Bicriteria Approximation Algorithms for Demand Matching

Yuchong Pan
MIT
yuchong@mit.edu

Michel X. Goemans
MIT
goemans@math.mit.edu

Abstract

The demand matching problem is a common generalization of the knapsack problem and the b-matching problem, in which integer demands on the edges are present. We present three (α, β)-bicriteria approximation algorithms for the demand matching problem, each of which outputs a subset of edges that violates each vertex capacity by at most β times the maximum demand, and whose total weight is at least a constant fraction $1/\alpha$ of the total weight of an optimal solution of the original instance.

In particular, the first algorithm is a combinatorial algorithm inspired by the well-known greedy algorithm of Dantzig for the fractional knapsack problem, which produces a $(2, 1)$-bicriteria approximation. Indeed, we prove more generally the existence of a $(k, 1)$-bicrteria approximation algorithm for the k-hypergraph demand matching problem, which includes the demand matching problem when $k = 2$. The second algorithm uses the iterative relaxation technique of Jain and a structural lemma of an extreme point of the natural LP relaxation of the demand matching problem, and produces a $(3/2, 1)$-bicriteria approximation. The third algorithm takes the better of two rounding strategies in the iterative relaxation framework when an odd cycle is present, and yields a $(4/3, 1)$-bicriteria approximation. The weight guarantees of all three approximation algorithms are with respect to the optimal value of the natural LP relaxation, where for the third algorithm we assume that the demand of each edge is at most the capacity of each of its endpoints.

1 Introduction

We consider the Demand Matching problem introduced by Shepherd and Vetta [25]. Let $G = (V, E)$ be a graph. Let $b : V \to \mathbb{N}$ denote *vertex capacities*. Let $d, w : E \to \mathbb{N}$ denote *edge demands* and *edge weights*, respectively. We say that a subset $M \subseteq E$ is a *demand matching* if $d(M \cap \delta(v)) \leq b(v)$ for all $v \in V$, where $\delta(v)$ denotes the set of edges of G incident to v. To emphasize the edge demands d and the vertex capacities b, we might say that M is (b, d)-*feasible*. The objective of the Demand Matching problem is to find a demand matching M such that $w(M)$ is maximized. A natural LP relaxation of the Demand Matching problem is the following:

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} x_e w(e) & & \text{(DM-LP)} \\ \text{subject to} \quad & \sum_{e \in \delta(v)} x_e d(e) \leq b(v) & \forall v \in V & \\ & x_e \in [0, 1] & \forall e \in E. & \end{aligned}$$

A well-known special case of the Demand Matching problem is the case where $d(e) = 1$ for all $e \in E$. This is the familiar Weighted b-Matching problem, which admits several polynomial-time algorithms [17, 21, 7, 3, 8]. In contrast, when G has only two vertices, the Demand Matching

problem reduces to the Knapsack problem. It follows that the Demand Matching problem is NP-hard.

The Demand Matching problem has been extensively studied, and several approximation algorithms are known. For $\alpha \geq 1$, we say that an algorithm is an *α-approximation* for the Demand Matching problem if, given an instance $(G = (V, E), b, d, w)$ with optimal value OPT, it outputs in polynomial time a (b, d)-feasible demand matching $M \subseteq E$ such that $w(M) \geq \mathsf{OPT}/\alpha$. We say that an approximation algorithm for the Demand Matching problem is *with respect to* (DM-LP) if OPT in the above definition can be replaced by the optimal value of (DM-LP).

Shepherd and Vetta [25] gave a 3.264-approximation algorithm for general graphs and a 2.764-approximation algorithm for bipartite graphs, both with respect to (DM-LP), and also showed that the Demand Matching problem is APX-hard even in bipartite graphs. Parekh [22] improved the upper bound on the integrality gap of (DM-LP) to 3, which matches the lower bound by Shepherd and Vetta [25]. For bipartite graphs, Singh and Wu [27] improved the upper bound on the integrality gap to 2.709, with a lower bound of 2.699.

The motivation of this work is to allow a (small) additive violation on each vertex capacity, in order to reduce the approximation factor on the weight. In other words, we give *resource augmentation* results, where we compare the quality of the outputs of our algorithms to instances with fewer resources (namely, smaller vertex capacities). Specifically, for $\alpha \geq 1$ and $\beta \geq 0$, we say that an algorithm is an *(α, β)-bicriteria approximation* for the Demand Matching problem if, given an instance $(G = (V, E), b, d, w)$ with optimal value OPT, it outputs in polynomial time a subset $M \subseteq E$ such that $w(M) \geq \mathsf{OPT}/\alpha$ and $d(M \cap \delta(v)) \leq b(v) + \beta \cdot d_{\max}$ for all $v \in V$, where $d_{\max} := \max_{e \in E} d(e)$.

Resource augmentation results have been established for some other combinatorial optimization problems including the Minimum Cost Bounded Degree Spanning Tree problem [13, 14, 23, 9, 4, 26], the Single-Source Unsplittable Flow problem [6, 12, 28, 18, 19, 29, 2, 16], and the k-Edge-Connected Spanning Subgraph problem [10, 20, 15]. These works have introduced a variety of techniques for obtaining resource augmentation results.

However, to the best of our knowledge, very few results of this type are known for the Demand Matching problem. One example is the work of Ahmadian and Friggstad [1], who studied a generalization of the Demand Matching problem and used iterative relaxation techniques to obtain a $(1, 2)$-bicriteria approximation algorithm. The main contribution of this paper is three bicriteria approximation algorithms for the Demand Matching problem.

Our first result is the analysis of a simple greedy approximation algorithm for the Demand Matching problem. Indeed, we give more generally a $(k, 1)$-bicriteria approximation algorithm for the k-Hypergraph Demand Matching problem, which we define below. In the k-Hypergraph Demand Matching problem, we are given a hypergraph $H = (V, \mathcal{E})$ such that each hyperedge contains at most k vertices, vertex capacities $b : V \to \mathbb{N}$, hyperedge demands $d : \mathcal{E} \to \mathbb{N}$, and hyperedge weights $w : \mathcal{E} \to \mathbb{N}$, and the objective is to find a subset $\mathcal{M} \subseteq \mathcal{E}$ such that $d(\mathcal{M} \cap \delta(v)) \leq b(v)$ for all $v \in V$, and that $w(\mathcal{M})$ is maximized; here $\delta(v)$ refers to the set of hyperedges containing v. When $k = 2$, this is the Demand Matching problem. A natural LP relaxation of the k-Hypergraph Demand Matching problem is the following:

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in \mathcal{E}} x_e w(e) && && (k\text{-HDM-LP}) \\ \text{subject to} \quad & \sum_{e \in \delta(v)} x_e d(e) \leq b(v) && \forall v \in V \\ & x_e \in [0, 1] && \forall e \in \mathcal{E}. \end{aligned}$$

The notions of α-approximation algorithms and (α, β)-bicriteria approximation algorithms for the k-Hypergraph Demand Matching problem are defined analogously to those for the Demand Matching problem, where we redefine $d_{\max} := \max_{e \in \mathcal{E}} d(e)$. Parekh [22] gave a deterministic $2k$-approximation algorithm for the k-Hypergraph Demand Matching problem with respect to (k-HDM-LP) (and more generally for the k-Column-Sparse Packing Integer Program problem), while the integrality gap is at least $2(k - 1 + 1/k)$.

Theorem 1.1. *There exists a combinatorial $(k, 1)$-bicriteria approximation algorithm for the k-Hypergraph Demand Matching problem with respect to* (k-HDM-LP)*. In particular, there is a combinatorial $(2, 1)$-bicriteria approximation algorithm for the Demand Matching problem with respect to* (DM-LP)*.*

Although our $(k, 1)$-bicriteria approximation algorithm is with respect to (k-HDM-LP), the algorithm is combinatorial and (k-HDM-LP) is not used in the algorithm; (k-HDM-LP) is used only in the analysis of the weight guarantee. Indeed, the algorithm is greedy and inspired by the well-known greedy algorithm by Dantzig [5] for the Fractional Knapsack problem.

Our second result is a $(3/2, 1)$-bicriteria approximation algorithm for the Demand Matching problem using the iterative relaxation technique of Jain [11] and the structure of an extreme point of (DM-LP).

Theorem 1.2. *There is a $(3/2, 1)$-bicriteria approximation algorithm for the Demand Matching problem with respect to* (DM-LP)*. For an instance (G, b, d, w) with G bipartite, this algorithm gives a $(1, 1)$-bicriteria approximation.*

Ahmadian and Friggstad [1] proved, for a generalization of the Demand Matching problem, a slightly weaker structure of a purely fractional extreme point $x^\star \in \mathbb{R}^E$ of (DM-LP), namely that $|\delta(v)| \leq x^\star(\delta(v)) + 2$ for *some* vertex $v \in V$, and used this structure to design a $(1, 2)$-bicriteria approximation algorithm also in the iterative relaxation framework of Jain [11]. We show, by a simple counting argument, that, if $x^\star \in \mathbb{R}^E$ is a purely fractional extreme point of (DM-LP) satisfying $|\delta(v)| \geq 2$ for every nonrelaxed vertex v, then $|\delta(v)| = 2$ for *every* nonrelaxed vertex v, and therefore its support graph is a vertex-disjoint union of odd cycles.

Finally, we say that an instance $(G = (V, E), b, d, w)$ of the Demand Matching problem satisfies the *no-bottleneck assumption* if, for each $e = uv \in E$, we have $d(e) \leq \min\{b(u), b(v)\}$. This assumption can be made without loss of generality without affecting the optimal value of the original instance, although it may affect the optimal value of the LP relaxation. For the Demand Matching problem with the no-bottleneck assumption, we combine the idea of the algorithm from Theorem 1.2 with an additional ingredient to give a $(4/3, 1)$-bicriteria approximation algorithm. In particular, when dealing with an odd cycle, we take the better of two rounding strategies, while in the algorithm from Theorem 1.2 we always discard an edge e with minimum $x^\star_e w(e)$ along the cycle.

Theorem 1.3. *For the Demand Matching problem with the no-bottleneck assumption, there is a $(4/3, 1)$-bicriteria approximation algorithm with respect to* (DM-LP)*.*

1.1 Organization

In section 2, we present a trivial greedy $(1, 1)$-bicriteria approximation algorithm for the Knapsack problem, and a greedy $(k, 1)$-bicriteria approximation algorithm for the k-Hypergraph Demand Matching problem, proving Theorem 1.1. In section 3, we prove a structural lemma for a purely fractional extreme point of (DM-LP), and give a $(3/2, 1)$-bicriteria approximation algorithm for

the Demand Matching problem, proving Theorem 1.2. In section 4, we give a $(4/3, 1)$-bicriteria approximation algorithm for the Demand Matching problem with the no-bottleneck assumption, proving Theorem 1.3. In appendix A, we show that the analysis of the approximation factor of k on the weight is tight for the greedy algorithm.

2 Greedy Algorithm

2.1 Greedy Algorithm for Knapsack

As a warm-up, we first show that a very simple greedy algorithm, presented in Algorithm 1, provides a $(1, 1)$-bicriteria approximation for the Knapsack problem.

Algorithm 1: A greedy algorithm for the Knapsack problem.

Input: $b \in \mathbb{N}$, and $d, w : [m] \to \mathbb{N}$.
Output: $M \subseteq [m]$.

```
M ← ∅.
Sort elements e ∈ [m] in a nonincreasing order of w(e)/d(e), breaking ties arbitrarily.
foreach e ∈ [m] in this order do
    if d(M) ≤ b then
        M ← M ∪ {e}.
    end
end
return M
```

It is clear that the subset $M \subseteq [m]$ output by Algorithm 1 satisfies the capacity guarantee $d(M) \leq b + d_{\max}$. For the weight guarantee, recall that the Fractional Knapsack problem admits a greedy exact algorithm by Dantzig [5]: sort all elements $e \in [m]$ in a nonincreasing order of $w(e)/d(e)$, and while the total weight of chosen fractional elements is less than b, take the largest possible fraction of each element in this order. In comparison, Algorithm 1 only differs in the fact that it takes the fractional element fully. Hence, the weight of the subset $M \subseteq [m]$ output by Algorithm 1 is at least the optimal value of the Fractional Knapsack problem, and thus Algorithm 1 provides a $(1, 1)$-bicriteria approximation for the Knapsack problem.

2.2 Greedy Algorithm for k-Hypergraph Demand Matching

Now, we present a greedy algorithm for the k-Hypergraph Demand Matching problem in Algorithm 2, which is inspired by the greedy algorithm for the Knapsack problem from section 2.1.

It is clear that the subset $\mathcal{M} \subseteq \mathcal{E}$ output by Algorithm 2 satisfies the capacity guarantee.

Lemma 2.1. *The subset $\mathcal{M} \subseteq \mathcal{E}$ output by Algorithm 2 satisfies $d(\mathcal{M} \cap \delta(v)) \leq b(v) + d_{\max}$ for all $v \in V$.*

We prove the weight guarantee for Algorithm 2.

Lemma 2.2. *Let $\mathcal{M} \subseteq \mathcal{E}$ be the output of Algorithm 2. Let $x^\star \in \mathbb{R}^{\mathcal{E}}$ be a feasible solution to* (k-HDM-LP). *Then*

$$k \cdot w(\mathcal{M}) \geq \sum_{e \in \mathcal{E}} x_e^\star w(e).$$

Algorithm 2: A greedy algorithm for the k-Hypergraph Demand Matching problem.

Input: a hypergraph $H = (V, \mathcal{E})$, $b : V \to \mathbb{N}$, and $d, w : \mathcal{E} \to \mathbb{N}$.
Output: $\mathcal{M} \subseteq \mathcal{E}$.

$\mathcal{M} \leftarrow \emptyset$.
Sort hyperedges $e \in \mathcal{E}$ in a nonincreasing order of $w(e)/d(e)$, breaking ties arbitrarily.
foreach $e \in \mathcal{E}$ *in this order* **do**
  if $d(\mathcal{M} \cap \delta(v)) \leq b(v)$ *for all* $v \in e$ **then**
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{e\}$.
  end
end
return $\mathcal{M}$

Proof. We say that the nonincreasing order of the hyperedges used in Algorithm 2 is the *greedy order*. For each $e \in \mathcal{E}$, let $\rho(e) := w(e)/d(e)$, and let $\mathsf{prev}(e)$ be the set of hyperedges preceding e in the greedy order. For each $e \in \mathcal{R} := \mathcal{E} \setminus \mathcal{M}$, fix $a(e) \in e$ so that, at the iteration where e is considered,

$$d(\mathcal{M} \cap \delta(a(e)) \cap \mathsf{prev}(e)) > b(a(e)).$$

For each $v \in V$, let

$$\mathcal{R}(v) := \{e \in \delta(v) : a(e) = v\}.$$

Let $V' := \{v \in V : \mathcal{R}(v) \neq \emptyset\}$. For each $v \in V'$, let e_v be the first hyperedge in $\mathcal{R}(v)$ in the greedy order, and let

$$\mathcal{P}(v) := \mathcal{M} \cap \delta(v) \cap \mathsf{prev}(e_v).$$

Fix $v \in V'$. Since $x^\star$ is feasible in (k-HDM-LP),

$$d(\mathcal{P}(v)) > b(v) \geq \sum_{e \in \delta(v)} x^\star_e d(e) \geq \sum_{e \in \mathcal{R}(v)} x^\star_e d(e) + \sum_{e \in \mathcal{P}(v)} x^\star_e d(e).$$

Hence,

$$\sum_{e \in \mathcal{R}(v)} x^\star_e d(e) < \sum_{e \in \mathcal{P}(v)} (1 - x^\star_e) d(e).$$

Since $\mathcal{P}(v) \subseteq \mathsf{prev}(e_v)$, we have $\rho(e) \geq \rho(e_v)$ for all $e \in \mathcal{P}(v)$. By the definition of e_v, we have $\rho(e) \leq \rho(e_v)$ for all $e \in \mathcal{R}(v)$. Since $\rho(e_v) = w(e_v)/d(e_v) > 0$,

$$\begin{aligned}\sum_{e \in \mathcal{R}(v)} x^\star_e w(e) = \sum_{e \in \mathcal{R}(v)} x^\star_e \rho(e) d(e) &\leq \rho(e_v) \sum_{e \in \mathcal{R}(v)} x^\star_e d(e) \\ &< \rho(e_v) \sum_{e \in \mathcal{P}(v)} (1 - x^\star_e) d(e) \leq \sum_{e \in \mathcal{P}(v)} (1 - x^\star_e) \rho(e) d(e) = \sum_{e \in \mathcal{P}(v)} (1 - x^\star_e) w(e).\end{aligned}$$

Therefore,

$$\sum_{e \in \mathcal{R}} x^\star_e w(e) = \sum_{v \in V'} \sum_{e \in \mathcal{R}(v)} x^\star_e w(e) < \sum_{v \in V'} \sum_{e \in \mathcal{P}(v)} (1 - x^\star_e) w(e) \leq k \sum_{e \in \mathcal{M}} (1 - x^\star_e) w(e),$$

where the last inequality follows from the fact that each hyperedge $e \in \mathcal{M}$ appears in $\mathcal{P}(v)$ for at most k vertices $v \in V'$, namely its endpoints. Hence,

$$\sum_{e\in\mathcal{E}} x^\star_e w(e) = \sum_{e\in\mathcal{R}} x^\star_e w(e) + \sum_{e\in\mathcal{M}} x^\star_e w(e)$$
$$< k \sum_{e\in\mathcal{M}} (1-x^\star_e) w(e) + \sum_{e\in\mathcal{M}} x^\star_e w(e) = \sum_{e\in\mathcal{M}} (k-(k-1)x^\star_e) w(e) \le k \cdot w(\mathcal{M}).$$

This completes the proof. □

Lemmas 2.1 and 2.2 together prove Theorem 1.1.

3 Iterative Relaxation Algorithm

For all $W \subseteq V$ and $F \subseteq E$, we use $\mathsf{LP}(W, F)$ to denote the following LP:

$$\begin{aligned} \text{maximize} \quad & \sum_{e\in F} x_e w(e) \\ \text{subject to} \quad & \sum_{e\in F\cap\delta(v)} x_e d(e) \le b(v) \qquad \forall v \in W \\ & x_e \in [0,1] \qquad \forall e \in F. \end{aligned}$$

First, we prove the following structure of a purely fractional extreme point. This structure is also stated in [25] with a slightly different formulation; we state and prove it here for the sake of completeness.

Lemma 3.1 (Shepherd and Vetta [25])**.** *Let $W \subseteq V$ and $F \subseteq E$. Let $x^\star \in \mathbb{R}^F$ be an extreme point of $\mathsf{LP}(W, F)$ such that $0 < x^\star_e < 1$ for all $e \in F$ and that $|F \cap \delta(v)| \ge 2$ for all $v \in W$. Then (V, F) is a vertex-disjoint union of odd cycles (plus isolated vertices).*

Proof. Since $0 < x^\star_e < 1$ for all $e \in F$, a rank argument (see, e.g., Theorem 5.7 of [24]) implies that $|W| \ge |F|$. Then

$$2|W| \le \sum_{v\in W} |F \cap \delta(v)| \le \sum_{v\in V} |F \cap \delta(v)| = 2|F| \le 2|W|.$$

Hence, $|F \cap \delta(v)| = 2$ for all $v \in W$, and $|F \cap \delta(v)| = 0$ for all $v \in V \setminus W$, and $|F| = |W|$. It follows that (V, F) is a vertex-disjoint union of cycles (plus isolated vertices).

If we consider the full-rank matrix of $\mathsf{LP}(W, F)$ whose columns are indexed by F and rows indexed by W, and scale each column $e \in F$ by $1/d(e)$, we see that any even cycle $C = (V_C, F_C)$ would make this matrix singular since the rows in V_C are not linearly independent. This proves that (V, F) is a vertex-disjoint union of odd cycles (plus isolated vertices). □

Lemma 3.1 suggests an algorithm in the iterative relaxation framework of Jain [11]. We present this algorithm in Algorithm 3.

We analyze the running time and the approximation guarantees of Algorithm 3.

Lemma 3.2. *Algorithm 3 terminates in polynomial time.*

Proof. In each iteration, Algorithm 3 either removes an edge from F, or removes a vertex from W. Hence, Algorithm 3 terminates in $|V| + |E|$ iterations. It is not hard to see that each iteration runs in polynomial time, completing the proof. □

Lemma 3.3. *Let $M \subseteq E$ be the output of Algorithm 3. Then $d(M \cap \delta(v)) \le b(v) + d_{\max}$ for all $v \in V$.*

Algorithm 3: An iterative relaxation algorithm for Demand Matching.

Input: a graph $G = (V, E)$, $b : V \to \mathbb{N}$, and $d, w : E \to \mathbb{N}$.
Output: $M \subseteq E$.

$M \leftarrow \emptyset$, $W \leftarrow V$, $F \leftarrow E$.
while $F \neq \emptyset$ **do**
  Let $x^\star \in \mathbb{R}^F$ be an extreme point optimal solution to $\mathsf{LP}(W, F)$.
  if $x^\star_e = 0$ *for some* $e \in F$ **then**
    $F \leftarrow F \setminus \{e\}$.
  else if $x^\star_e = 1$ *for some* $e = uv \in F$ **then**
    $M \leftarrow M \cup \{e\}$, $F \leftarrow F \setminus \{e\}$, $b(u) \leftarrow b(u) - d(e)$, $b(v) \leftarrow b(v) - d(e)$.
  else if $|F \cap \delta(v)| \leq 1$ *for some* $v \in W$ **then**
    $W \leftarrow W \setminus \{v\}$.
  else
    // $0 < x^\star_e < 1$ `for all` $e \in F$ `and` $|F \cap \delta(v)| \geq 2$ `for all` $v \in W$
    Lemma 3.1 implies that (V, F) is a vertex-disjoint union of odd cycles.
    foreach *odd cycle* $C = (V_C, F_C)$ *in the vertex-disjoint union* **do**
      $e_0 \leftarrow \arg\min_{e \in F_C} x^\star_e w(e)$.
      $F \leftarrow F \setminus \{e_0\}$.
    end
  end
end
return M

Proof. Let $v \in V$. Consider the iteration where v is removed from W. Let $M' \subseteq M$ be the set of edges added to M prior to this iteration. Let $W' \subseteq V$ and $F' \subseteq E$ be the subset of vertices not discarded and the subset of edges not discarded prior to this iteration, respectively. Let $x^\star \in \mathbb{R}^{F'}$ be an extreme point optimal solution to $\mathsf{LP}(W', F')$. It is not hard to show by induction that

$$d(M' \cap \delta(v)) + \sum_{e \in F' \cap \delta(v)} x^\star_e d(e) \leq b(v).$$

Since v is removed from W in this iteration, we have $|F' \cap \delta(v)| \leq 1$. Since $M \subseteq M' \cup F'$,

$$\begin{aligned} d(M \cap \delta(v)) &\leq d(M \cap \delta(v)) + d(F' \cap \delta(v)) \leq b(v) - \sum_{e \in F' \cap \delta(v)} x^\star_e d(e) + d(F' \cap \delta(v)) \\ &= b(v) + \sum_{e \in F' \cap \delta(v)} (1 - x^\star_e) d(e) \leq b(v) + |F' \cap \delta(v)| \cdot 1 \cdot d_{\max} \leq b(v) + d_{\max}. \end{aligned}$$

This completes the proof. □

Lemma 3.4. *Let $M \subseteq E$ be the output of Algorithm 3. Then $w(M)$ is at least $2/3$ times the optimal value of* (DM-LP)*. If G is bipartite, then $w(M)$ is at least the optimal value of* (DM-LP)*.*

Proof. Consider an iteration of Algorithm 3. Let $M' \subseteq M$ be the set of edges added to M prior to this iteration. Let $W' \subseteq V$ and $F' \subseteq E$ be the subset of vertices not discarded and the subset of edges not discarded prior to this iteration, respectively. Let $x^\star \in \mathbb{R}^{F'}$ be an extreme point optimal solution to $\mathsf{LP}(W', F')$. It is not hard to see that removing an edge $e \in F'$ with $x^\star_e = 0$ or moving an edge $e \in F'$ with $x^\star_e = 1$ into M' does not decrease the weight of fixed edges plus the optimal

value of the LP. If we drop a constraint corresponding to some vertex $v \in W'$, the optimal value of $\mathsf{LP}(W' \setminus \{v\}, F')$ is at least the optimal value of $\mathsf{LP}(W', F')$. Therefore, it suffices to consider the case where (V, F') is a vertex-disjoint union of odd cycles.

If G is bipartite, then (V, F') is also bipartite, which implies that there exists no odd cycle in (V, F'). Hence, $w(M)$ is at least the optimal value of (DM-LP) if G is bipartite.

Let $C = (V_C, F'_C)$ be an odd cycle from the vertex-disjoint union. Let $e_0 := \arg\min_{e \in F'_C} x^\star_e w(e)$. Since $|F'_C| \geq 3$,

$$x^\star_{e_0} w(e_0) \leq \frac{1}{3} \sum_{e \in F'_C} x^\star_e w(e).$$

Since at most one edge is removed from each odd cycle, the total weight of all removed edges in this case is at most $1/3$ times the optimal value of (DM-LP). Combined with the other three cases, this proves that $w(M)$ is at least $2/3$ times the optimal value of (DM-LP). □

Lemmas 3.2 to 3.4 together prove Theorem 1.2.

4 Better-of-the-Two Algorithm

In this section, we present a better-of-the-two approximation algorithm in Algorithm 4 for the Demand Matching problem with the no-bottleneck assumption. This algorithm is similar to Algorithm 3 except for the handling of odd cycles. In particular, when the support graph of the extreme point is a vertex-disjoint union of odd cycles as guaranteed by Lemma 3.1, we employ an alternative rounding strategy by taking all edges along the cycle and discarding any chosen edge that is incident to at least one vertex along one of these odd cycles, and then take the better of these two rounding strategies.

We analyze the running time and the approximation guarantees of Algorithm 4.

Lemma 4.1. *Algorithm 4 terminates in polynomial time.*

Proof. In each iteration, Algorithm 4 either removes at least one edge from F, or removes a vertex from W. Hence, Algorithm 4 terminates in $|V| + |E|$ iterations. It is not hard to see that each iteration runs in polynomial time, completing the proof. □

Lemma 4.2. *Let $M \subseteq E$ be the output of Algorithm 4. Then $d(M \cap \delta(v)) \leq b(v) + d_{\max}$ for all $v \in V$.*

Proof. By Lemma 3.3, it suffices to prove the case where M is the subset produced by the new rounding strategy. Let $v \in V$. If v is ever removed from W in Algorithm 4, then $d(M \cap \delta(v)) \leq b(v) + d_{\max}$ by the same argument as in the proof of Lemma 3.3. Hence, it suffices to assume that v belongs to an odd cycle in the vertex-disjoint union guaranteed by Lemma 3.1.

Let $e, f \in E$ be the two edges incident to v in the odd cycle. Since the no-bottleneck assumption is satisfied, we have $\max\{d(e), d(f)\} \leq b(v)$. Since Algorithm 4 discards all edges incident to v that are already chosen,

$$d(M \cap \delta(v)) = d(e) + d(f) \leq b(v) + d_{\max}.$$

This completes the proof. □

Lemma 4.3. *Let $M \subseteq E$ be the output of Algorithm 4. Then $w(M)$ is at least $3/4$ times the optimal value of* (DM-LP).

Algorithm 4: A better-of-the-two algorithm for the DEMAND MATCHING problem.

Input: a graph $G = (V, E)$, $b : V \to \mathbb{N}$, and $d, w : E \to \mathbb{N}$, such that the no-bottleneck assumption is satisfied.

Output: $M \subseteq E$.

Let $M_1 \subseteq E$ be the output of Algorithm 3 on input (G, b, d, w).
$M_2 \leftarrow \emptyset$, $W \leftarrow V$, $F \leftarrow E$.
while $F \neq \emptyset$ **do**
    Let $x^\star \in \mathbb{R}^F$ be an extreme point optimal solution to $\mathsf{LP}(W, F)$.
    if $x^\star_e = 0$ *for some* $e \in F$ **then**
        $F \leftarrow F \setminus \{e\}$.
    else if $x^\star_e = 1$ *for some* $e = uv \in F$ **then**
        $M_2 \leftarrow M_2 \cup \{e\}$, $F \leftarrow F \setminus \{e\}$, $b(u) \leftarrow b(u) - d(e)$, $b(v) \leftarrow b(v) - d(e)$.
    else if $|F \cap \delta(v)| \leq 1$ *for some* $v \in W$ **then**
        $W \leftarrow W \setminus \{v\}$.
    else
        // $0 < x^\star_e < 1$ `for all` $e \in F$ `and` $|F \cap \delta(v)| \geq 2$ `for all` $v \in W$
        Lemma 3.1 implies that (V, F) is a vertex-disjoint union $\mathcal{C}$ of odd cycles.
        $R \leftarrow \{e \in M_2 : e \text{ is incident to some vertex } v \in \bigcup_{(V_C, F_C) \in \mathcal{C}} V_C\}$.
        $M_2 \leftarrow (M_2 \setminus R) \cup \bigcup_{(V_C, F_C) \in \mathcal{C}} F_C$.
        $F \leftarrow \emptyset$.
    end
end
return $\arg\max_{M \in \{M_1, M_2\}} w(M)$

Proof. Let $M_1 \subseteq E$ be the subset produced by the rounding strategy of Algorithm 3. Let $M_2 \subseteq E$ be the subset produced by the new rounding strategy. Then

$$w(M) = \max\{w(M_1), w(M_2)\} \geq \frac{3}{4}w(M_1) + \frac{1}{4}w(M_2).$$

Let LPOPT be the optimal value of (DM-LP). Let $\mathcal{C}$ be the vertex-disjoint union of odd cycles guaranteed by Lemma 3.1. Let $W' \subseteq V$ and $F' \subseteq E$ be the subset of vertices not discarded and the subset of edges not discarded prior to this iteration, respectively. Let $x^\star \in \mathbb{R}^{F'}$ be an extreme point optimal solution to $\mathsf{LP}(W', F')$. Let

$$\mu := \sum_{(V_C, F'_C) \in \mathcal{C}} \min_{e \in F'_C} x^\star_e w(e),$$
$$\nu := \sum_{(V_C, F'_C) \in \mathcal{C}} \sum_{e \in F'_C} w(e).$$

The proof of Lemma 3.4 shows that

$$w(M_1) \geq \mathsf{LPOPT} - \mu,$$

and that

$$\mu \leq \frac{1}{3} \sum_{(V_C, F'_C) \in \mathcal{C}} \sum_{e \in F_C} x^\star_e w(e) \leq \frac{1}{3} \sum_{(V_C, F'_C) \in \mathcal{C}} \sum_{e \in F_C} w(e) = \frac{\nu}{3}.$$

Let $M'_2 \subseteq M_2$ be the subset of edges already chosen prior to this iteration. Let $R \subseteq M'_2$ be the subset of edges incident to some vertex $v \in \bigcup_{(V_C, F'_C) \in \mathcal{C}} V_C$. Since $R \subseteq M'_2$,

$$w(M_2) = w(M'_2) - w(R) + \nu \geq \nu.$$

Hence,

$$w(M) \geq \frac{3}{4}w(M_1) + \frac{1}{4}w(M_2) \geq \frac{3}{4}(\mathsf{LPOPT} - \mu) + \frac{1}{4}\nu = \frac{3}{4}\mathsf{LPOPT} + \frac{1}{4}(\nu - 3\mu) \geq \frac{3}{4}\mathsf{LPOPT}.$$

This completes the proof. □

Lemmas 4.1 to 4.3 together imply Theorem 1.3.

References

[1] S. Ahmadian and Z. Friggstad. Further approximations for demand matching: Matroid constraints and minor-closed graphs. *arXiv preprint arXiv:1705.10396*, 2017.

[2] D. Alemán-Espinosa and N. Kumar. Unsplittable multicommodity flows in outerplanar graphs. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 385–399. Springer, 2025.

[3] R. P. Anstee. A polynomial algorithm for b-matchings: an alternative approach. *Information Processing Letters*, 24(3):153–157, 1987.

[4] K. Chaudhuri, S. Rao, S. Riesenfeld, and K. Talwar. What would Edmonds do? Augmenting paths and witnesses for degree-bounded MSTs. *Algorithmica*, 55(1):157–189, 2009.

[5] G. B. Dantzig. Discrete-variable extremum problems. *Operations research*, 5(2):266–288, 1957.

[6] Y. Dinitz, N. Garg, and M. X. Goemans. On the single-source unsplittable flow problem. *Combinatorica*, 19(1):17–41, 1999.

[7] H. N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 448–456, 1983.

[8] A. M. H. Gerards. Matching. *Handbooks in operations research and management science*, 7: 135–224, 1995.

[9] M. X. Goemans. Minimum bounded degree spanning trees. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 273–282. IEEE, 2006.

[10] D. E. Hershkowitz, N. Klein, and R. Zenklusen. Ghost value augmentation for k-edge-connectivity. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1853–1864, 2024.

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

[12] S. G. Kolliopoulos and C. Stein. Approximation algorithms for single-source unsplittable flow. *SIAM Journal on Computing*, 31(3):919–946, 2001.

[13] J. Könemann and R. Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 537–546, 2000.

[14] J. Könemann and R. Ravi. Primal-dual meets local search: approximating MST's with nonuniform degree bounds. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 389–395, 2003.

[15] N. Kumar and C. Swamy. Almost tight additive guarantees for k-edge-connectivity. *arXiv preprint arXiv:2506.20906*, 2025.

[16] M. Majthoub Almoghrabi, M. Skutella, and P. Warode. Integer and unsplittable multiflows in series-parallel digraphs. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 427–441. Springer, 2025.

[17] A. B. Marsh III. *Matching algorithms*. PhD thesis, The Johns Hopkins University, 1979.

[18] M. Martens, F. Salazar, and M. Skutella. Convex combinations of single source unsplittable flows. In *European Symposium on Algorithms*, pages 395–406. Springer, 2007.

[19] S. Morell and M. Skutella. Single source unsplittable flows with arc-wise lower and upper bounds. *Mathematical Programming*, 192(1):477–496, 2022.

[20] Z. Nutov and R. Cohen. Bicriteria approximation for k-edge-connectivity. In *33rd Annual European Symposium on Algorithms (ESA 2025)*, pages 66–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2025.

[21] M. W. Padberg and M. R. Rao. Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, 7(1):67–80, 1982.

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

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

[24] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Springer, 2003.

[25] F. B. Shepherd and A. Vetta. The demand-matching problem. *Mathematics of Operations Research*, 32(3):563–578, 2007.

[26] M. Singh and L. C. Lau. Approximating minimum bounded degree spanning trees to within one of optimal. *Journal of the ACM (JACM)*, 62(1):1–19, 2015.

[27] M. Singh and H. Wu. Nearly tight linear programming bounds for demand matching in bipartite graphs, 2012.

[28] M. Skutella. Approximating the single source unsplittable min-cost flow problem. *Mathematical Programming*, 91(3):493–514, 2002.

[29] V. Traub, L. V. Koch, and R. Zenklusen. Single-source unsplittable flows in planar graphs. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 639–668. SIAM, 2024.

A Tightness of the Weight Guarantee for the Greedy Algorithm

In this appendix, we show that the analysis of the approximation factor of k on the weight in the proof of Lemma 2.2 is tight for Algorithm 2.

Lemma A.1. *Let $\varepsilon > 0$. Then there exists an instance $(H = (V, \mathcal{E}), b, d, w)$ of the k-Hypergraph Demand Matching problem for which Algorithm 2 outputs a subset $\mathcal{M} \subseteq \mathcal{E}$ such that $w(\mathcal{M})$ is at most $(1/k + \varepsilon)$ times the optimal value of the instance. In other words, Algorithm 2 cannot guarantee an approximation factor better than k on the weight, even when an additive violation of $d_{\max}$ on each vertex capacity is allowed. Furthermore, this instance satisfies the no-bottleneck assumption, i.e., $d(e) \leq b(v)$ for all $e \in \mathcal{E}$ and $v \in e$.*

Proof. Let $H = (V, \mathcal{E})$ be the hypergraph defined as follows. Let $U := \{u_1, \ldots, u_{k+1}\}$ be a set of $k+1$ distinct vertices. For each $i \in [k+1]$, let $V_i := \{v_{i,1}, \ldots, v_{i,k-1}\}$ be a set of $k-1$ distinct vertices, and let $e_i := U \setminus \{u_i\}$ and $f_i := \{u_i, v_{i,1}, \ldots, v_{i,k-1}\}$ be two hyperedges of size k. Let $\mathcal{E}_0 := \{e_1, \ldots, e_{k+1}\}$ and $\mathcal{F} := \{f_1, \ldots, f_{k+1}\}$. Define $V := U \cup V_1 \cup \ldots \cup V_{k+1}$ and $\mathcal{E} := \mathcal{E}_0 \cup \mathcal{F}$. Each vertex in U belongs to exactly k hyperedges in $\mathcal{E}_0$, and to exactly one hyperedge in $\mathcal{F}$.

Let $D \in \mathbb{N}$ be such that $(k+1)/(k(kD-1)) \leq \varepsilon$. Let $B := kD - 1$. Define $b(v) := B$ for all $v \in V$. Define $d(e) := D$ for all $e \in \mathcal{E}_0$, and $d(e) := B$ for all $e \in \mathcal{F}$. Define $w(e) := D + 1$ for all $e \in \mathcal{E}_0$ and $w(e) := B$ for all $e \in \mathcal{F}$. One can readily check that this instance (H, b, d, w) satisfies the no-bottleneck assumption.

We analyze the behavior of Algorithm 2. We have $w(e)/d(e) = (D+1)/D > 1$ for all $e \in \mathcal{E}_0$, and $w(e)/d(e) = B/B = 1$ for all $e \in \mathcal{F}$. Algorithm 2 first considers all hyperedges in $\mathcal{E}_0$ (in an arbitrary order), and accepts all of $\mathcal{E}_0$ since each vertex in U belongs to exactly k hyperedges in $\mathcal{E}_0$.

At this point, the load at each vertex in U is $kD > B$. Algorithm 2 then considers all hyperedges in $\mathcal{F}$ (in an arbitrary order), and rejects all of $\mathcal{F}$ since the load at each vertex in U has already exceeded its capacity. Hence, Algorithm 2 outputs $\mathcal{M} = \mathcal{E}_0$ with $w(\mathcal{M}) = (k+1)(D+1)$.

In contrast, an optimal solution of this instance selects all of $\mathcal{M}^\star = \mathcal{F}$ with $w(\mathcal{M}^\star) = (k+1)B = (k+1)(kD-1)$. Hence,

$$\frac{w(\mathcal{M})}{w(\mathcal{M}^\star)} = \frac{(k+1)(D+1)}{(k+1)(kD-1)} = \frac{1}{k} + \frac{k+1}{k(kD-1)} \leq \frac{1}{k} + \varepsilon.$$

It follows that $w(\mathcal{M}) \leq (1/k + \varepsilon) \cdot w(\mathcal{M}^\star)$, completing the proof. □