SANDIA REPORT

SAND2012-8235
Unlimited Release
Printed September 2012

LDRD Final Report: Combinatorial Optimization with Demands

Ojas Parekh

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2012-8235
Unlimited Release
Printed September 2012

LDRD Final Report: Combinatorial Optimization with Demands

Ojas Parekh
Discrete Mathematics and Complex Systems (01465)
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1326
odparek@sandia.gov

Robert D. Carr (01465)
David Pritchard (Princeton University)

Abstract

This report summarizes the research conducted and results obtained under the Laboratory-Directed Research and Development (LDRD) project, "Combinatorial Optimization with Demands," from October 2010 to September 2012. Complex resource allocation problems are ubiquitous in mission-driven applications. We investigated resource allocation problems endowed with additional parameters called *demands*, which allow for richer modeling. Demand-endowed resource allocation problems are considerably harder to solve than their traditional counterparts, and we developed a conceptually simple but powerful new framework called *iterative packing* which allowed us to address the added complexity of demands. By leveraging iterative packing, we designed *approximation algorithms* to solve a large class of hard resource allocation problems with demands. Approximation algorithms are efficient heuristics which also offer a performance guarantee on the worst case deviation from the cost of an optimal solution. Our algorithms offer better performance guarantees over previously known algorithms; moreover, we are able to show that our algorithms provide performance guarantees that are likely close to best possible.

Acknowledgments

We thank Cindy Phillips for insightful and entertaining discussions. We thank Brian Barrett and Rich Field for maintaining the LaTeX template on which this report is based.

Contents

This page intentionally left blank.

1 Introduction

In many combinatorial optimization and resource allocation scenarios, many commodities or resources are discounted when purchased in bulk, and others such as piping and cabling may only be available in a small number of types of varying capacity and cost. Such scenarios result in a nonlinear relationship between cost and capacity, which may be modeled by infusing traditional combinatorial optimization models with a feature called *demands*. Thus the challenge is to design algorithms for models endowed with demands while leveraging the rich and immense mathematical infrastructure developed for more traditional non-demand linear and combinatorial optimization. Even without considering economies of scale or demands, most of the underlying problems we consider are already NP-hard; a means of mitigating this is designing an approximation algorithm, an efficient heuristic algorithm that provides a worst-case performance guarantee relative to the cost of an optimal solution.

Our main results are simple but high-quality approximation algorithms for a large class of resource allocation problems that significantly advance the state of the art. In the remainder of this section we provide a brief and self-contained introduction to designing approximation algorithms for combinatorial optimization problems, highlighting the main challenges as well as the general technique on which our algorithms are based.

1.1 Integer programming

A technique that has proven remarkably effective in solving resource allocation and more general combinatorial optimization problems is *integer programming*. An integer programming problem, or *integer program* (IP), takes the following form:

$$\max \{c \cdot x \mid Ax \leq b \text{ and } x \text{ is an integer vector}\}, \tag{1.1}$$

where the *constraint matrix* A and vectors b and c define the problem instance. We require that the entries of A, b, and c are rational and, by scaling, may assume without loss of generality that they are integer. A *feasible solution* x is any integer vector that satisfies the linear constraints $Ax \leq b$, while an *optimal solution* is a feasible solution x that also maximizes the *objective function*, $c \cdot x$ over all feasible solutions. Thus we may think of solving an IP as finding an integer vector x maximizing $c \cdot x$ subject to the constraints $Ax \leq b$.

Although perhaps not apparent, integer programs are extremely effective in modeling combinatorial optimization problems. For example, one may express the well-known and fundamental maximum weight matching problem as a simple IP. A *matching* in a graph $G = (V,E)$ is simply a set of edges such that no two edges in the set share a common vertex. If we are given weights, c on the edges, the maximum weight matching problem seeks to find a matching of maximum possible

weight in G. An IP for maximum weight matching follows:

$$\begin{aligned}
\text{Maximize} \quad & \sum_{e \in E} c_e x_e & & \\
\text{subject to} \quad & \sum_{e \in \delta(v)} x_e \leq 1 & \forall v \in V & \quad (1.2) \\
& 0 \leq x_e \leq 1 & \forall e \in E & \quad (1.3) \\
& x_e \text{ integer} & \forall e \in E. & \quad (1.4)
\end{aligned}$$

The above IP is presented in a common but slightly different form than (1.1). In particular the rows of A are expressed in a compact quantified form, i.e. (1.2) expands to $|V|$ inequalities. Thus the constraint matrix A corresponding to the above IP would have $|V| + 2|E|$ rows, since (1.3) corresponds to both $-x_e \leq 0$ and $x_e \leq 1$. To see that the above IP indeed captures maximum weight matching, first note that each entry of the vector x corresponds to an edge in G, and the constraints (1.3) and (1.4) ensure that for each edge e, $x_e \in \{0,1\}$. This allows us to treat any feasible solution, x as a vector indicating which edges to select in a matching, i.e. precisely those edges with $x_e = 1$ are selected. By this correspondence, every feasible solution represents a matching since the constraints (1.2) ensure that for every vertex v, no more than one edge containing it is selected (the notation $\delta(v)$ refers to the set of edges adjacent to v).

Integer programming provides a rich medium for expressing combinatorial optimization problems, and for our purposes, we may simply define a *combinatorial optimization problem* as one which can be expressed as an IP. Unfortunately an IP formulation of a combinatorial optimization problem does not immediately lend itself to an efficient algorithm; solving an IP in general is NP-hard. However, by dropping the requirement that x be an integer vector from (1.1), we drastically reduce the complexity of the problem:

$$\max \{c \cdot x \mid Ax \leq b \text{ and } x \text{ is a } \textbf{real} \text{ vector}\}. \quad (1.5)$$

This problem is called *linear programming* and the above is a *linear program* (LP). Surprisingly, any LP can be solved in polynomial time, while as noted above, solving IPs is NP-hard. Nevertheless, exact IP algorithms that may take exponential time in the worst case are perhaps the most widespread and effective general means of solving practical instances of combinatorial optimization problems.

1.2 Demands

Integer Programs derived from combinatorial optimization problem are typically dominated with constraints that have coefficients that are ether 0 are 1. This is an artifact of modeling combinatorial structures. For example, consider the constraints (1.2) from our maximum weight matching IP. This constraint captures the combinatorial structure of a matching, i.e. that no two selected edges should share a vertex. We formulated this as a linear constraint by insisting that at most one edge adjacent to each vertex is selected, which results in a constraint with 0 or 1 coefficients. It turns out that such 0-1 constraints are expressive enough to model a variety of combinatorial structures;

moreover, these constraints are easier for general purpose IP solvers to handle than those with large coefficients.

On the other hand, allowing coefficients beyond 0 or 1 does add an added level of richness. For example, consider a simple constraint that says that we must select at most ten items among a larger collection:

$$\sum_i x_i \leq 10, \tag{1.6}$$

where we assume that each variable is also constrained to be 0 or 1. Note that (1.2) is a version of this type of constraint. As one may imagine, such constraints are quite common.

Now if we allow the coefficients of such a constraint to take potentially larger values, d_i, then we get:

$$\sum_i d_i x_i \leq 10. \tag{1.7}$$

This also has a natural interpretation: each item i has a cost d_i associated with it, and we may select any combination of items with a total cost of at most 10. Thus this imposes a natural budget constraint on some resource, such as money, time, fuel, workforce, etc. This type of constraint is also known as a *knapsack* constraint.

We call the quantities d_i *demands*, and we may consider the demand-endowed version of any natural 0-1 combinatorial optimization problem. We saw above that endowing the 0-1 constraint, (1.6) with demands results in the knapsack problem, (1.7). We may also consider endowing the maximum weight matching problem with demands. This results in a problem called *demand matching* that is a natural generalization of both matching and knapsack, for which we obtain a new approximation algorithm.

Demands are instrumental in modeling economies of scale, indivisible bundles of goods, all or nothing scenarios in routing problems as well as other types of resource allocation constraints. Unfortunately problems with demand constraints are considerably harder to solve than their 0-1 counterparts. Although we could simply ignore the underlying combinatorial structure of a demand problem and try using a general purpose IP solver, this is likely to result in poor performance. We show that for a large class of resource allocation problems called *sparse packing problems*, that we are able to solve the demand version of the problem while still leveraging underlying combinatorial structure.

1.3 Approximation algorithms

All the combinatorial optimization problems we consider are NP-hard, hence it is unreasonable to expect an efficient algorithm that always produces an optimal solution. However we may set our sights on efficient algorithms that always produce an approximately optimal solution. Such algorithms are commonly known as heuristics, yet one issue with such algorithms is that they provide no guarantee on how close the solutions they produce are to an optimal solution. *Approximation*

algorithms remedy this by providing a mathematical guarantee on their worst case performance relative to the optimal value.

The fact that high quality approximation algorithms exist is surprising, because for NP-hard problems it is also typically NP-hard to even compute just the value of an optimal solution rather than the solution itself. Although their design is generally driven by the same sorts of intuitive insights as heuristics, approximation algorithms are often more complicated in order to provide the provable guarantee on their performance. This along with the observation that many heuristics empirically tend to produce nearly optimal solutions on practical instances are common criticisms of approximation algorithms. Yet designing approximation algorithms is an important endeavor because one often gains a great deal of mathematical insight into a problem which may offer suggestions for simpler and easier to implement heuristics. Also the performance guarantee offered by an approximation algorithms is a worst-case bound, and the algorithm may perform vastly better on practical instances. One advantage of the iterative packing framework which we developed and the resulting approximation algorithms is that they are very easy to implement.

Approximation algorithms are typically parameterized by their performance guarantee. We formally define this notion below.

Definition 1.1 (Approximation algorithm)**.** For a constant $\alpha \geq 1$, an *α-approximation algorithm* is a polynomial-time algorithm that for any problem instance I, produces a solution of objective value at least $\frac{1}{\alpha} \cdot \text{OPT}_I$, where OPT_I is the optimal objective value for the instance I.

Thus an α-approximation algorithm is able to always produce a solution of weight at least $\frac{1}{\alpha}$ of the best possible. Since we seek to maximize the objective function, the smaller the value of α, the better the algorithm. A 1-approximation is simply an exact algorithm. We usually call α the *performance ratio* or *performance guarantee*. Many NP-hard combinatorial optimization problems exhibit an interesting phenomenon where a nearly optimal approximation algorithm can actually be used to solve the problem exactly. These problems are called APX-hard, and for such problems there exists some threshold $t > 1$ such that there can be no α-approximation for $\alpha \leq t$ (unless P=NP) [5]. Thus for APX-hard problems, a goal is to find the best possible approximation guarantee (assuming P$\neq$ NP). This all begs the question: how does one devise an approximation algorithm without being able to compute OPT_I (since, as mentioned above, even computing OPT_I is itself often NP-hard)?

1.4 LP relaxations and rounding

In order to derive an α-approximation algorithm for a problem, by Definition (1.1) we need to show that for any instance I, the algorithm produces a solution S_I such that:

$$\text{weight}(S_I) \geq \frac{1}{\alpha} \cdot OPT_I.$$

To do so it suffices to find an bound B_I for each instance such that:

1. $B_I \geq OPT_I$, and
2. $\text{weight}(S_I) \geq \frac{1}{\alpha} \cdot B_I$.

Then we would have $\text{weight}(S_I) \geq \frac{1}{\alpha} \cdot B_I \geq \frac{1}{\alpha} \cdot OPT_I$ as desired. In other words, if we are able to prove a performance guarantee with respect to an *upper bound* on OPT_I, then we get one with respect to OPT_I as well. This may seem counterintuitive since proving a performance guarantee with respect to an upper bound, which may be larger, seems harder that the original task; however, this circumvents the potential NP-hardness of computing OPT_I. A key to effectively leveraging such a bound is being able to show for any instance I, the upper bound is close to OPT_I.

One systematic way of finding such upper bounds is by considering the *LP relaxation* of a combinatorial optimization problem formulated as an IP. An LP relaxation is simply the LP obtained from an IP by dropping the restriction that x be an integer vector. For instance the LP relaxation of the maximum weight matching IP from Section (1.1) is obtained by dropping the integrality constraint, (1.4). LPs can be solved efficiently not just theoretically, as noted, but also in practice by using, for example, the celebrated Simplex Algorithm. Since we obtain an LP relaxation by expanding or *relaxing* the feasible region, any feasible solution for the IP is also feasible for the LP relaxation (but not vice versa), so solving the LP relaxation yields a solution x^* whose weight is at least as large as the optimal IP solution, i.e. OPT_I.

Thus solving the LP relaxation gives us both an upper bound on OPT_I as well as a solution x^*; however, the latter may not be a bona fide solution to the underlying combinatorial optimization problem since it may take on fractional values. Recall that for matching, we interpreted $x_e = 1$ as selecting the edge e; how should we interpret $x^*_e = \frac{1}{3}$? One natural idea is to round such fractional values to integers and then interpret the resulting solution. This is certainly a viable approach; however, doing so is nontrivial. Rounding up is beneficial since it increases the objective value; however, such a solution may no longer be feasible (e.g. for matching we may round up two fractional edges touching the same vertex). If we round down, we're not guaranteed a good quality approximate solution; indeed if all the fractional values are below 1, we would obtain the empty solution which has weight 0.

LP relaxation's are a powerful approach but achieving a good performance ratio is often more of an inspired art. Some of the more elegant breakthroughs in approximation algorithms in last decade have been LP relaxation based approaches. However, there is a natural barrier for the performance guarantee of an LP relaxation based approximation algorithm, which we simply call an *LP-relative approximation algorithm*. Since such algorithms are using the optimal LP relaxation value as an upper bound rather than the actual value of OPT_I, the performance guarantee one is able to establish is limited by this upper bound, even though the algorithm may perform better in reality. A natural way of quantifying this limit is by considering the *integrality gap* of an LP relaxation.

Definition 1.2 (Integrality gap)**.** The integrality gap, g of an LP relaxation is the worst case gap between the optimal values of the relaxation and the IP taken over all instances, I:

$$g = \max_I \frac{\text{OPT}^*_I}{\text{OPT}_I},$$

where OPT_I^* is the optimal value of the LP relaxation for I.

As an example, consider the LP relaxation for the maximum matching problem. Our instance I is simply a triangle where each edge has weight 1. It is apparent that any matching on I has weight 1 since only one edge may be selected. However, the LP can achieve a larger weight by "cheating" and using fractional values. The fractional solution which assigns a value of $\frac{1}{2}$ to each of the three edges is feasible for the LP relaxation since at each vertex v, $\sum_{e \in \delta(v)} x_e = \frac{1}{2} + \frac{1}{2} = 1$. That is, it is enough that the edges fractionally sum to at most 1. Thus this LP solution has total weight $3 \cdot \frac{1}{2} = \frac{3}{2}$. This example shows that the integrality gap is at least $\frac{\text{OPT}_I^*}{\text{OPT}_I} = \frac{3/2}{1} = 3/2$. Could there be another instance with a larger gap?

In order to answer this question we need an upper bound on the integrality gap g. We showed $g \geq \frac{3}{2}$ above, so if we could show $g \leq \frac{3}{2}$ we would know the integrality gap for maximum weight matching. The latter requires showing that for any instance, its gap is not large. One way of doing this is, for any instance, actually producing a solution of large weight relative to the optimal LP relaxation solution. This is precisely what an LP-relative approximation algorithm does, since it uses the optimal LP value as its bound. Thus if we are able to derive an LP-relative α-approximation algorithm with a matching integrality gap (i.e. $g = \alpha$), then we know that no better approximation algorithm is possible without using a stronger upper bound than the LP relaxation.

Results which establish the integrality gaps of LP relaxations are interesting in their own right. Using our new iterative packing framework, we have been able to design approximation algorithms that also establish integrality gaps, i.e. our algorithms give a previously unknown upper bound on g with a matching lower bound. Iterative packing is a conceptually simple technique, and its power in producing high quality approximation algorithms that are able to establish integrality gaps is surprising. We describe iterative packing and motivate it with an example in the next section and then present our results in the following sections.

2 Iterative packing

A conceptually simple yet powerful method called iterative rounding, introduced by Jain just over a decade ago [47], has been surprisingly effective in elegantly producing breakthrough approximation algorithms for a variety of discrete optimization problems and resolving open problems [47, 65, 57]. Iterative rounding is designed to work on covering problems, and unfortunately, applying the method to even non-demand versions of packing problems such as those arising in resource allocation is difficult. The main thrust of our work is a new method called *iterative packing* inspired by iterative rounding that works for packing problems, even with demands.

Our main insight, which differentiates our approach from previous ones, is to iteratively build a collection of feasible integral solutions, that serve as a sparse approximate convex decomposition of the current fractional solution. A *convex decomposition* of a fractional LP solution x^* is a weighted average of integral solutions:

$$x^* = \sum_{i \in \mathscr{I}} \lambda_i \chi^i,$$

where each χ^i is an integral IP solution to the problem, and $\sum_i \lambda_i = 1$ and $\lambda_i \geq 0$ for all i. One may think of the multipliers λ_i as a weighting of the IP solution χ^i. Since the λ_i are nonnegative and sum to 1, the fractional solution x^* is then a weighted average of the integral χ^i.

A convex decomposition is a powerful property, since if we are given one for a fractional solution, we essentially have an **exact** solution to our problem. This is because no matter what the objective function c is, we must have that one of the χ^i has objective value as good as x^*. To see this note that:

$$c \cdot x^* = \sum_{i \in \mathscr{I}} \lambda_i (c \cdot \chi^i);$$

now the scalar, $c \cdot x^*$ on the left is just a weighted average of the scalars, $c \cdot \chi^i$, hence one of the latter must have value at least $c \cdot x^*$. If x^* is an optimal LP solution, then for some j, we have $c \cdot \chi^j \geq c \cdot x^* \geq \mathrm{OPT}_I$, hence χ^j must be an optimal integral solution.

Thus hoping for an convex decomposition is too much for NP-hard problems; however, we may expect an *α-approximate convex decomposition*, for some $\alpha \geq 1$:

$$\frac{1}{\alpha} x^* = \sum_{i \in \mathscr{I}} \lambda_i \chi^i.$$

The idea behind such a decomposition is that by scaling the fractional solution x^* down by $\frac{1}{\alpha}$, it becomes easier to produce a convex decomposition; however, scaling down decreases the overall objective value by a factor of α. Using a similar argument as above, one can see that being able to generate an α-approximate convex decomposition in polynomial time implies a simple α-approximation algorithm: select the χ^i with the largest objective value as the solution. Thus approximate convex decompositions are powerful because they are essentially collections of solutions with the property that no matter which objective function one selects, at least one of the

solutions in the collection is a good approximate solution. This in some sense gives us an objective-oblivious representation of an approximate solution, which has applications in other contexts such a multi-objective optimization. Iterative packing produces such an approximate convex decomposition an an iterative fashion.

2.1 Related work

Singh and Lau [65] were the first to extend Jain's celebrated iterative rounding technique [47] to address packing constraints. Their approach obtains an approximate solution that marginally violates the packing constraints by iteratively removing packing constraints involving only a small number of variables. They were able to apply this elegant idea to resolve an open question concerning minimum cost degree-bounded spanning trees. More recently, Chan and Lau [19] employed an interesting combination of an iterative approach and the fractional local ratio method [8] to give the first approximation algorithm for the k-hypergraph matching problem that matches the integrality gap of the natural LP formulation, which had previousy been established as $k-1+1/k$ [36].

Pseudo-greedy methods similar to iterative packing have been successfully applied to packing and coloring problems. Chekuri, Mydlarz, and Shepherd [23] used such a technique to obtain a 4-approximation for multicommodity flows on trees. Bar-Yehuda et al. [8] gave both an iterative packing like and local ratio algorithms for approximating independent sets in t-interval graphs. Feige and Singh [32] applied this type of technique for weighted edge coloring of bipartite graphs.

Iterative packing can be seen as an extension and unification of the above type of techniques into a single framework. Moreover, akin to the iterative rounding method for covering problems, iterative packing explicitly identifies elements with large fractional values to obtain better approximation ratios. Other aspects of the framework include leveraging a specific ordering of the elements and starting with a nontrivial convex decomposition. This combination of ideas allows iterative packing to obtain approximation ratios approaching the integrality gap of the underlying LP formulation. We illustrate a basic version of iterative packing on the maximum weight matching problem in Section 2.2.

2.2 An example: maximum weight matching

We illustrate iterative packing on the maximum weight matching problem. Although this is a simple application, it serves well to illustrate the method. Consider the natural LP relaxation for the maximum matching problem from Section 1.1 on a graph $G=(V,E)$:

$$\begin{aligned} \text{Maximize} \quad & \sum_{e\in E} c_e x_e && P_M(G) \\ \text{subject to} \quad & \sum_{e\in\delta(v)} x_e \leq 1 \quad \forall v\in V && (2.1) \\ & 0\leq x_e \leq 1 \quad \forall e\in E. && (2.2) \end{aligned}$$

Given a feasible fractional solution x^* for the above LP, the iterative packing procedure obtains an α-approximate convex decomposition of x^*,

$$\frac{1}{\alpha}x^* = \sum_{i \in \mathscr{I}} \lambda_i \chi^i, \tag{2.3}$$

for some $\alpha \geq 1$, where each χ^i is an integral feasible solution – i.e. a matching (and $\sum_i \lambda_i = 1$ and $\lambda_i \geq 0$ for all i). Iterative packing in its most basic form directly produces a sparse decomposition, namely one with $|\mathscr{I}| \leq |E| + 1$. Even when this is not the case, we can apply elementary linear algebra to retain at most $|E| + 1$ solutions (more generally $n + 1$, where $x^* \in \mathbb{R}^n$). A procedure to accomplish the latter is related to the classical Carathéodory's Theorem.

The basic idea behind iterative packing is simple. We start with the trivial fractional solution, $x = 0$ which has a trivial approximate convex decomposition for any value of α: $\frac{1}{\alpha}x = \lambda_1 \cdot \chi^1$, where $\lambda_1 = 1$ and $\chi^1 = 0$. Now we select some edge e and insert into each χ^i into which it fits (i.e. χ^i is still feasible after adding e). We keep doing this until we have inserted e into a collection of solutions whose multipliers sum exactly to $\frac{1}{\alpha}x_e$. This ensures that the equation of (2.3) corresponding to e is satisfied. In order to do this, we may have to add a new solution to our existing convex decomposition. We then select another edge and repeat. in this manner, we iterate through the edges and grow an attempt to grow an approximate convex decomposition. We may fail and not be able to insert an edge into enough solutions; however, the idea is that if α is large enough, then each fractional edge is scaled by $\frac{1}{\alpha}$ to a value small enough for iterative packing to succeed. Analyzing the algorithm entails finding a value of α that allows the algorithm to succeed and is also small enough to give an good performance guarantee.

This describes iterative packing in a bottom-up perspective; to actually prove the bounds we require, it will be easier to describe the algorithm from a top-down perspective. Continuing with our example, we first show that choosing $\alpha = 2$ suffices. This yields a 2-approximation while also showing that the integrality gap of $P_M(G)$ is at most 2. We then show that we may actually select α to be as low as $\frac{3}{2}$; to do this, we leverage the fact that extreme points of $P_M(G)$ must contain an edge e with $x_e \geq 1/2$ (in fact this holds for all e). As we observed in Section 1.4, this establishes the integrality gap of $P_M(G)$. Although this result is well known, this is interesting, since much like iterative rounding, iterative packing offers insight into how large fractional components can facilitate the approximation of packing problems. Extending these ideas allowed us to derive results for more general packing problems that were not previously known.

We start with a fractional solution x^* and:

1. Remove an edge e (without otherwise modifying the instance)
2. Recursively obtain an α-approximate convex decomposition of the resulting fractional solution, $\bar{x}^*$
3. Pack e into precisely a $\frac{1}{\alpha}x^*_e$ fraction of the integral solutions.

The key, of course, is showing that the last step can always be performed successfully. For this to work, we require that for any fractional (or perhaps extreme point) solution x^* there exists an

$e \in E$ with

$$\sum_{i \in \mathscr{I}_e} \lambda_i \geq \frac{1}{\alpha} x_e^* \ , \tag{2.4}$$

where $\frac{1}{\alpha}\bar{x}^* = \sum_{i \in \mathscr{I}} \lambda_i \chi^i$ is an arbitrary approximate convex decomposition of the residual solution, $\bar{x}^*$, and $i \in \mathscr{I}_e$ indicates that χ^i is able to accommodate the edge e (i.e. $\chi^i \cup e$ is still a valid matching).

Although we may well be able to pack e into a fraction of the integral solutions that is larger than an $\frac{1}{\alpha}x_e^*$, to maintain our implicit inductive hypothesis we must ensure that e is packed into exactly an $\frac{1}{\alpha}x_e^*$ fraction of solutions. To accomplish this, we may have to clone some solution χ^i, insert e into exactly one of the two copies of χ^i, and distribute the multiplier λ_i among the copies so that e appears in the requisite fraction of solutions. The base case, which contains no edges, selects the empty solution with a multiplier of 1. Thus if (2.4) holds universally for a particular value of α, then we can efficiently obtain an α-approximate convex decomposition of x^* consisting of at most $|E|+1$ integral solutions. Selecting the best of these gives us the corresponding α-approximation algorithm as previously observed.

To see that (2.4) holds when $\alpha = 2$, consider some fractional solution x^* and an arbitrary edge $e = uv \in E$ with $x_e^* > 0$. Obtaining $\bar{x}^*$ as above by deleting e, we have that

$$\max\{\bar{x}^*(\delta(u)), \bar{x}^*(\delta(v))\} \leq 1 - x_e^* \ ,$$

hence in any convex decomposition $\frac{1}{\alpha}\bar{x}^*$, at most a $\frac{2}{\alpha}(1-x_e^*)$ fraction of the χ^i do not accomodate e, hence we require $1 - \frac{2}{\alpha}(1-x_e^*) \geq \frac{1}{\alpha}x^*{}_e$, which is equivalent to

$$\alpha \geq 2 - x_e^* \tag{2.5}$$

Thus by selecting $\alpha = 2$, we may successfully pack any edge $0 \leq x_e^* \leq 1$ in the last step of our algorithm. However, by selecting a large edge at each iteration we can improve the bound. It is well known that extreme points of $P_M(G)$ are $1/2$-integral, i.e. they have values either 0, $\frac{1}{2}$, or 1, so we may actually take $\alpha = 2 - \frac{1}{2} = 3/2$.

Note that although establishing the performance guarantee of the algorithm requires a bit of analysis in showing that a small value of α allows iterative packing to succeed, the algorithm itself is quite simple. For the more general packing problems we outline in the next few sections, we will need to refine the basic method. For example, we might wonder whether iterating over the edges in a particular order affords us anything. We will show that selecting a proper ordering of the edges is crucial for iterative packing to succeed when demands are introduced. Another area of refinement is the recursion in step (2): how far down do we go? In the above example, the base case was an empty solution, but we might also stop earlier and use some other technique for the base case.

For each of the following sections, we will define the problem that we solved and outline our results in the context of previous work. We will briefly describe the enhancements to iterative packing required to obtain our results, but we will leave full technical details to the papers we have already published describing our work [59, 16, 60].

3 Packing in hypergraphs

3.1 Problem statement

One natural generalization of the maximum weight matching problem is to consider the problem in the context of hypergraphs rather than graphs. One may motivate a hypergraph by considering what a graph where each edge is allowed to have more than two vertices might look like. A *hypergraph* $\mathscr{H}$ is defined on a finite set of *vertices*, V, just as a standard graph. However, instead of edges, a hypergraph contains *hyperedges*, where each hyperedge, S is simply a subset of vertices (standard edges are subsets of exactly two vertices). A k-hypergraph is a hypergraph in which every hyperedge has at most k vertices; so a 2-hypergraph is nothing more than a standard graph (with possible single-vertex "half edges").

One may define a *hypergraph matching* analogously to a graph matching as a collection of hyperedges such that no two share a common vertex. Our focus is on generalizations of hypergraph matching. We consider the case where each vertex v has a capacity b_v, and we allow up to b_v hyperedges incident upon v. This generalizes the well-known b-matching problem in graphs.

Another way of approaching the problem is to consider an integer program, (1.1) where the matrix A and vector b are nonnegative. Such problems are called *packing* problems. If we further restrict A to have entries that are either 0 or 1, then we obtain exactly the maximum weight hypergraph b-matching problem described above. The columns of A correspond to the hyperedges, while its row correspond to vertices. If we consider a sparse matrix A in which no column has more than k nonzero entries, then we obtain the maximum weight k-hypergraph b-matching problem. The 2-hypergraph 1-matching problem is equivalent to the maximum weight matching problem in graphs. Although the b-matching problem in graphs is solvable in polynomial time, k-hypergraph b-matching is NP-hard for $k \geq 3$ even when $b = 1$.

3.2 Related work

Matching problems in k-hypergraphs are well-studied. A celebrated result of Füredi, Kahn, and Seymour [36] established the integrality gap of the natural relaxation at $k-1+1/k$ for k-hypergraphs matching, with an improvement to $k-1$ for k-partite k-uniform hypergraphs. On the algorithmic side for any fixed $\varepsilon > 0$, the best known performance ratios are $(\frac{k}{2}+\varepsilon)$ for the unweighted version by Hurkens and Schrijver [45] and $(\frac{k+1}{2}+\varepsilon)$ for the weighted version by Berman [10]. On the other hand, Hazan, Safra and Schwartz [44] showed that the problem is hard to approximate within a factor of $\Omega(\frac{k}{\log k})$ unless P = NP.

The above algorithms are based on local search and do not provide a bound on the integrality gap of the natural relaxation. Chan and Lau [19] recently gave a $(k-1+1/k)$-approximation algorithm based on the fractional local ratio method; their approximation guarantee matches the integrality gap of the natural relaxation and offers an improvement for $k = 3$. They also give linear and semidefinite formulations with an improved gap.

Unfortunately, none of the above results extend to b-matching in k-uniform hypergraphs. For this problem fewer approximation results are known. Most relevant to our work are a greedy $k+1$-approximation by Krysta [55] and a primal-dual k-approximation by Young and Koufogiannakis [54]. An algorithm with guarantee $(\frac{k+3}{2}+\varepsilon)$ is implicit in the recent work of Feldman et al. [33] and Ward [66] on k-exchange systems; however, its running time has an exponential dependence on k and a pseudo-polynomial dependence on b. In addition to being a true polynomial time algorithm with respect to k and b, our result provides a better approximation for $k \leq 4$.

3.3 Results

We derived a $k-1+1/k$-approximation algorithm for k-hypergraph b-matching [60]. Our algorithm also establishes that the integrality gap of the natural LP relaxation for this problem is $k-1+1/k$ for any prime k; though this is conjectured to be the case for all k. Thus algorithm obtains the best possible performance ratio using the natural LP relaxation, and an improvement would require a new upper bound.

Devising our algorithm required a number of refinements to the vanilla iterative packing approach. We needed to perform iterative packing based on a carefully selected ordering of the hyperedges that allowed us to prove our improved bound. We also needed stronger inductive conditions on the approximate convex decomposition that our iterative packing algorithm produced. Although the algorithm is fairly simple to implement, the analysis required handling many technical details and was nontrivial.

4 Demand packing in graphs and hypergraphs

4.1 Problem statement

We also considered a demand version of the k-hypergraph b-matching problem, which we refer to as the *k-hypergraph demand matching* (k-HDM) problem. From the IP perspective, the problem we considered is solving any packing integer program (with possibly non 0 or 1 coefficients) with at most k nonzeros per column; we impose the additional restriction that all the nonzero values in any particular column are the same. The more general problem in which the nonzero values in a column are allowed to take on different values is called *k-column-sparse packing integer programming* (k-CS-PIP). The motivation for the problem is perhaps more natural from the graph perspective.

Consider the b-matching problem in graphs, except that now we also associate a demand d_e with each edge e. For b-matching, a feasible solution is one that has at most b_v edges adjacent to vertex v. For the demand version this condition becomes: a feasible solution is one where the sum of the demands of the edges adjacent to v is at most b_v (i.e., F is a feasible set of edges if $\sum_{e \in F \cap \delta(v)} d_e \leq b_v$ for all v). Thus feasibility is dictated by a knapsack type constraint at each vertex. One can view this problem as a natural common generalization of both knapsack and hypergraph matching problems.

The addition of demands renders the problem much more difficult to approximate, and prior to our work it was not clear whether a performance guarantee comparable to the $k - 1 + 1/k$ for the non-demand version was achievable. Moreover, even the *demand matching* problem, obtained when they hypergraph is a graph (i.e. $k = 2$), is NP-hard [64].

4.2 Related work

Shepherd and Vetta introduced the demand matching problem in 2002 and observed connections to designing CLOS switching networks [64]. They gave a deterministic 3.5-approximation algorithm and randomized 3.264-approximation algorithm for demand matching. Chakrabarty and Pritchard [62] recently gave a deterministic 4-approximation algorithm and randomized 3.764-approximation algorithm for the more general 2-CS-PIP problem. Shepherd and Vetta also established a lower bound of 3 on the integrality gap of the natural LP for demand matching.

For the more general k-hypergraph demand matching problem, Chekuri, Mydlarz, and Shepherd [23] presented an approximation algorithm with $O(k)$ guarantee for the restricted version in which the maximum demand of any hyper edge is at most $\min_v b_v$. Their result is part of a framework which they developed based on work of Kolliopoulos and Stein [53] that relates the integrality gap of a demand-endowed packing problem to its unit demand counterpart.

While Chekuri *et al.* [21] observed an $8k$-approximation for k-HDM, a recent flurry of work has also yielded $O(k)$-approximations for the more general k-CS-PIP problem. Pritchard initiated

the improvements with an iterative rounding based $2^k k^2$-approximation [61], which was improved to an $O(k^2)$-approximation by Chekuri, Ene, and Korula (see [62] and [6]) and Chakrabarty and Pritchard [62]. Most recently, Bansal *et al.* [6] devised a deterministic $8k$-approximation and a randomized $(ek + o(k))$-approximation.

4.3 Results

Our first result for these problems is a deterministic $2k$-approximation for k-hypergraph demand matching. The integrality gap of this relaxation is at least $2k - 1$ [6], hence our result essentially closes the gap. We show that our result is obtained by a simple modification to the standard iterative packing approach, namely iterate over the edges in order of decreasing demand. No other modifications are necessary. The fact that such a simple algorithm is able to achieve the integrality gap is remarkable considering the difficulties that demands present. Our result is also interesting because we give a performance guarantee that is only a factor of 2 off from the ones we were able to derive for the non-demand version of the problem.

Our second results settles an open problem and establishes the integrality gap for both demand matching and the more general 2-CS-PIP by giving a 3-approximation algorithm whose performance guarantee matches the lower bound given by Shepherd and Vetta. This required a more sophisticated invocation of iterative packing with a different base case. We essentially used iterative packing to handling a troublesome configuration of edges and used an existing algorithm to finish the job. This type of hybrid approach presents interesting possibilities for future work.

The results are detailed in a single-authored paper by the PI in the prestigious peer-reviewed Integer Programming and Combinatorial Optimization conference [59].

5 Capacitated 2-edge-connected spanning subgraph

5.1 Problem statement

The 2-edge-connected spanning subgraph (2-ECSS) problem captures the essence of network design: given a graph with edge costs, we seek to select, at minimum cost, a subset of edges spanning all the vertices that contains at least two edge-disjoint paths between any pair of vertices. This type of requirement is natural in the design of networks that are robust in light of attacks or failures. This problem is a fundamental combinatorial optimization problem, in part because of its close relation to the traveling salesman problem (TSP). Indeed under metric costs, the approximability of the two problems has both been observed and conjectured to behave similarly [39].

We consider a generalization of 2-ECSS which endows it with economies of scale. In addition to purchasing a single edge uv, our generalization allows the bulk purchase of bundles of edges between u and v at a discounted rate; i.e., purchasing a bundle is never more expensive than purchasing the same number of individual edges. Since we require 2-edge-connectivity, a bundle need not contain more than two edges. Moreover multiple bundles between a pair of vertices are redundant as we need only offer the single cheapest bundle. Thus our generalization offers three installation options between each pair of vertices: (i) no edge, (ii) a single edge, or (iii) a bundle of two edges. This type of problem falls under the banner of capacitated [18, 15] or buy-at-bulk [4, 37, 22] network design. We follow Chakrabarty et al. [18] and refer to our problem as the capacitated 2-ECSS (cap-2-ECSS) problem, where each edge has a capacity that represents the size of the associated bundle; e.g., a bundle of two edges is represented as an edge of capacity 2.

This problem differs from those we have discussed thus far in that it is a covering rather than a packing problem. Here, since our objective is minimization, we must redefine what we mean by an α-approximation algorithm (recall Defintion 1.1). In the context of minimization problems, an α-approximation algorithm produces a solution of cost at most $\alpha \cdot \mathrm{OPT}_I$ for any instance I. We still have that $\alpha = 1$ is an exact algorithm and that the smaller the value of α, the better the performance of the algorithm. We define integrality gaps accordingly. The notion of capacities here is analogous to that of demands in the context of packing problems that we have discussed.

5.2 Related work

The standard 2-ECSS problem admits a 2-approximation algorithm, and several algorithms of different flavors are known [46, 51, 52]. Under the restriction of metric costs, 2-ECSS is closely related to (symmetric) TSP, and Christofides' celebrated heuristic for metric TSP [24] delivers a 3/2-approximation algorithm for metric 2-ECSS as well. In fact for metric costs the natural LP relaxation for 2-ECSS is equivalent to the subtour relaxation [39, 67], which is a well known and standard TSP LP relaxation (e.g. Schrijver, Chapter 58 [63]). Wolsey showed that Christofides' algorithm is also a 3/2-approximation with respect to this LP bound [68]. The integrality gap is known to be at least 4/3, and it is a well known conjecture that it is precisely 4/3 (e.g. Goe-

mans [38]).

Another closely related variant is one which allows selecting multiple copies of any edge (we need at most two). The TSP counterpart to this multi-2-ECSS problem is the graphical TSP [25], which allows doubling edges and seeks to obtain a minimum cost spanning connected eulerian subraph (i.e. all vertex degrees are even). Both multi-2-ECSS and graphical TSP admit 3/2-approxmations as well [13]. For these problems the metric and multiedge variants are equivalent from an optimization and approximation point of view.

Interestingly, in our case of cap-2-ECSS such an equivalence is not apparent, since the usual technique of shortcutting does not immediately apply to capacitated edges. Thus multi-2-ECSS is a more appropriate pedigree for cap-2-ECSS rather than the more commonly formulated metric 2-ECSS. Capacitated network design has attracted a great deal of interest recently, and many generalizations of cap-2-ECSS have been studied, including larger degrees of connectivity, multiple commodities, and richer notions of economies of scale; see [4, 18, 37, 22] and the references therein for a sample of recent work. In particular Chakrabarty et al. [18] give logarithmic approximation algorithms for capacitated variants of k-ECSS.

5.3 Results

We show that, somewhat surprisingly, the capacitated extension of the natural LP relaxation for (multi-)2-ECSS has an integrality gap of at least 2. We offer a strengthened formulation, and demonstrate its integrality gap is between 3/2 and 5/3. Our result is constructive and we give a Christofides like 5/3-approximation algorithm. Our result is interesting because in addition to giving a good approximation guarantee, we show how existing approaches like the Chrstiofides heuristic may be extended to address capacities/demands.

This work was conducted jointly with Robert Carr (01465) and has been accepted for publication in the high-impact journal, Operations Research Letters [16].

6 Future work

In addition to establishing the state of the art in terms of theoretical performance guarantees, we believe our algorithms have the potential for significant practical impact. In fact this was part of the motivation in engineering a simple but effective framework like iterative packing. Our algorithms are easy to implement; however, many integer programming applications arising at the labs have complicated constraints in addition to types of packing or resource allocation constraints our work addresses. A direction for future research is to investigate how our work can be brought to bear on such applications. However, our work still has the potential for immediate impact in that the iterative packing framework can be integrated within Sandia's PICO IP solver to provide upper bounds for problems with a packing component.

7 Conclusion

Our most significant contribution is devising the iterative packing framework which provides a unified approach for designing approximation algorithms for packing problems. Moreover, the performance guarantees offered by these algorithms are very good and are able to match the integrality gap of the natural LP relaxation, indicating that improving our results is challenging and may not even be possible.

The work conducted under this LDRD has resulted in several publications in top peer-reviewed Operations Research venues. In particular we produced: 1 peer-reviewed conference paper, 1 journal paper, 1 submitted peer-reviewed conference paper, and 1-2 additional papers in preparation.

References

[1] *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*. IEEE Computer Society, 2006.

[2] *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*. IEEE Computer Society, 2010.

[3] Anthony Alexander, Sylvia Boyd, and Paul Elliot-Magwood. On the integrality gap of the 2-edge connected subgraph problem. Technical Report TR-2006-04, SITE, University of Ottawa, 2006.

[4] Spyridon Antonakopoulos, Chandra Chekuri, F. Bruce Shepherd, and Lisa Zhang. Buy-at-bulk network design with protection. *Math. Oper. Res.*, 36(1):71–87, 2011.

[5] Giorgio Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1999.

[6] Nikhil Bansal, Nitish Korula, Viswanath Nagarajan, and Aravind Srinivasan. On k-column sparse packing programs. In Eisenbrand and Shepherd [31], pages 369–382.

[7] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph (Seffi) Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *J. ACM*, 48:1069–1090, September 2001.

[8] Reuven Bar-Yehuda, Magnús M. Halldórsson, Joseph Naor, Hadas Shachnai, and Irina Shapira. Scheduling split intervals. *SIAM J. Comput.*, 36(1):1–15, 2006.

[9] Reuven Bar-Yehuda and Dror Rawitz. Using fractional primal-dual to schedule split intervals with demands. *Discrete Optimization*, 3(4):275–287, 2006.

[10] Piotr Berman. A $d/2$ approximation for maximum weight independent set in d-claw free graphs. In Halldórsson [42], pages 214–219.

[11] Robert E. Bixby, E. Andrew Boyd, and Roger Z. Ríos-Mercado, editors. *Integer Programming and Combinatorial Optimization, 6th International IPCO Conference, Houston, Texas, USA, June 22-24, 1998, Proceedings*, volume 1412 of *Lecture Notes in Computer Science*. Springer, 1998.

[12] C. Boucher and D. Loker. Expected approximation guarantees for the demand matching problem. Technical Report CS-2006-33, University of Waterloo, School of Computer Science, 2006.

[13] Sylvia Boyd and Amy Cameron. A 3/2-approximation algorithm for the multi-two-edge connected subgraph problem. Technical Report TR-2008-01, SITE, University of Ottawa, 2008.

[14] Robert Carr and R. Ravi. A new bound for the 2-edge connected subgraph problem. In Bixby et al. [11], pages 112–125.

[15] Robert D. Carr, Lisa Fleischer, Vitus J. Leung, and Cynthia A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *SODA*, pages 106–115, 2000.

[16] Robert D. Carr and Ojas Parekh. Approximating a capacitated 2-edge-connected spanning subgraph problem. *Operations Research Letters*, 2012. 13 pages, accepted for publication.

[17] Robert D. Carr and Santosh Vempala. Randomized metarounding. *Random Struct. Algorithms*, 20(3):343–352, 2002.

[18] Deeparnab Chakrabarty, Chandra Chekuri, Sanjeev Khanna, and Nitish Korula. Approximability of capacitated network design. In Günlük and Woeginger [41], pages 78–91.

[19] YukHei Chan and LapChi Lau. On linear and semidefinite programming relaxations for hypergraph matching. *Mathematical Programming*, pages 1–26, 2011.

[20] Moses Charikar, editor. *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*. SIAM, 2010.

[21] Chandra Chekuri, Alina Ene, and Nitish Korula. Unsplittable flow in paths and trees and column-restricted packing integer programs. In Dinur et al. [27], pages 42–55.

[22] Chandra Chekuri, Mohammad Taghi Hajiaghayi, Guy Kortsarz, and Mohammad R. Salavatipour. Approximation algorithms for non-uniform buy-at-bulk network design. In *FOCS* [1], pages 677–686.

[23] Chandra Chekuri, Marcelo Mydlarz, and F. Bruce Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Trans. Algorithms*, 3(3):27, 2007. Preliminary version appeared in *Proc. 30th ICALP*, pages 410–425, 2003.

[24] N. Christofides. Worst case analysis of a new heuristic for the traveling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.

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

[26] Camil Demetrescu and Magnús M. Halldórsson, editors. *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, volume 6942 of *Lecture Notes in Computer Science*. Springer, 2011.

[27] Irit Dinur, Klaus Jansen, Joseph Naor, and José D. P. Rolim, editors. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21-23, 2009. Proceedings*, volume 5687 of *Lecture Notes in Computer Science*. Springer, 2009.

[28] Christoph Dürr and Thomas Wilke, editors. *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th - March 3rd, 2012, Paris, France*, volume 14 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[29] Jack Edmonds. Maximum matching and a polyhedron with 0,1 vertices. *J. Research National Bureau Standards, Sect. B*, 69:185–204, 1965.

[30] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[31] Friedrich Eisenbrand and F. Bruce Shepherd, editors. *Integer Programming and Combinatorial Optimization, 14th International Conference, IPCO 2010, Lausanne, Switzerland, June 9-11, 2010. Proceedings*, volume 6080 of *Lecture Notes in Computer Science*. Springer, 2010.

[32] Uriel Feige and Mohit Singh. Edge coloring and decompositions of weighted graphs. In Halperin and Mehlhorn [43], pages 405–416.

[33] Moran Feldman, Joseph Naor, Roy Schwartz, and Justin Ward. Improved approximations for k-exchange systems - (extended abstract). In Demetrescu and Halldórsson [26], pages 784–798.

[34] Amos Fiat and Peter Sanders, editors. *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*. Springer, 2009.

[35] Lisa Fleischer, Michel X. Goemans, Vahab S. Mirrokni, and Maxim Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 611–620, New York, NY, USA, 2006. ACM.

[36] Zoltán Füredi, Jeff Kahn, and Paul D. Seymour. On the fractional matching polytope of a hypergraph. *Combinatorica*, 13(2):167–180, 1993.

[37] Ashish Goel and Ian Post. One tree suffices: A simultaneous o(1)-approximation for single-sink buy-at-bulk. In *FOCS* [2], pages 593–600.

[38] Michel X. Goemans. Worst-case comparison of valid inequalities for the tsp. *Math. Program.*, 69:335–349, 1995.

[39] Michel X. Goemans and Dimitris J. Bertsimas. Survivable networks, linear programming relaxations and the parsimonious property. *Math. Program.*, 60:146–166, 1993.

[40] Ralph E. Gomory. An algorithm for integer solutions to linear programs. In *Recent advances in mathematical programming*, pages 269–302. McGraw-Hill, New York, 1963.

[41] Oktay Günlük and Gerhard J. Woeginger, editors. *Integer Programming and Combinatoral Optimization - 15th International Conference, IPCO 2011, New York, NY, USA, June 15-17, 2011. Proceedings*, volume 6655 of *Lecture Notes in Computer Science*. Springer, 2011.

[42] Magnús M. Halldórsson, editor. *Algorithm Theory - SWAT 2000, 7th Scandinavian Workshop on Algorithm Theory, Bergen, Norway, July 5-7, 2000, Proceedings*, volume 1851 of *Lecture Notes in Computer Science*. Springer, 2000.

[43] Dan Halperin and Kurt Mehlhorn, editors. *Algorithms - ESA 2008, 16th Annual European Symposium, Karlsruhe, Germany, September 15-17, 2008. Proceedings*, volume 5193 of *Lecture Notes in Computer Science*. Springer, 2008.

[44] Elad Hazan, Shmuel Safra, and Oded Schwartz. On the complexity of approximating -set packing. *Computational Complexity*, 15(1):20–39, 2006.

[45] Cor A. J. Hurkens and Alexander Schrijver. On the size of systems of sets every t of which have an sdr, with an application to the worst-case ratio of heuristics for packing problems. *SIAM J. Discrete Math.*, 2(1):68–72, 1989.

[46] Kamal Jain. Factor 2 approximation algorithm for the generalized steiner network problem. In *FOCS*, pages 448–457, 1998.

[47] Kamal Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21:39–60, 2001.

[48] Joanna Jedrzejowicz and Andrzej Szepietowski, editors. *Mathematical Foundations of Computer Science 2005, 30th International Symposium, MFCS 2005, Gdansk, Poland, August 29 - September 2, 2005, Proceedings*, volume 3618 of *Lecture Notes in Computer Science*. Springer, 2005.

[49] David S. Johnson and Uriel Feige, editors. *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*. ACM, 2007.

[50] Idit Keidar, editor. *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings*, volume 5805 of *Lecture Notes in Computer Science*. Springer, 2009.

[51] Samir Khuller. Approximation algorithms for finding highly connected subgraphs. In Dorit S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*. PWS Publishing Co., Boston, Mass., 1996.

[52] Samir Khuller and Uzi Vishkin. Biconnectivity approximations and graph carvings. *J. ACM*, 41(2):214–235, 1994.

[53] Stavros G. Kolliopoulos and Clifford Stein. Approximating disjoint-path problems using packing integer programs. *Mathematical Programming*, 99(1):63–87, 2004.

[54] Christos Koufogiannakis and Neal E. Young. Distributed fractional packing and maximum weighted b-matching via tail-recursive duality. In Keidar [50], pages 221–238.

[55] Piotr Krysta. Greedy approximation via duality for packing, combinatorial auctions and routing. In Jedrzejowicz and Szepietowski [48], pages 615–627.

[56] Lap Chi Lau, Joseph Naor, Mohammad R. Salavatipour, and Mohit Singh. Survivable network design with degree or order constraints. *SIAM J. Comput.*, 39(3):1062–1087, 2009.

[57] Lap-Chi Lau, R. Ravi, and Mohit Singh. *Iterative Methods in Combinatorial Optimization*. Cambridge University Press, New York, NY, USA, 1st edition, 2011.

[58] Ron Lavi and Chaitanya Swamy. Truthful and near-optimal mechanism design via linear programming. In *FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 595–604, Washington, DC, USA, 2005. IEEE Computer Society.

[59] Ojas Parekh. Iterative packing for demand and hypergraph matching. In *Proc. of the 15th International Conference on Integer Programming and Combinatorial Optimization*, pages 349–361. Springer-Verlag, 2011.

[60] Ojas Parekh and David Pritchard. Approximation algorithms for generalized hypergraph matching problems. 12 pages, submitted to the Symposium on Theoretical Aspects of Computer Science, 2013.

[61] David Pritchard. Approximability of sparse integer programs. In Fiat and Sanders [34], pages 83–94.

[62] David Pritchard and Deeparnab Chakrabarty. Approximability of sparse integer programs. To appear in *Algorithmica*, 19 pages, 2010.

[63] Alexander Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer-Verlag, Berlin, 2003.

[64] F. Bruce Shepherd and Adrian Vetta. The demand-matching problem. *Mathematics of Operations Research*, 32(3):563–578, August 2007.

[65] Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In Johnson and Feige [49], pages 661–670.

[66] Justin Ward. A $(k+3)/2$-approximation algorithm for monotone submodular k-set packing and general k-exchange systems. In Dürr and Wilke [28], pages 42–53.

[67] David P. Williamson. Analysis of the held-karp heuristic for the traveling salesman problem. Master's thesis, MIT, Cambridge, MA, June 1990.

[68] Laurence A. Wolsey. Heuristic analysis, linear programming and branch and bound. *Math. Program. Study*, 13:121–134, 1980.

DISTRIBUTION:

1	MS 1326	Ojas Parekh, 01465
1	MS 1326	Robert D. Carr, 01465
1	MS 1326	M. Daniel Rintoul, 01465
1	MS 0899	Technical Library, 9536 (electronic copy)
1	MS 0359	D. Chavez, LDRD Office, 1911

Sandia National Laboratories