Further Approximations for Demand Matching: Matroid Constraints and Minor-Closed Graphs[*]

Sara Ahmadian[1] and Zachary Friggstad[†2]

1 **Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Canada**
sahmadian@uwaterloo.ca

2 **Department of Computing Science, University of Alberta, Edmonton, Canada**
zacharyf@ualberta.ca

Abstract

We pursue a study of the Generalized Demand Matching problem, a common generalization of the b-Matching and Knapsack problems. Here, we are given a graph with vertex capacities, edge profits, and asymmetric demands on the edges. The goal is to find a maximum-profit subset of edges so the demands of chosen edges do not violate the vertex capacities. This problem is **APX**-hard and constant-factor approximations are already known.

Our main results fall into two categories. First, using iterated relaxation and various filtering strategies, we show with an efficient rounding algorithm that if an additional matroid structure $\mathcal{M}$ is given and we further only allow sets $F \subseteq E$ that are independent in $\mathcal{M}$, the natural LP relaxation has an integrality gap of at most $\frac{25}{3} \approx 8.333$. This can be further improved in various special cases, for example we improve over the 15-approximation for the previously-studied Coupled Placement problem [Korupolu et al. 2014] by giving a 7-approximation.

Using similar techniques, we show the problem of computing a minimum-cost base in $\mathcal{M}$ satisfying vertex capacities admits a $(1, 3)$-bicriteria approximation: the cost is at most the optimum and the capacities are violated by a factor of at most 3. This improves over the previous $(1, 4)$-approximation in the special case that $\mathcal{M}$ is the graphic matroid over the given graph [Fukanaga and Nagamochi, 2009].

Second, we show Demand Matching admits a polynomial-time approximation scheme in graphs that exclude a fixed minor. If all demands are polynomially-bounded integers, this is somewhat easy using dynamic programming in bounded-treewidth graphs. Our main technical contribution is a sparsification lemma that allows us to scale the demands of some items to be used in a more intricate dynamic programming algorithm, followed by some randomized rounding to filter our scaled-demand solution to one whose original demands satisfy all constraints.

1998 ACM Subject Classification F.2.2 Computations on Discrete Structures, G.2.1 Combinatorial Algorithms, G.1.6 Optimization

Keywords and phrases Approximation Algorithms, Column-Restricted Packing, Demand Matching, Matroids, Planar Graphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.55

[*] A full version of the paper is available at https://arxiv.org/abs/1705.10396.
[†] This research was undertaken, in part, thanks to funding from the Canada Research Chairs program and an NSERC Discovery Grant.

1 Introduction

Many difficult combinatorial optimization problems involve resource allocation. Typically, we have a collection of resources, each with finite supply or **capacity**. Additionally there are tasks to be accomplished, each with certain requirements or **demands** for various resources. Frequently the goal is to select a maximum value set of tasks and allocate the required amount of resources to each task while ensuring we have enough resources to accomplish the chosen tasks. This is a very well-studied paradigm: classic problems include Knapsack, Maximum Matching, and Maximum Independent Set, and more recently-studied problems include Unsplittable Flow [1] and Coupled Placement [9]. In general, we cannot hope to get non-trivial approximation algorithms for these problems. Even the simple setting of Maximum Independent Set is inapproximable [8, 18], so research frequently focuses on well-structured special cases.

Our primary focus is when each task requires at most two different resources. Formally, in Generalized Demand Matching (GDM) we are given a graph $G = (V, E)$ with, perhaps, parallel edges. The vertices should be thought of as resources and the tasks as edges. Each $v \in V$ has a capacity $b_v \geq 0$ and each $uv \in E$ has demands $d_{u,e}, d_{v,e} \geq 0$ and a value $p_{uv} \geq 0$. A subset $M \subseteq E$ is *feasible* if $d_v(\delta(v) \cap M) \leq b_v$ for each $v \in V$ (we use $d_v(S)$ as shorthand for $\sum_{e \in S} d_{v,e}$ when $S \subseteq \delta(v)$). We note that the simpler term Demand Matching (DM) is used when $d_{u,e} = d_{v,e}$ for each edge $e = uv$ (e.g. [15, 17]).

DM is well-studied from the perspective of approximation algorithms. It is fairly easy to get constant-factor approximations and some work has been done refining these constants. Moreover the integrality gap of a natural LP relaxation is also known to be no worse than a constant (see the related work section). On the other hand, DM is **APX**-hard [15].

Our main results come in two flavours. First, we look to a generalization we call Matroidal Demand Matching ($\mathsf{GDM_M}$). Here, we are given the same input as in GDM but there is also a matroid $\mathcal{M} = (E, \mathcal{I})$ over the edges E with independence system $\mathcal{I} \subseteq 2^E$ that further restricts feasibility of a solution. A set $F \subseteq E$ is feasible if it is feasible as a solution to the underlying GDM problem and also $F \in \mathcal{I}$. We assume $\mathcal{M}$ is given by an efficient independence oracle. Our algorithms will run in time that is polynomial in the size of G and the maximum running time of the independence oracle.

As a special case, $\mathsf{GDM_M}$ includes the previously-studied Coupled Placement problem. In Coupled Placement, we are given a bipartite graph $G = (V, E)$ with vertex capacities. The tasks are not individual edges, rather for each task j and each $e = uv \in E$ we have demands $d^j_{u,e}, d^j_{v,e}$ placed on the respective endpoints u, v for placing j on edge e. Finally, each task j has a profit p_j and the goal is to select a maximum-profit subset of tasks j and, for each chosen task j, assign j to an edge of G so vertex capacities are not violated. We note that an edge may receive many different tasks. This can be viewed as an instance of $\mathsf{GDM_M}$ by creating parallel copies of each edge $e \in E$, one for each task j with corresponding demand values and profit for j and letting $\mathcal{M}$ be the partition matroid ensuring we take at most one edge corresponding to any task.

For another interesting case, consider an instance where, in addition to tasks requiring resources from a shared pool, each also needs to be connected to a nearby power outlet. We can model such an instance by letting $\mathcal{M}$ be a transversal matroid over a bipartite graph where tasks form one side, outlets form the other side, and an edge indicates the edge can reach the outlet.

In fact $\mathsf{GDM_M}$ can be viewed as a packing problem with a particular submodular objective function. These are studied in [2] so the problem is not new; our results are improved approximations. Our techniques also apply to give bicriteria approximations for the variant of $\mathsf{GDM_M}$ where we must pack a cheap base of the matroid while obeying congestion bounds. In the special case where $\mathcal{M}$ is the graphic matroid over G itself (i.e. the Minimum Bounded-Congestion Spanning Tree problem), we get an improved bicriteria approximation.

Second, we study GDM in special graph classes. In particular, we demonstrate a PTAS in families of graphs that exclude a fixed minor. This is complemented by showing that even DM is strongly NP-hard in simple planar graphs, thereby ruling out a fully-polynomial time approximation scheme (FPTAS) in simple planar graphs unless **P** = **NP**.

1.1 Statements of Results and Techniques

We first establish some notation. For a matroid $\mathcal{M} = (E, \mathcal{I})$, we let $r_{\mathcal{M}} : 2^E \to \mathbb{Z}_{\geq 0}$ be the rank function for $\mathcal{M}$. We omit the subscript $\mathcal{M}$ if the matroid is clear from the context. For $v \in V$ we let $\delta(v)$ be all edges having v as one endpoint; for $F \subseteq E$ we let $\delta_F(v)$ denote $\delta(v) \cap F$. For a vector of values x indexed by a set S, we let $x(A) = \sum_{i \in A} x_i$ for any $A \subseteq S$. A **polynomial-time approximation scheme** (PTAS) is an approximation algorithm that accepts an additional parameter $\epsilon > 0$. It finds a $(1+\epsilon)$-approximation in time $O(n^{f(\epsilon)})$ for some function f (where n is the size of the input apart from ϵ), so the running time is polynomial for any constant $\epsilon > 0$. An FPTAS is a PTAS with running time being polynomial in $\frac{1}{\epsilon}$ and n.

We say an instance of GDM has a **consistent ordering of edges** if E can be ordered such that the restriction of this ordering to each set $\delta(v)$ has these edges $e \in E$ appear in nondecreasing order of demands $d_{v,e}$. For example, DM itself has a consistent ordering of demands, just sort edges by their demand values. This more general case was studied in [13]. We say the instance is **conflict-free** if for any $e, f \in E$ we have that $\{e, f\}$ does not violate the capacity of any vertex.

In the first half of our paper, we mostly study the following linear-programming relaxation of $\mathsf{GDM_M}$. Here, $r : 2^E \to \mathbb{Z}$ is the rank function for $\mathcal{M}$.

$$\max : \left\{ \sum_{e \in E} p_e x_e : \sum_{e \in \delta(v)} d_{v,e} x_e \leq b_v \ \forall v \in V, \quad x(A) \leq r(A) \ \ \forall A \subseteq E, \quad x \geq 0 \right\} \qquad (\textbf{LP-M})$$

Note $x(\{e\}) \leq 1$ is enforced for each $e \in E$ as $r(\{e\}) \leq 1$. It is well-known that the constraints can be separated in polynomial time when given an efficient independence oracle for $\mathcal{M}$, so we can find an extreme point optimum solution to (**LP-M**) in polynomial time.

Throughout, we assume each edge is feasible by itself. This is without loss of generality: an edge that is infeasible by itself can be discarded[1]. We first prove the following.

▶ **Theorem 1.** *Let* OPT(**LP-M**) *denote the optimum solution value of* (**LP-M**). *If* $d_{v,e} \leq b_v$ *for each* $v \in V, e \in \delta(v)$ *then we can find, in polynomial time, a feasible solution* $\mathsf{M} \subseteq E$ *such that* OPT(**LP-M**)$/p(\mathsf{M})$ *(and, thus, the integrality gap) is at most:*

- $\frac{25}{3}$ *in general graphs*
- 7 *in bipartite graphs*
- 5 *if the instance has a consistent ordering of edges*
- 4 *if the instance is conflict-free*
- $1 + O(\epsilon^{1/3})$ *if* $d_{v,e} \leq \epsilon \cdot b_v$ *for each* $v \in V, e \in \delta(v)$ *(i.e. edges are* ϵ-***small****)*

[1] This is a standard step when studying packing LPs, even the natural Knapsack LP relaxation has an unbounded integrality gap if we consider items that do not fit by themselves.

These bounds also apply to graphs with parallel edges, so we get a 7-approximation for Coupled Placement, which beats the previously-stated 15-approximation in [9].

We prove all bounds in Theorem 1 using the same framework: iterated relaxation to find some $\mathsf{M}' \in \mathcal{I}$ with $p(\mathsf{M}') \geq OPT_{LP}$ that may violate some capacities by a controlled amount, followed by various strategies to pare the solution down to a feasible solution. We note constant-factor approximations for GDM_M were already implicit in [2], the bounds in Theorem 1 improve over their bounds and are relative to (**LP-M**) whereas [2] involves multilinear extensions of submodular functions.

Our techniques can also be used to address a variant of GDM_M. The input is the same, except we are required to select a base of $\mathcal{M}$. The goal is to find a minimum-value base satisfying the vertex capacities. More formally, let Minimum Bounded-Congestion Matroid Basis be given the same way as in GDM_M, except the goal is to find a minimum-cost base B of $\mathcal{M}$ satisfying the vertex capacities (i.e. the cheapest base that is a solution to the GDM_M problem).

When all demands are 1, this is the Minimum Bounded-Degree Matroid Basis problem which, itself, contains the famous Minimum Bounded-Degree Spanning Tree problem. As an important special case, we let Minimum Bounded-Congestion Spanning Tree denote the problem when $k = 2$ with arbitrary demands where $\mathcal{M}$ is the graphic matroid over G. Even determining if there is a feasible solution is **NP**-hard, so we settle with approximations that may violate the capacities a bit. Consider the following LP relaxation, which we write when G can even be a hypergraph.

$$\min : \left\{ \sum_{e \in E} p_e x_e : \sum_{e \in \delta(v)} d_{v,e} x_e \leq b_v \ \forall v \in V, \quad x(A) \leq r(A) \ \forall A \subseteq E, \quad x(E) = r(E), \quad x \geq 0 \right\} \tag{\textbf{LP-B}}$$

As a side effect of how we prove Theorem 1, we also prove the following.

▶ **Corollary 2.** *If G is a hypergraph where each edge has size at most k, then in polynomial time we can either determine there is no integral point in* (**LP-B**) *or we can find a base* B *of $\mathcal{M}$ such that $p(\mathsf{B}) \leq \mathrm{OPT}(\textbf{LP-B})$ and $d_v(\delta_\mathsf{B}(v)) \leq b_u + k \cdot \max_{e \in \delta(v)} d_{v,e}$ for each $v \in V$.*

▶ **Theorem 3.** *There is a $(1, 1+k)$-bicriteria approximation for* Minimum Bounded-Congestion Matroid Basis.

In particular, there is a $(1, 3)$-bicriteria approximation for Minimum Bounded-Congestion Spanning Tree, beating the previous best $(1, 4)$-bicriteria approximation [5]. Theorem 3 matches the bound in [9] for the special case of Coupled Placement in k-partite hypergraphs, but in a more general setting.

One could also ask if we can generalize Theorem 1 to hypergraphs. An $O(k)$-approximation is already known [2] and the integrality gap of (**LP-M**) is $\Omega(k)$ even without matroid constraints, so we could not hope for an asymptotically better approximation. We remind the reader that our focus in GDM_M is improved constants in the case of graphs ($k = 2$).

Our second class of results are quite easy to state. We study GDM in families of graphs that exclude a fixed minor. It is easy to see GDM is strongly **NP**-hard in planar graphs if one allows parallel edges as it is even strongly **NP**-hard with just two vertices, *e.g.* see [6, 11]. We show the presence of parallel edges is not the only obstacle to getting an FPTAS for GDM (or even DM) in planar graphs.

▶ **Theorem 4.** DM *is NP-hard in simple, bipartite planar graphs even if all demands, capacities, values, and vertex degrees are integers bounded by a constant.*

We then present our main result in this vein, which gives a PTAS for GDM in planar graphs among other graph classes.

▶ **Theorem 5.** GDM *admits a PTAS in families of graphs that exclude a fixed minor.*

This is obtained through the usual reduction to bounded-treewidth graphs [4]. We would like to scale demands to be polynomially-bounded integers, as then it is easy to solve the problem using dynamic programming over the tree decomposition. But packing problems are too fragile for scaling demands naively: an infeasible solution may be regarded as feasible in the scaled instance.

We circumvent this issue with a sparsification lemma showing there is a near-optimal solution M' where, for each vertex v, after packing a constant number of edges across v the remaining edges in $\delta_{\mathsf{M}'}(v)$ have very small demand compared with even the residual capacity. Our dynamic programming algorithm then guesses these large edges in each bag of the tree decomposition and packs the remaining edges according to scaled values. The resulting solution may be slightly infeasible, but the blame rests on our scaling of *small* edges and certain pruning techniques can be used to whittle this solution down to a feasible solution with little loss in the profit.

1.2 Related Work

DM (the case with symmetric demands) is well-studied. Shepherd and Vetta initially give a 3.264-approximation in general graphs and a 2.764-approximation in bipartite graphs [15]. These are all with respect to the natural LP relaxation, namely (**LP-M**) with matroid constraints replaced by $x_e \leq 1, \forall e \in E$. They also prove that DM is **APX**-hard even in bipartite graphs and give an FPTAS in the case G is a tree.

Parekh [13] improved the integrality gap bound for general graphs to 3 in cases of GDM that have a consistent ordering of edges. Singh and Wu improve the gap in bipartite graphs to 2.709 [17]. The lower bound on the integrality gap for general graphs is 3 [15], so the bound in [13] is tight. In bipartite graphs, the gap is at least 2.699 [17].

Bansal, Korula, Nagarajan, and Srinivasan study the generalization of GDM to hypergraphs [2]. They show if each edge has at most k endpoints, the integrality gap of the natural LP relaxation is $\Theta(k)$. They also prove that a slight strengthening of this LP has a gap of at most $(e + o(1)) \cdot k$. Even more relevant to our results is that they prove if the value function over the edges is submodular, then rounding a relaxation based on the multilinear extension of submodular functions yields a $\left(\frac{e^2}{e-1} + o(1)\right) \cdot k$-approximation. For $k = 2$, this immediately gives a constant-factor approximation for $\mathsf{GDM_M}$ by considering the submodular objective function $f : 2^E \to \mathbb{R}$ given by $f(S) = \max\{p(S') : S' \subseteq S, S' \in \mathcal{I}\}$.

They briefly comment on the case $k = 2$ in their work and say that even optimizations to their analysis for this special case yields only a 11.6-approximation for DM (i.e. without a matroid constraint). So our $\frac{25}{3}$-approximation for $\mathsf{GDM_M}$ is an improvement over their work. They also study the case where $d_{v,e} \leq \epsilon \cdot b_v$ for each $v \in V$ and each hyperedge $e \in \delta(v)$ and present an algorithm for GDM with submodular objective functions whose approximation guarantee tends to $\frac{4e^2}{e-1}$ as $\epsilon \to 0$ (with k fixed).

As noted earlier, our results yield improvements for two specific problems. First, our 7-approximation for $\mathsf{GDM_M}$ in bipartite graphs improves over the 15-approximation for Coupled Placement [9]. The generalization of Coupled Placement to k-partite hypergraphs is also studied in [9] where they obtain an $O(k^3)$-approximation, but this was already inferior to the $O(k)$-approximation in [2] when viewing it as a submodular optimization problem with packing constraints.

Second, our work also applies to the Minimum-Congestion Spanning Tree problem, defined earlier. Determining if there is even a feasible solution is **NP**-hard as this models the Hamiltonian Path problem. A famous result of Singh and Lau shows if all demands are 1 (so we want to bound the degrees of the vertices) then we can find a spanning tree with cost at most the optimum cost (if there is any solution) that violates the degree bounds additively by +1 [16]. In the case of arbitrary demands, the best approximation so far is a $(1, 4)$-approximation [5]: it finds a spanning tree whose cost is at most the optimal cost and violates the capacities by a factor of at most 4. It is known that obtaining a $(1, c)$-approximation is **NP**-hard for any $c < 2$ [7].

2 Approximation Algorithm for Demand Matching Problem

Here we present approximation algorithms for $\mathsf{GDM_M}$ and prove Theorem 1 and Corollary 2. Our algorithm consists of two phases: the iterative relaxation phase and the pruning phase. The first finds a set $\mathsf{M}' \in \mathcal{I}$ with $p(\mathsf{M}') \geq \text{OPT}(\textbf{LP-M})$ that places demand at most $b_v + 2 \cdot \max_{e \in \delta_{\mathsf{M}'}(v)} d_{v,e}$ on each $v \in V$. The second prunes M' to a feasible solution, different pruning strategies are employed to prove the various bounds in Theorem 1.

2.1 Iterative Relaxation Phase

This part is presented for the more general case of hypergraphs where each edge has at most k endpoints. Our $\mathsf{GDM_M}$ results in Theorem 1 pertain to $k = 2$, but we will use properties of this phase in our proof of Corollary 2. The algorithm starts with (**LP-M**) and iteratively removes edge variables and vertex capacities.

We use the following notation. For some $W \subseteq V, F \subseteq E$, a matroid $\mathcal{M}'$ with ground set F, and values $b'_v, v \in W$ we let $\textbf{LP-M}[W, F, \mathcal{M}', b']$ denote the LP relaxation we get from (**LP-M**) over the graph (V, F) with matroid $\mathcal{M}'$ where we drop capacity constraints for $v \in V - W$ and use capacities b'_v for $v \in W$.

Note that the relevant graph for $\textbf{LP-M}[W, F, \mathcal{M}', b']$ still has all vertices V, it is just that some of the capacity constraints are dropped. Also, for a matroid $\mathcal{M}'$ and an edge $e \in F$ we let $\mathcal{M}' - e$ be the matroid obtained by deleting e and, if $\{e\}$ is independent in $\mathcal{M}'$, we let $\mathcal{M}'/e$ be the matroid obtained by contracting e (i.e. a set A is independent in $\mathcal{M}'/e$ if and only if $A \cup \{e\}$ is independent in $\mathcal{M}'$).

Algorithm 1 describes the steps in the iterated relaxation phase. Correct execution and termination are consequences of the following two lemmas. Their proofs are standard for iterated techniques and are deferred to the full version.

▶ **Lemma 6.** *Throughout the execution of the algorithm, whenever $\mathcal{M}'$ is contracted by e we have $\{e\}$ is independent (i.e. e is not a loop) in $\mathcal{M}'$.*

▶ **Lemma 7.** *The algorithm terminates in polynomial time and the returned set M' is an independent set in $\mathcal{M}$ with $p(\mathsf{M}') \geq \text{OPT}(\textbf{LP-M})$. Furthermore, if at any point $W' = \emptyset$ then the corresponding extreme point solution x^* is integral.*

The last statement in the lemma emphasizes the last case in the body of the loop cannot be encountered if $W' = \emptyset$.

Next, we prove M' is a feasible demand matching with respect to capacities $b_v + k \cdot \max_{e \in \delta(v)} d_{e,v}$ for each $v \in V$ by utilizing the following claim.

▶ **Claim 8.** *In any iteration, if $0 < x^*_e < 1$ for each $e \in F$ then $|\delta_F(v)| \leq x^*(\delta_F(v)) + k$ for some $v \in W$.*

Algorithm 1 Iterated Relaxation Procedure for GDM_M.

```
$W \leftarrow V, F \leftarrow E, \mathcal{M}' \leftarrow \mathcal{M}$
$b'_v \leftarrow b_v$ for each $v \in V$
$\mathsf{M}' \leftarrow \emptyset$
while $F \neq \emptyset$ do
    solve LP-M$[W, F, \mathcal{M}', b']$ to get an optimum extreme point $x^*$
    if $x^*_e = 0$ for some $e \in F$ then
        $F \leftarrow F - \{e\}$
        $\mathcal{M}' \leftarrow \mathcal{M}' - e$                              ▷ fix $x^*_e$ to 0 from now on
    else if $x^*_e = 1$ for some $e \in F$ then
        $F \leftarrow F - \{e\}$
        $\mathcal{M}' \leftarrow \mathcal{M}'/e$
        $\mathsf{M}' \leftarrow \mathsf{M}' \cup \{e\}$                              ▷ fix $x^*_e$ to 1 from now on
        $b'_v \leftarrow b'_v - d_{v,e}$ for each endpoint $v$ of $e$      ▷ permanently allocate space for $e$
    else
        let $v$ be any vertex in $W$ with minimum value $|\delta_F(v)| - x^*(\delta_F(v))$
        $W \leftarrow W - \{v\}$                              ▷ drop the capacity constraint for $v$
return $\mathsf{M}'$
```

Proof. Let $A_1 \subsetneq A_2 \subsetneq \ldots \subsetneq A_t \subseteq F$ be any *maximal-length chain of tight sets*. That is, $x^*(A_i) = r_{\mathcal{M}'}(A_i)$ for each A_i in the chain. Then the indicator vectors $\chi_{A_i} \in \{0,1\}^F$ of the sets A_i are linearly independent and every other $A \subseteq F$ with $x^*(A_i) = r_{\mathcal{M}'}(A)$ has $\chi_A \in \text{span}\{\chi_{A_i} : 1 \leq i \leq t\}$. This can be proven by using uncrossing techniques that exploit submodularity of $r_{\mathcal{M}'}$, see Chapter 5 of [10].

Now, as $A_{i-1} \subsetneq A_i$ for $1 < i \leq t$ and $x^*_e > 0$ for each $e \in F$, we see $r_{\mathcal{M}'}(A_i) = x^*(A_i) > x^*(A_{i-1}) = r_{\mathcal{M}'}(A_{i-1})$. Since the ranks are integral and $r_{\mathcal{M}'}(A_1) \neq 0$ (as $A_1 \neq \emptyset$), then $r_{\mathcal{M}}(A_i) \geq i$ for all $1 \leq i \leq t$.

Note that $|F| \leq t + |W|$ because of the number of non-zero (fractional) variables is at most the size of a basis for the tight constraints. We have

$$\begin{aligned}\sum_{v \in W} |\delta_F(v)| - x^*(\delta_F(v)) &\leq \sum_{v \in V} |\delta_F(v)| - x^*(\delta_F(v)) \leq k \cdot (|F| - x^*(F)) \\ &\leq k \cdot (|F| - r_{\mathcal{M}'}(A_t)) \leq k \cdot (|F| - t) \leq k \cdot |W|.\end{aligned}$$

The second bound holds because each edge has at most k endpoints, so it can contribute $1 - x^*_e \geq 0$ at most k times throughout the sum. Thus, some $v \in W$ satisfies the claim. ◀

▶ **Lemma 9.** *Algorithm 1 returns a set* $\mathsf{M}' \in \mathcal{I}$ *such that* $d_v(\delta_{\mathsf{M}'}(v) - L(v)) \leq b_v$ *where* $L(v)$ *denotes the* $\min\{k, |\delta_{\mathsf{M}'}(v)|\}$ *edges* $e \in \delta_{\mathsf{M}'}(v)$ *with greatest demand* $d_{v,e}$ *across* v.

Proof. We know $\mathsf{M}' \in \mathcal{I}$ by Lemma 7. Consider an iteration where a vertex $v \in W$ is removed from W. Claim 8 shows $|\delta_F(v)| \leq x^*(\delta_F(v)) + k$.

Let $F^k_v = \{e_1, \ldots, e_k\}$ be the k edges of this iteration in $\delta_F(v)$ having largest demand (if $|\delta_F(v)| < k$ then let $F^k_v = \delta_F(v)$). Then

$$\sum_{e \in \delta_F(v) - F^k_v} d_{v,e} \leq \sum_{e \in \delta_F(v)} d_{v,e} \cdot x^*_{e,v} \leq b'_v.$$

The first bound follows because if we shift x^*-values from larger- to smaller-demand edges the value $\sum_{e \in \delta_F(v)} d_{v,e} x^*_{e,v}$ does not increase. We can continue to do this until each $e \in \delta_F(v) - F^k_v$ has one unit of x^*-mass because $|\delta_F(v)| - k \leq x^*(\delta_F(v))$.

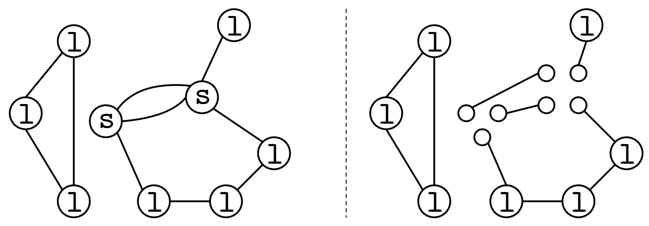

Figure 1 **Left**: The graph with vertex labels `s` and `l` and edges A. **Right**: The graph $\overline{G}$ obtained by "shattering" the `s` vertices. Notice the maximum degree is 2, the `ss` edges are isolated, and the `sl` edges lie on paths.

At this point of the algorithm, we have $d_v(\delta_{\mathsf{M}}(v)) = b_v - b'_v$ (letting M denote the set M' from the current iteration). So $d_v(\delta_F(v) - F_v^k) + d_v(\delta_{\mathsf{M}}(v)) \leq b_v$. We conclude by noting the edges returned by the algorithm contains only edges in $\mathsf{M} \cup F$ so $d_v(\mathsf{M}' - L(v)) \leq b_v$. ◀

Proof of Corollary 2. If there is no feasible solution to (**LP-M**), then there can be no integral solution. Otherwise, we use the same iterated relaxation technique as in Algorithm 1, except on (**LP-B**), whose polytope is the restriction of the polytope from (**LP-M**) to the base polytope of $\mathcal{M}$ (which is also integral, Corollary 40.2d of [14]).

All arguments are proven in essentially the same way. So we can find, in polynomial time, a base B with $p(\mathsf{B}) \leq \mathrm{OPT}(\textbf{LP-B})$ where $d_v(\delta_{\mathsf{B}}(v)) \leq b_v + k \cdot \max_{e \in \delta_{\mathsf{B}}(v)} d_{v,e}$. ◀

2.2 Pruning phase

We focus on $\mathsf{GDM}_{\mathsf{M}}$ ($k = 2$) in this section and show how to prune a set $\mathsf{M}' \subseteq E$ satisfying the properties of Lemma 9 to a feasible solution $\mathsf{M} \subseteq \mathsf{M}'$ while controlling the loss in its value. Each part of Theorem 1 is proved through the following lemmas. In each, for a vertex $v \in V$ we let $L(v)$ be the two edges with highest d_v-value in $\delta_{\mathsf{M}'}(v)$ (or $L(v) = \delta_{\mathsf{M}'}(v)$ if $|\delta_{\mathsf{M}'}(v)| \leq 1$). We also let $S(v) = \delta_{\mathsf{M}'}(v) - L(v)$ be the remaining edges. Note $d_v(\delta_{S(v)}(v)) \leq b_v$.

▶ **Lemma 10.** *For arbitrary graph G and arbitrary demands, we can find a feasible demand matching $\mathsf{M} \subseteq \mathsf{M}'$ with $p(\mathsf{M}) \geq p(\mathsf{M}') \cdot \frac{3}{25}$.*

Proof. For each vertex v, label v randomly with `s` with probability α or with `l` with probability $1 - \alpha$ (for α to be chosen later). Say $e \in \mathsf{M}'$ *agrees* with the labelling for an endpoint v if either $e \in S(v)$ and v is labelled `s`, or $v \in L(v)$ and v is labelled `l`. Let $\mathsf{A} \subseteq \mathsf{M}'$ be the edges agreeing with the labelling on both endpoints.

Modify the graph (V, A) by replacing each $v \in V$ labelled `s` with $|\delta_{\mathsf{A}}(v)|$ vertices and reassigning the endpoint v of each $e \in \delta_{\mathsf{A}}(v)$ to one of these vertices in a one-to-one fashion. See Figure 1 for an illustration. Call this new graph $\overline{G}$.

Each vertex in $\overline{G}$ has degree at most 2 so $\overline{G}$ decomposes naturally into paths and cycles. Each path with ≥ 2 edges can be decomposed into 2 matchings and each cycle can be decomposed into 3 matchings. Randomly choose one such matching for each path and cycle to keep and discarding the remaining edges on these paths and cycles. Note edges uv of $\overline{G}$ where u and v both had degree 1 are not discarded.

Let M be the resulting set of edges, viewed in the original graph G. Note that M is feasible: any vertex labelled `s` already had its capacity satisfied by A because $\delta_{\mathsf{A}}(v) \subseteq S(v)$. Any vertex labelled `l` has at most one of its incident edges in A chosen to stay in M.

Let $e = uv \in \mathsf{M}'$, we place a lower bound on $\mathbf{Pr}[e \in \mathsf{M}]$ by analyzing a few cases.

- If $e \in S(u) \cap S(v)$, then $\mathbf{Pr}[e \in \mathsf{M}] = \mathbf{Pr}[e \in \mathsf{A}] = \alpha^2$.

- If $e \in S(u) \cap L(v)$ or vice-versa then $\mathbf{Pr}[e \in \mathsf{M}] = \alpha \cdot (1-\alpha)/2$ (note e does not lie on a cycle in $\overline{G}$ since one endpoint is labelled $\mathbf{s}$).
- If $e \in L(u) \cap L(v)$ then $\mathbf{Pr}[e \in \mathsf{M}] = (1-\alpha)^2/3$.

Choosing $\alpha = 2/5$, we have $\mathbf{E}[p(\mathsf{M})] \geq p(\mathsf{M}') \cdot \frac{3}{25}$. ◀

We can efficiently derandomize this technique as follows. First, we use a pairwise independent family of random values to generate a probability space over labelings of V with $O(|V|)$ events such that the distribution of labels over pairs $u, v \in V$ is the same as with independently labelling the vertices. See Chapter 11 of [12] for details of this technique. For each such labelling, we decompose the paths and cycles of $\overline{G}$ into matchings and keep the most profitable matching from each path and cycle instead of randomly picking one.

▶ **Lemma 11.** *For a conflict-free instance of* GDM_M*, we can find a feasible solution* $\mathsf{M} \subseteq \mathsf{M}'$ *with* $p(\mathsf{M}) \geq \frac{p(\mathsf{M}')}{4}$.

Proof. The set A from the proof of Lemma 10 is already feasible so it does not need to be pruned further. In this case, choose $\alpha = 1/2$. ◀

▶ **Lemma 12.** *If the given graph G is bipartite, then we can find a feasible solution* $\mathsf{M} \subseteq \mathsf{M}'$ *with* $p(\mathsf{M}) \geq \frac{p(\mathsf{M}')}{7}$.

Proof. Say V_L, V_R are the two sides of V. We first partition M' into 4 groups:

$$\{uv \in \mathsf{M}' : uv \in S(u) \cap S(v)\},$$
$$\{uv \in \mathsf{M}' : uv \in L(u) \cap L(v)\},$$
$$\{uv \in \mathsf{M}' : uv \in S(u) \cap L(v)\},$$
$$\{uv \in \mathsf{M}' : uv \in L(u) \cap S(v)\}.$$

The first set is feasible. The latter three sets can each be partitioned into two feasible sets as follows. For one of these sets, form $\overline{G}$ as in the proof of Lemma 10. Each cycle can also be decomposed into two matchings because G, thus $\overline{G}$, is bipartite. Between all sets listed above, we have partitioned M' into 7 feasible sets. Let M be one with maximum profit. ◀

▶ **Lemma 13.** *For an arbitrary graph $G = (V, E)$ with a consistent ordering on edges, we can find a feasible demand matching* $\mathsf{M} \subseteq \mathsf{M}'$ *with* $p(\mathsf{M}) \geq \frac{p(\mathsf{M}')}{5}$.

Proof. We partition M' into five groups in this case. Consider the edges in decreasing order of the consistent ordering. When edge $e = uv$ is considered, assign it to a group that does not include edges in $L(u) \cup L(v)$ that come before e in the ordering. As $|L(u) \cup L(v)| \leq 4$, the edges can be partitioned into five groups this way. Each group A is a feasible demand matching since $\delta_\mathsf{A}(v) \subseteq S(v)$ or $|\delta_\mathsf{A}(v)| = 1$ for each vertex v. Now let M be the group with maximum profit, so $p(\mathsf{M}) \geq \frac{p(\mathsf{M}')}{5}$. ◀

▶ **Lemma 14.** *If $d_{v,e} \leq \epsilon \cdot b_v$ for each $v \in V, e \in \delta(v)$, we can find a feasible demand matching* $\mathsf{M} \subseteq \mathsf{M}'$ *with* $p(\mathsf{M}) \geq (1 - O(\epsilon^{1/3})) \cdot p(\mathsf{M}')$.

This is proven using a common randomized pruning procedure so the proof is skipped in this extended abstract. See, for example, [3] for a similar treatment in another packing problem.

3 Demand Matching in Excluded-Minor Families

In this section we prove GDM admits a PTAS in graphs that exclude a fixed graph as a minor. Our proof of Theorem 4 (the **NP**-hardness) appears in the full version. Throughout we let OPT denote the optimum solution value to the given GDM instance.

Let H be a graph and let $\mathcal{G}_H$ be all graphs that exclude H as a minor. Our PTAS uses the following decomposition.

▶ **Theorem 15** (Demaine, Hajiaghayi, and Kawarabayashi [4]). *There is a constant c_H depending only on H such that for any k and any $G \in \mathcal{G}_H$, the vertices V of G can be partitioned into $k+1$ disjoint sets so that the union of any k of these sets induce a graph with treewidth bounded by $c_H \cdot k$. Such a partition can be found in time that is polynomial in $|V|$.*

Using this decomposition, we have a PTAS for GDM when $G \in \mathcal{G}_H$ if we have a PTAS for GDM in bounded-treewidth graphs.

Intuition for our approach is given at the end of Section 1.1. We assume, for simplicity, that all $d_{v,e}$-values are distinct so we can naturally speak of *the* largest demands in a set. This is without loss of generality, we could scale demands and capacities by a common value so they are integers and then subtract $\frac{2i+j}{3|E|^2}$ from the j'th endpoint of the i'th edge according to some arbitrary ordering. Such a perturbation does not change feasibility of solutions as the total amount subtracted from all edges is < 1.

3.1 A Sparsification Lemma

We present our sparsification lemma, which even holds for general instances of GDM_M.

▶ **Lemma 16** (Sparsification Lemma). *For each $\epsilon > 0$ there is a feasible solution $\mathsf{M} \subseteq E$ with the following properties.*

- $p(\mathsf{M}) \geq (1-2\epsilon) \cdot \mathrm{OPT}$
- *for each $v \in V$, there is some $M_v \subseteq \mathsf{M}$ with $|M_v| \leq 1/\epsilon^2$ such that $d_{v,e} \leq \epsilon \cdot (b_v - d_v(\delta_{M_v}(v)))$ for all $e \in \delta_{\mathsf{M}-M_v}(v)$*

Think of M_v as the "large" edges in $\delta_\mathsf{M}(v)$ and $\delta_{\mathsf{M}-M_v}(v)$ as the "small" edges in $\delta_\mathsf{M}(v)$. Note that some $e \in \mathsf{M}$ may be designated large on one endpoint and small on the other.

Proof. Let M^* be an optimum solution. For each $v \in V$, if $|\delta_{\mathsf{M}^*}(v)| \geq 1/\epsilon^2$ then let L_v be the $1/\epsilon^2$ edges in $\delta_{\mathsf{M}^*}(v)$ with greatest d_v-demand and R_v be a random subset of L_v of size $1/\epsilon$. If $|\delta_{\mathsf{M}^*}(v)| < 1/\epsilon^2$, simply let $L_v = \delta_{\mathsf{M}^*}(v)$ and $R_v = \emptyset$.

Set $\mathsf{M} = \mathsf{M}^* - \cup_{v \in V} R_v$ and for each $v \in V$ set $M_v = \mathsf{M} \cap L_v$. Clearly M is feasible as it is a subset of the optimum solution. For each $e = uv \in \mathsf{M}^*$, e lies in R_u or R_v with probability at most ϵ each, so $\mathbf{Pr}[e \notin \mathsf{M}] \leq 2\epsilon$. Thus, $\mathbf{E}[p(\mathsf{M})] \geq (1-2\epsilon) \cdot \mathrm{OPT}$.

Now we focus on proving the second property for M. Let v be an arbitrary vertex in V. By construction $|M_v| \leq |L_v| \leq 1/\epsilon^2$. If $|R_v| = 0$ then $\delta_{\mathsf{M}-M_v}(v) = \emptyset$, otherwise, $|R_v| = 1/\epsilon$ and for each remaining $e \in \delta_{\mathsf{M}-M_v}(v)$, we note that $d_{v,e} + d_v(\delta_{M_v}(v)) + \sum_{e' \in R_v} d_{v,e'} \leq b_v$ because the terms represent a subset of edges of M^* incident to v. Rearranging and using the fact that $d_{v,e'} \geq d_{v,e}$ for any $e' \in R_v$ shows $\frac{1}{\epsilon} \cdot d_{v,e} \leq b_v - d_v(\delta_{M_v}(v))$. ◀

This motivates the following notion of a relaxed solution.

▶ **Definition 17.** An ϵ-relaxed solution is a subset $\mathsf{M} \subseteq E$ along with sets $M_v \subseteq \delta_\mathsf{M}(v)$ with $|M_v| \leq 1/\epsilon^2$ for each $v \in V$ such that the following hold. First, let $\overline{b}_v = b_v - d_v(\delta_{M_v}(v))$ for each $v \in V$. Next, for each $e \in \delta_{\mathsf{M}-M_v}(v)$, let $d'_{v,e}$ be the value of $d_{v,e}$ rounded down to the nearest integer multiple of $\frac{\epsilon}{|E|}\overline{b}_v$. Then the following must hold:

- **Large Edge Feasibility**: $d_v(\delta_{M_v}(v)) \leq b_v$ for each $v \in V$.
- **Small Edges**: $d_{v,e} \leq \epsilon \overline{b}_v$ for each $v \in V$ and each $e \in \delta_{\mathsf{M}-M_v}(v)$.
- **Discretized Small Edge Feasibility**: $d'_v(\delta_{\mathsf{M}-M_v}(v)) \leq \overline{b_v}$ for each $v \in V$.

The set M in an ϵ-relaxed solution is not necessarily a feasible GDM solution under the original demands d. As we will see shortly, it can be pruned to get a feasible solution without losing much value. Note the scaling from d to d' for some of the edges e in the definition is done independently for each endpoint of e: the demand at different endpoints may be shifted down by different amounts.

Sometimes we informally say just a set $\mathsf{M} \subseteq E$ itself is an ϵ-relaxed solution even if we do not explicitly mention the corresponding M_v sets.

▶ **Lemma 18.** *Let* M *be an* ϵ*-relaxed solution with maximum possible value* $p(\mathsf{M})$*. Then* $p(\mathsf{M}) \geq (1 - 2\epsilon) \cdot OPT$*.*

Proof. The set M and its corresponding M_v subsets from Lemma 16 suffices. ◀

▶ **Lemma 19.** *Given any* ϵ*-relaxed solution* $\mathsf{M} \subseteq E$*, we can efficiently find some* $\mathsf{M}' \subseteq \mathsf{M}$ *that is a feasible* GDM *solution with* $p(\mathsf{M}') \geq (1 - O(\epsilon^{1/3})) \cdot p(\mathsf{M})$*.*

The idea is that the $\{0, 1\}$ indicator vector of M is almost a feasible solution to (**LP-M**) with the trivial matroid $\mathcal{I} = 2^E$ in the residual instance after all "large" edges are packed so it can be pruned to a feasible solution while losing very little value by appealing to the last bound in Theorem 1. There is a minor subtlety in how to deal with edges that are both "small" and "large". The proof is deferred to the full version.

3.2 A Dynamic Programming Algorithm

Suppose $G = (V, E)$ has treewidth at most τ and that we are given a tree decomposition $\mathcal{T} = (\mathcal{B}, E_{\mathcal{T}})$ of G where each $B \in \mathcal{B}$ has $|B| \leq \tau + 1$. Recall this means the following:

1. For each $v \in V$, the set of bags $\mathcal{B}_v = \{B \in \mathcal{B} : v \in B\}$ form a connected subtree of $\mathcal{T}$.
2. For each $uv \in E$, there is at least one bag $B \in \mathcal{B}$ with $u, v \in B$.

Let $B^r \in \mathcal{B}$ be some arbitrarily chosen *root* bag. View $\mathcal{T}$ as being rooted at B^r. We may assume that each $B \in \mathcal{B}$ has at most two children. In fact, it simplifies our recurrence a bit to assume each $B \in \mathcal{B}$ is either a leaf in $\mathcal{T}$ or has precisely two children. This is without loss of generality. Arbitrarily order the children of a non-leaf vertex so one is the *left* child and one is the *right* child. For a bag B, let $\mathcal{T}_B$ be the subtree of $\mathcal{T}$ rooted at B (so $\mathcal{T}_{B^r} = \mathcal{T}$).

For each $v \in V$, say $\overline{B}_v$ is the bag containing v that is closest to the root B^r. Note for $uv \in E$ with $\overline{B}_u \neq \overline{B}_v$ that one of $\overline{B}_u$ or $\overline{B}_v$ lies on the path between the other and B^r (by the properties of tree decompositions). For each $B \in \mathcal{B}$ and each $v \in B$, we partition a subset of the edges of $\delta(v)$ into four groups:

- $\delta^{\texttt{here}}(v : B) = \{uv \in \delta(v) : \overline{B}_u = B\}$.
- $\delta^{\texttt{left}}(v : B) = \{uv \in \delta(v) : \overline{B}_u$ lies in the left subtree of $B\}$.
- $\delta^{\texttt{right}}(v : B) = \{uv \in \delta(v) : \overline{B}_u$ lies in the right subtree of $B\}$.
- $\delta^{\texttt{up}}(v : B) = \{uv \in \delta(v) : \overline{B}_u$ lies between B and $B^r\}$.

The only other edges $uv \in \delta(v)$ not accounted for here do not have $\overline{B}_u$ in either $\mathcal{T}_B$ or between B and B^r. We note if $B = \overline{B}_v$, then every edge in $\delta(v)$ lies in one of the four groups and for any $uv \in \delta^{\texttt{up}}(v : B)$ we must have $u \in B$ (otherwise no bag contains u and v, which is impossible since $uv \in E$) and, consequently, $\overline{B}_u$ lies between B and B^r. This will be helpful to remember when we describe the recurrence.

Dynamic Programming States

Let $\Delta := \{\texttt{here}, \texttt{left}, \texttt{right}, \texttt{up}\}$ be the set of "directions" used above. The DP states are given by tuples Φ with the following components.

- A bag $B \in \mathcal{B}$.
- For each $v \in B$, a subset $M_v \subseteq \delta(v)$ with $|M_v| \leq 1/\epsilon^2$.
- For each $v \in B$ and $\kappa \in \Delta$, an integer $a_{v,\kappa} \in \{0, \ldots, |E|/\epsilon\}$ such that $\sum_{\kappa \in \Delta} a_{v,\kappa} \leq \frac{|E|}{\epsilon}$.

The number of such tuples is at most $|\mathcal{B}| \cdot |E|^{O(\tau/\epsilon^2)} \cdot (|E|/\epsilon)^{O(\tau)}$, which is polynomial in G when τ and ϵ are regarded as constants. The idea behind $a_{v,\kappa}$ is that it describes how to reserve the discretized d'_v-demand for edges $uv \in \delta^\kappa(v : B) - M_v$. Of course, other edges in $\delta(v)$ not in the partitions $\delta^\kappa(v : B)$ may be in an optimal ϵ-relaxed solution. They will either be explicitly guessed in M_v or will be considered in a state higher up the tree by the time the bag $\overline{B}_v$ is processed.

Dynamic Programming Values

For each such tuple $\Phi = (B; \langle M_v \rangle_{v \in B}; \langle a_{v,\kappa} \rangle_{v \in B, \kappa \in \Delta})$, we let $f(\Phi)$ denote the maximum total value of an ϵ-relaxed solution $\mathsf{M}' \subseteq E$ (with corresponding large sets M'_v for $v \in V$) satisfying the following properties. We slightly abuse notation and say $v \in \mathcal{T}_B$ for some $v \in V$ if v lies in some bag of the subtree $\mathcal{T}_B$.

- Each $uv \in \mathsf{M}'$ has at least one endpoint in $\mathcal{T}_B$.
- $M'_v = M_v$ for each $v \in B$.
- Each $uv \in \mathsf{M}'$ with both $\overline{B}_u, \overline{B}_v \notin \mathcal{T}_B$ lies in $M'_u \cup M'_v$.
- For $v \in B$ let $\overline{b_v} = b_v - d_v(\delta_{M'_v}(v))$. For $\kappa \in \Delta$ and $v \in B$, it must be that $d'_v(\delta^\kappa(v : B) \cap \mathsf{M}' - M'_v) \leq a_{v,\kappa} \cdot \frac{\epsilon}{|E|} \cdot \overline{b_v}$ where $d'_{v,e}$ is the largest integer multiple of $\frac{\epsilon}{|E|} \cdot \overline{b_v}$ that is at most $d_{v,e}$ for $e \in \delta_{\mathsf{M}' - M'_v}(v)$.

The last point is a bit technical. Intuitively, it says the scaled demand of small edges incident to v coming from some direction $\kappa \in \Delta$ fit in the capacity of v reserved for that direction.

If there is no such F, we say $f(\Phi) = -\infty$. Note the maximum of $f(\Phi)$ over all configurations Φ for the root bag B_r is the maximum value over all ϵ-relaxed solutions.

3.2.1 The Recurrence

For the sake of space, details behind the recurrence are deferred to the full version. We just outline the main ideas. A tuple Φ is a base case if the bag B is a leaf of $\mathcal{T}$. In this case, only edges in some $\delta^\kappa(v : B)$ set for $\kappa \in \{\texttt{here}, \texttt{up}\}$ are considered (there are none in the directions `left`, `right`). We find the optimal way to pack such edges that are not part of a "large" set M_v while ensuring the d'_v-demands do not violate the residual capacities $\overline{b_v}$ and, in particular, for each direction κ we ensure this packing does not violate the part of the residual capacity for that direction allocated by the $a_{v,\kappa}$ values. This subproblem is just the Multi-Dimensional Knapsack problem with $2|B|$ knapsacks. A standard pseudopolynomial-time algorithm can be used to solve it as the scaled demands are from a polynomial-size discrete range.

For the recursive step, we try all pairs of configurations $\Phi^{\texttt{left}}, \Phi^{\texttt{right}}$ that are "consistent" with Φ. Really this just means they agree on the sets M_v for shared vertices v and they agree on how much demand $a_{v,\kappa}$ should be allocated for each direction. For each such consistent pair, we pack small edges in $\delta^{\texttt{here}}(v : B)$ and $\delta^{\texttt{up}}(v : B)$ optimally such that their scaled demands do not violate the $a_{v,\kappa}$-capacities, again using Multi-Dimensional Knapsack.

References

1 Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Andreas Wiese. A mazing $2 + \varepsilon$ approximation for unsplittable flow on a path. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 26–41. Society for Industrial and Applied Mathematics, 2014.

2 Nikhil Bansal, Nitish Korula, Viswanath Nagarajan, and Aravind Srinivasan. Solving packing integer programs via randomized rounding with alterations. *Theory of Computing*, 8(1):533–565, 2012.

3 Gruia Calinescu, Amit Chakrabarti, Howard Karloff, and Yuval Rabani. An improved approximation algorithm for resource allocation. *ACM Transactions on Algorithms*, 7, 2011.

4 Erik D. Demaine, Mohammad Taghi Hajiaghayi, and Ken-ichi Kawarabayashi. Algorithmic graph minor theory: Decomposition, approximation, and coloring. In *Proceedings of the Forty-sixth Annual IEEE Symposium on Foundations of Computer Science*, pages 637–646. IEEE, 2005.

5 Takuro Fukunaga and Hiroshi Nagamochi. Network design with weighted degree constraints. *Discrete Optimization*, 7(4):246–255, 2010.

6 Georgii Gens and Evgenii Levner. Complexity of approximation algorithms for combinatorial problems: a survey. *ACM SIGACT News*, 12(3):52–65, 1980.

7 Mohammad Ghodsi, Hamid Mahini, Kian Mirjalali, Shayan Oveis Gharan, Morteza Zadimoghaddam, et al. Spanning trees with minimum weighted degrees. *Information Processing Letters*, 104(3):113–116, 2007.

8 Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

9 Madhukar Korupolu, Adam Meyerson, Rajmohan Rajaraman, and Brian Tagiku. Coupled and k-sided placements: generalizing generalized assignment. *Mathematical Programming*, 154(1-2):493–514, 2015.

10 Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative methods in combinatorial optimization*, volume 46. Cambridge University Press, 2011.

11 Michael J. Magazine and Maw-Sheng Chern. A note on approximation schemes for multidimensional knapsack problems. *Mathematics of Operations Research*, 9(2):244–247, 1984.

12 Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

13 Ojas Parekh. Iterative packing for demand and hypergraph matching. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 349–361. Springer, 2011.

14 Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2003.

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

16 Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, pages 661–670. ACM, 2007.

17 Mohit Singh and Hehui Wu. Nearly tight linear programming bounds for demand matching in bipartite graphs. `http://cgi.cs.mcgill.ca/~hehui/paper/Demand_matching.pdf`, 2012.

18 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, pages 681–690. ACM, 2006.