, D. A. Spielman, and S.-H. Teng. Lower-stretch spanning trees. *SIAM J. Comput.*, 38(2):608–628, 2008.

[55] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. System Sci.*, 69(3):485–497, 2004.

[56] U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM J. Comput.*, 31(4):1090–1118, 2002.

[57] M. Feldman, G. Kortsarz, and Z. Nutov. Improved approximating algorithms for directed steiner forest. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 922–931, 2009.

[58] L. Fleischer. A 2-approximation for minimum cost 0, 1, 2 vertex connectivity. In *Proceedings, MPS Conference on Integer Programming and Combinatorial Optimization*, pages 115–129, 2001.

[59] L. Fleischer, K. Jain, and D. P. Williamson. Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems. *J. Comput. System Sci.*, 72(5):838–867, 2006.

[60] L. R. Ford, Jr. and D. R. Fulkerson. Maximal flow through a network. *Canad. J. Math.*, 8:399–404, 1956.

[61] A. Frank. Packing paths, circuits and cuts - a survey. In B. Korte, L. Lovasz, H. J. Promel, and A. Schrijver, editors, *Paths, Flows and VLSI-Layout*, volume 9 of *Algorithms and Combinatorics*, pages 47–100. Springer, 1990.

[62] A. Frank. Increasing the rooted-connectivity of a digraph by one. *Math. Programming*, 84(3):565–576, 1999.

[63] A. Frank, T. Király, and M. Kriesell. On decomposing a hypergraph into k connected sub-hypergraphs. *Discrete Appl. Math.*, 131(2):373–383, 2003.

[64] D. R. Fulkerson. Blocking and anti-blocking pairs of polyhedra. *Math. Programming*, 1:168–194, 1971.

[65] M. Fürer and B. Raghavachari. An NC approximation algorithm for the minimum degree spanning tree problem. In *Proc. of the 28th Annual Allerton Conf. on Communication, Control and Computing*, pages 274–281, 1990.

[66] M. Fürer and B. Raghavachari. Approximating the minimum-degree Steiner tree to within one of optimal. *J. Algorithms*, 17(3):409–423, Nov. 1994.

[67] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding and its applications to approximation algorithms. *J. ACM*, 53(3):324–360 (electronic), 2006.

[68] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, San Francisco, 1979.

[69] N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group Steiner tree problem. *J. Algorithms*, 37(1):66–84, 2000. (Preliminary version in *9th SODA*, pages 253–259, 1998).

[70] A. Goel and D. Estrin. Simultaneous optimization for concave costs: single sink aggregation or single source buy-at-bulk. *Algorithmica*, 43(1-2):5–15, 2005.

[71] M. X. Goemans. Minimum bounded degree spanning trees. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 273–282, 2006.

[72] M. X. Goemans, A. V. Goldberg, S. A. Plotkin, D. B. Shmoys, É. Tardos, and D. P. Williamson. Improved approximation algorithms for network design problems. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 223–232, 1994.

[73] M. X. Goemans and Y. Myung. A catalog of Steiner tree formulations. *Networks*, 23:19–28, 1993.

[74] M. X. Goemans and D. P. Williamson. Approximating minimum-cost graph problems with spanning tree edges. *Operations Research Letters*, 16(4):183–189, 1994.

[75] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24:296–317, 1995.

[76] M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*. PWS Publishing, 1997.

[77] M. Grötschel, A. Martin, and R. Weismantel. Packing Steiner trees: a cutting plane algorithm and computational results. *Math. Programming*, 72:125–145, 1996.

[78] M. Grötschel, A. Martin, and R. Weismantel. Packing Steiner trees: polyhedral investigations. *Math. Programming*, 72:101–123, 1996.

[79] M. Grötschel, C. Monma, and M. Stoer. *Handbook in Operations Research and Management Science*, chapter Design of Survivable Networks. Noth-Holland, 1994.

[80] S. Guha, A. Meyerson, and K. Mungala. A constant factor approximation for the single sink edge installation problems. In *Proceedings, ACM Symposium on Theory of Computing*, pages 383–388, 2001.

[81] A. Gupta. Steiner points in tree metrics don't (really) help. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 220–227, Philadelphia, PA, 2001. SIAM.

[82] A. Gupta, M. T. Hajiaghayi, and H. Räcke. Oblivious network design. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 970–979, 2006.

[83] A. Gupta, R. Krishnaswamy, and R. Ravi. Online and stochastic survivable network design. In *Proceedings, ACM Symposium on Theory of Computing*, pages 685–694, 2009.

[84] A. Gupta, A. Kumar, J. M. Kleinberg, R. Rastogi, and B. Yener. Provisioning a Virtual Private Network: A network design problem for multicommodity flow. In *Proceedings, ACM Symposium on Theory of Computing*, pages 389–398, 2001.

[85] A. Gupta, A. Kumar, and T. Roughgarden. Simpler and better approximation algorithms for network design. In *Proceedings, ACM Symposium on Theory of Computing*, pages 365–372, 2003.

[86] A. Gupta, I. Newman, Y. Rabinovich, and A. Sinclair. Cuts, trees and ℓ_1-embeddings of graphs. *Combinatorica*, 24(2):233–269, 2004. (Preliminary version in 40th FOCS, 1999.).

[87] A. Gupta, M. Pál, R. Ravi, and A. Sinha. Boosted sampling: Approximation algorithms for stochastic optimization problems. In *Proceedings, ACM Symposium on Theory of Computing*, pages 417–426, 2004.

[88] E. Halperin and R. Krauthgamer. Polylogarithmic inapproximability. In *Proceedings of the thirty-fifth ACM symposium on Theory of computing*, pages 585–594. ACM Press, 2003.

[89] C. Harrelson, K. Hildrum, and S. Rao. A polynomial-time tree decomposition to minimize congestion. In *Proceedings, ACM Symposium on Parallelism in Algorithms and Architectures*, pages 34–43, 2003.

[90] A. Hayrapetyan, C. Swamy, and E. Tardos. Network design for information networks. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 933–942, 2005.

[91] S. Hougardy and H. J. Prömel. A 1.598 approximation algorithm for the Steiner problem in graphs. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 448–453, 1999.

[92] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem*. Number 53 in Annals of Discrete Mathematics. Elsevier Science Publishers B. V., 1992.

[93] K. Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[94] K. Jain, I. I. Mandoiu, V. V. Vazirani, and D. P. Williamson. A primal-dual schema based approximation algorithm for the element connectivity problem. *J. Algorithms*, 45(1):1–15, 2002.

[95] V. Jarník. O jistém problému minimálním. *Práca Moravské Přírodovědecké Společnosti*, 6:57–63, 1930. In Czech.

[96] T. Jenkyns. The efficiency of the greedy algorithm. In *Proceedings of the 7th Southeastern Conference on Combinatorics, Graph theory and Computing*, pages 341–350, 1976.

[97] D. R. Karger and M. Minkoff. Building Steiner trees with incomplete global knowledge. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 613–623, 2000.

[98] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, NY, 1972.

[99] M. Karpinski and A. Zelikovsky. New approximation algorithms for the Steiner tree problems. *J. Combinatorial Optimization*, 1(1):47–65, 1997.

[100] P. N. Klein and R. Ravi. When cycles collapse: A general approximation technique for constrained two-connectivity problems. In *Proceedings, MPS Conference on Integer Programming and Combinatorial Optimization*, pages 39–55, 1993.

[101] J. Könemann, D. Pritchard, and K. Tan. A partition-based relaxation for Steiner trees. *Math. Programming*, 2009.

[102] J. Könemann and R. Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. *SIAM J. Comput.*, 31(6):1783–1793, 2002.

[103] J. Könemann and R. Ravi. Primal-dual meets local search: Approximating MST's with nonuniform degree bounds. *SIAM J. Comput.*, 34(3):763–773, 2005.

[104] G. Konjevod, R. Ravi, and F. S. Salman. On approximating planar metrics by tree metrics. *Inform. Process. Lett.*, 80(4):213–219, 2001.

[105] B. Korte and D. Hausmann. An analysis of the greedy algorithm for independence systems. *Annals of Discrete Mathematics*, 2:65–74, 1978.

[106] B. Korte, L. Lovász, and R. Schrader. *Greedoids*. Springer-Verlag, New York, 1991.

[107] B. Korte, H. J. Prömel, and A. Steger. Steiner trees in VLSI-layout. In B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, editors, *Paths, Flows, and VLSI-Layout*, pages 185–214. Springer-Verlag, Berlin, Germany, 1990.

[108] B. Korte and J. Vygen. *Combinatorial Optimization*. Springer, New York, fourth edition, 2008.

[109] G. Kortsarz, R. Krauthgamer, and J. R. Lee. Hardness of approximation for vertex-connectivity network design problems. *SIAM J. Comput.*, 33(3):704–720, 2004.

[110] G. Kortsarz and Z. Nutov. Approximating minimum-cost connectivity problems. In T. F. Gonzalez, editor, *Approximation Algorithms and Meta-heuristics*. Chapman & Hall, 2007.

[111] M. Kriesell. Edge-disjoint trees containing some given vertices in a graph. *J. Combin. Theory Ser. B*, 88(1):53–65, 2003.

[112] J. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings, American Mathematical Society*, 7:48–50, 1956.

[113] L. C. Lau. Packing steiner forests. In *Proceedings, MPS Conference on Integer Programming and Combinatorial Optimization*, pages 362–376, 2005.

[114] L. C. Lau. *On approximate min-max theorems for graph connectivity problems*. PhD thesis, University of Toronto, Toronto, Canada, 2006.

[115] L. C. Lau. An approximate max-steiner-tree-packing min-steiner-cut theorem*. *Combinatorica*, 27(1):71–90, 2007.

[116] L. C. Lau, S. Naor, M. Salvatipour, and M. Singh. Survivable network design with degree and order constraints. In *Proceedings, ACM Symposium on Theory of Computing*, 2007.

[117] L. C. Lau and M. Singh. Additive approximation for bounded degree survivable network design. In *Proceedings, ACM Symposium on Theory of Computing*, pages 759–768, 2008.

[118] E. Lawler. *Combinatorial Optimization: Networks and Matroids*. Saunders College Publishing, Fort Worth, 1976.

[119] W. Mader. A reduction method for edge–connectivity in graphs. *Annals of Discrete Mathematics*, 3:145–164, 1978.

[120] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt, III. Bicriteria network design problems. *J. Algorithms*, 28(1):142–171, 1998.

[121] V. Melkonian and É. Tardos. Algorithms for a network design problem with crossing supermodular demands. *Networks*, 43(4):256–265, 2004.

[122] M. Mendel and A. Naor. Maximum gradient embeddings and monotone clustering. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 10th International Workshop, APPROX 2007, and 11th International Workshop, RANDOM 2007, Princeton, NJ, USA, August 20-22, 2007, Proceedings*, pages 242–256, 2007.

[123] K. Menger. Zur allgemeinen kurventheorie. *Fund. Math.*, 10:95–115, 1927.

[124] A. Meyerson, K. Munagala, and S. Plotkin. Cost-distance: Two metric network design. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 624–630, 2000.

[125] M. Mitzenmacher and E. Upfal. *Probability and computing*. Cambridge University Press, Cambridge, 2005. Randomized algorithms and probabilistic analysis.

[126] R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge University Press, Cambridge, 1995.

[127] V. Nagarajan, R. Ravi, and M. Singh. Personal communication. 2009.

[128] C. S. J. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *J. London Math. Soc.*, 36:445–450, 1961.

[129] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. I. *Math. Programming*, 14(3):265–294, 1978.

[130] Z. Nutov. An almost $O(\log k)$-approximation for k-connected subgraphs. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 912–921, 2009.

[131] Z. Nutov. A note on rooted survivable networks. *Inform. Process. Lett.*, 109(19):1114–1119, 2009.

[132] T. Polzin and S. V. Daneshmand. A comparison of Steiner tree relaxations. In *Proceedings of the Combinatorial Optimization Symposium (COS-98)*, volume 112, 1-3 of *Discrete Applied Mathematics*, pages 241–262, 2001.

[133] T. Polzin and S. V. Daneshmand. On Steiner trees and minimum spanning trees in hypergraphs. *Oper. Res. Lett.*, 31(1):12–20, 2003.

[134] R. C. Prim. Shortest connection networks and some generalizations. *Bell Systems Technical Journal*, pages 1389–1401, November 1957.

[135] H. J. Prömel and A. Steger. *The Steiner Tree Problem — A Tour through Graphs, Algorithms, and Complexity*. Vieweg Verlag, Braunschweig-Wiesbaden, 2002.

[136] H. Räcke. Minimizing congestion in general networks. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 43–52, 2002.

[137] H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings, ACM Symposium on Theory of Computing*, pages 255–264, 2008.

[138] B. Raghavachari. Algorithms for finding low degree structures. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*. PWS Publishing, 1997.

[139] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *J. Comput. System Sci.*, 37(2):130–143, 1988. Twenty-Seventh Annual IEEE Symposium on the Foundations of Computer Science (Toronto, ON, 1986).

[140] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[141] R. Ravi. Rapid rumor ramification: Approximating the minimum broadcast time. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 202–213, 1994.

[142] R. Ravi. Matching based augmentations for approximating connectivity problems. In *Proceedings, Latin American Theoretical Informatics Symposium*, pages 13–24, 2006.

[143] R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt. Many birds with one stone: Multi-objective approximation algorithms. In *Proceedings, ACM Symposium on Theory of Computing*, pages 438–447, 1993.

[144] R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt, III. Approximation algorithms for degree-constrained minimum-cost network-design problems. *Algorithmica*, 31(1):58–78, 2001.

[145] R. Ravi and F. S. Salman. Approximation algorithms for the traveling purchaser problem and its variants in network design. In *Algorithms—ESA '99 (Prague)*, volume 1643 of *Lecture Notes in Comput. Sci.*, pages 29–40. Springer, Berlin, 1999.

[146] R. Ravi and M. Singh. Delegate and conquer: An LP-based approximation algorithm for minimum degree MSTs. In *Proceedings, International Colloquium on Automata, Languages and Processing*, pages 169–180, 2006.

[147] R. Ravi and D. P. Williamson. An approximation algorithm for minimum-cost vertex-connectivity problems. *Algorithmica*, 18(1):21–43, 1997.

[148] R. Ravi and D. P. Williamson. Erratum: An approximation algorithm for minimum-cost vertex-connectivity problems. *Algorithmica*, 34(1):98–107, 2002.

[149] G. Robins and A. Zelikovsky. Tighter bounds for graph steiner tree approximation. *SIAM J. Discrete Math.*, 19(1):122–134, 2005.

[150] J. D. P. Rolim and L. Trevisan. A case study of de-randomization methods for combinatorial approximation algorithms. In *Network design: connectivity and facilities location (Princeton, NJ, 1997)*, volume 40 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 319–334. Amer. Math. Soc., Providence, RI, 1998.

[151] A. Schrijver. *Combinatorial optimization*. Springer, New York, 2003.

[152] M. Singh and L. C. Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In *Proceedings, ACM Symposium on Theory of Computing*, 2007.

[153] A. Srinivasan. A survey of the role of multicommodity flow and randomization in network design and routing. In *Randomization methods in algorithm design (Princeton, NJ, 1997)*, volume 43 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 271–302. Amer. Math. Soc., Providence, RI, 1999.

[154] A. Srinivasan. Distributions on level-sets with applications to approximation algorithms. In *Proceedings, IEEE Symposium on Foundations of Computer Science*, pages 588–597, 2001.

[155] A. Srinivasan. New approaches to covering and packing problems. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms*, pages 567–576, 2001.

[156] C. Swamy and A. Kumar. Primal-dual algorithms for connected facility location problems. *Algorithmica*, 40(4):245–269, 2004.

[157] K. Talwar. Single-sink buy-at-bulk LP has constant integrality gap. In *Proceedings, MPS Conference on Integer Programming and Combinatorial Optimization*, volume 2337 of *Lecture Notes in Computer Science*, pages 475–486, 2002.

[158] K. Talwar. Bypassing the embedding: Algorithms for low-dimensional metrics. In *Proceedings, ACM Symposium on Theory of Computing*, pages 281–290, 2004.

[159] W. T. Tutte. On the problem of decomposing a graph into n-connected factors. *J. London Math. Society*, 36:221–230, 1961.

[160] V. V. Vazirani. *Approximation Algorithms*. Springer, second edition, 2003.

[161] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings, ACM Symposium on Theory of Computing*, pages 67–74, 2008.

[162] D. Warme. *Spanning Trees in Hypergraphs with Applications to Steiner Trees*. PhD thesis, University of Virginia, 1998.

[163] D. B. West. *Introduction to Graph Theory*. Prenctice Hall, Upper Saddle River, 2nd edition, 2001.

[164] D. P. Williamson. The primal-dual method for approximation algorithms. *Math. Programming*, 91(3):447–478, 2002.

[165] D. P. Williamson, M. X. Goemans, M. Mihail, and V. V. Vazirani. A primal-dual approximation algorithm for generalized steiner network problems. *Combinatorica*, 15(3):435–454, 1995.

[166] R. T. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Math. Programming*, 28:271–287, 1984.

[167] B. Y. Wu, G. Lancia, V. Bafna, K.-M. Chao, R. Ravi, and C. Y. Tang. A polynomial-time approximation scheme for minimum routing cost spanning trees. *SIAM J. Comput.*, 29(3):761–778, 2000. (Preliminary version in *9th SODA*, 1998).

[168] A. Z. Zelikovsky. An $11/6$-approximation algorithm for the network Steiner problem. *Algorithmica*, 9:463–470, 1993.