Scalable Auction Algorithms for Bipartite Maximum Matching Problems

Quanquan C. Liu
Northwestern University, Evanston, IL, USA

Yiduo Ke
Northwestern University, Evanston, IL, USA

Samir Khuller
Northwestern University, Evanston, IL, USA

Abstract

Bipartite maximum matching and its variants are well-studied problems under various models of computation with the vast majority of approaches centering around various methods to find and eliminate augmenting paths. Beginning with the seminal papers of Demange, Gale and Sotomayor [DGS86] and Bertsekas [Ber81], bipartite maximum matching problems have also been studied in the context of *auction algorithms*. These algorithms model the maximum matching problem as an auction where one side of the bipartite graph consists of bidders and the other side consists of items; as such, these algorithms offer a very different approach to solving this problem that do not use classical methods. Dobzinski, Nisan and Oren [DNO14] demonstrated the utility of such algorithms in distributed, interactive settings by providing a simple and elegant $O(\log n/\varepsilon^2)$ round *maximum cardinality bipartite matching (MCM)* algorithm that has small round and communication complexity and gives a $(1-\varepsilon)$-approximation for any (not necessarily constant) $\varepsilon > 0$. They leave as an open problem whether an auction algorithm, with similar guarantees, can be found for the *maximum weighted bipartite matching (MWM)* problem. Very recently, Assadi, Liu, and Tarjan [ALT21] extended the utility of auction algorithms for MCM into the semi-streaming and massively parallel computation (MPC) models, by cleverly using maximal matching as a subroutine, to give a new auction algorithm that uses $O(1/\varepsilon^2)$ rounds and achieves the state-of-the-art bipartite MCM results in the streaming and MPC settings.

In this paper, we give new auction algorithms for maximum weighted bipartite matching (MWM) and maximum cardinality bipartite b-matching (MCbM). Our algorithms run in $O\left(\log n/\varepsilon^8\right)$ and $O\left(\log n/\varepsilon^2\right)$ rounds, respectively, in the distributed setting. We show that our MWM algorithm can be implemented in the distributed, interactive setting using $O(\log^2 n)$ and $O(\log n)$ bit messages, respectively, directly answering the open question posed by Demange, Gale and Sotomayor [DNO14]. Furthermore, we implement our algorithms in a variety of other models including the the semi-streaming model, the shared-memory work-depth model, and the massively parallel computation model. Our semi-streaming MWM algorithm uses $O(1/\varepsilon^8)$ passes in $O(n \log n \cdot \log(1/\varepsilon))$ space and our MCbM algorithm runs in $O(1/\varepsilon^2)$ passes using $O\left(\left(\sum_{i\in L} b_i + |R|\right)\log(1/\varepsilon)\right)$ space (where parameters b_i represent the degree constraints on the b-matching and L and R represent the left and right side of the bipartite graph, respectively). Both of these algorithms improves *exponentially* the dependence on ε in the space complexity in the semi-streaming model against the best-known algorithms for these problems, in addition to improvements in round complexity for MCbM. Finally, our algorithms eliminate the large polylogarithmic dependence on n in depth and number of rounds in the work-depth and massively parallel computation models, respectively, improving on previous results which have large polylogarithmic dependence on n (and exponential dependence on ε in the MPC model).

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases auction algorithms, maximum weight bipartite matching, maximum b-matching, distributed blackboard model, parallel work-depth model, streaming model, massively parallel computation model

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.28

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2307.08979

Funding We gratefully acknowledge support from NSF-Award 2216970 (IDEAL Institute).

1 Introduction

One of the most basic problems in combinatorial optimization is that of bipartite matching. This central problem has been studied extensively in many fields including operations research, economics, and computer science and is the cornerstone of many algorithm design courses and books. There is an abundance of existing classical and recent theoretical work on this topic [25, 12, 13, 19, 20, 27, 29, 30, 6, 11, 31]. Bipartite maximum matching and its variants are commonly taught in undergraduate algorithms courses and are so prominent to be featured regularly in competitive programming contests. In both of these settings, the main algorithmic solutions for maximum cardinality matching (MCM) and its closely related problems of maximum weight matching (MWM) are the Hungarian method using augmenting paths and reductions to maximum flow. Although foundational, such approaches are sometimes difficult to generalize to obtain efficient solutions in other *scalable models of computation*, e.g. distributed, streaming, and parallel models.

Although somewhat less popularly known, the elegant and extremely simple *auction-based maximum cardinality and maximum weighted matching algorithms* of Demange, Gale, and Sotomayor [10] and Bertsekas [8] solve the maximum cardinality/weighted matching problems in bipartite graphs. Their MCM auction algorithms denote vertices on one side of the bipartite input as *bidders* and the other side as *items*. Bidders are maintained in a queue and while the queue is not empty, the first bidder from the queue *bids* on an item with minimum price (breaking ties arbitrarily) from its neighbors. This bidder becomes the new owner of the item. Each time an item is reassigned to a new bidder, its price increases by some (not necessarily constant) $\varepsilon > 0$. If the assigned item still has price less than 1, the bidder is added again to the end of the queue. Setting $\varepsilon = \frac{1}{n+1}$ results in an algorithm that gives an exact maximum cardinality matching in $O(mn)$ time, where m and n refer to the number of edges and vertices respectively. Such an algorithm intuitively takes advantage of the fact that bidders prefer items in low demand (smaller price); naturally, such items should also be matched in a maximum cardinality matching.

One of the bottlenecks in the original auction algorithm is the need to maintain bidders in a queue from which they are selected, one at a time, to bid on items. Such a bottleneck is a key roadblock to the scalability of such algorithms. More recently, Dobzinski, Nisan, and Oren [11] extended this algorithm to the approximation setting for any (not necessarily constant) $\varepsilon > 0$. They give a simple and elegant randomized $(1 - \varepsilon)$-approximation algorithm for bipartite MCM in $O\left(\frac{\log n}{\varepsilon^2}\right)$ rounds of communication for any $\varepsilon > 0$. Furthermore, they illustrate an additional advantage for this algorithm beyond its simplicity. They show that in a distributed, interactive, blackboard setting, their auction MCM multi-round interactive algorithm uses less communication bits than traditional algorithms for this problem. This interactive setting is modeled via simultaneous communication protocols where agents simultaneously send a single message in each round to a central coordinator and some state is computed by the central coordinator after each round of communication. The goal in this model is to limit the total number of bits sent in all of the agents' messages throughout the duration of the algorithm. They leave as an open question whether an interactive, approximation auction algorithm that uses approximately the same number of rounds and bits of communication can be found for the maximum *weighted* bipartite matching problem.

Such an approach led to the recent simple and elegant paper of Assadi, Liu, and Tarjan [6] that adapted their algorithm to the semi-streaming setting and removed the $\log n$ factor in the semi-streaming setting from the number of passes to give an algorithm that finds an $(1-\varepsilon)$-approximate maximum cardinality matching in $O\left(1/\varepsilon^2\right)$ passes, where in each pass a maximal matching is found. Furthermore, they showed implementations of their algorithm in the massively parallel computation (MPC) model, achieving the best-known bounds in both of these settings. In this paper, we extend their algorithm to other variants of the problem on bipartite graphs, including maximum weight matching and maximum cardinality b-matching and achieve novel improvements in a variety of *scalable models*. The maximum cardinality b-matching problem (MCBM) is a well-studied generalization of MCM. In MCBM, each vertex is given an integer budget b_v where each vertex can be matched to at most b_v of their neighbors; a matching of maximum cardinality contains the maximum possible number of edges in the matching. The b-matching problem generalizes a number of real-life allocation problems such as server to client request serving, medical school residency matching, ad allocation, and many others. Although the problem is similar to MCM, often obtaining efficient algorithms for this problem requires non-trivial additional insights. As indicated in Ghaffari et al [16] b-matching problems can be considerably harder than matching.

Summary of Results. In this paper, we specifically give the following results. Our auction algorithms and their analyses are described in detail in Section 3 and Section 4.

▶ **Theorem 1** (Maximum Weight Bipartite Matching). *There exists an auction algorithm for maximum weight bipartite matching (MWM) that gives a $(1-\varepsilon)$-approximation for any $\varepsilon > 0$ and runs in $O\left(\frac{\log n \cdot \log(1/\varepsilon)}{\varepsilon^8}\right)$ rounds of communication (with high probability) and with $O\left(\log^2 n\right)$ bits per message. This algorithm can be implemented in the multi-round, semi-streaming model using $O\left(n \cdot \log n \cdot \log(1/\varepsilon)\right)$ space and $O\left(\frac{\log(1/\varepsilon)}{\varepsilon^7}\right)$ passes. This algorithm can be implemented in the work-depth model in $O\left(\frac{m \cdot \log(1/\varepsilon)}{\varepsilon^6}\right)$ work and $O\left(\frac{\log n \cdot \log(1/\varepsilon)}{\varepsilon^8}\right)$ depth. Finally, our algorithm can be implemented in the MPC model using $O(\log(1/\varepsilon)/\varepsilon^7)$ rounds, $O(n)$ space per machine, and $O\left(\frac{m \log(1/\varepsilon) \log n}{\varepsilon}\right)$ total space.*

The best-known algorithms in the semi-streaming model for the maximum weight bipartite matching problem are the $(1/\varepsilon)^{O(1/\varepsilon^2)}$ pass, $O\left(n \operatorname{poly}(\log(n)) \operatorname{poly}(1/\varepsilon)\right)$ space algorithm of Gamlath et al. [15] and the $O\left(\frac{\log(1/\varepsilon)}{\varepsilon^2}\right)$ pass, $O\left(\frac{n \log n}{\varepsilon^2}\right)$ space algorithm of Ahn and Guha [1]. To the best of our knowledge, our result is the first to achieve sub-polynomial dependence on $1/\varepsilon$ in the space for the MWM problem in the semi-streaming model. Thus, we improve the space bound exponentially compared to the previously best-known algorithms in the streaming model. The best-known algorithms in the distributed and work-depth models required $\operatorname{poly}(\log n)$ in the number of rounds and depth, respectively [21]; in the MPC setting, the best previously known algorithms have exponential dependence on ε [15]. We eliminate such dependencies in our paper and our algorithm is also simpler. A summary of previous results and our results can be found in Table 1.

▶ **Theorem 2** (Maximum Cardinality Bipartite b-Matching). *There exists an auction algorithm for maximum cardinality bipartite b-matching (MCBM) that gives a $(1-\varepsilon)$-approximation for any $\varepsilon > 0$ and runs in $O\left(\frac{\log n}{\varepsilon^2}\right)$ rounds of communication. This algorithm can be implemented in the multi-round, semi-streaming model using $O\left(\left(\sum_{i \in L} b_i + |R|\right) \log(1/\varepsilon)\right)$ space and $O\left(\frac{1}{\varepsilon^2}\right)$ passes. Our algorithm can be implemented in the shared-memory work-depth model in $O\left(\frac{\log^3 n}{\varepsilon^2}\right)$ depth and $O\left(\frac{m \log n}{\varepsilon^2}\right)$ total work.*

The best-known algorithms for maximum cardinality bipartite b-matching in the semi-streaming model is the $O\left(\frac{\log n}{\varepsilon^3}\right)$ pass, $\widetilde{O}\left(\frac{\sum_{i\in L\cup R} b_i}{\varepsilon^3}\right)$ space algorithm of Ahn and Guha [1]. In the general, non-bipartite setting (a harder setting than what we consider), a very recent $(1-\varepsilon)$-approximation algorithm of Ghaffari, Grunau, and Mitrović [16] runs in $\exp\left(2^{O(1/\varepsilon)}\right)$ passes and $\widetilde{O}\left(\sum_{i\in L\cup R} b_i + \text{poly}(1/\varepsilon)\right)$ space. Here, we also improve the space exponentially in $1/\varepsilon$ and, in addition, improve the number of passes by an $O(\log n)$ factor. More details comparing our results to other related works are given in Section 1.1. Due to the space constraints, most proofs in the following sections are deferred to the full version of our paper. Our results as well as comparisons with previous work are given in Table 1.

Table 1 We assume the ratio between the largest weight edge and smallest weight edge in the graph is $\text{poly}(n)$. Results for general graphs are labeled with (general); results that are specifically for bipartite graphs do not have a label. Upper bounds are given in terms of $O(\cdot)$ and lower bounds are given in terms of $\Omega(\cdot)$. "Space p.m." stands for space per machine. The complexity measures for the "blackboard distributed" setting is the total communication (over all rounds and players) in bits.

Model		Previous Results		Our Results	
Blackboard Distributed	MWM	$\Omega(n \log n)$ (trivial)	[11]	$O\left(\frac{n \log^3(n)\cdot\log(1/\varepsilon)}{\varepsilon^9}\right)$	Theorem 11
	MCBM	$\Omega(nb \log n)$	trivial	$O\left(\frac{nb \log^2 n}{\varepsilon^2}\right)$	Theorem 28
Streaming	MWM	$O\left(\frac{\log(1/\varepsilon)}{\varepsilon^2}\right)$ pass $O\left(\frac{n \log n}{\varepsilon^2}\right)$ space	[1]	$O\left(\frac{\log(1/\varepsilon)}{\varepsilon^7}\right)$ pass $O\left(n \cdot \log n \cdot \log(1/\varepsilon)\right)$ space	Theorem 13
	MCBM	$O(\log n/\varepsilon^3)$ pass $\widetilde{O}\left(\frac{\sum_{i\in L\cup R} b_i}{\varepsilon^3}\right)$ space	[2]	$O\left(\frac{1}{\varepsilon^2}\right)$ pass $O\left(\left(\sum_{i\in L} b_i + \lvert R\rvert\right)\log(1/\varepsilon)\right)$ space	Theorem 30
MPC	MWM	$O_\varepsilon(\log \log n)$ rounds $O_\varepsilon(n \,\text{poly}(\log n))$ space p.m.	[15] (general)	$O\left(\frac{\log(1/\varepsilon)\cdot\log\log n}{\varepsilon^7}\right)$ rounds $O(n)$ space p.m.	Theorem 17
Parallel	MWM	$O\left(m \cdot \text{poly}\left(1/\varepsilon, \log n\right)\right)$ work $O\left(\text{poly}\left(1/\varepsilon, \log n\right)\right)$ depth	[21] (general)	$O\left(\frac{m \log(1/\varepsilon)}{\varepsilon^6}\right)$ work $O\left(\frac{\log n\cdot\log(1/\varepsilon)}{\varepsilon^8}\right)$ depth	Theorem 15
	MCBM	N/A	N/A	$O\left(\frac{m \log n}{\varepsilon^2}\right)$ work $O\left(\frac{\log^3 n}{\varepsilon^2}\right)$ depth	Theorem 31

Concurrent, Independent Work. In concurrent, independent work, Zheng and Henzinger [34] study the maximum weighted matching problem in the sequential and dynamic settings using auction-based algorithms. Their simple and elegant algorithm makes use of a sorted list of items (by utility) for each bidder and then matches the bidders one by one individually (in round-robin order) to their highest utility item. They also extend their algorithm to give dynamic results. Due to the sequential nature of their matching procedure, they do not provide any results in scalable models such as the streaming, MPC, parallel, or distributed models.

1.1 Other Related Works

There has been no shortage of work done on bipartite matching. In addition to the works we discussed in the introduction, there has been a number of other relevant works in this general area of research. Here we discuss the additional works not discussed in Section 1. These

include a plethora of results for $(1-\varepsilon)$-approximate maximum cardinality matching as well as some additional results for MWM and b-matching. Most of these works use various methods to find augmenting paths with only a few works focusing on auction-based techniques. We hope that our paper further demonstrates the utility of auction-based approaches in this setting and will lead to additional works in this area in the future. Although our work focuses on the bipartite matching problem, we also provide the best-known bounds for the matching problem on general graphs here, although this is a harder problem than our setting. We separate these results into the bipartite matching results, the general matching results, and lower bounds.

General Matching. A number of works have considered MCM in the streaming setting, providing state-of-the-art bounds in this setting. Fischer et al. [14] gave a deterministic $(1-\varepsilon)$-approximate MWM algorithm in general graphs in the semi-streaming model that uses poly$(1/\varepsilon)$ passes, improving exponentially on the number of passes of Lotker et al. [28]. Very recently, Assadi et al. [4] provided a semi-streaming algorithm in optimal $O(n)$ space and $O\left(\log n \log(1/\varepsilon)/\varepsilon\right)$ passes. They also provide a MWM algorithm that also runs in $O(n)$ space but requires $\tilde{\Omega}(n/\varepsilon)$ passes. Please refer to these papers and references therein for older results in this area. Ahn and Guha [2] also considered the general weighted non-bipartite maximum matching problem in the semi-streaming model and utilize linear programming approaches for computing a $(2/3-\varepsilon)$-approximation and $(1-\varepsilon)$-approximation that uses $O(\log(1/\varepsilon)/\varepsilon^2)$ passes, $O\left(n \cdot \left(\frac{\log(1/\varepsilon)}{\varepsilon^2} + \frac{\log n/\varepsilon}{\varepsilon}\right)\right)$ space, and $O\left(\frac{\log n}{\varepsilon^4}\right)$ passes, $O\left(\frac{n \log n}{\varepsilon^4}\right)$ space, respectively.

Bipartite Matching. Ahn and Guha [2] also extended their results to the bipartite MWM and b-Matching settings with small changes. Specifically, in the MWM setting, they give a $O(\log(1/\varepsilon)/\varepsilon^2)$ pass, $O(n \cdot ((\log(1/\varepsilon))/\varepsilon^2 + (\log n/\varepsilon)/\varepsilon))$ space algorithm. For maximum cardinality b-matching, they give a $O(\log n/\varepsilon^3)$ pass and $\widetilde{O}\left(\frac{\sum_{i \in L \cup R} b_i}{\varepsilon^3}\right)$ space algorithm. For exact bipartite MWM in the semi-streaming model, Liu et al. [26] gave the first streaming algorithm to break the n-pass barrier in the exact setting; it uses $\tilde{O}(n)$ space and $\tilde{O}(\sqrt{m})$ passes using interior point methods, SDD system solvers, and various other techniques to output the optimal matching with high probability. Work on bipartite MWM prior to [26] either required $\Omega(n \log n)$ passes[22] or only found approximate solutions [1, 2, 23].

Lower Bounds. Several papers have looked at matching problems from the lower bound side. Konrad et al. [24] considered the communication complexity of graph problems in a blackboard model of computation (for which the simultaneous message passing model of Dobzinski et al. [11] is a special variant). Specifically, they show that any non-trivial graph problem on n vertices require $\Omega(n)$ bits [24] in communication complexity. In a similar model called the *demand query model*, Nisan [32] showed that any *deterministic* algorithm that runs in $n^{o(1)}$ rounds where in each round at most $n^{1.99}$ demand queries are made, cannot find a MCM within a $n^{o(1)}$ factor of the optimum. This is in contrast to *randomized* algorithms which can make such an approximation using only $O(\log n)$ rounds. For streaming matching algorithms, Assadi [3] provided a conditional lower bound ruling out the possibilities of small constant factor approximations for two-pass streaming algorithms that solve the MCM problem. Such a lower bound also necessarily extends to MWM and MCBM. Goel et al. [17] provided a $n^{1+\Omega(1/\log\log n)}$ lower bound for the one-round message complexity of bipartite $(2/3+\varepsilon)$-approximate MCM (this also naturally extends to a space

lower bound). For older papers on these lower bounds, please refer to references cited within each of the aforementioned cited papers. Finally, Assadi et al. [5] showed that any streaming algorithm that approximates MCM requires either $n^{\Omega(1)}$ space or $\Omega(\log(1/\varepsilon))$ passes.

Unweighted to Weighted Matching Transformations. Current transformations for transforming unweighted to weighted matchings all either:

- lose a factor of 2 in the approximation factor [18, 33], or
- increase the running time of the algorithm by an exponential factor in terms of $1/\varepsilon$, specifically, a factor of $\varepsilon^{-O(1/\varepsilon)}$ [7].

Thus, we cannot use such default transformations from unweighted matchings to weighted matchings in our setting since all of the complexity measures in this paper have only *polynomial* dependence on ε and all guarantee $(1-\varepsilon)$-approximate matchings. However, we do make use of *weighted to weighted matching transformations* provided our original weighted matching algorithms have only *polylogarithmic* dependence on the maximum ratio between edge weights in the graph. Such transformations from weighted to weighted matchings do not increase the approximation factor and also allows us to *eliminate the polylogarithmic dependence on the maximum ratio of edge weights.*

2 Preliminaries

This paper presents algorithms for bipartite matching under various settings. The input consists of a bipartite graph $G = (L \cup R, E)$. We denote the set of neighbors of any $i \in L, j \in R$ by $N(i), N(j)$, respectively. We present $(1-\varepsilon)$-approximation algorithms where $\varepsilon \in (0,1)$ is our approximation parameter. All notations used in all of our algorithms in this paper are given in Table 2. The specified weight of an edge (i,j) will become the valuation of the bidder i for item j. Due to space constraints, we defer most of our proofs to the full version of our paper.

2.1 Scalable Model Definitions

In addition, we consider a number of scalable models in our paper including the ***blackboard distributed*** model, the ***semi-streaming*** model, the ***massively parallel computation (MPC)*** model, and the ***parallel shared-memory work-depth*** model.

Blackboard distributed model. We use the blackboard distributed model as defined in [11]. There are n *players*, one for each vertex of the left side of our bipartite graph (we assume wlog that the left side of the graph contains more vertices). The players engage in a fixed communication protocol using messages sent to a central coordinator. In other words, players write on a common "blackboard." Each players can receive a (not necessarily identical) message in each round from the coordinator. Players communicate using *rounds* of communication where in each round the player sends a message (of some number of bits) to the central coordinator. In every round, players choose to send messages depending solely on the contents of the blackboard and their private information. Termination of the algorithm and the final matching are determined by the central coordinator and the contents of the blackboard. The measure of complexity is the number of rounds of the algorithm and the message size sent by each player in each round. One can also measure the total number of bits send by all messages by multiplying the two.

Table 2 Table of Notations.

Symbol	Meaning
ε	approximation parameter
L, R	bidders, items, resp. WLOG $\|L\| \leq \|R\|$
i, j, i', j'	$i \in L, j \in R, i' \in L', j' \in R'$, i' (resp. j') indicates copy of i (resp. j)
p_j	current price of item j
D_i	demand set of bidder i
(i, a_i)	bidder $i \in L$ and currently matched item a_i
o_i	the item matched to bidder i in OPT
u_i	the utility of bidder i which is calculated by $1 - p_{a_i}$
$v_i(j)$	the valuation of bidder i for item j, i.e. the weight of edge (i, j)
C_i, C_j	copies of bidder $i \in L$, copies of item $j \in R$, resp.
L', R'	$L' = \bigcup_{i \in L} C_i$, $R' = \bigcup_{j \in R} C_j$
E', G'	$E' = \{(i^{(k)}, j^{(l)}) \mid (i, j) \in E, k \in [b_i], l \in [b_j]\}$, $G' = (L' \cup R', E')$
$c_{i'}$	price cutoff for bidder i'
N	ratio of the maximum weighted edge over the minimum weighted edge
v_i	the valuation function for bidder i
G_d	induced subgraph consisting of $(\cup_{i \in L} D_i) \cup L$
$\widehat{M}_d$	a non-duplicate maximal matching in G'_d
M'_d, M_d	produced matching in G', corresponding matching in G, resp.
$M_{\max}$	matching with largest cardinality produced

Semi-streaming model. In this paper, we use the semi-streaming model with arbitrary ordered edge insertions. Edges are arbitrarily (potentially adversarially) ordered in the stream. For this paper, we only consider insertion-only streams. The space usage for semi-streaming algorithms is bounded by $\widetilde{O}(n)$. The relevant complexity measures in this model are the number of passes of the algorithm and the space used.

Massively parallel computation (MPC) model. The massively parallel computation (MPC) model is a distributed model where different machines communicate with each other via a communication network. There are M machines, each with S space, and these machines communicate with each using Q rounds of communication. The initial graph is given in terms of edges and edges are partitioned arbitrarily across the machines. The relevant complexity measures are the total space usage $(M \cdot S)$, space per machine S, and number of rounds of communication Q.

Parallel shared-memory work-depth model. The parallal shared-memory work-depth model is a parallel model where different processors can process instructures in parallel and read and write from the same shared-memory. The relevant complexity measures for an algorithm in this model are the *work* which is the total amount of computation performed by the algorithm and the *depth* which is the longest chain of sequential dependencies in the algorithm.

3 An Auction Algorithm for $(1 - \varepsilon)$-Approximate Maximum Weighted Bipartite Matching

We present the following auction algorithm for maximum (weighted) bipartite matching (MWM) that is a generalization of the simple and elegant algorithm of Assadi et al. [6] to the weighted setting. Our generalization requires several novel proof techniques and recovers

the round guarantee of Assadi et al. [6] in the maximum cardinality matching setting when the weights of all edges are 1. Furthermore, we answer an open question posed by Dobzinski et al. [11] for developing a $(1-\varepsilon)$-approximation auction algorithm for maximum *weighted* bipartite matching for which no prior algorithms are known. Throughout this section, we denote the maximum ratio between two edge weights in the graph by R. Our algorithm can also be easily extended into algorithms in various scalable models:

- a semi-streaming algorithm which uses $O\left(n \cdot \log n \cdot \log(1/\varepsilon)\right)$ space (the number of vertices in the bipartite graph) and which requires $O(\log(1/\varepsilon)/\varepsilon^7)$ rounds,
- a shared-memory parallel algorithm using $O\left(\frac{n \log n}{\varepsilon^9}\right)$ work and $O\left(\frac{1}{\varepsilon^8}\right)$ depth, and
- an MPC algorithm using $O(\log(1/\varepsilon)/\varepsilon^7)$ rounds, $O(n)$ space per machine, and $O\left(\frac{n \log n}{\varepsilon}\right)$ total space.

In contrast, the best-known semi-streaming MWM algorithm of Ahn and Guha [1] requires $\widetilde{O}(\log(1/\varepsilon)/\varepsilon^2)$ passes and $\widetilde{O}\left(\frac{n \log n}{\varepsilon^2}\right)$ space. Our paper shows a $\widetilde{O}(1/\varepsilon^5)$ round algorithm that instead uses $O(n \log(1/\varepsilon))$ space. Since $\varepsilon = \Omega(1/n)$ (or otherwise we obtain an exact maximum weight matching), our algorithm works in the semi-streaming model for all possible values of ε whereas Ahn and Guha [1] no longer works in semi-streaming when ε is small enough.

Our algorithm follows the general framework given in [6]. However, both our algorithm and our analysis require additional techniques. The main hurdle we must overcome is the fact that the weights may be much *larger* than the number of bidders and items. In that case, if we use the MCM algorithm trivially in this setting, the number of rounds can be very large, proportional to $\frac{w_{\max}}{\varepsilon^2}$ where $w_{\max}$ is the maximum weight of the edge. We avoid this problem in our algorithm, instead obtaining only $\operatorname{poly}\log n$ and ε dependence in the number of rounds. Our main result in this section is the following (recall from Section 1).

▶ **Theorem 1** (Maximum Weight Bipartite Matching). *There exists an auction algorithm for maximum weight bipartite matching (MWM) that gives a $(1-\varepsilon)$-approximation for any $\varepsilon > 0$ and runs in $O\left(\frac{\log n \cdot \log(1/\varepsilon)}{\varepsilon^8}\right)$ rounds of communication (with high probability) and with $O\left(\log^2 n\right)$ bits per message. This algorithm can be implemented in the multi-round, semi-streaming model using $O\left(n \cdot \log n \cdot \log(1/\varepsilon)\right)$ space and $O\left(\frac{\log(1/\varepsilon)}{\varepsilon^7}\right)$ passes. This algorithm can be implemented in the work-depth model in $O\left(\frac{m \cdot \log(1/\varepsilon)}{\varepsilon^6}\right)$ work and $O\left(\frac{\log n \cdot \log(1/\varepsilon)}{\varepsilon^8}\right)$ depth. Finally, our algorithm can be implemented in the MPC model using $O(\log(1/\varepsilon)/\varepsilon^7)$ rounds, $O(n)$ space per machine, and $O\left(\frac{m \log(1/\varepsilon) \log n}{\varepsilon}\right)$ total space.*

Before we give our algorithm, we give some notation used in this section.

Notation. The input bipartite graphs is represented by $G = (L \cup R, E)$ where L is the set of bidders and R is the set of items. Let $N(v)$ denote the neighbors of node $v \in L \cup R$. We use the notation $i \in L$ to denote bidders and $j \in R$ to denote items. For a bidder $i \in L$, the *valuation* of i for items in R is defined as the function $v_i : R \to \mathbb{Z}_{\geq 0}$ where the function outputs a non-negative integer. If $v_i(j) > 0$, for any $j \in R$, then $j \in N(i)$. Each bidder can match to at most one item. We denote the bidder item pair by (i, a_i) where a_i is the matched item and $a_i = \perp$ if i is not matched to any item. For any agent i where $a_i \neq \perp$, the *utility* of a bidder i given its matched item a_i is $u_i \triangleq v_i(a_i) - p_{a_i}$ where p_{a_i} is the current price of item a_i. For an agent i where $a_i = \perp$, the utility of agent i is 0. We denote an optimum matching by OPT. We use the notation $i \in \text{OPT}$ to denote a bidder who is matched in OPT and o_i to denote the item matched to bidder i in OPT.

Input Specifications. In this section, we assume all weights are poly(n) where $n = |L| + |R|$. We additionally assume the following characteristics about our inputs because we can perform a simple pre-processing of our graph to satisfy these specifications. Provided an input graph $G = (L \cup R, E)$ with weights $v_i(j)$ for every edge $(i, j) \in E$, we find the maximum weight among all the weights of the edges, $w_{\max} = \max_{(i,j)\in E}(v_i(j))$. We rescale the weights of all the edges by $\frac{1}{w_{\max}}$ and remove all edges with rescaled weight $< \varepsilon^{\lceil \log_\varepsilon m \rceil + 1}$. This upper bound of $\varepsilon^{\lceil \log_\varepsilon m \rceil + 1}$ is crucial in our analysis.

In other words, we create a new graph $G' = (L \cup R, E')$ with the same set of bidders L and items R. We associate the new weight functions v'_i with each bidder $i \in L$ where $(i, j) \in E'$ if $v_i(j) \geq w_{\max} \cdot \varepsilon^{\lceil \log_\varepsilon m \rceil + 1}$ and $v'_i(j) = v_i(j)/w_{\max}$ for each $(i, j) \in E'$. Provided that finding the maximum weight edge can be done in $O(1)$ rounds in the blackboard distributed and MPC models, $O(1)$ passes in the streaming model, and $O(n+m)$ work and $O(1)$ depth in the parallel model, we assume the input to our algorithms is G' (instead of the original graph G). In other words, we assume all inputs $G = (V, E)$ to our algorithm have scaled edge weights and $v_i(j)$ for $i \in L, j \in R$ are functions that return the scaled edge weights in the rest of this section.

3.1 Detailed Algorithm

We now present our auction algorithm for maximum weighted bipartite matching in Algorithm 1. The algorithm works as follows. Recall that we also assume the input to our algorithm is the scaled graph. This means that the maximum weight of the scaled edges is 1 and there exists at least one edge with weight 1; hence, the maximum weight matching will have value at least 1. We also initialize the tuples that keep track of matched items. Initially, no items are assigned to bidders (Algorithm 1) and the prices of all items are set to 0 (Algorithm 1).

We perform $\lceil \frac{\log^2(N)}{\varepsilon^4} \rceil$ phases of bidding (Algorithm 1). In each phase, we form the *demand set* D_i of each unmatched bidder i. The demand set is defined to be the set of items with non-zero utility which have *approximately* the maximum utility value for bidder i (Algorithm 1). This procedure is different from both MCM and MCBM (where no slack is needed in creating the demand set) but we see in the analysis that we require this slack in the maximum utility value to ensure that enough progress is made in each round. Then, we create the induced subgraph consisting of all unmatched bidders and their demand sets (Algorithm 1). We find an arbitrary maximal matching in this created subgraph (Algorithm 1) by first finding the maximal matching in order of decreasing buckets (from highest – bucket with the largest weights – to lowest). This means that we call our maximal matching algorithm $O(\log(N))$ times first on the induced subgraph consisting of the highest bucket, removing the matches, and then on the induced subgraph of the remaining edges plus the next highest bucket, and so on. We use the folklore distributed maximal matching algorithm where in each round, a bidder uniformly-at-random picks a neighbor to match; this algorithm is also used in [11] for the maximal matching step. This simple algorithm terminates in $O(\log n)$ rounds with high probability using $O(\log n)$ communication complexity. Such randomization is necessary to obtain $O(\log n)$ rounds using $O(\log n)$ communication complexity.

We rematch items according to the new matching (Algorithm 1). We then increase the price of each rematched item. The price increase depends on the weight of the matched edge to the item; higher weight matched edges have larger increases in price than smaller weight edges. Specifically, the price is increased by $\varepsilon \cdot v_i(a_i)$ where $v_i(a_i)$ is the weight of the newly matched edge between i and a_i (Algorithm 1). The intuition behind this price increase is that we want to increase the price proportional to the weight gained from the matching since

Algorithm 1 Auction Algorithm for Maximum Weighted Bipartite Matching.

Input: A scaled graph $G = (L \cup R, E)$, parameter $0 < \varepsilon < 1$, and the scaling factor $w_{\max}$.
Output: An $(1 - 6\varepsilon)$-approximate maximum weight bipartite matching.

For each bidder $i \in L$, set (i, a_i) to $a_i = \perp$.
For each item $j \in R$, set $p_j = 0$.
for $d = 1, \ldots, \lceil \frac{\log^2(N)}{\varepsilon^4} \rceil$ **do**
  for each unmatched bidder $i \in L$ **do**
    Let $U_i \triangleq \max_{j \in N(i), v_i(j) - p_j > 0} (v_i(j) - p_j)$.
    Let $D_i \triangleq \{j \in R \mid p_j < v_i(j), v_i(j) - p_j \geq U_i - \varepsilon \cdot v_i(j)\}$.
  Create the subgraph G_d as the subgraph consisting of $\left(\bigcup_{i \in L} D_i\right) \cup L$ and all edges.
  Find any arbitrary maximal matching M_d of G_d in order of highest bucket to lowest.
  for $(i, j) \in M_d$ **do**
    match j to i by setting $a_i = j$ and $a_{i'} = \perp$ for the previous owner i' of j.
    Increase the price of j to $p_j \leftarrow p_j + \varepsilon \cdot v_i(j)$.
  Let M' be the matched edges in this current iteration.
Return the matching $M = \arg\max_{M'} \left(w_{\max} \cdot \sum_{i \in L} v_i(a_i)\right)$ as the approximate maximum weight matching and $(i, a_i) \in M$ as the matched edges.

the price increase takes away from the overall utility of our matching. If not much weight is gained from the matching, then the price should not increase by much; otherwise, if a large amount of weight is gained from the matching, then we can afford to increase the price by a larger amount. We see later on in our analysis that this allows us to bucket the items according to their matched edge weight into $\lceil \log_{(1/\varepsilon)} m \rceil$ buckets. Such bucketing is useful in ensuring that we have sufficiently many happy bidders with a sufficiently large total matched weight. Finally, we return all matched items and bidders as our approximate matching and the sum of the weights of the matched items as the approximate weight. Obtaining the maximum weight of the matching in the original, unscaled graph is easy. We multiply the edge weights by $w_{\max}$ and the sum of these weights is the total weight of our approximate matching (Algorithm 1).

3.2 Analysis

In this section, we prove the approximation factor and round complexity of our algorithm. We use the same definition of happy that is defined in [6].

▶ **Definition 3** (ε-Happy [6]). *A bidder i is ε-happy if $u_i \geq v_i(j) - p_j - \varepsilon$ for every $j \in R$.*

▶ **Definition 4** (Unhappy). *A bidder i is **unhappy** at the end of round d if they are unmatched and their demand set is non-empty.*

Note that a happy bidder may never be unhappy and vice versa. For this definition, we assume that the demand set of a bidder can be computed at any point in time (not only when the Algorithm computers it).

Approach. The main challenge we face in our MWM analysis is that it is no longer sufficient to just show at least $(1 - \varepsilon)$-fraction of bidders in OPT are happy in order to obtain the desired approximation. Consider this simple example. Suppose a given instance has an optimum solution OPT with six matched bidders where one bidder is matched to an item

via a weight-1 edge. It also has five additional bidders matched to items via weight-$\frac{1}{\sqrt{n}}$ edges. Suppose we set $\varepsilon = 1/6$ to be a constant. Then, requiring 5/6-fraction of the bidders in OPT to be happy is not sufficient to get a 5/6-factor approximation. Suppose the five bidders matched with edges of weight $\frac{1}{\sqrt{n}}$ are the happy bidders. This is sufficient to satisfy the condition that 5/6-fraction of the bidders in OPT are happy. However, the total combined weight of the matching in this case is $\frac{5}{\sqrt{n}}$ while the weight of the optimum matching is $\left(1 + \frac{5}{\sqrt{n}}\right)$. The returned matching is then a $\frac{5}{\sqrt{n}}$-approximate MWM and for large n, this is much less than the desired 5/6-factor approximation.

Instead, we require a specific fraction of the total *weight* of the optimum solution, W_{OPT}, to be matched in our returned matching. We ensure this new requirement by considering two types of unhappy bidders. Type 1 unhappy bidders are bidders who are unhappy in round $k-1$ and remain unmatched in round k. Type 2 unhappy bidders are bidders who are unhappy in round $k-1$ and become matched in round k. We show that there exists a round where the following two conditions are satisfied:

1. We bucket the bidders in OPT according to the weight of their matched edge in OPT such that bidders matched with similar weight edges are in the same bucket; there exists a round where at most (ε^2)-fraction of the bidders in each bucket are Type 1 unhappy.
2. We charge the weight a Type 2 unhappy bidder obtains in round k to the bidder in round $k-1$; there exists a round $k-1$ where a total of at most $\varepsilon \cdot W_{\text{OPT}}$ weight is charged to Type 2 unhappy bidders.

Simultaneously satisfying both of the above conditions is enough to obtain our desired approximation. The rest of this section is devoted to showing our precise analysis using the above approach.

Detailed Analysis. Recall that we defined the utility of agent i to be the value of the item matched to her minus its price $u_i = v_i(a_i) - p_{a_i}$. In this section, we use the definition of ε-happy from Definition 3.

A similar observation to the observation made in [6] about the happiness of matched bidders can also be made in our case; however, since we are dealing with edge weights, we need to be careful to increment our prices in terms of the newly matched edge weight. In other words, two different bidders could be ε_1-happy and ε_2-happy after incrementing the price of their respective items by ε_1 and ε_2 where $\varepsilon_1 \neq \varepsilon_2$; the incremented prices ε_1 and ε_2 depend on the matched edge weights of the items assigned to the bidders. We prove the correct happiness guarantees given by our algorithm below.

▶ **Observation 5.** *At the end of every round, matched bidder i with matched edge (i, a_i) where a_i is priced at p_{a_i} are $(2\varepsilon \cdot v_i(a_i))$-happy. At the end of every round, unmatched bidders with empty demand sets D_i are ε-happy.*

For the weighted case, we need to consider what we call *weight buckets*. We define these weight buckets with respect to the optimum matching OPT. Recall our notation where $i \in \text{OPT}$ is a bidder who is matched in OPT and o_i is the matched item of the bidder in OPT. Bidder i is in the b-th weight bucket if $\varepsilon^{b-1} \leq v_i(o_i) < \varepsilon^{b-2}$.

▶ **Observation 6.** *All bidders $i \in OPT$ in bucket b satisfy $\varepsilon^{b-1} \leq v_i(o_i) < \varepsilon^{b-2}$.*

We now show that if a certain number of bidders in OPT are happy in our matching, then we obtain a matching with sufficiently large enough weight. However, our guarantee is somewhat more intricate than the guarantee provided in [6]. We show that in $\lceil \frac{\log^2(N)}{\varepsilon^4} \rceil$

rounds, there exists one round d where a set of sufficient conditions are satisfied to obtain our approximation guarantee. To do this, we introduce two types of unhappy bidders. Specifically, Type 1 and Type 2 unhappy bidders.

Each *unhappy* bidder results in some loss of total matched weight. However, at the end of round $k-1$ it is difficult to determine the exact amount of weight lost to unhappy bidders. Thus, in our analysis, we determine the amount of weight lost to unhappy bidders at the end of round $k-1$ in round k. The way that we determine the weight lost in round $k-1$ is by *retroactively* categorizing an unhappy bidder in round $k-1$ as a Type 1 or Type 2 unhappy bidder *depending on what happens in round k*. Thus, for our analysis, we categorize the bidders into categories of unhappy bidders for the *previous* round.

A ***Type* 1** unhappy bidder in round $k-1$ is a bidder i that remains unmatched at the end of round k. In other words, a Type 1 unhappy bidder was unhappy in round $k-1$ and either remains unhappy in round k or becomes happy because it does not have any demand items anymore (and remains unmatched). A ***Type* 2** unhappy bidder i in round $k-1$ is a bidder who was unhappy in round $k-1$ but is matched to an item in round k. Thus, a Type 2 unhappy bidder i in round $k-1$ becomes happy in round k because a new item is matched to i. Both types of bidders are crucial to our analysis given in the proof of Lemma 7 since they contribute differently to the potential amount of value that could be matched by our algorithm.

In the following lemma, let OPT be the optimum matching in graph G and $W_{\text{OPT}} = \sum_{i \in \text{OPT}} v_i(o_i)$. Let B_b be the set of bidders $i \in \text{OPT}$ in weight bucket b. If a Type 2 unhappy bidder i gets matched to a_i in round k, we say the weight $v_i(a_i)$ is ***charged*** to bidder i in round $k-1$. We denote this charged weight as $c_i(a_i)$ when performing calculations for round $k-1$.

▶ **Lemma 7.** *Provided $G = (L \cup R, E)$ and an optimum weighted matching OPT with weight $W_{OPT} = \sum_{i \in OPT} v_i(o_i)$, if in some round d of Algorithm 1 of Algorithm 1 both of the following are satisfied,*

1. *at most $\varepsilon^2 \cdot |B_b|$ of the bidders in each bucket b are Type 1 unhappy and*
2. *at most $\varepsilon \cdot W_{OPT}$ weight is charged to Type 2 unhappy bidders,*

then the matching in G has weight at least $(1-6\varepsilon) \cdot W_{OPT}$.

Proof. In such an iteration r, let HAPPY denote the set of all happy bidders. For any bidder $i \in \text{HAPPY} \cap \text{OPT}$, by Definition 3 and Observation 5, $u_i \geq v_i(o_i) - p_{o_i} - 2\varepsilon \cdot v_i(a_i)$ where o_i is the item matched to i in OPT and a_i is the item matched to i from our matching.

Before we go to the core of our analysis, we first make the observation that we can, in general, disregard prices of the items in our analysis. Let M be our matching. The sum of the utility of every matched bidder in our matching can be upper and lower bounded by the following expression:

$$\sum_{i \in M} (v_i(a_i) - p_{a_i}) \geq \sum_{i \in \text{OPT} \cap \text{HAPPY}} u_i \geq \sum_{i \in \text{OPT} \cap \text{HAPPY}} (v_i(o_i) - p_{o_i} - 2\varepsilon \cdot v_i(a_i)) \,.$$

As in the maximum cardinality matching case, all items with non-zero price are matched to a bidder. We can then simplify the above expression to give

$$\sum_{i\in M} v_i(a_i) - \sum_{j\in R} p_j \geq \sum_{i\in \text{OPT}\cap \text{Happy}} v_i(o_i) - \sum_{i\in \text{OPT}\cap \text{Happy}} p_{o_i} - \sum_{i\in \text{OPT}\cap \text{Happy}} 2\varepsilon \cdot v_i(a_i) \tag{1}$$

$$\sum_{i\in M\setminus(\text{OPT}\cap \text{Happy})} v_i(a_i) + \sum_{i\in \text{OPT}\cap \text{Happy}} (1+2\varepsilon)v_i(a_i) - \sum_{j\notin\{o_i | i\in \text{OPT}\cap \text{Happy}\}} p_j \geq \sum_{i\in \text{OPT}\cap \text{Happy}} v_i(o_i) \tag{2}$$

$$\sum_{i\in M} (1+2\varepsilon)v_i(a_i) - \sum_{j\notin\{o_i | i\in \text{OPT}\cap \text{Happy}\}} p_j \geq \sum_{i\in \text{OPT}\cap \text{Happy}} v_i(o_i). \tag{3}$$

Equation (1) follows from the fact that all non-zero priced items are matched. Equation (2) follows from separating $\text{OPT}\cap\text{Happy}$ from the left hand side and moving the summation of the $2\varepsilon \cdot v_i(a_i)$ values over $\text{OPT}\cap\text{Happy}$ from the right hand side to the left hand side. Finally, Equation (3) follows because $\sum_{i\in M}(1+2\varepsilon)v_i(a_i)$ upper bounds the left hand side expression for $\sum_{i\in M\setminus(\text{OPT}\cap\text{Happy})} v_i(a_i) + \sum_{i\in\text{OPT}\cap\text{Happy}}(1+2\varepsilon)v_i(a_i)$.

Let Unhappy_1 denote the set of Type 1 unhappy bidders and Unhappy_2 denote the set of Type 2 unhappy bidders. We let $c_i(a_i)$ be the weight charged to bidder i in Unhappy_2 in the next round. Recall that each bidder in Unhappy_2 is matched in the next round.

For each bucket, b, we can show the following using our assumption that at most $\varepsilon^2\cdot|B_b|$ of the bidders in bucket b are Type 1 unhappy,

$$\sum_{i\in B_b\cap\text{Happy}} v_i(o_i) \geq \sum_{i\in B_b\setminus\text{Unhappy}_2} v_i(o_i) - \varepsilon^2\cdot\varepsilon^{b-2}\cdot|B_b| \tag{4}$$

$$\geq \sum_{i\in B_b\setminus\text{Unhappy}_2} v_i(o_i) - \varepsilon\cdot\varepsilon^{b-1}\cdot|B_b| \tag{5}$$

$$\geq \sum_{i\in B_b\setminus\text{Unhappy}_2} v_i(o_i) - \sum_{i\in B_b}\varepsilon\cdot v_i(o_i). \tag{6}$$

Equation (4) shows that one can lower bound the sum of the optimum values of all happy bidders in bucket b by the sum of the optimum values of all bidders who are not Type-2 unhappy minus some factor. First, $\sum_{i\in B_b\setminus UNHAPPY_2} v_i(o_i)$ is the sum of the optimum values of all bidders in bucket b *except* for the Type-2 unhappy bidders. Now, we need to subtract the maximum sum of values given to the Type-1 unhappy bidders. We know that bucket b has at most $\varepsilon^2\cdot|B_b|$ Type-1 unhappy bidders. Each of these bidders could be assigned an optimum item with value at most ε^{b-2} (by Observation 6). Thus, the maximum value lost to Type-1 unhappy bidders is $\varepsilon^2\cdot\varepsilon^{b-2}\cdot|B_b|$, leading to Equation (4). Thus, the maximum value of weight lost to all Type 1 unhappy bidders in bucket b is $\varepsilon^2\cdot\varepsilon^{b-2}\cdot|B_b|$. Then, Equation (6) follows because $v_i(o_i)\geq\varepsilon^{b-1}$ for all $i\in B_b$. This means that $\sum_{i\in B_b} v_i(o_i)\geq\varepsilon^{b-1}\cdot|B_b|$.

Summing Equation (6) over all buckets b we obtain

$$\sum_{i\in\text{OPT}\cap\text{Happy}} v_i(o_i) \geq \sum_{i\in\text{OPT}\setminus\text{Unhappy}_2} v_i(o_i) - \sum_{i\in\text{OPT}}\varepsilon\cdot v_i(o_i). \tag{7}$$

We now substitute our expression obtained in Equation (7) into Equation (3),

$$\sum_{i\in M}(1+2\varepsilon)v_i(a_i) - \sum_{j\notin\{o_i|i\in\text{OPT}\cap\text{Happy}\}} p_j \geq \sum_{i\in\text{OPT}\setminus\text{Unhappy}_2} v_i(o_i) - \sum_{i\in\text{OPT}}\varepsilon\cdot v_i(o_i). \tag{8}$$

The last thing that we need to show is a bound on the weight lost due to bidders in $\text{OPT} \cap \text{UNHAPPY}_2$. We now consider our second assumption which states that at most $\varepsilon \cdot W_{\text{OPT}}$ weight is charged to Type 2 unhappy bidders. Since all bidders $i \in \text{UNHAPPY}_2$ become happy in the next round, we can bound the weights charged to the Type 2 unhappy bidders using Observation 5 by

$$\sum_{i \in \text{OPT} \cap \text{UNHAPPY}_2} c_i(a_i) \geq \sum_{i \in \text{OPT} \cap \text{UNHAPPY}_2} \left(v_i(o_i) - p_{o_i} - 2\varepsilon \cdot c_i(a_i)\right). \tag{9}$$

Note first that $\sum_{j \notin \{o_i | i \in \text{OPT} \cap \text{HAPPY}\}} p_j \geq \sum_{i \in \text{OPT} \cap \text{UNHAPPY}_2} p_{o_i}$ since $\text{OPT} \setminus (\text{OPT} \cap \text{HAPPY})$ includes $\text{OPT} \cap \text{UNHAPPY}_2$ so we can remove the prices from these bounds in Equation (10). We add Equation (9) to Equation (8) and use our assumptions to obtain

$$\sum_{i \in M} (1+2\varepsilon) v_i(a_i) + \sum_{i \in \text{OPT} \cap \text{UNHAPPY}_2} c_i(a_i) \geq \sum_{i \in \text{OPT}} (1-\varepsilon) \cdot v_i(o_i) - \sum_{i \in \text{OPT} \cap \text{UNHAPPY}_2} 2\varepsilon \cdot c_i(a_i) \tag{10}$$

$$\sum_{i \in M} (1+2\varepsilon) v_i(a_i) \geq \sum_{i \in \text{OPT}} (1-\varepsilon) \cdot v_i(o_i) - \sum_{i \in \text{OPT} \cap \text{UNHAPPY}_2} (1+2\varepsilon) \cdot c_i(a_i) \tag{11}$$

$$\geq \left(\sum_{i \in \text{OPT}} (1-\varepsilon) \cdot v_i(o_i) \right) - (1+2\varepsilon) \cdot \varepsilon \cdot W_{\text{OPT}} \tag{12}$$

$$\sum_{i \in M} v_i(a_i) \geq \frac{(1-4\varepsilon)}{(1+2\varepsilon)} \cdot \sum_{i \in \text{OPT}} v_i(o_i) \tag{13}$$

$$\sum_{i \in M} v_i(a_i) \geq (1-6\varepsilon) W_{\text{OPT}}. \tag{14}$$

Equation (10) follows from summing $\sum_{i \in \text{OPT}} (1-\varepsilon) \cdot v_i(o_i) = \sum_{i \in \text{OPT} \setminus \text{UNHAPPY}_2} v_i(o_i) + \sum_{i \in \text{OPT} \cap \text{UNHAPPY}_2} v_i(o_i) - \sum_{i \in \text{OPT}} \varepsilon \cdot v_i(o_i) = \sum_{i \in \text{OPT}} (1-\varepsilon) \cdot v_i(o_i)$. Equation (11) follows from moving $\sum_{i \in \text{OPT} \cap \text{UNHAPPY}_2} c_i(a_i)$ to the right hand side. Equation (12) follows from substituting our assumption that $\sum_{i \in \text{OPT} \cap \text{UNHAPPY}_2} c_i(a_i) \leq \varepsilon \cdot W_{\text{OPT}}$. Equation (13) follows from simple manipulations and since $W_{\text{OPT}} = \sum_{i \in \text{OPT}} v_i(o_i)$. Finally, Equation (14) follows because $\frac{(1-4\varepsilon)}{(1+2\varepsilon)} \geq (1-6\varepsilon)$ for all $\varepsilon > 0$ and gives the desired approximation given in the lemma statement. ◀

We show that the conditions of Lemma 7 are satisfied for at least one round if the algorithm is run for at least $\lceil \frac{\log^2(N)}{\varepsilon^4} \rceil$ rounds. We prove this using potential functions similar to the potential functions used for MCM. We first bound the maximum value of these potential functions.

▶ **Lemma 8.** *Define the potential function* $\Phi_{items} \triangleq \sum_{j \in R} p_j$. *Then the upper bound for this potential is* $\Phi_{items} \leq W_{OPT}$.

Proof. We show that the potential function Φ_{items} is always upper bounded by W_{OPT} via a simple proof by contradiction. Suppose that $\Phi_{items} > W_{\text{OPT}}$, then, we show that the matching obtained by our algorithm has weight greater than W_{OPT}, a contradiction. For a bidder/item pair, (i, a_i), the weight of edge (i, a_i) is at least $p_{a_i} - 2\varepsilon \cdot v_i(a_i)$. Let p'_{a_i} be the price of a_i before the last reassignment of a_i to i. Furthermore, since i picked a_i, it must mean that $v_i(a_i) > p'_{a_i}$ since a_i would not be included in D_i otherwise. This means that the sum of the weights of all the matched edges is at least $\sum_{(i,a_i)} v_i(a_i) > \sum_{(i,a_i)} p'_{a_i} \geq \Phi_{items} > W_{\text{OPT}}$ by our assumption that $\Phi_{items} > W_{\text{OPT}}$. Thus, we obtain that we get a matching with greater weight than the optimum weight matching, a contradiction. ◀

▶ **Lemma 9.** *There exists a phase* $d \leq \frac{\log^2(N)}{\varepsilon^4}$ *wherein* both *of the following statements are satisfied:*

1. *At most* $\varepsilon^2 \cdot |B_b|$ *bidders in bucket* b *are Type* 1 *unhappy for all buckets* b*;*
2. *The set of all Type* 2 *unhappy bidders results in a loss of less than* $\varepsilon \cdot W_{OPT}$ *weight (in charged weight) where* W_{OPT} *is the optimum weight attainable by the matching.*

Recall that we assign each bidder to a weight bucket using the weight assigned to the bidder in OPT.

Using the above lemmas, we can prove our main theorem that our algorithm gives a $(1-7\varepsilon)$-approximate maximum weight bipartite matching in $O\left(\frac{\log^3(N)\cdot\log(n)}{\varepsilon^4}\right)$ distributed rounds using $O(\log n)$ communication complexity.

▶ **Theorem 10.** *Algorithm 1 returns a* $(1-7\varepsilon)$*-approximate maximum weight bipartite matching* M *in* $O\left(\frac{\log^3(N)\cdot\log n}{\varepsilon^4}\right)$ *rounds whp using* $O(\log n)$ *bits of communication per message in the broadcast model.*

Reducing the Round Complexity. We can use the following transformation from Gupta-Peng [18] to reduce the round complexity at an increase in the communication complexity. For completeness, we give the theorem for the transformation the full version of our paper.

▶ **Theorem 11.** *There exists a* $(1-\varepsilon)$*-approximate distributed algorithm for maximum weight bipartite matching that runs in either:*

- $O\left(\log n \cdot \frac{\log(1/\varepsilon)}{\varepsilon^7}\right)$ *rounds of communication using* $O\left(\frac{\log^2 n}{\varepsilon}\right)$ *bits of communication, or*
- $O\left(\log n \cdot \frac{\log(1/\varepsilon)}{\varepsilon^8}\right)$ *rounds of communication using* $O\left(\log^2 n\right)$ *bits of communication*

where we assume the maximum ratio between weights of edges in the input graph is $\text{poly}(n)$*. In the blackboard model, this requires a total of* $O\left(\frac{n\log^3 n\log(1/\varepsilon)}{\varepsilon^8}\right)$ *bits of communication.*

3.3 Semi-Streaming Implementation

The implementation of this algorithm in the semi-streaming model is very similar to the implementation of the MCM algorithm of Assadi et al. [6].

▶ **Lemma 12.** *Given a weighted graph* $G = (V, E)$ *as input in an arbitrary edge-insertion stream where all weights are at most* $\text{poly}(n)$*, there exists a semi-streaming algorithm which uses* $O\left(\frac{\log^2(N)}{\varepsilon^4}\right)$ *passes and* $O\left(n \cdot \log n \cdot \log(1/\varepsilon)\right)$ *space that computes a* $(1-\varepsilon)$*-approximate maximum weight bipartite matching for any* $\varepsilon > 0$*.*

Reducing the Number of Passes. We use the transformation of [18] as stated the full version of our paper to eliminate our dependence on n within our number of rounds. The transformation is as follows. For each instance of $(1+\varepsilon)$-MWM, we maintain the prices in our algorithm for each of the nodes involved in each of the copies of our algorithm. When an edge arrives in the stream, we first partition it into the relevant level of the appropriate copy of the structure.

▶ **Theorem 13.** *There exists a* $(1-\varepsilon)$*-approximate streaming algorithm for maximum weight bipartite matching that uses* $O\left(\frac{\log(1/\varepsilon)}{\varepsilon^7}\right)$ *passes in* $O\left(n \cdot \log n \cdot \log(1/\varepsilon)\right)$ *space.*

3.4 Shared-Memory Parallel Implementation

The implementation of this algorithm in the shared-memory work-depth model follows almost directly from our auction algorithm. We show the following lemma when directly implementing our auction algorithm.

▶ **Lemma 14.** *Given a weighted graph $G = (V, E)$ as input where all weights are at most* $\text{poly}(n)$*, there exists a shared-memory parallel algorithm which uses $O\left(\frac{m \log^2 n}{\varepsilon^4}\right)$ work and $O\left(\frac{\log^4 n}{\varepsilon^4}\right)$ depth that computes a $(1-\varepsilon)$-approximate maximum weight bipartite matching for any $\varepsilon > 0$.*

Using the transformations, we can reduce the number of rounds for our shared-memory parallel algorithms.

▶ **Theorem 15.** *Given a weighted graph $G = (V, E)$ as input where all weights are at most* $\text{poly}(n)$*, there exists a shared-memory parallel algorithm which uses $O\left(\frac{m \log(1/\varepsilon)}{\varepsilon^6}\right)$ work and $O\left(\frac{\log(n) \cdot \log(1/\varepsilon)}{\varepsilon^8}\right)$ depth that computes a $(1-\varepsilon)$-approximate maximum weight bipartite matching for any $\varepsilon > 0$.*

3.5 MPC Implementation

We implement our auction algorithm in the MPC model below.

▶ **Lemma 16.** *Given a weighted graph $G = (V, E)$ as input where all weights are at most* $\text{poly}(n)$*, there exists a MPC algorithm using $O\left(\frac{\log^2(N) \cdot \log\log n}{\varepsilon^4}\right)$ rounds, $O(n)$ space per machine, and $O(n \log(1/\varepsilon) + m)$ total space that computes a $(1-\varepsilon)$-approximate maximum weight bipartite matching for any $\varepsilon > 0$.*

As before, we can improve the complexity of our MPC algorithm using the transformations the full version of our paper.

▶ **Theorem 17.** *Given a weighted graph $G = (V, E)$ as input where all weights are at most* $\text{poly}(n)$*, there exists a MPC algorithm using $O\left(\frac{\log(1/\varepsilon) \cdot \log\log n}{\varepsilon^7}\right)$ rounds, $O(n)$ space per machine, and $O\left(\frac{m \log(1/\varepsilon) \log n}{\varepsilon}\right)$ total space that computes a $(1-\varepsilon)$-approximate maximum weight bipartite matching for any $\varepsilon > 0$.*

4 A $(1-\varepsilon)$-approximation Auction Algorithm for b-Matching

We show in this section that we also obtain an auction-based algorithm for MCBM by extending the auction-based algorithm of [6]. This algorithm also leads to better streaming algorithms for this problem. We use the techniques introduced in the auction-based MCM algorithm of Assadi, Liu, and Tarjan [6] as well as new techniques developed in this section to obtain a $(1-\varepsilon)$-approximation algorithm for bipartite maximum cardinality b-matching. The maximum cardinality b-matching problem is defined in Definition 18.

▶ **Definition 18** (Maximum Cardinality Bipartite b-Matching (MCBM))**.** *Given an undirected, unweighted, bipartite graph $G = (L \cup R, E)$ and a set of values $\{b_v \leq |R| \mid v \in L \cup R\}$, a* ***maximum cardinality b-matching*** *(MCBM) finds a matching of maximum cardinality between vertices in L and R where each vertex $v \in L \cup R$ is matched to at most b_v other vertices.*

The key difference between our algorithm for b-matching and the MCM algorithm of [6] is that we have to account for when more than one item is assigned to each bidder in L; in fact, up to b_i items in R can be assigned to any bidder $i \in L$. This one to many relationship calls for a different algorithm and analysis. The crux of our algorithm in this section is to create b_i copies of each bidder i and b_j copies of each item j. Then, copies of items maintain their own prices and copies of bidders can each choose at most one item. We define some notation to describe these copies. Let C_i be the set of copies of bidder i and C_j be the set of copies of item j. Then, we denote each copy of i by $i^{(k)} \in C_i$ for $k \in [b_i]$ and each copy of j by $j^{(k)} \in C_j$ for $k \in [b_j]$. As before, we denote a bidder and their currently matched item by $\left(i^{(k)}, a_{i^{(k)}}\right)$.

In MCBM, we require that the set of all items chosen by different copies of the same bidder to include at most one copy of each item. In other words, we require if $j^{(k)} \in \bigcup_{i' \in C_i} a_{i'}$, then no other $j^{(l)} \in \bigcup_{i' \in C_i} a_{i'}$ for any $j^{(k)}, j^{(l)} \in C_j$ and $k \neq l$. This *almost* reduces to the problem of finding a maximum cardinality matching in a $\sum_{i \in L} b_i + \sum_{j \in R} b_j$ sized bipartite graph but *not quite*. Specifically, the main challenge we must handle is when multiple copies of the same bidder want to be matched to copies of the *same* item. In this case, we cannot match any of these bidder copies to copies of the same item and thus must somehow handle the case when there exist items of lower price but we cannot match them.

In addition to handling the above hard case, as before, the crux of our proof relies on a variant of the ε-happy definition and the definitions of appropriate potential functions.

Recall from the MCM algorithm of [6] that an ε-happy bidder has utility that is at least the utility gained from matching to any other item (up to an additive ε). Such a definition is insufficient in our setting since it may be the case that matching to a copy of an item that is already matched to a different copy of the same bidder results in lower cost. However, such a match is not helpful since any number of matches between copies of the same bidder and copies of the same item contributes a value of one to the cardinality of the eventual matching.

Our algorithm solves all of the above challenges and provides a $(1-\varepsilon)$-approximate MCBM in asymptotically the same number of rounds as the MCM algorithm of [6]. We describe our auction based algorithm for MCBM next and the precise pseudocode is given in Algorithm 2. Our algorithm uses the parameters defined in Table 2. We show the following results using our algorithm. We discuss semi-streaming implementations of our algorithm in Section 4.3. Let L be the half with fewer numbers of nodes.

▶ **Theorem 2** (Maximum Cardinality Bipartite b-Matching). *There exists an auction algorithm for maximum cardinality bipartite b-matching (MCBM) that gives a $(1-\varepsilon)$-approximation for any $\varepsilon > 0$ and runs in $O\left(\frac{\log n}{\varepsilon^2}\right)$ rounds of communication. This algorithm can be implemented in the multi-round, semi-streaming model using $O\left(\left(\sum_{i \in L} b_i + |R|\right) \log(1/\varepsilon)\right)$ space and $O\left(\frac{1}{\varepsilon^2}\right)$ passes. Our algorithm can be implemented in the shared-memory work-depth model in $O\left(\frac{\log^3 n}{\varepsilon^2}\right)$ depth and $O\left(\frac{m \log n}{\varepsilon^2}\right)$ total work.*

4.1 Algorithm Description

The algorithm works as follows. We assign to each bidder, i, b_i unmatched slots and the goal is to fill all slots (or as many as possible). For each bidder $i \in L$ and each item $j \in R$, we create b_i and b_j copies, respectively, and assign these copies to new sets L' and R', respectively (Algorithm 2). This step of the algorithm changes slightly in our streaming implementation. For each bidder and item with an edge between them $(i, j) \in E$, we create

Algorithm 2 Auction Algorithm for Bipartite b-Matching.

Input: Graph $G = (L \cup R, E)$ and parameter $0 < \varepsilon < 1$.
Output: An $(1 - \varepsilon)$-approximate maximum cardinality bipartite b-matching.

Create L', R', E' and graph G'. (Defined in Table 2.)
For each $i' \in L'$, set $(i', \perp)$ where $a_{i'} = \perp$, $c_{i'} \leftarrow 0$.
For each $j' \in R'$, set $p_{j'} = 0$.
Set $M_{\max} \leftarrow \emptyset$.
for $d = 1, \ldots, \lceil \frac{2}{\varepsilon^2} \rceil$ **do**
  For each unmatched bidder $i' \in L'$, find $D_{i'} \leftarrow$ FindDemandSet$(G', i', c_{i'})$ [Algorithm 3].
  Create G'_d.
  Find any arbitrary non-duplicate maximal matching $\widehat{M}_d$ of G_d.
  for $(i', j') \in \widehat{M}_d$ **do**
    Set $a_{i'} = j'$ and $a_{i_{prev}} = \perp$ for the previous owner i_{prev} of j'.
    Increase $p_{j'} \leftarrow p_{j'} + \varepsilon$.
  For each $i' \in L'$ with $D_{i'} \neq \emptyset$ and $a_{i'} = \perp$, increase $c_{i'} \leftarrow c_{i'} + \varepsilon$.
  Using M'_d compute M_d where for each $(i', j') \in M'_d$, add (i, j) to M_d if $(i, j) \notin M_d$.
  If $|M_d| > |M_{\max}|$, $M_{\max} \leftarrow M_d$.
Return $M_{\max}$.

Algorithm 3 FindDemandSet$(G' = (L' \cup R', E'), i', c_{i'})$.

Let $N'(i') = \{j' \in R' \mid j^{(l)} \neq a_{i^{(k)}} \forall i^{(k)} \in C_i, \forall j^{(l)} \in C_j \wedge p_{j'} \geq c_{i'} \forall j' \in C_j\}$.
$D_{i'} \leftarrow \arg\min_{j' \in N'(i'), p_{j'} < 1} (p_{j'})$.
Return $D_{i'}$.

a biclique between C_i and C_j; the edges of all created bicliques is the set of edges E'. The graph $G' = (L' \cup R', E')$ is created as the graph consisting of nodes in $L' \cup R'$ and edges in E'. As before, we initialize each bidder's assigned item to $\perp$ (Algorithm 2). Then, we set the price for each copy in R' to 0 (Algorithm 2).

In our MCBM algorithm, we additionally set a price cutoff for each bidder $c_{i'}$ initialized to 0 (Algorithm 2). Such a cutoff helps us to prevent bidding on lower price items previously not bid on because they were matched to another copy of the same bidder. More details on how the cutoff prevents bidders from bidding against themselves can be found in the proof of Lemma 25. We maintain the maximum cardinality matching we have seen in $M_{\max}$ (Algorithm 2). We perform $\lceil \frac{2}{\varepsilon^2} \rceil$ rounds of assigning items to bidders (Algorithm 2). For each round, we first find the demand set for each unmatched bidder $i' \in L'$ using Algorithm 3 (Algorithm 2). The demand set is defined with respect to the cutoff price $c_{i'}$ and the set of items assigned to other copies of bidder i. The demand set considers all items $j' \in R'$ that are neighbors of i' where no copy of j, $j^{(k)} \in C_j$, is assigned to any copies of i and $p_{j'} \geq c_{i'}$ (Algorithm 3, Algorithm 3). From this set of neighbors, the returned demand set is the set of item copies with the minimum price in $N'(i')$ (Algorithm 3).

Using the induced subgraph of $\left(\bigcup_{i' \in L'} D_{i'}\right) \cup L'$ (Algorithm 2), we greedily find a maximal matching while avoiding assigning copies of the same item to copies of the same bidder (Algorithm 2). We call such a maximal matching that does not assign more than one copy of the same item to copies of the same bidder to be a ***non-duplicate*** maximal matching. This greedy matching prioritizes the unmatched items by first matching the unmatched items and then matching the matched items. We can perform a greedy matching by matching an

edge if the item is unmatched and no copies of the bidder it will match to is matched to another copy of the item. For each newly matched item (Algorithm 2), we rematch the item to the newly matched bidder (Algorithm 2). We increase the price of the newly matched item (Algorithm 2). For each remaining unmatched bidder, we increase the cutoff price by ε (Algorithm 2).

We compute the corresponding matching in the original graph using M'_d (Algorithm 2) by including one edge (i, j) in the matching if and only if there exists at least one bidder copy $i' \in C_i$ matched to at least one copy of the item $j' \in C_j$. Finally, we return the maximum cardinality $M_{\max}$ matching from all iterations as our $(1-\varepsilon)$-approximate maximum cardinality b-matching (Algorithm 2).

4.2 Analysis

In this section, we analyze the approximation error of our algorithm and prove that it provides a $(1-\varepsilon)$-approximate maximum cardinality b-matching.

Approach. We first provide an intuitive explanation of the approach we take to perform our analysis and then we give our precise analysis. Here, we describe both the challenges in performing the analysis and explain our choice of certain methods in the algorithm to facilitate our analysis. We especially highlight the parts of our algorithm and analysis that differ from the original MCM algorithm of [6]. First, in order to show the approximation factor of our algorithm, we require that the utility obtained by a large number of matched bidders from our algorithm is greater than the corresponding utility from switching to the optimum items in the optimum matching. For b-matching, any combination of matched items and bidder copies satisfy this criteria. Furthermore, matching multiple item copies of the same item to bidder copies of the same bidder does not increase the utility of the bidder. Thus, we look at matchings where at most one copy of each bidder is matched to at most one copy of each item. Recall our definition of ε-happy given in Definition 3 and we let HAPPY be the set of bidders satisfying that definition.

For b-matching, each bidder i is matched to a set of at most b_i items. Let $(i, O_i) \in \text{OPT}$ denote the set of items $O_i \subseteq R$ matched to bidder i in OPT. Recall [6] that the proof requires $u_i \geq 1 - p_{o_i} - \varepsilon$ for every bidder $i \in \text{HAPPY} \cap \text{OPT}$ to show that $\sum_{i \in L} u_i \geq \sum_{i \in \text{HAPPY} \cap \text{OPT}} 1 - p_{o_i} - \varepsilon$. Using our bidder copies, C_i, the crux of our analysis proof is to show that for every $(i, O_i) \in \text{OPT}$, we can *assign* the items in O_i to the set of happy bidder copies in C_i such that each happy bidder copy receives a unique item, denoted by $r_{i'}$, and $c_{i'} \leq p_{\min, r_{i'}}$ where $p_{\min, r_{i'}}$ is the price of the minimum priced copy of $r_{i'}$. Using this assignment, we are able to show once again that $\sum_{i' \in L'} u_{i'} \geq \sum_{i' \in \text{HAPPY} \cap \text{OPT}} 1 - p_{\min, r_{i'}} - \varepsilon$. This requires a precise definition of $\text{HAPPY} \cap \text{OPT}$. Let $S_i \subseteq C_i$ be the set of all happy bidders in C_i. Recall that the optimum solution gives a matching between a bidder $i \in L$ and potentially multiple items in R; we turn this matching into an optimum matching in G'. If $|S_i| \leq |O_i|$, then all happy copies in S_i are in OPT; otherwise, we pick an arbitrary set of $|O_i|$ happy bidder copies in S_i to be in OPT. Then, the summation is determined based on this set of happy bidder copies in $\text{HAPPY} \cap \text{OPT}$.

Once we have shown this, the only other remaining part of the proof is to show that in the $\lceil \frac{2}{\varepsilon^2} \rceil$ rounds that we run the algorithm the potential increases by ε for every unhappy bidder in OPT for each round that the bidder is unhappy. As in the case for MCM, the price of an item increases whenever it becomes re-matched. Hence, Π_{items} increases by ε each time a bidder who was happy becomes unhappy. To ensure that $\Pi_{bidders}$ increases by ε

for each bidder who was unhappy and remains unhappy, we set a ***cutoff*** price that increases by ε for each round where a bidder remains unhappy. Thus, this cutoff guarantees that $\Pi_{bidders}$ increases by ε each time.

Detailed Analysis. Now we show our detailed analysis that formalizes our approach described above. We first show that our algorithm maintains both Invariant 20 and Invariant 21. We also show our algorithm obeys the following invariant.

▶ **Invariant 19.** *The set of matched items of all copies of any bidder $i \in L$ contains at most one copy of each item. In other words, $\left|\bigcup_{i' \in C_i} a_{i'} \cap C_j\right| \leq 1$ for all $j \in R$.*

We restate two invariants used in [6] below. We prove that our Algorithm 2 also maintains these two invariants.

▶ **Invariant 20** (Non-Zero Price Matched [6])**.** *Any item j with positive price $p_j > 0$ is matched.*

▶ **Invariant 21** (Maximum Utility [6])**.** *The total utility of all bidders is at most the cardinality of the matching minus the total price of the items.*

▶ **Lemma 22.** *Algorithm 2 maintains Invariant 19, Invariant 20, and Invariant 21.*

We follow the style of analysis outlined in [6] by defining appropriate definitions of ε-happy and appropriate potential functions Π_{items} and $\Pi_{bidders}$. In the case of b-matching, we modify the definition of ε-happy in this setting to be the following.

▶ **Definition 23** ((ε, c)-Happy)**.** *A bidder $i' \in L'$ is (ε, c)-happy (at the end of a round) if $u_{i'} \geq 1 - p_{j'} - \varepsilon$ for all neighbors in the set $N'(i')$ where $N'(i')$ is as defined in Algorithm 3 of Algorithm 3 (i.e. contains all neighboring items j' where $p_{j'} \geq c_{i'}$ and no copy of the neighbor is matched to another copy of i').*

At the end of each round, it is easy to show that all matched i' and i' whose demand sets $D_{i'}$ are empty are $(\varepsilon, c_{i'})$-happy.

▶ **Lemma 24.** *At the end of any round, if bidder i' is matched or if their demand set is empty, $D_{i'} = \emptyset$, then i' is $(\varepsilon, c_{i'})$-happy.*

In addition to the new definition of happy, we require another crucial observation before we prove our approximation guarantee. Specifically, we show that for any set of bidder copies C_i and any set of $|C_i|$ items $I \subseteq R$, Lemma 24 is sufficient to imply there exists at least one assignment of items in I to happy bidders in S_i such that each item is assigned to at most one bidder and each happy bidder is assigned at least one item where the minimum price of the item is at least the cutoff price of the bidder.

▶ **Lemma 25.** *For a set of bidder copies C_i and any set $I \subseteq R$ of $|C_i|$ items where $(i, j) \in E$ for all items $j \in I$, there exists at least one assignment of items in I to bidders in C_i, where we denote the item assigned to copy i' by $r_{i'}$, that satisfy the following conditions:*

1. *The assignment is a one-to-one mapping between bidders in C_i and items in I.*
2. *Any item j matched to i' is assigned to i'.*
3. *Let $r^*_{i'}$ be the lowest cost copy of item $r_{i'}$, $r^*_{i'} = \arg\min_{j' \in C_{r_{i'}}} (p_{j'})$; then $p_{r^*_{i'}} \geq c_{i'}$ for all $i' \in C_i$.*

We now perform the approximation analysis. Suppose as in the case of MCM, we have at least $(1-\varepsilon)|\text{OPT}|$ happy bidders in OPT (i.e. $|\textsc{Happy} \cap \text{OPT}| \geq (1-\varepsilon)|\text{OPT}|$), then we show that we can obtain a $(1-\varepsilon)$-approximate MCBM. Let OPT be an optimum MCBM matching and $|\text{OPT}|$ be the cardinality of this matching.

▶ **Lemma 26.** *Assuming $|\textsc{Happy} \cap OPT| \geq (1-\varepsilon)|OPT|$, then we obtain a $(1-2\varepsilon)$-approximate MCBM.*

The potential argument proof is almost identical to that for MCM provided our use of $c_{i'}$. Specifically, as in the case for MCM, we use the same potential functions and using these potential functions, we show that our algorithm terminates in $O\left(\frac{1}{\varepsilon^2}\right)$ rounds. The key difference between our proof and the proof of MCM explained in [6] is our definition of $\Pi_{bidders}$ which is precisely defined in the proof of Lemma 27 below.

▶ **Lemma 27.** *In $\lceil \frac{2}{\varepsilon^2} \rceil$ rounds, there exists at least one round where $|OPT \cap \textsc{Happy}| \geq (1-\varepsilon)|OPT|$.*

Using the above lemmas, we can prove the round complexity of Theorem 2 to be $O\left(\frac{1}{\varepsilon^2}\right)$ by Lemma 26 and Lemma 27.

▶ **Theorem 28.** *There exists an auction algorithm for maximum cardinality bipartite b-matching (MCBM) that gives a $(1-\varepsilon)$-approximation for any $\varepsilon > 0$ and runs in $O\left(\frac{\log n}{\varepsilon^2}\right)$ rounds of communication using $O(b \log n)$ bits per message in the blackboard distributed model. In total, the number of bits used by the algorithm is $O\left(\frac{nb \log^2 n}{\varepsilon^2}\right)$.*

4.3 Semi-Streaming Implementation

We now show an implementation of our algorithm to the semi-streaming setting and show the following lemma which proves the semi-streaming portion of our result in Theorem 2. We are guaranteed $\varepsilon \geq \frac{1}{2n^2}$; otherwise, an exact matching is found. In order to show the space bounds, we use an additional lemma below that upper and lower bounds the prices of any copies of the same item in R'.

▶ **Lemma 29.** *For any $j \in R$, let $j_{\min}$ be the minimum priced copy in C_j and $j_{\max}$ be the maximum priced copy in C_j. Then, $p_{j_{\max}} - p_{j_{\min}} \leq \varepsilon$.*

Using the above, we prove our desired bounds on the number of passes and the space used.

▶ **Theorem 30.** *There exists a semi-streaming algorithm for maximum cardinality bipartite b-matching that uses $O\left(\frac{1}{\varepsilon^2}\right)$ rounds and $\widetilde{O}\left(\left(\sum_{i \in L} b_i + |R|\right) \log(1/\varepsilon)\right)$ space where L is the side with the smaller number of nodes in the input graph.*

We note that the space bound is necessary in order to report the solution. (There exists a given input where reporting the solution requires $\widetilde{O}\left(\left(\sum_{i \in L} b_i + |R|\right) \log(1/\varepsilon)\right)$ space.) Thus, our algorithm is tight with respect to this notion.

4.4 Shared-Memory Parallel Implementation

We now show an implementation of our algorithm to the shared-memory parallel setting. The main challenge for this setting is obtaining an algorithm for obtaining non-duplicate maximal matchings. To obtain non-duplicate maximal matchings, we just need to modify

the maximal matching algorithm of [9] to obtain a maximal matching with the non-duplicate characteristic. Namely, the modification we make is to consider all copies of a node to be neighbors of each other. Since there can be at most n copies of a node, this increases the degree of each node by at most n. Hence, the same analysis as the original algorithm still holds in this new setting.

▶ **Theorem 31.** *There exists a shared-memory parallel algorithm for maximum cardinality bipartite b-matching that uses* $O\left(\frac{\log^3 n}{\varepsilon^2}\right)$ *depth and* $O\left(\frac{m \log n}{\varepsilon^2}\right)$ *total work where* L *is the side with the smaller number of nodes in the input graph.*

References

1 Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. In *Proceedings of the 38th International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'11, pages 526–538, Berlin, Heidelberg, 2011. Springer-Verlag.

2 Kook Jin Ahn and Sudipto Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. *ACM Trans. Parallel Comput.*, 4(4), January 2018. doi:10.1145/3154855.

3 Sepehr Assadi. A two-pass (conditional) lower bound for semi-streaming maximum matching. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 708–742. SIAM, 2022. doi:10.1137/1.9781611977073.32.

4 Sepehr Assadi, Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. Semi-streaming bipartite matching in fewer passes and optimal space. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 627–669. SIAM, 2022.

5 Sepehr Assadi, Gillat Kol, Raghuvansh R. Saxena, and Huacheng Yu. Multi-pass graph streaming lower bounds for cycle counting, max-cut, matching size, and other problems. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 354–364, 2020. doi:10.1109/FOCS46700.2020.00041.

6 Sepehr Assadi, S. Cliff Liu, and Robert E. Tarjan. *An Auction Algorithm for Bipartite Matching in Streaming and Massively Parallel Computation Models*, pages 165–171. SIAM, 2021. doi:10.1137/1.9781611976496.18.

7 Aaron Bernstein, Aditi Dudeja, and Zachary Langley. A framework for dynamic matching in weighted graphs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 668–681, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451113.

8 Dimitri P Bertsekas. A new algorithm for the assignment problem. *Mathematical Programming*, 21(1):152–171, 1981.

9 Guy E. Blelloch, Jeremy T. Fineman, and Julian Shun. Greedy sequential maximal independent set and matching are parallel on average. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 308–317, 2012.

10 Gabrielle Demange, David Gale, and Marilda Sotomayor. Multi-item auctions. *Journal of political economy*, 94(4):863–872, 1986.

11 Shahar Dobzinski, Noam Nisan, and Sigal Oren. Economic efficiency requires interaction. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 233–242, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2591796.2591815.

12 Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards B*, 69:125–130, 1965.

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

14 Manuela Fischer, Slobodan Mitrović, and Jara Uitto. Deterministic $(1+\varepsilon)$-approximate maximum matching with poly$(1/\varepsilon)$ passes in the semi-streaming model and beyond. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 248–260, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3520039.

15 Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted matchings via unweighted augmentations. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, pages 491–500, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3293611.3331603.

16 Mohsen Ghaffari, Christoph Grunau, and Slobodan Mitrović. Massively parallel algorithms for b-matching. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2022.

17 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. *On the communication and streaming complexity of maximum bipartite matching*, pages 468–485. SIAM, 2012. doi:10.1137/1.9781611973099.41.

18 Manoj Gupta and Richard Peng. Fully dynamic (1+ e)-approximate matchings. In *FOCS*, pages 548–557. IEEE Computer Society, 2013.

19 Nicholas J. A. Harvey. Algebraic structures and algorithms for matching and matroid problems. *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 531–542, 2006.

20 John E. Hopcroft and Richard M. Karp. A n5/2 algorithm for maximum matchings in bipartite. In *12th Annual Symposium on Switching and Automata Theory (swat 1971)*, pages 122–125, 1971. doi:10.1109/SWAT.1971.1.

21 Shang-En Huang and Hsin-Hao Su. (1-ϵ)-approximate maximum weighted matching in poly$(1/\epsilon,$ log n) time in the distributed and parallel settings. *CoRR*, abs/2212.14425, 2022. doi:10.48550/arXiv.2212.14425.

22 Arun Jambulapati, Yang P. Liu, and Aaron Sidford. Parallel reachability in almost linear work and square root depth. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1664–1686, 2019. doi:10.1109/FOCS.2019.00098.

23 Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1679–1697. SIAM, 2013.

24 Christian Konrad, Peter Robinson, and Viktor Zamaraev. Robust lower bounds for graph problems in the blackboard model of communication. *CoRR*, abs/2103.07027, 2021. arXiv:2103.07027.

25 D. König. Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. *Mathematische Annalen*, 77:453–465, 1916. URL: http://eudml.org/doc/158740.

26 S Cliff Liu, Zhao Song, and Hengjie Zhang. Breaking the n-pass barrier: A streaming algorithm for maximum weight bipartite matching. *arXiv preprint*, 2020. arXiv:2009.06106.

27 Yang P. Liu and Aaron Sidford. *Faster Energy Maximization for Faster Maximum Flow*, pages 803–814. Association for Computing Machinery, New York, NY, USA, 2020. doi:10.1145/3357713.3384247.

28 Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. Improved distributed approximate matching. *J. ACM*, 62(5), November 2015. doi:10.1145/2786753.

29 Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 253–262, 2013. doi:10.1109/FOCS.2013.35.

30 Silvio Micali and Vijay V. Vazirani. An $o(\sqrt{|v|}\cdot|e|)$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*, pages 17–27, 1980. doi:10.1109/SFCS.1980.12.

31 M. Mucha and P. Sankowski. Maximum matchings via gaussian elimination. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 248–255, 2004. doi:10.1109/FOCS.2004.40.

32 Noam Nisan. The demand query model for bipartite matching. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 592–599. SIAM, 2021.

33 Daniel Stubbs and Virginia Vassilevska Williams. Metatheorems for dynamic weighted matching. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 58:1–58:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ITCS.2017.58.

34 Da Wei Zheng and Monika Henzinger. Multiplicative auction algorithm for approximate maximum weight bipartite matching. *arXiv preprint*, 2023. arXiv:2301.09217.