Approximate Bipartite b-Matching using Multiplicative Auction[⋆]

Bhargav Samineni[1], S M Ferdous[2], Mahantesh Halappanavar[2], and Bala Krishnamoorthy[3]

[1] The University of Texas at Austin, sbharg@utexas.edu
[2] Pacific Northwest National Laboratory, {sm.ferdous,hala}@pnnl.gov
[3] Washington State University, kbala@wsu.edu

Abstract. Given a bipartite graph $G(V = (A \cup B), E)$ with n vertices and m edges and a function $b\colon V \to \mathbb{Z}_+$, a *b-matching* is a subset of edges such that every vertex $v \in V$ is incident to at most $b(v)$ edges in the subset. When we are also given edge weights, the Max Weight b-Matching problem is to find a b-matching of maximum weight, which is a fundamental combinatorial optimization problem with many applications. Extending on the recent work of Zheng and Henzinger (IPCO, 2023) on standard bipartite matching problems, we develop a simple *auction* algorithm to approximately solve Max Weight b-Matching. Specifically, we present a multiplicative auction algorithm that gives a $(1 - \varepsilon)$-approximation in $O(m\varepsilon^{-1} \log \varepsilon^{-1} \log \beta)$ worst case time, where β the maximum b-value. Although this is a $\log \beta$ factor greater than the current best approximation algorithm by Huang and Pettie (Algorithmica, 2022), it is considerably simpler to present, analyze, and implement.

Keywords: b-Matching · Auctions · Approximation Algorithms

1 Introduction

Matching, also known as the *assignment* problem, in a bipartite graph is one of the most fundamental discrete optimization problems, which is rich in theory, algorithms, and applications. A weighted matching (also known as the linear sum assignment problem) in a bipartite graph aims to find a pairing of vertices between the partitions with the maximum sum of edge weights in the matching. There are many classic applications of maximum weighted bipartite matchings (MWM) in resource allocation and assignments [13,32], and we also observe many emerging applications [1,6,16,18,33,36,42,44].

Classic *exact* algorithms for MWM include the celebrated primal-dual Hungarian algorithm [29,35], which is expensive and has little to no parallelism. Alternative *approximate* approaches include a class of algorithms called *auction* algorithms [9,10,15], which treat the matching process as a welfare-maximizing

[⋆] The manuscript is accepted as a refereed paper in the 2024 INFORMS Optimization Society conference.

allocation of *objects* to *bidders*, where objects and bidders are vertices in the bipartitions of the graph. Auction algorithms assign prices to the objects and let *eligible* bidders bid on objects with the maximum utility. The bidder outbids any previous bids for the object making the previous bidder eligible again. By formulating the bid values in a certain manner, the auction eventually terminates at an equilibrium where all the bidders are *approximately* happy with the object they win. Auction based approaches are easier to present, analyze, and implement as they only involve performing a series of simple local updates with good empirical performance [4, 39, 40]. Often, the runtime complexity of the classic auction algorithms is pseudo-polynomial as they depend on the maximum weight of the graph. There is a renewed interest in auction algorithms focusing on improving the runtime and developing algorithms in *scalable computational models* such as distributed, streaming, and parallel models [5, 30, 45]. In particular, a recent result of Zheng and Henzinger [45] shows a multiplicative auction algorithm achieving a $(1-\varepsilon)$-approximate matching in $O(m\varepsilon^{-1}\log\varepsilon^{-1})$ time, which matches the more complicated state-of-the-art algorithm of Duan and Pettie [17].

In this paper, our focus is on the bipartite b-matching problem, which generalizes a matching by allowing each vertex $v \in V$ to be incident to at most $b(v)$ edges in the matching. Weighted bipartite b-matchings ($\mathsf{MW}b\text{-}\mathsf{M}$) are particularly suited for recommendation and assignment applications where multiple choices are preferred. This naturally occurs in modern applications like movie recommendations, route suggestions, and ad allocations. Consequently, $\mathsf{MW}b\text{-}\mathsf{M}$ has been used in protein structure alignment [28], computer vision [8], estimating text similarity [37], reviewers assignment in peer-review systems [14, 31, 43], and diverse assignment [2, 3]. Although $\mathsf{MW}b\text{-}\mathsf{M}$ has been extensively explored in different algorithmic paradigms, little is known in terms of auction algorithms.

To extend the auction paradigm to b-matchings, we must deal with the issue of objects being matched to multiple bidders. Indeed, this breaks the analogy of an auction, as it makes little sense for multiple people to win a single object. To deal with this, we make a simple modification: instead of allowing up to $b(j)$ bidders to bid on and win a single object j, we create a set $M(j)$ of $b(j)$ identical *copies* of j for bidders to bid on. Additionally, instead of an object j explicitly maintaining a price, each copy of j maintains its own price. The auction now works by matching bidders to copies of an object, potentially outbidding the current match, and then updating the prices of the object copies.

Using this modification and adapting the multiplicative auction algorithm of Zheng and Henzinger [45], we design an $(1-\varepsilon)$-approximate auction algorithm for $\mathsf{MW}b\text{-}\mathsf{M}$. In particular, we interpret the auction process for b-matching in a primal-dual linear programming framework. In Section 4, we describe *approximate* complementary slackness conditions and prove that any primal-dual variable pairs that obey them have a desired approximation guarantee. In Section 5, we present a $(1-\varepsilon)$-approximation algorithm based on this analysis and the modifications discussed earlier by extending the recent multiplicative auction algorithm of Zheng and Henzinger [45] to $\mathsf{MW}b\text{-}\mathsf{M}$. The worst case running time is shown as $O(m\varepsilon^{-1}\log\varepsilon^{-1}\log\beta)$, which is a $\log\beta$ factor greater than the running time of the state-of-the-art approximation algorithm by Huang and Pettie [25] when restricted to bipartite graphs. While our algorithm has a runtime dependence on β, it is reasonable to assume that $\beta = O(1)$ for many real-world applications. Also, the algorithm of Huang and Pettie [25] is relatively complicated to analyze and implement even for bipartite graphs, as it is based on the scaling framework [17, 22]. In contrast, the multiplicative auction algorithm we present is considerably simpler in both regards.

2 Preliminaries

Let $G = (V = (A \cup B), E, w)$ be a simple undirected bipartite graph with bipartitions A and B, $n := |V|$ vertices, $m := |E|$ edges, and weights $w\colon E \to \mathbb{R}_{\geq 0}$. For a vertex $v \in V$, denote by $\deg(v)$ and $N(v)$ the number of edges it is incident to and its set of neighbors in G, respectively. For a subset of edges $H \subseteq E$, let $\deg_H(v)$ denote the number of edges in H that v is incident to. We define $\Delta := \max_{v \in V} \deg(v)$. Given a function $b\colon V \to \mathbb{Z}_+$, a b-matching (also known as f-matching or degree-constrained subgraph) is a subset of edges $F \subseteq E$ such that $\deg_F(v) \leq b(v)$ for all $v \in V$, where we can assume without loss of generality $1 \leq b(v) \leq \deg(v)$. We denote $\beta := \max_{v \in V} b(v)$. A vertex v is *saturated* by F if $\deg_F(v) = b(v)$, and it is *unsaturated* by F if $\deg_F(v) < b(v)$. Additionally, we let $F(v)$ denote the set of vertices v is matched to under F. For a real-valued function f defined on the elements of a set Y, we use the standard summing notation $f(Y) := \sum_{y \in Y} f(y)$. Without loss of generality, we assume $b(A) \leq b(B)$; thus the size of any b-matching is at most $b(A) \leq \frac{b(V)}{2}$. The Max Weight b-Matching (MWb-M) problem is to find a b-matching F that maximizes $w(F)$ given a weighted bipartite graph $G = (V = (A \cup B), E, w)$ and function $b\colon V \to \mathbb{Z}_+$ as input.

3 Related Work

The auction approach for Max Weight Matching (MWM) can be attributed to Demange et al. [15] and Bertsekas [9, 10], who also extended it to the Assignment, Transportation, and general network flow problems [11, 12]. Recently, Assadi et al. [5] gave an auction algorithm for $(1-\varepsilon)$-approximate Max Cardinality Matching that yields algorithms in the semi-streaming [19] and MPC [26] models of computation. Liu et al. [30] extended on this work to develop auction algorithms for $(1-\varepsilon)$-approximate MWM and Max Cardinality b-Matching that work in various scalable models of computation. We note that the algorithm of Liu et al. [30] is the only auction approach for b-matching that we are aware of, and it works only for the unweighted version of the problem. Additionally, Zheng and Henzinger [45] developed multiplicative auction algorithms to give a $O(m\varepsilon^{-1} \log \varepsilon^{-1})$ time auction algorithm for $(1-\varepsilon)$-approximate MWM.

Several algorithms for special cases of MWb-M exist. For integral edge weights, Gabow and Tarjan [23] developed exact scaling algorithms by reducing to finding a perfect b-matching on a bipartite multigraph, while Huang and Kavitha [24] give an exact $O(b(V)^{1/2}mW)$ time algorithm by decomposing into W unweighted subproblems. In general graphs, finding a max weight b-matching is also well studied. Gabow [21] gave an exact $O(b(V)\min\{m\log n, n^2\})$ time algorithm. Bayati et al. [7] proposed an exact algorithm based on belief propagation. Huang and Pettie [25] presented a $(1-\varepsilon)$-approximate scaling algorithm with running time $O(m\alpha(m,n)\varepsilon^{-1}\log\varepsilon^{-1})$, where $\alpha(m,n)$ is the inverse Ackermann function. For bipartite graphs, the running time reduces to $O(m\varepsilon^{-1}\log\varepsilon^{-1})$. There are also several $\frac{1}{2}$ and $(\frac{2}{3}-\varepsilon)$-approximation algorithms designed for b-matching [27,34,38] in the sequential and parallel models. Finding a max weight b-matching can also be reduced to a standard matching problem [20,25,41]. However, the reduction is not approximation preserving, and may drastically increase the size of the graph.

4 Primal-Dual Analysis for *b*-Matchings

Given a graph $G = (V = (A \cup B), E, w)$, we refer to vertices in A as *bidders* and vertices in B as *objects*. For each edge $(i,j) \in E$, define an indicator variable $x(i,j) \in \{0,1\}$. The LP-Relaxation of MWb-M and its dual are then given by

$$\begin{array}{llll}
\max & \sum_{(i,j)\in E} w(i,j)x(i,j) & & \\
\text{s.t.} & \sum_{j\in N(i)} x(i,j) \le b(i) & \forall i \in A \\
 & \sum_{i\in N(j)} x(i,j) \le b(j) & \forall j \in B \\
 & 0 \le x(i,j) \le 1 & \forall (i,j) \in E.
\end{array}
\qquad
\begin{array}{lll}
\min & \sum_{i\in A} b(i)\pi(i) + \sum_{j\in B} b(j)p(j) + \sum_{(i,j)\in E} z(i,j) & \\
\text{s.t.} & \pi(i) + p(j) + z(i,j) \ge w(i,j) & \forall (i,j) \in E \\
 & \pi(i) \ge 0 & \forall i \in A \\
 & p(j) \ge 0 & \forall j \in B \\
 & z(i,j) \ge 0 & \forall (i,j) \in E.
\end{array}$$

The dual variables z are defined for every edge, while the dual variables π and p are defined for vertices in A and B, respectively. We refer to the dual $\pi(i)$ for a bidder i as its *profit* and the dual $p(j)$ for an object j as its *price*.

Property 1 (Complementary Slackness). Let x and (π, p, z) be feasible primal and dual solutions, and let F be the b-matching induced by x. Then these are *optimal* solutions if and only if the following conditions hold:

1) $x(i,j) > 0 \implies \pi(i) + p(j) + z(i,j) = w(i,j)$ for all $(i,j) \in E$.
2) $x(i,j) < 1 \implies z(i,j) = 0$ for all $(i,j) \in E$.
3) $\sum_{j\in N(i)} x(i,j) < b(i) \implies \pi(i) = 0$ for all $i \in A$.
4) $\sum_{i\in N(j)} x(i,j) < b(j) \implies p(j) = 0$ for all $j \in B$.

The first two conditions can be restated as $\pi(i) + p(j) \le w(i,j)$ if $(i,j) \in F$ and $\pi(i) + p(j) \ge w(i,j)$ if $(i,j) \notin F$, respectively, while the last two conditions indicate any unsaturated vertices have an optimal dual value of zero.

By Properties 1.1 and 1.2, maintaining the edge duals z is redundant as their minimizing value can be given explicitly by $z(i,j) = \max\{w(i,j) - \pi(i) - p(j), 0\}$.

We can also partially restate the conditions for all $(i, j) \in F$ as

$$w(i,j) - p(j) \geq \pi(i) \geq \max\left\{ \max_{k \in N(i) \setminus F(i)} \{w(i,k) - p(k)\}, 0 \right\}, \tag{1}$$

where $F(i)$ is the set of objects i is matched to under F. The upper and lower bound on $\pi(i)$ follows from the slackness condition when $(i, j) \in F$ and when $(i, j) \notin F$, respectively, along with the non-negativity constraints of the dual variables. Additionally, owing to Property 1.3, if a bidder i is unsaturated in an optimal solution, then $\pi(i) = 0$ which implies $\max_{k \in N(i) \setminus F(i)} \{w(i,k) - p(k)\} \leq 0$. Hence, if we are given some arbitrary prices p, a feasible value of π respecting Property 1 is implicitly given by the lower bound of Eq. (1).

This motivates the naive auction based approach, where both the primal and dual problems are simultaneously solved, but only the primal and price variables are maintained explicitly. Consider a set of matched edges F and prices p. Eq. (1) motivates the construction of F and p such that every edge $(i, j) \in F$ satisfies

$$w(i,j) - p(j) \geq \max\left\{ \max_{k \in N(i) \setminus F(i)} \{w(i,k) - p(k)\}, 0 \right\} =: \pi(i).$$

We call such an edge *happy*. For a bidder i, if all edges $(i, j) \in F$ are happy and either i is saturated by F and $\pi(i) \geq 0$ or i is unsaturated by F and $\pi(i) = 0$, then we also call i happy. If all bidders are happy under F, then F is also happy. If F is happy, a feasible b-matching, and together with prices p satisfies Property 1.4 (i.e. unsaturated objects have a dual value of zero), then it must be optimal since it satisfies every condition of Property 1.

4.1 ε-Happiness

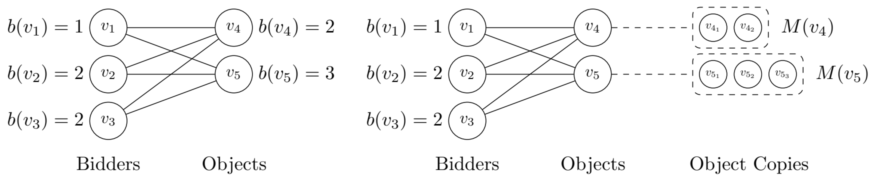
We can relax the notion of happiness to an *approximate* sense by maintaining some multiplicative slack $(1 - \varepsilon)$ for some $\varepsilon > 0$.

Definition 1 (ε-Happiness). *For $F \subseteq E$ and prices p, an edge $(i, j) \in F$ is ε-happy if*

$$w(i,j) - p(j) \geq \max\left\{ \max_{k \in N(i) \setminus F(i)} \{(1-\varepsilon)w(i,k) - p(k)\}, 0 \right\} =: \pi(i).$$

A bidder i is ε-happy if all edges $(i, j) \in F$ are ε-happy and either i is saturated by F and $\pi(i) \geq 0$ ***or*** *i is unsaturated by F and $\pi(i) = 0$.*

Note that F being ε-happy satisfies Property 1.3. If F is ε-happy, a feasible b-matching, and together with prices p satisfies Property 1.4, then its weight falls within a $(1 - \varepsilon)$ factor of the max weight b-matching. To show this, we use a lemma from Huang and Pettie [25] which follows from simple primal-dual arguments. We note that the original lemma is for b-matchings in general graphs, but we specialize it for bipartite graphs and our notation here.



(a) The original graph. **(b)** Adding copies of objects.

Fig. 1: The set of copies associated with each object are given by dashed boxes. During the auction process, bidders bid on and increase the price of a specific copy of an object in order to match to it.

Lemma 1 ([25, Lemma 5]). *Let F be a b-matching and π, p the dual variables for vertices in A and B, respectively. If all unsaturated vertices have dual values of zero, each unmatched edge $(i, j) \notin F$ satisfies $\pi(i) + p(j) \geq (1 - \delta_1)w(i, j)$, and each matched edge $(i, j) \in F$ satisfies $\pi(i) + p(j) \leq (1 + \delta_2)w(i, j)$, then $w(F) \geq (1 - \delta_1)(1 + \delta_2)^{-1}w(F^*)$ where F^* is a max weight b-matching.*

Lemma 2. *Let F, p be some feasible b-matching and prices, respectively, such that F is ε-happy and F, p satisfy Property 1.4. Then $w(F) \geq (1 - \varepsilon)w(F^*)$.*

Proof. For $(i, j) \in F$, we have that $\pi(i) + p(j) \leq w(i, j) - p(j) + p(j) = w(i, j)$ since (i, j) is ε-happy. For $(i, j) \notin F$, we have that

$$\begin{aligned}\pi(i) + p(j) &= \max\left\{\max_{k \in N(i) \setminus F(i)} \{(1 - \varepsilon)w(i, k) - p(k)\}, 0\right\} + p(j) \\ &\geq (1 - \varepsilon)w(i, j) - p(j) + p(j) = (1 - \varepsilon)w(i, j)\end{aligned}$$

since i is ε-happy and $j \in N(i) \setminus F(i)$. By Property 1.4 and F being ε-happy, the dual values of unsaturated vertices are zero. The lemma follows by applying Lemma 1 with $\delta_1 = \varepsilon$, $\delta_2 = 0$. □

5 A Multiplicative Auction Algorithm

The notion of ε-happiness gives us a framework to build an algorithm with approximation guarantees as long as we can maintain that matched edges are ε-happy. However, updating the prices of objects is non-trivial since an object may be matched to multiple bidders, and updating the prices carelessly may break ε-happiness for a matched edge. To resolve this, we introduce the idea of object copies. Specifically, we associate each object $j \in B$ with a set of object copies $M(j) = \{c_1, \ldots, c_{b(j)}\}$, where each object copy $c \in M(j)$ maintains its own price $p_j(c)$. When a bidder wants to match with an object j, it must choose a specific object copy $c \in M(j)$ to bid on and be assigned to. Note that an object copy can be assigned to at most one bidder.

We visualize the role of multiple copies of an object in our algorithm in Figure 1. In this example, an eligible bidder, say v_2, would search through the copies of v_4 and v_5, and then bid on the ones that offer it the best utilities in each. The other eligible bidders would work similarly, potentially outbidding and unmatching someone else on the same object copy. This process continues until an equilibrium is reached. This modification allows us to maintain the overall auction process described for matching problems, with the only changes being that bidders may now bid on multiple things and have to increase the price of a specific copy of j instead of j directly to match to it. The addition of these object copies motivates a slight change to ε-happiness, which we call *strong ε-happiness*.

Definition 2 (Strong ε-Happiness). *For $F \subseteq E$ and prices for each object, an edge $(i, j) \in F$, where i is assigned to $c \in M(j)$, is strongly ε-happy if,*

$$w(i,j) - p_j(c) \geq \max\left\{\max_{k \in N(i)\setminus F(i),\, l \in M(k)} \{(1-\varepsilon)w(i,k) - p_k(l)\}, 0\right\} =: \pi(i).$$

We can also show that strong ε-happiness implies ε-happiness if we set the price of an object j as the minimum of the prices of its object copies.

Proposition 1. *Let F be strongly ε-happy. If we set $p(j) = \min_{c \in M(j)}\{p_j(c)\}$ as the price for each object $j \in B$, then F is ε-happy.*

Proof. Consider an edge $(i, j) \in F$, where i is assigned to $c \in M(j)$. Then,

$$\begin{aligned} w(i,j) - p(j) \geq w(i,j) - p_j(c) &\geq \max\left\{\max_{k \in N(i)\setminus F(i),\, l \in M(k)} \{(1-\varepsilon)w(i,k) - p_k(l)\}, 0\right\} \\ &= \max\left\{\max_{k \in N(i)\setminus F(i)} \{(1-\varepsilon)w(i,k) - p(k)\}, 0\right\} \end{aligned}$$

where the last equality follows from the fact that

$$(1-\varepsilon)w(i,k) - p(k) = (1-\varepsilon)w(i,k) - \min_{l \in M(k)} p_k(l) = \max_{l \in M(k)} (1-\varepsilon)w(i,k) - p_k(l),$$

for all objects $k \in N(i)$.

We now describe our algorithm that adapts the multiplicative auction algorithm of Zheng and Henzinger [45] to b-matchings.

5.1 Weight Preprocessing

We first pre-process the edge weights as described in [45]. If we are given an approximate parameter $\varepsilon' \in (0, 1)$, we can afford to round and scale the edge weights since we are seeking a $(1 - \varepsilon')$-approximate solution. We remove any edge of weight less than $\frac{\varepsilon'}{b(V)}W$ since including $\frac{1}{2}b(V)$ such edges in a b-matching[4]

[4] Note that the size of any b-matching is upper bounded by $b(A) \leq \frac{1}{2}b(V)$

would not even add $\frac{\varepsilon'}{2}W \leq \frac{\varepsilon'}{2}w(F^*)$ to its weight. Hence, we can scale the original weights by $\frac{1}{\varepsilon' W}b(V)$ so that the maximum edge weight is $\frac{b(V)}{\varepsilon'}$ and the minimum edge weight is at least 1. We slightly abuse notation and denote these pre-processed edge weights with the function w. Next, we round down the weights to the nearest integer power of $(1+\varepsilon)$, where $\varepsilon = \frac{1}{2}\varepsilon'$. To do this, we define $\textsc{iLog}(x) = \lfloor \log_{1+\varepsilon}(x) \rfloor$ and the new weights as $\tilde{w}(i,j) = (1+\varepsilon)^{\textsc{iLog}(w(i,j))}$ for each edge $(i,j) \in E$. We remark that finding a $(1-\varepsilon)$-approximate solution with respect to $\tilde{w}$ gives a $(1-\varepsilon')$-approximate solution with respect to w.

Proposition 2. *Let w and $\tilde{w}$ be the weights before and after rounding. To find a $(1-\varepsilon')$-approximate solution with respect to w, it suffices to find a $(1-\varepsilon)$-approximate solution with respect to $\tilde{w}$.*

Proof. Note that $(1+\varepsilon)^{-1}w(i,j) < \tilde{w}(i,j) \leq w(i,j)$ by definition of the rounding. If F^* and $\tilde{F}^*$ are optimal solutions with respect to w and $\tilde{w}$ and F a $(1-\varepsilon)$-approximate solution with respect to $\tilde{w}$, then

$$w(F) \geq \tilde{w}(F) \geq (1-\varepsilon)\tilde{w}(\tilde{F}^*) \geq (1-\varepsilon)\tilde{w}(F^*) > \frac{1-\varepsilon}{1+\varepsilon}w(F^*) \geq (1-2\varepsilon)w(F^*).$$

Since $\varepsilon = \varepsilon'/2$, we get that $w(F) \geq (1-\varepsilon')w(F^*)$. □

We also take note of two important integers when rounding, $s_{\max} = \textsc{iLog}(W) = \textsc{iLog}(b(V)/2\varepsilon) = O(\varepsilon^{-1}\log(b(V)\varepsilon^{-1}))$ and $s_{\min}$, the smallest integer such that $(1+\varepsilon)^{-s_{\min}} \leq \varepsilon$. A simple analysis shown in [45] gives that $s_{\min} = \Theta(\varepsilon^{-1}\log\varepsilon^{-1})$.

5.2 The Algorithm

We present the pseudocode of our algorithm in Algorithm 1. For each object $j \in B$, we initialize its set of copies and set their prices to zero. For each bidder $i \in A$, we build a queue Q_i that contains pairs of the form $(r,(i,j))$ for each $j \in N(i)$ and each integer *index* $r_{ij} - s_{\min} \leq r \leq r_{ij} = \textsc{iLog}(w(i,j))$, where the pairs in Q_i are ordered in non-increasing order based on the index r starting from the top of the queue. To efficiently build the queues, we sort the pairs associated with all the edges using a global bucket sort (lines 8 to 12). With this, we can populate Q_i for each $i \in A$ by going through the indices in decreasing order and inserting any relevant pairs (lines 13 to 15). We also maintain for each $i \in A$ an integer r_i corresponding to the most recent popped index from Q_i.

We maintain a list I of bidders that are not strongly ε-happy. While this list is not empty, we remove a bidder i from it and call the function $\textsc{AssignAndBid}(i)$. This function pops pairs from Q_i until it saturates i or empties Q_i and along the way matches to certain objects. More accurately, suppose a pair $(r,(i,j))$ is popped. If $(i,j) \notin F$, then i will attempt to match with the cheapest object copy $c \in M(j)$ given that $\tilde{w}(i,j) - p_j(c)$ is above a certain threshold. This threshold guarantees that matching to j is a "safe" choice, and we do not have to worry about a better choice coming up later in the queue. If i is matched to c, we add the tuple $\langle j,c\rangle$ to a temporary list T and update the b-matching F to indicate

Algorithm 1 Multiplicative Auction

```
    Input: $G = (V = (A \cup B), E)$, weights $\tilde{w}\colon E \to \mathbb{R}_{\geq 0}$, vertex capacities $b\colon V \to \mathbb{Z}_+$, and $\varepsilon \in (0, \frac{1}{2})$
    Output: A set of edges $F$ such that $F$ is a strongly $\varepsilon$-happy $b$-matching
 1: $I \leftarrow A$, $F \leftarrow \emptyset$
 2: $r_i \leftarrow 0, Q_i \leftarrow \emptyset$ for all $i \in A$
 3: $L_r \leftarrow \emptyset$ for all $r$ from $s_{\max}$ to $-s_{\min}$
 4: for $j \in B$ do
 5:     $M(j) \leftarrow \{c_1, \ldots, c_{b(j)}\}$
 6:     $p_j(c) \leftarrow 0$ for all $c \in M(j)$
 7: // Initialization Phase
 8: for $(i,j) \in E$ do
 9:     $r_{ij} \leftarrow$ iLog$(\tilde{w}(i,j))$
10:     $r_i \leftarrow \max\{r_i, r_{ij}\}$
11:     for $r$ from $r_{ij}$ to $r_{ij} - s_{\min}$ do
12:         $L_r \leftarrow L_r \cup (r, (i,j))$
13: for $r$ from $s_{\max}$ to $-s_{\min}$ do
14:     for $(r, (i,j)) \in L_r$ do
15:         $Q_i$.push$((r, (i,j)))$
16: // Auction Phase
17: while $I \neq \emptyset$ do
18:     Choose some $i \in I$
19:     AssignAndBid$(i)$
20: return $F$
21: procedure AssignAndBid$(i)$
22:     $T \leftarrow \emptyset$                // Temporary List
23:     while $Q_i \neq \emptyset$ and $|F(i)| < b(i)$ do
24:         $(r, (i,j)) \leftarrow Q_i$.pop(), $r_i \leftarrow r$
25:         if $j \in F(i)$ then
26:             Let $c \in M(j)$ be the object copy $i$ is assigned to
27:             if $\tilde{w}(i,j) - p_j(c) \geq (1+\varepsilon)^r$ then
28:                 $T \leftarrow T \cup \{\langle j, c\rangle\}$
29:         else
30:             $c \leftarrow \arg\min_{c' \in M(j)} p_j(c')$
31:             if $\tilde{w}(i,j) - p_j(c) \geq (1+\varepsilon)^r$ then
32:                 Match$(i, \langle j, c\rangle)$, $T \leftarrow T \cup \{\langle j, c\rangle\}$
33:     for $\langle j, c\rangle \in T$ do
34:         $\gamma_{i,c} \leftarrow \tilde{w}(i,j) - p_j(c) - (1-\varepsilon)(1+\varepsilon)^{r_i+1}$     // Bid value
35:         $p_j(c) \leftarrow p_j(c) + \gamma_{i,c}$
36:     $I \leftarrow I \setminus \{i\}$
37: procedure Match$(i, \langle j, c\rangle)$
38:     if $c$ is already assigned to another bidder $y \neq i$ then
39:         $F \leftarrow F \setminus \{(y,j)\}$
40:         if $Q_y \neq \emptyset$ then $I \leftarrow I \cup \{y\}$
41:     $F \leftarrow F \cup \{(i,j)\}$
```

this. If c was previously assigned to some other bidder y, we remove the relevant edge from F and add y back to I if $Q_y \neq \emptyset$ as it may not be strongly ε-happy. Otherwise, if $(i,j) \in F$, where i is assigned to $c \in M(j)$, we add the tuple $\langle j, c\rangle$ to T if the value $\tilde{w}(i,j) - p_j(c)$ is also above a certain threshold. Once i becomes saturated or Q_i is empty, we calculate bids for all the tuples in T based on the current r_i value and perform a price update on all the chosen object copies. Finally, we can remove i from I as we can show it is strongly ε-happy.

5.3 Invariants and Analysis

Throughout the runtime of the algorithm, we maintain the following invariants.

Invariant 1. *Fix any $i \in A$. For all $k \in N(i) \setminus F(i)$ and $l \in M(k)$, $\tilde{w}(i,k) - p_k(l) < \max\{(1+\varepsilon)^{r_i+1}, (1+\varepsilon)^{r_{ik}-s_{\min}}\}$.*

Proof. This is true at the start since $F = \emptyset$, $r_i = \max_{j \in N(i)}\{\text{iLog}(\tilde{w}(i,j))\}$, and all object copy prices are set to zero so $\tilde{w}(i,j) - p_j(c) = \tilde{w}(i,j) < (1+\varepsilon)^{r_i+1}$ for all $j \in N(i)$ and $c \in M(j)$. We show in Invariant 2 that throughout the algorithm the prices of object copies are monotonically increasing. Thus, it suffices to show the inequality holds whenever r_i changes. Note that r_i can only ever decrease. If r_i changes to some value r, then we can guarantee there exists no pairs $(r', (i,k))$ where $r' \geq r+1 > r = r_i$ and $k \in N(i) \setminus F(i)$ in Q_i as such pairs must have been popped and discarded. Additionally, we can guarantee that a lower bound on the indices of any pair popped for a specific k is $r_{ik} - s_{\min}$. Thus, there exists no $k \in N(i) \setminus F(i)$ and $l \in M(k)$ with $\tilde{w}(i,k) - p_k(l) \geq \{(1+\varepsilon)^{r_i+1}, (1+\varepsilon)^{r_{ik}-s_{\min}}\}$ by construction and hence the claim follows. □

Invariant 2. *The prices of object copies are monotonically increasing.*

Proof. Fix some bidder i. Suppose $(r, (i,j))$ was popped from Q_i and i chooses to add some tuple $\langle j, c\rangle$ to T. By construction, $\tilde{w}(i,j) - p_j(c) \geq (1+\varepsilon)^r$. Thus,

$$\gamma_{i,c} = \tilde{w}(i,j) - p_j(c) - (1-\varepsilon)(1+\varepsilon)^{r_i+1} \geq (1+\varepsilon)^r - (1-\varepsilon^2)(1+\varepsilon)^{r_i} > 0$$

where the last inequality follows from the fact that $r_i \leq r$. □

Invariant 3. *At the end of a call to* AssignAndBid(i)*, i is strongly ε-happy.*

Proof. Consider a tuple $\langle j, c\rangle \in T$ during this iteration. Right before we updated $p_j(c)$, Invariant 1 implies that for all $k \in N(i) \setminus F(i)$ and $l \in M(k)$, $\tilde{w}(i,k) - p_k(l) < \max\{(1+\varepsilon)^{r_i+1}, (1+\varepsilon)^{r_{ik}-s_{\min}}\}$. Partition $N(i) \setminus F(i)$ into the set of objects L_1 that are less than the first argument of the max and the set L_2 that are less than the second argument. For $k \in L_2$, we have that for all $l \in M(k)$

$$\tilde{w}(i,k) - p_k(l) < (1+\varepsilon)^{r_{ik}-s_{\min}} \leq \varepsilon(1+\varepsilon)^{r_{ik}} = \varepsilon\tilde{w}(i,k),$$

which implies $(1-\varepsilon)\tilde{w}(i,k) - p_k(l) < 0$. For $k \in L_1$, we have that for all $l \in M(k)$,

$$(1-\varepsilon)\tilde{w}(i,k) - p_k(l) < (1-\varepsilon)(\tilde{w}(i,k) - p_k(l)) < (1-\varepsilon)(1+\varepsilon)^{r_i+1}.$$

By construction of the bid, we have that $\tilde{w}(i,j) - p_j(c) = (1-\varepsilon)(1+\varepsilon)^{r_i+1}$. Thus

$$\tilde{w}(i,j) - p_j(c) \geq \max\left\{\max_{k \in N(i)\setminus F(i),\, l \in M(k)} \{(1-\varepsilon)\tilde{w}(i,k) - p_k(l)\}, 0\right\}$$

and the edge (i,j) is strongly ε-happy.

Now consider each edge (i,g), where i is assigned to $h \in M(g)$, that was already included in F at the start of the call but $\langle g, h\rangle \notin T$ at the end of the call. By the previous analysis, we know that when i last increased $p_g(h)$, the edge (i,g) must have been strongly ε-happy. By Invariant 2 we know that the prices of object copies are monotonically increasing throughout the algorithm, so the edge (i,g) must still be strongly ε-happy.

If at the end of the call i is saturated, then i must be strongly ε-happy. However, if i is unsaturated then it must be that Q_i is empty. In this case for each $k \in N(i) \setminus F(i)$ it must be that the entry $(r_{ik} - s_{\min}, (i,k))$ must have been popped from Q_i, which indicates that for all $l \in M(k)$,

$$\tilde{w}(i,k) - p_k(l) < (1+\varepsilon)^{r_{ij}-s_{\min}} \leq \varepsilon(1+\varepsilon)^{r_{ij}} = \varepsilon\tilde{w}(i,k).$$

This gives $(1-\varepsilon)\tilde{w}(i,k) - p_k(l) < 0$, so $\pi(i) = 0$ and i is still strongly ε-happy. □

By Invariant 3, when Algorithm 1 terminates, the b-matching F returned must be strongly ε-happy. Termination is guaranteed since for each bidder $i \in A$, Q_i is a fixed length and in case it is emptied, i must be strongly ε-happy. We can also guarantee that if an object $j \in B$ is unsaturated at termination, then $\min_{c \in M(j)} \{p_j(c)\} = 0$. If we set $p(j) = \min_{c \in M(j)} \{p_j(c)\}$, using Proposition 1 and Lemma 2, F must be a $(1-\varepsilon)$-approximate with respect to the weights $\tilde{w}$. By Proposition 2, F is then $(1-\varepsilon')$-approximate with respect to w.

For runtime analysis, we divide the algorithm into two phases: initialization and auction. The initialization phase populates Q_i for each $i \in A$, which requires bucket sorting $ms_{\min}$ pairs and takes $O(ms_{\min}+(s_{\max}+s_{\min})) = O(m\varepsilon^{-1}\log\varepsilon^{-1})$ time. The auction phase involves calls to AssignAndBid(i) for each $i \in A$, which is dominated by the size of Q_i and the time it takes to find a minimum price object copy and update its price. If we use a min priority queue to maintain an ordering of $M(j)$ for each $j \in B$, where the price $p_j(c)$ of an object copy $c \in M(j)$ is its priority, then these operations takes $O(1)$ and $O(\log\beta)$ time, respectively. We may need to do a price update for each pair in Q_i, so the total amount of work done in all calls to AssignAndBid(i) is $O(\deg(i)\, s_{\min}\log\beta)$. Summing over all bidders $i \in A$, the total amount of work done in the auction phase is $O(ms_{\min}\log\beta) = O(m\varepsilon^{-1}\log\varepsilon^{-1}\log\beta)$. Since the weight preprocessing described in Section 5.1 takes $O(m)$ time, we have the following.

Theorem 1. *There exists a multiplicative auction algorithm for MWb-M that gives a $(1-\varepsilon)$-approximate solution in $O(m\varepsilon^{-1}\log\varepsilon^{-1}\log\beta)$ time.*

6 Conclusions

In this paper, we present a near-linear $(1-\varepsilon)$-approximate multiplicative auction algorithm for MWb-M. However, the algorithm's runtime has a dependence on β, the max b-value. While it is reasonable to assume $\beta = O(1)$ in many practical applications, a direction for further research would be to remove this dependence on β to have runtime parity with the algorithm of Huang and Pettie [25].

References

1. Abeywickrama, T., Liang, V., Tan, K.L.: Optimizing bipartite matching in real-world applications by incremental cost computation. Proceedings of the VLDB Endowment **14**(7), 1150–1158 (2021)
2. Ahmadi, S., Ahmed, F., Dickerson, J.P., Fuge, M., Khuller, S.: An algorithm for multi-attribute diverse matching. In: Proc. of the 29th International Joint Conference on Artificial Intelligence (IJCAI 2020). pp. 3–9 (2020). https://doi.org/10.24963/ijcai.2020/1
3. Ahmed, F., Dickerson, J.P., Fuge, M.: Diverse weighted bipartite b-matching. In: Proc. of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017). pp. 35–41 (2017). https://doi.org/10.24963/ijcai.2017/6
4. Alfaro, C.A., Perez, S.L., Valencia, C.E., Vargas, M.C.: The assignment problem revisited. Optimization Letters **16**(5), 1531–1548 (2022). https://doi.org/10.1007/s11590-021-01791-4

5. Assadi, S., Liu, S.C., Tarjan, R.E.: An auction algorithm for bipartite matching in streaming and massively parallel computation models. In: Proc. of the 4th Symposium on Simplicity in Algorithms (SOSA 2021). pp. 165–171 (2021). https://doi.org/10.1137/1.9781611976496.18
6. Azad, A., Buluç, A., Li, X.S., Wang, X., Langguth, J.: A distributed-memory algorithm for computing a heavy-weight perfect matching on bipartite graphs. SIAM Journal on Scientific Computing **42**(4), C143–C168 (2020)
7. Bayati, M., Borgs, C., Chayes, J., Zecchina, R.: Belief-propagation for weighted b-matchings on arbitrary graphs and its relation to linear programs with integer solutions. SIAM Journal on Discrete Mathematics **25**(2), 989–1011 (2011). https://doi.org/10.1137/090753115
8. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. IEEE transactions on pattern analysis and machine intelligence **24**(4), 509–522 (2002)
9. Bertsekas, D.P.: A distributed algorithm for the assignment problem. Lab. for Information and Decision Systems Working Paper, MIT (1979), https://web.mit.edu/dimitrib/www/Bertsekas_Auction_Distributed_1979.pdf
10. Bertsekas, D.P.: A new algorithm for the assignment problem. Mathematical Programming **21**(1), 152–171 (Dec 1981). https://doi.org/10.1007/BF01584237, http://link.springer.com/10.1007/BF01584237
11. Bertsekas, D.P.: The auction algorithm: A distributed relaxation method for the assignment problem. Annals of Operations Research **14**(1), 105–123 (1988). https://doi.org/10.1007/BF02186476
12. Bertsekas, D.P.: Auction algorithms for network flow problems: A tutorial introduction. Computational Optimization and Applications **1**(1), 7–66 (1992). https://doi.org/10.1007/BF00247653
13. Burkard, R.E., Dell'Amico, M., Martello, S.: Assignment Problems. SIAM (2009). https://doi.org/10.1137/1.9781611972238, https://doi.org/10.1137/1.9781611972238
14. Charlin, L., Zemel, R.: The Toronto paper matching system: An automated paper-reviewer assignment system. In: ICML 2013 Workshop on Peer Reviewing and Publishing Models (2013), https://openreview.net/pdf?id=caynafZAnBafx
15. Demange, G., Gale, D., Sotomayor, M.: Multi-item auctions. Journal of Political Economy **94**(4), 863–872 (1986). https://doi.org/10.1086/261411
16. Deng, D., Kim, A., Madden, S., Stonebraker, M.: Silkmoth: an efficient method for finding related sets with maximum matching constraints. Proceedings of the VLDB Endowment **10**(10), 1082–1093 (2017)
17. Duan, R., Pettie, S.: Linear-time approximation for maximum weight matching. Journal of the ACM **61**(1), 1–23 (2014). https://doi.org/10.1145/2529989
18. Duff, I.S., Koster, J.: On algorithms for permuting large entries to the diagonal of a sparse matrix. SIAM J. Matrix Anal. Appl. **22**(4), 973–996 (2001). https://doi.org/10.1137/S0895479899358443, https://doi.org/10.1137/S0895479899358443
19. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: On graph problems in a semi-streaming model. Theoretical Computer Science **348**(2-3), 207–216 (2005). https://doi.org/10.1016/j.tcs.2005.09.013
20. Ferdous, S.: ALGORITHMS FOR DEGREE-CONSTRAINED SUBGRAPHS AND APPLICATIONS. Ph.D. thesis, Purdue University Graduate School (2021)
21. Gabow, H.N.: An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In: Proc. of the 15th ACM Symposium on

Theory of Computing (STOC 1983). pp. 448–456 (1983). https://doi.org/10.1145/800061.808776

22. Gabow, H.N.: Scaling algorithms for network problems. Journal of Computer and System Sciences **31**(2), 148–168 (1985). https://doi.org/10.1016/0022-0000(85)90039-X
23. Gabow, H.N., Tarjan, R.E.: Faster scaling algorithms for network problems. SIAM Journal on Computing **18**(5), 1013–1036 (1989). https://doi.org/10.1137/0218069
24. Huang, C.C., Kavitha, T.: New algorithms for maximum weight matching and a decomposition theorem. Mathematics of Operations Research **42**(2), 411–426 (2017). https://doi.org/10.1287/moor.2016.0806
25. Huang, D., Pettie, S.: Approximate generalized matching: f-matchings and f-edge covers. Algorithmica **84**(7), 1952–1992 (2022). https://doi.org/10.1007/s00453-022-00949-5
26. Karloff, H., Suri, S., Vassilvitskii, S.: A model of computation for MapReduce. In: Proc. of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA 2010). pp. 938–948 (2010). https://doi.org/10.1137/1.9781611973075.76
27. Khan, A., Pothen, A., Mostofa Ali Patwary, M., Satish, N.R., Sundaram, N., Manne, F., Halappanavar, M., Dubey, P.: Efficient approximation algorithms for weighted b-matching. SIAM Journal on Scientific Computing **38**(5), S593–S619 (2016). https://doi.org/10.1137/15M1026304
28. Krissinel, E., Henrick, K.: Secondary-structure matching (ssm), a new tool for fast protein structure alignment in three dimensions. Acta Crystallographica Section D: Biological Crystallography **60**(12), 2256–2268 (2004). https://doi.org/10.1107/S0907444904026460
29. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly **2**(1–2), 83–97 (1955). https://doi.org/10.1002/nav.3800020109
30. Liu, Q.C., Ke, Y., Khuller, S.: Scalable auction algorithms for bipartite maximum matching problems. In: Proc. of the 26th International Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2023). pp. 28:1–28:24 (2023). https://doi.org/10.4230/LIPIcs.APPROX/RANDOM.2023.28
31. Liu, X., Suel, T., Memon, N.: A robust model for paper reviewer assignment. In: Proceedings of the 8th ACM Conference on Recommender systems. pp. 25–32 (2014)
32. Lovász, L., Plummer, M.D.: Matching theory, vol. 367. American Mathematical Soc. (2009)
33. Mehta, A., et al.: Online matching and ad allocation. Foundations and Trends® in Theoretical Computer Science **8**(4), 265–368 (2013)
34. Mestre, J.: Greedy in approximation algorithms. In: Proc. of the 14th European Symposium on Algorithms (ESA 2006). pp. 528–539 (2006). https://doi.org/10.1007/11841036_48
35. Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the society for industrial and applied mathematics **5**(1), 32–38 (1957)
36. Olschowka, M., Neumaier, A.: A new pivoting strategy for gaussian elimination. Linear Algebra and its Applications **240**, 131–151 (1996)
37. Pang, L., Lan, Y., Guo, J., Xu, J., Wan, S., Cheng, X.: Text matching as image recognition. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 30 (2016)
38. Pothen, A., Ferdous, S., Manne, F.: Approximation algorithms in combinatorial scientific computing. Acta Numerica **28**, 541–633 (2019)

39. Ramshaw, L., Tarjan, R.E.: On minimum-cost assignments in unbalanced bipartite graphs. HP Labs, Palo Alto, CA, USA, Tech. Rep. HPL-2012-40R1 **20** (2012), https://www.hpl.hp.com/techreports/2012/HPL-2012-40R1.pdf
40. Sathe, M., Schenk, O., Burkhart, H.: An auction-based weighted matching implementation on massively parallel architectures. Parallel Computing **38**(12), 595–614 (2012). https://doi.org/10.1016/j.parco.2012.09.001
41. Shiloach, Y.: Another look at the degree constrained subgraph problem. Information Processing Letters **12**(2), 89–92 (1981). https://doi.org/10.1016/0020-0190(81)90009-0
42. Simonovsky, M., Komodakis, N.: Graphvae: Towards generation of small graphs using variational autoencoders. In: Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27. pp. 412–422. Springer (2018)
43. Tang, W., Tang, J., Tan, C.: Expertise matching via constraint-based optimization. In: Proc. of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2010). vol. 1, pp. 34–41 (2010). https://doi.org/10.1109/WI-IAT.2010.133
44. Zeakis, A., Skoutas, D., Sacharidis, D., Papapetrou, O., Koubarakis, M.: Tokenjoin: Efficient filtering for set similarity join with maximumweighted bipartite matching. Proceedings of the VLDB Endowment **16**(4), 790–802 (2022)
45. Zheng, D.W., Henzinger, M.: Multiplicative auction algorithm for approximate maximum weight bipartite matching. In: Proc. of the 24th Conference on Integer Programming and Combinatorial Optimization (IPCO 2023). pp. 453–465 (2023). https://doi.org/10.1007/978-3-031-32726-1_32