Access to Data and Number of Iterations: Dual Primal Algorithms for Maximum Matching under Resource Constraints

KOOK JIN AHN and SUDIPTO GUHA, University of Pennsylvania

17

In this article, we consider graph algorithms in models of computation where the space usage (random accessible storage, in addition to the read-only input) is sublinear in the number of edges m and the access to input is constrained. These questions arise in many natural settings, and in particular in the analysis of streaming algorithms, MapReduce or similar algorithms, or message passing distributed computing that model constrained parallelism with sublinear central processing.

We focus on weighted nonbipartite maximum matching in this article. For any constant $p > 1$, we provide an iterative sampling-based algorithm for computing a $(1 - \epsilon)$ -approximation of the weighted nonbipartite maximum matching that uses $O(p/\epsilon)$ rounds of sampling, and $O(n^{1+1/p})$ space. The results extend to b -Matching with small changes. This article combines adaptive sketching literature and fast primal-dual algorithms based on relaxed Dantzig-Wolfe decision procedures. Each round of sampling is implemented through linear sketches and can be executed in a single round of streaming or two rounds of MapReduce. The article also proves that nonstandard linear relaxations of a problem, in particular penalty-based formulations, are helpful in reducing the adaptive dependence of the iterations.

CCS Concepts: • **Theory of computation** → **MapReduce algorithms**; **Sketching and sampling**; **Graph algorithms analysis**; • **Mathematics of computing** → **Matchings and factors**;

Additional Key Words and Phrases: Maximum matching, primal dual algorithms

ACM Reference format:

Kook Jin Ahn and Sudipto Guha. 2018. Access to Data and Number of Iterations: Dual Primal Algorithms for Maximum Matching under Resource Constraints. *ACM Trans. Parallel Comput.* 4, 4, Article 17 (January 2018), 40 pages.

<https://doi.org/10.1145/3154855>

1 INTRODUCTION

Access to input is increasingly becoming an important and expensive resource in many computing environments that seek to solve large-scale optimization problems. These environments are increasingly distributed and iterative. The main objective in these environments is to use at most a small constant number of iterative steps/rounds and as small a state as possible between

This work was supported by NSF Awards 1117216, 1546151.

Authors' addresses: K. J. Ahn, Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043; email: kookjin@seas.upenn.edu; S. Guha, Penn Department of Computer and Information Science, 3330 Walnut Street, Philadelphia, PA 19104; email: sudipto@seas.upenn.edu.

Current address: K. J. Ahn is currently at kookjin@google.com. This work was done while the author was at the University of Pennsylvania.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2018 ACM 2329-4949/2018/01-ART17 \$15.00

<https://doi.org/10.1145/3154855>

subsequent iterations. The computation at a particular round depends on the computation in previous rounds and thus the algorithms are adaptive. The existence of such constant round iterative near-optimal algorithms is typically not known for many combinatorial optimization problems. We note that many of these distributed iterative computing environments, for example, streaming, MapReduce and its variants, and message passing distributed algorithms, differ considerably in their details. While most algorithms focus on specific models, larger abstractions that span multiple models are not readily available. In the context of this article, we use the fact that environments that are widely adopted typically allow sampling and computation of inner products with pseudorandom vectors (linear sketches). Furthermore, the considerations of computational environments under specific resource constraints such as communication, space, or access to data have not been widely addressed in modeling combinatorial optimization problems. We are often left with formulations that were designed for a different computational environment in mind.

Nonbipartite matching has long been a standard bearer of combinatorial optimization problems. In this article, we focus on weighted nonbipartite maximum b -matching problems and devise new relaxations that are useful for resource-constrained models. At a high level, our algorithm can be viewed as a recursive sampling algorithm. In each iteration, we redefine the weights of the datapoints using functions that are efficient to represent, communicate, and compute. On a more technical level, we use inner-product-based sampling so that we can reduce the adaptivity of the algorithm, and thereby the number of iterations in comparison to standard iterative approaches for solving these problems.

A sampling-based algorithm for finding the edges in a maximum matching is natural and intuitive: we iteratively sample edges; we show that the sampled edges contain a large matching or we refine the weights and repeat. While the high-level description is natural, the details of the analysis, for example, the demonstration of a large matching or refinements of weights, are nonobvious at best. In a recent result (Lattanzi et al. 2011), it was shown that for an n node m edge graph, we can find an $O(1)$ approximation for bipartite graphs using $O(p)$ rounds of edge filtering via map-reduce and $O(n^{1+1/p})$ centralized space (for any constant $p > 1$). However, it was also shown that the approximation bound achieved in the actual implementation was better than the worst case. This raises the natural question: is a $(1 - \epsilon)$ -approximation for any constant $\epsilon > 0$ achievable in a small number of rounds? In this article, we provide such an algorithm.

Note that although we discuss matchings, these questions are applicable to a broader set of problems; in many scenarios, iterative algorithms converge faster and better than deemed by the worst-case guarantees. Unfortunately, there are few systematic techniques that allow us to analyze iterative algorithms. One well-known example is linear (or convex) programming—and for maximum weighted matching there are LP relaxations that are exact. Therefore, any algorithm that provides a $(1 - \epsilon)$ approximation must also (possibly implicitly) provide a bound for the underlying LP, which is provided in Definition 1.

Definition 1. Let $\|U\|_b = \sum_{i \in U} b_i$ and $\mathcal{O} = \{U \mid \|U\|_b \text{ is odd}\}$. For standard matching all $b_i = 1$. Let $B = \sum_i b_i$. For a graph $G = (V, E)$ consider

$$\begin{aligned}
 \beta^* &= \max \sum_{(i,j) \in E} w_{ij} y_{ij} & (LP1) \\
 \sum_{j:(i,j) \in E} y_{ij} &\leq b_i & \forall i \in V \quad (\text{Vertex Constraints}) \\
 \sum_{(i,j) \in E: i,j \in U} y_{ij} &\leq \lfloor \|U\|_b / 2 \rfloor & \forall U \in \mathcal{O} \quad (\text{Odd Set Constraints}) \\
 y_{ij} &\geq 0 & \forall (i,j) \in E.
 \end{aligned}$$

The variables y_{ij} indicate the presence of the edge (i, j) in the matching. The bipartite case does not need the odd-set constraints. To achieve a $(1 - \epsilon)$ -approximation in the nonbipartite setting, the number of constraints in **LP1** can be reduced to $n^{O(1/\epsilon)}$ by considering $\mathcal{O}_s = \{U \in \mathcal{O} \mid \|U\|_b \leq 4/\epsilon\}$. However, $n^{O(1/\epsilon)}$ is still large.

There has been an enormous amount of research on solving LPs efficiently starting from Khachian's early result (Khachiyan 1978), for example, the multiplicative weight update framework (Littlestone and Warmuth (1994) and many others), positive linear programming (Luby and Nisan 1993), fractional packing and covering (Plotkin et al. (1995) and subsequent results), matrix games (Grigoriadis and Khachiyan 1995), and many similar descriptions that exist in different literature across different subfields (see the surveys Arora et al. (2012) and Foster and Vohra (1999)). None of the existing methods allow a constant number of iterations. Note that even though many frameworks such as Nemirovski (2005) and Nesterov (2005) highlight fast convergence, the description of these algorithms are over a unit ball. It is unclear how to efficiently represent nonbipartite matching on the unit ball while guaranteeing a $(1 - \epsilon)$ -approximation. Note that augmentation-path-based techniques for matching, which maintain a primal feasible solution, require either random access or many (superconstant) iterations.

Most of these methods described in Arora et al. (2012) and Foster and Vohra (1999) maintain multipliers, one for each constraint,¹ and seek to optimize a linear combination (using the respective multipliers) of the constraints. Therefore, multiple constraints are reduced to a single constraint. This is similar to a Dantzig-Wolfe-type decomposition, since the resulting object is typically a simpler problem. However, maintaining $n^{1/\epsilon}$ multipliers for the $n^{1/\epsilon}$ constraints is a challenging task—they can be maintained implicitly as was done in Ahn and Guha (2014), but the dependence on $\log n$ is somewhat fundamental for any black-box solver such as described in Arora et al. (2012) and Foster and Vohra (1999). Methods that only maintain dual multipliers typically require $\Omega(\rho\epsilon^{-2} \log M)$ iterations for M constraints, where ρ is the width parameter (determines how changes in the variables affect the constraints, defined more precisely in the sequel). In fact, $\Omega(\rho\epsilon^{-2} \log M)$ is a lower bound for random constraint matrices shown in Klein and Young (1999).

The aforementioned suggests that we should also consider the dual of the **LP1**, which is given in **LP2**: The variables x_i correspond to the vertex constraints and z_U correspond to the odd-set constraints in **LP1**. It may appear that Dantzig decompositions would help because we have fewer constraints—and that is correct. However, the width parameter of **LP2** is dependent on the weight of the edges.

$$\begin{aligned}
 \beta^* = \min \quad & \sum_i b_i x_i + \sum_{U \in \mathcal{O}} \lfloor \|U\|_b / 2 \rfloor z_U \\
 & x_i + x_j + \sum_{U \in \mathcal{O}; i, j \in U} z_U \geq w_{ij} \quad \forall (i, j) \in E \\
 & x_i, z_U \geq 0 \quad \forall i \in V, U \in \mathcal{O}
 \end{aligned} \tag{LP2}$$

Consider the following graph where all $b_i = 1$. It is clear that to get a $(1 - \epsilon)$ approximation, we must consider the odd set corresponding to the whole triangle; the bipartite relaxation has value $1 + 5\epsilon$, whereas the integral solution has value 1. An assignment of $z_U = 1$ to the entire graph using the relaxation **LP2** is a valid solution, but the width parameter becomes $O(1/\epsilon)$ based on the edge of weight 10ϵ .

¹From the perspective of space efficiency, methods such as Bienstock and Iyengar (2004), Nemirovski (2005), and Nesterov (2005), which maintain both primal and dual multipliers, are dominated by storing all the edges in the graph (same as number of primal variables).

In this simple example, we can set $z_U = 10\epsilon$ and $x_i = 1 - 10\epsilon$ for the vertex at the apex. It is unclear if such reassignments exist in a large graph. And even for unweighted graphs, note that $n^{1/\delta}$ different z_U affect a single edge and the width is large. Viewed another way, if a problem definition requires a constraint matrix of size $n^{1/\epsilon} \times m$ and we wish to solve that problem in $o(m, n^{1/\epsilon})$ space and constant number of rounds, then simple transformations will not suffice and newer ideas are necessary.

1.1 Models and Related Work

Matching has been well studied in the context of streaming algorithms. In such a setting, the input corresponds to a sequence of edges and associated weights (i, j, w_{ij}) , in arbitrary order, and the random access memory (RAM) is sublinear in the input size. The computation proceeds in passes. The algorithm is typically measured on how close is the output solution to the optimum solution (its approximation ratio), the amount of memory, and the number of passes—smaller is more desirable in all these measures. Observe that iterative sampling or sketching is easy to compute in the streaming model. For weighted nonbipartite matching, no previous result was known in the streaming model where the number of iterations is independent of the problem parameters. For unweighted cardinality matching, McGregor (2005) provided an algorithm with $2^{O(1/\epsilon)}$ iterations using $\tilde{O}(n)$ space (where we ignore polylogarithmic factors and polynomial dependence on $1/\epsilon$) in this model. The best-known single-round results that actually produce a matching are an $O(1)$ approximation using a single round (Eggert et al. 2009; Epstein et al. 2010; Feigenbaum et al. 2005; Zelke 2008), which applies to both bipartite and nonbipartite graphs. For bipartite graphs a $(1 - \epsilon)$ -approximation in $O(\epsilon^{-2} \log \frac{1}{\epsilon})$ rounds (Ahn and Guha 2013) and a $(1 - \epsilon)$ approximation in $O(\epsilon^{-2})$ rounds in a vertex arrival model (assuming order on the list of input edges) (Kapralov 2013) were known. In Guruswami and Onak (2013), it is shown that a space bound of $n^{1+\Omega(1/p)}/p^{O(1)}$ is necessary for a p round communication protocol to find the exact maximum in bipartite graphs.

Another well-studied model is the map-reduce model with some centralized storage. The computation proceeds by applying the map and reduce operators to the arbitrary order of the edges. The map operation is a uniform operation that is applied to all edges to partition them. The reduce operator applies to each partition and produces a small amount of information, which is sent back to the central coordinator (see Lattanzi et al. (2011) and references therein). Iterative sampling or sketching is not difficult to achieve in this model. The approximation for bipartite/nonbipartite graphs using $O(p)$ rounds of sampling and $\tilde{O}(n^{1+1/p})$ space in Lattanzi et al. (2011) corresponds to iterative sampling (and filtering). For bipartite graphs, $(1 - \epsilon)$ approximation is achievable using several algorithms (see Bahmani et al. (2014)), but they require $\Omega(\log n)$ rounds.

Matching has also been studied in the context of distributed computing (see Mansour and Vardi (2013) for a survey of results). The congested clique model (Drucker et al. 2014) is of interest to this article; in that model, n participants have local (possibly shared inputs) and any node can send a bounded-length message to any other node in a round.

1.2 Results, Main Ideas, and Roadmap

The results in this article are as follows: (i) a result on space efficient computation of matchings (ii) a refinement of the notion of adaptivity into “adaptivity at sampling” and “adaptivity at use,”

(iii) new relaxations for matchings, and (iv) a dual-primal framework. The framework incorporates (ii) and produces a primal feasible optimal solution even though it attempts to solve the dual. We now discuss these contributions and the related work.

Results on Small-Space Matchings: The main algorithmic result we prove herein is:

THEOREM 1.1. *For any constant $0 < \epsilon \leq \frac{1}{16}$ and $p > 1$, we provide a $(1 - \epsilon)$ approximation scheme for the weighted nonbipartite matching problem using $O(p/\epsilon)$ rounds of adaptive sketching/sampling that can be implemented in MapReduce and $O(n^{1+1/p})$ centralized space. The space requirement increases to $O(n^{1+1/p} \log B)$ if $B = \sum_i b_i$ is super polynomial in n . The running time is $O(m \text{ poly}(\epsilon^{-1}, \log n) \log B)$.*

The algorithm is sampling based as promised and is described at a high level in Algorithm 1. The algorithm maintains a certificate of optimality and therefore if we stumble upon a good matching, we will terminate early. We provide a warmup result in Section 4 for bipartite matching, which uses $O(1/\epsilon)$ rounds and can be sublinear in the number of edges (e.g., when $m > n^{1.1}$), which was also not known heretofore. In the algorithm provided in this article, the probability of sampling each edge (i, j) depends on (along with i and j) the z_U values of different odd sets U of size at most $1/\epsilon$ (and containing both i, j). The number of such odd sets with $z_U > 0$ is at most $O(\epsilon^{-5} (\log B) (\log^2 n) \log^2 \frac{1}{\epsilon})$.

ALGORITHM 1: An Algorithm for Maximum Matching

- 1: Start with an initial sampling distribution over the edges.
 - 2: **while** we do not have a certificate of $(1 - \epsilon)$ approximation **do**
 - 3: Sample $O(n^{1+1/p})$ edges, subdivided into $t = O(\frac{1}{p\epsilon} \log n)$ independent parts, say, $S_1, S_2, \dots, S_t$.
 - 4: Use S_1 to simulate a step of solving a dual LP of matching. Then use S_1 to refine or adjust S_2 and then use S_2 . In general, use $S_1, \dots, S_q$ to refine S_{q+1} and use S_{q+1} .
 - 5: We prove that either we succeed in the refinement and use of S_{q+1} or produce an explicit dual (of the dual) certificate—which is a large explicit primal solution, in this case the desired matching.
 - 6: **end while**
-

The full $O(n^{1+1/p})$ space is not needed to define the value of the multiplier for an edge, and this fact is useful in distributed settings. The linear sketches can be viewed as requiring each vertex to sketch its neighborhood $n^{1/p}$ times. Theorem 1.1 shows that if we view the nodes in the graph as players and each player sees only the edges adjacent to it, then in the congested clique model, we can compute a $(1 - \epsilon)$ approximation for the maximum weighted nonbipartite b -matching problem using $O(p/\epsilon)$ rounds and $O(n^{1/p})$ size messages per vertex. Note that the communication graph and the input graph are different in this model.

Adaptivity at Sampling Versus Adaptivity at Use: An iterative algorithm, where the computation in iteration j depends on the result of iteration $j - 1$, is defined as an adaptive algorithm. The degree of adaptivity corresponds to the longest chain of dependencies introduced between the iterations; often this is the same as the number of rounds. For primal-dual algorithms that approximately solve LPs, the computation is adaptive and requires $\Omega(\log n)$ rounds that define a linear chain.

In contrast, the while loop in Algorithm 1 is executed for at most $O(p/\epsilon)$ steps. Given the subdivision of each sample into $O(\frac{1}{p\epsilon} \log n)$ parts, we have an algorithm that uses $O(\epsilon^{-2} \log n)$ iterations, but the number of rounds of sampling is only $O(p/\epsilon)$. The adaptivity of the sampling process is different from the adaptive behavior of the primal-dual process; in the latter, the computation in an iteration strongly depends on the computation in the previous iteration. However, the

Fig. 1. Adaptivity at sampling versus adaptivity at use.

sampling process is the process that accesses data and therefore the number of rounds of data access is $O(p/\epsilon)$. This differentiation is key to this article and is likely to be used in many other settings. One such setting, albeit in retrospect, is the dynamic streaming connectivity algorithm in Ahn et al. (2012a, 2012b), where the linear sketches² were computed in parallel in one round *but used sequentially* in $O(\log n)$ steps of postprocessing to produce a spanning tree. In this article, we show that $S_1, \dots, S_t$ are relatives of cut-sparsifiers. In particular, we define:

Definition 2 (The Deferred Cut-Sparsifier Problem). We are given a weighted graph $G = (V, E, u)$, but the weights are **not revealed** to us. Instead, we are given $\{\zeta_{ij}\}$ with the promise that $\zeta_{ij}/\chi \leq u_{ij} \leq \zeta_{ij}\chi$. We have to produce a smaller summary data structure $\mathcal{D}$ storing only some of the edges. Once $\mathcal{D}$ is constructed, **then** the exact weights u_{ij} of only those edges stored in our structure $\mathcal{D}$ are revealed to us. We output a sparsifier $H = (V, E', u^s)$.

Cut-sparsifiers, introduced in Benczúr and Karger (1996) are combinatorial objects that preserve every cut to within $1 \pm \epsilon$ factor. Cut sparsifiers are relevant in the context of matching because if we focus on the odd set constraints of LP1, we observe

$$\sum_{(i,j) \in E: i \in U, j \notin U} y_{ij} \leq \lfloor \|U\|_b/2 \rfloor \iff \frac{1}{2} \sum_{i \in U} \left(\sum_{(i,j) \in E} y_{ij} \right) - \left(\sum_{(i,j) \in E: i \in U, j \notin U} y_{ij} \right) \leq \lfloor \|U\|_b/2 \rfloor,$$

where the last inequality is a linear combination of cuts. However, at the same time, the use of sparsifier is nontrivial because no sparsifier can preserve differences of cuts approximately (since that would answer the sign of the difference exactly). Naive use of sparsifiers does not help for matchings; the largest matching in a graph has no connection to large matchings in the sparsifier of that graph.

As mentioned in Algorithm 1, we sparsify the dual weights of (a suitably modified) dual of matching problem LP2 (the dual of the dual is intentional). The intuition behind the two notions of “adaptivity at sampling” and “adaptivity at use” is best illustrated by the schematic in Figure 1. Suppose that the true dual weights at time t are $\mathbf{u}(t)$; then the primal-dual algorithm can be thought of as progressing through the sequence $\mathbf{u}(1), \tilde{\mathbf{x}}(1), \mathbf{u}(2), \tilde{\mathbf{x}}(2), \mathbf{u}(3), \tilde{\mathbf{x}}(3), \dots$, as shown in the left part of Figure 1 (without the step labeled sketch). The Oracle corresponds to the mechanism that provides a (an infeasible) primal solution $\tilde{\mathbf{x}}(i)$ that updates the dual weights to $\mathbf{u}(i)$ in the primal-dual algorithm. If we prove that $\mathbf{u}$ can be sparsified but do not prove any further properties, the overall process remains as adaptive as before; that is, $\mathbf{u}(1) \rightarrow \mathbf{u}(2) \rightarrow \mathbf{u}(3)$ changes to $\mathbf{u}(1) \rightarrow \mathbf{u}^s(1) \rightarrow \mathbf{u}(2) \rightarrow \mathbf{u}^s(2) \rightarrow \mathbf{u}(3)$, which are identical from the perspective of the number of adaptive steps required to compute $\mathbf{u}(t)$ or $\mathbf{u}^s(t)$. This is shown in the left part of the Figure 1.

²In this case, the input is an oriented vertex-edge adjacency matrix. The sketch is computed first, and subsequently an adversary provides a cut. We then sample an edge across that cut (if one exists, or determine that no such edge exists) with high probability.

Deferred sparsifiers constructed via sampling allow us to bypass that dependency chain mentioned earlier. The sparsifier construction is broken into two parts—the first part of the construction can be made nonadaptive and in parallel. Then the second part can be performed sequentially, but over a small subset of edges stored in memory. We construct $\zeta(1), \dots, \zeta(t)$ in parallel, which eventually gives us the deferred sparsifiers $\mathcal{D}(1), \dots, \mathcal{D}(t)$. We derive the actual sparsifier $\mathbf{u}^s(1)$ and the update $\tilde{\mathbf{x}}(1)$. Instead of explicitly computing $\mathbf{u}(2)$, we refine the weights of $\mathcal{D}(2)$ and produce $\mathbf{u}^s(2)$ directly. Therefore, we sequentially compute $\mathbf{u}^s(1) \rightarrow \mathbf{u}^s(2) \rightarrow \dots \rightarrow \mathbf{u}^s(t)$ and are able to make t simultaneous steps without further access to data. This is shown in the right part of Figure 1. However, such a reduction of adaptivity is only feasible for specific linear programming relaxations. We should choose that relaxation with care and we discuss one such novel relaxation for matching.

New Relaxations for Nonbipartite (and Bipartite) Matchings: Assume that the edge weights are at least 1 and rounded to integral powers of $(1 + \epsilon)$. This can be achieved without much difficulty in all the computation models considered. Let $\hat{w}_k = (1 + \epsilon)^k$ and $\hat{E}_k$ be the set of edges (i, j) with that weight. Then the (dual of the) maximum b -matching is given by LP3. Observe that LP3 is very similar to LP4, where we are considering a “layered” variant— $x_{i(k)}$ corresponds to the cost of vertex i in level k . $x_i = \max_k x_{i(k)}$ is the contribution of vertex i to the objective. However, the cost of each set U in level ℓ is $z_{U,\ell}$ and the contribution of a set U is *additive*! An edge (i, j) is covered from the cost of the two vertices i, j (specifically their cost in level k) or the sum of the costs of all U (odd, containing both i, j) of all layers *below or equal to* k .

$$\begin{aligned}
 \min \quad & \sum_i b_i x_i + \sum_{U \in \mathcal{O}} \left\lfloor \frac{\|U\|_b}{2} \right\rfloor \sum_{\ell} z_{U,\ell} & (\text{LP3}) \\
 & x_{i(k)} + x_{j(k)} + \sum_{\ell \leq k} \left(\sum_{U \in \mathcal{O}; i, j \in U} z_{U,\ell} \right) \geq \hat{w}_k & \forall (i, j) \in E_k \\
 & 2x_{i(k)} + \sum_{\ell \leq k} \left(\sum_{U \in \mathcal{O}; i \in U} z_{U,\ell} \right) \leq 3\hat{w}_k & \forall i, k \\
 & x_i - x_{i(k)} \geq 0 & \forall i, k \\
 & x_{i(k)}, x_i, z_{U,\ell} \geq 0 & \forall i, k, U, \ell
 \end{aligned}$$

THEOREM 1.2 [PROVED IN SECTION 2]. For $0 < \epsilon \leq 1/16$, let the optimum solution of LP3 when we only use the variables corresponding to $U \in \mathcal{O}_s$ be $\tilde{\beta}$, and then $\beta^* \leq \tilde{\beta} \leq (1 + \epsilon)\beta^*$.

We only focus on $\mathcal{O}_s$ in this article, but note that including the other variables can decrease $\tilde{\beta}$ but it will always be above β^* . It is interesting that vertices and odd-sets are treated differently in LP3—LP3 is likely to be of interest independently of resource constraints.

$$\begin{aligned}
 \min \quad & \sum_i b_i x_i + \sum_{U \in \mathcal{O}} \left\lfloor \|U\|_b / 2 \right\rfloor z_U & (\text{LP4}) \\
 & x_i + x_j + \sum_{U \in \mathcal{O}; i, j \in U} z_U \geq 1 & \forall (i, j) \in E \\
 & 2x_i + \sum_{U: i \in U} z_U \leq 3 & \forall i \in V \\
 & x_i, z_U \geq 0 & \forall i \in V, U \in \mathcal{O}
 \end{aligned}
 \quad \left| \quad
 \begin{aligned}
 \max \quad & \sum_{(i,j) \in E} y_{ij} - 3 \sum_i \mu_i & (\text{LP5}) \\
 & \sum_{j: (i,j) \in E} y_{ij} - 2\mu_i \leq b_i & \forall i \in V \\
 & \sum_{(i,j) \in E; i, j \in U} y_{ij} - \sum_{i \in U} \mu_i \leq \left\lfloor \frac{\|U\|_b}{2} \right\rfloor & \forall U \in \mathcal{O} \\
 & y_{ij}, \mu_i \geq 0 & \forall i \in V, \forall (i, j) \in E
 \end{aligned}$$

The intuition behind the new relaxation is already illustrated when all edge weights are 1 and we can drop the index ℓ for simplicity. The relaxation simplified relaxation is LP4 and its dual is LP5. The formulation LP5 allows each vertex to be fractionally matched to $b_i + 2\mu_i$ edges, yet the

overall objective is charged for this flexibility—this is a classic penalty-based formulation. It can be shown (and we do, for the general weighted case, through ideas based on the proof of total dual integrality) that the objective function has not increased from LP1 (for $w_{ij} = 1$). Moreover, in LP4, subject to $2x_i + \sum_{U:i \in U} z_U \leq 3$ and nonnegativity,

$$x_i + x_j + \sum_{U \in \mathcal{O}; i, j \in U} z_U \leq 6, \quad (1)$$

or in other words, the right-hand side of the equation is now independent of any problem parameters! This corresponds to showing bounded width and helps us reduce the number of iterations. In other words, if we wish to solve the dual using a primal-dual formulation, we can encode the width parameter as a constraint such as Equation (1). Since we are adding constraints to the dual, we have a penalty-based formulation for the original primal problem. Such penalty-based formulations allow flexibility in how we get to a near-optimum solution (the trajectory in a primal-dual algorithm) and are likely to be useful in other problems.

A Dual-Primal Framework: With the insight of width constraints in dual corresponding to penalty-based primals, we proceed to provide a black-box framework that will use all the ideas we have discussed thus far, namely, (a) deferred computation, (b) starting from a dual, and (c) expressing width as constraints. Suppose that we wish to solve PRIMAL,

$$\beta^* = \max \mathbf{c}^T \mathbf{y}; \mathbf{y} \in \{\mathcal{A}^T \mathbf{y} \leq \mathbf{b}, \mathbf{y} \geq \mathbf{0}\}. \quad (\text{PRIMAL})$$

Suppose that we wish to apply the primal-dual method to a modification of the dual $\mathbf{A}\mathbf{x} \leq \mathbf{c}$, where $\mathbf{Q}$ is an auxiliary polytope. Then a feasible solution to $\{\mathbf{b}^T \mathbf{x} \leq \beta, \mathbf{A}\mathbf{x} \geq (1 - 3\epsilon)\mathbf{c}, \mathbf{x} \in \mathbf{Q}, \mathbf{x} \geq \mathbf{0}\}$ implies $\beta^* \leq \beta / (1 - a_1\epsilon)$. Intuitively, if we have a current primal solution to PRIMAL of value β and we show that the dual has a small solution (compared to β), then we have an optimal solution. If we fail to show that the dual has a small solution, then that proof must (at least implicitly) provide a certificate that we have a larger solution to PRIMAL. Suppose that we can extract an explicit certificate—multipliers for the constraints in the dual can be interpreted as candidate primal solutions! We use this connection. If we keep trying to prove the dual is small and keep failing, then the best explicit certificate found so far will keep increasing the value β and eventually we will succeed in proving that the dual (and therefore also the primal) is near optimal. Note that we solved the primal by focusing on the dual $\mathbf{A}\mathbf{x} \leq \mathbf{c}$. This is the starting point of the framework.

Suppose that we have width constraints given by matrix $\mathbf{P}_o$, where ρ_o is a constant > 1 .

$$\{\mathbf{P}_o \mathbf{x} \leq 2\mathbf{q}_o, \mathbf{x} \in \mathbf{Q}, \mathbf{x} \geq \mathbf{0}\} \quad \Rightarrow \quad \mathbf{A}\mathbf{x} \leq \rho_o \mathbf{c}$$

The primal-dual framework maintains weights $\mathbf{u}$ and requires an Oracle to provides us $\mathbf{x}$,

$$\mathbf{u}^T \mathbf{A}\mathbf{x} \geq (1 - \epsilon/2) \mathbf{u}^T \mathbf{c} \quad \text{s.t.} \quad \mathbf{x} \in \{\mathbf{P}_o \mathbf{x} \leq 2\mathbf{q}_o, \mathbf{x} \in \mathbf{Q}, \mathbf{x} \geq \mathbf{0}\}, \quad (\text{OUTER})$$

intuitively; the framework is describing the fact that approximately satisfying a collection of constraints is as easy as approximately satisfying the linear combination. The question that arises is: *How can we solve this linear combination without complete access to A?*

The answer, of course, is sparsification! If instead of $\mathbf{u}$ we could use a sparsified set of multipliers $\mathbf{u}^s$, then we would only need to remember a few rows of $\mathbf{A}$ at a time. In the context of matching, $\mathbf{A}$ is the dual with m constraints (and many variables) and the sparsification reduces the number of constraints to $\tilde{O}(n)$. Since the multipliers $\mathbf{u}$ do not usually change quickly in a primal-dual framework, we actually have a promise problem and can use *deferred sparsifiers* as discussed earlier. Deferred sparsification now reduces the number of adaptive accesses to $\mathbf{A}$, but each time we construct several independent deferred sparsifiers in parallel. Note that the explicit certificate we extract corresponds to nonzero values in the sparsifier, and therefore we extract a sparse solution!

Finally, the sparsifiers provide one other benefit. Instead of solving OUTER, we can now apply recursion and solve OUTER approximately by an inner primal-dual algorithm. The difficult part, however, is to ensure that when these inner iterations fail, we have to provide an infeasibility certificate of $\{\mathbf{b}^T \mathbf{x} \leq \beta, \mathbf{A}\mathbf{x} \leq \mathbf{c}\}$ directly even though we only have access to the rows of $\mathbf{A}$ corresponding to $\mathbf{u}^s$. Given the number of different pieces that need to be addressed simultaneously, we abstract the framework and a corresponding theorem.

Definition 3. The problem $\beta^* = \max \mathbf{c}^T \mathbf{y}; \mathbf{y} \in \{\mathcal{A}^T \mathbf{y} \leq \mathbf{b}, \mathbf{y} \geq \mathbf{0}\}$ is defined to be (ρ_0, ρ_i) -**Dual-Primal amenable** if there exist $\mathbf{A} \in \mathbb{R}_+^{m \times N}$, $\mathbf{b}, \mathbf{x} \in \mathbb{R}_+^N$, $\mathbf{P}_o \in \mathbb{R}_+^{\tilde{n}_o \times N}$, $\mathbf{q}_o \in \mathbb{R}_+^{\tilde{n}_o}$, $\mathbf{P}_i \in \mathbb{R}_+^{\tilde{n}_i \times N}$, $\mathbf{q}_i \in \mathbb{R}_+^{\tilde{n}_i}$, convex polytope $\mathbf{Q}$ with $\mathbf{0} \in \mathbf{Q}$, and an absolute constant a_1 such that the following hold simultaneously:

- (d1) (DUAL FEASIBILITY.) A feasible solution to $\{\mathbf{b}^T \mathbf{x} \leq \beta, \mathbf{A}\mathbf{x} \geq (1 - 3\epsilon)\mathbf{c}, \mathbf{x} \in \mathbf{Q}, \mathbf{x} \geq \mathbf{0}\}$ implies $\beta^* \leq \beta/(1 - a_1\epsilon)$.
- (d2) (OUTER WIDTH.) $\{\mathbf{P}_o \mathbf{x} \leq 2\mathbf{q}_o, \mathbf{x} \in \mathbf{Q}, \mathbf{x} \geq \mathbf{0}\}$ implies $\mathbf{A}\mathbf{x} \leq \rho_o \mathbf{c}$.
- (d3) (INNER WIDTH.) $\{\mathbf{P}_i \mathbf{x} \leq \mathbf{q}_i, \mathbf{x} \in \mathbf{Q}, \mathbf{x} \geq \mathbf{0}\}$ implies $\mathbf{P}_o \mathbf{x} \leq \rho_i \mathbf{q}_o$.
- (d4) (A DEFERRED $\mathbf{u}$ -SPARSIFIER.) Given $\mathbf{u}, \mathbf{v} \in \mathbb{R}_+^m$, and the promise that all nonzero $\mathbf{u}_\ell$ satisfy $1/\mathcal{L}_0 \leq \mathbf{v}_\ell/\gamma \leq \mathbf{u}_\ell \leq \gamma \mathbf{v}_\ell \leq \mathcal{L}_0$ for each $\ell \in [m]$ given fixed $\gamma, \mathcal{L}_0 \geq 1$, we can construct a data structure $\mathcal{D}$ that samples a subset of indices in $[m]$ of size $\tilde{n}_s$ based on the $\mathbf{v}$ values, and stores the indices. After $\mathcal{D}$ has been constructed, the exact values of those stored entries of $\mathbf{u}$ are revealed and the data structure constructs a nonnegative vector $\mathbf{u}^s$ such that for some property $\mathbb{G}(\mathbf{u}^s, \mathbf{x})$, which is convex in $\mathbf{x}$ and $\mathbf{x} = \mathbf{0}$ satisfies $\mathbb{G}(\mathbf{u}^s, \mathbf{x})$, we have

$$(\mathbf{u}^s)^T \mathbf{A}\mathbf{x} \geq \left(1 - \frac{\epsilon}{8}\right) (\mathbf{u}^s)^T \mathbf{c}, \quad \mathbb{G}(\mathbf{u}^s, \mathbf{x}), \mathbf{x} \geq \mathbf{0} \Rightarrow \mathbf{u}^T \mathbf{A}\mathbf{x} \geq \left(1 - \frac{\epsilon}{2}\right) \mathbf{u}^T \mathbf{c}. \quad (\text{SWITCH})$$

- (d5) (INITIAL SOLUTION.) We can efficiently find a solution to INITIAL for some $a_\epsilon \geq 2$:

$$\begin{cases} \mathbf{A}\mathbf{x}_0 \geq (1 - \epsilon_0)\mathbf{c} \\ \frac{\beta^*}{a_\epsilon} \leq \beta_0 = \mathbf{b}^T \mathbf{x}_0 < \frac{\beta^*}{2} \\ \mathbf{P}_o \mathbf{x}_0 \leq 2\mathbf{q}_o \\ \mathbf{P}_i \mathbf{x}_0 \leq \mathbf{q}_i \\ \mathbf{x}_0 \in \mathbf{Q}, \mathbf{x}_0 \geq \mathbf{0}. \end{cases} \quad (\text{INITIAL})$$

Observe that we can set $\rho_o = \rho_i = \infty$, but unbounded values of ρ_o, ρ_i will not have any algorithmic consequence. For bounded ρ_o, ρ_i , the matrices $\mathbf{A}$ and $\mathcal{A}$ of course have to be related—but that relationship is not necessarily a simple representation (such as column sampling). The main consequence of (ρ_o, ρ_i) -dual-primal amenability is as follows:

THEOREM 1.3. Suppose for a (ρ_o, ρ_i) -dual-primal amenable system PRIMAL there exists a MICROORACLE, which, given an absolute constant $a_2 > 0, \epsilon$ such that $0 < \epsilon \leq 1, \mathbf{u}^s \in \mathbb{R}_+^m, \zeta \in \mathbb{R}_+^{\tilde{n}_o}$, and $\varrho > 0$, provides either:

- (i) a feasible solution for PRIMAL with $\mathbf{c}^T \mathbf{y} \geq (1 - a_2\epsilon)\beta$ such that for all $\ell \in [m], \mathbf{y}_\ell > 0$ implies $\mathbf{u}_\ell^s > 0$, that is, a solution only involving the primal variables corresponding to the sampled constraints in the deferred sparsifier, or
- (ii) a solution $\mathbf{x}$ of LAGINNER, where the number of nonzero entries of $\mathbf{x}$ is at most $\tilde{n}_x$.

$$\begin{aligned} (\mathbf{u}^s)^T \mathbf{A}\mathbf{x} - \varrho \zeta^T \mathbf{P}_o \mathbf{x} &\geq \left(1 - \frac{1}{16}\right) (\mathbf{u}^s)^T \mathbf{c} - \varrho \zeta^T \mathbf{q}_o \\ \mathbb{G}(\mathbf{u}^s, \mathbf{x}) & \\ \tilde{\mathbf{Q}}(\beta) = \{\mathbf{b}^T \mathbf{x} \leq \beta, \mathbf{P}_i \mathbf{x} \leq \mathbf{q}_i, \mathbf{x} \in \mathbf{Q}, \mathbf{x} \geq \mathbf{0}\} & \end{aligned} \quad (\text{LAGINNER})$$

Then we can find a $(1 - (1 + a_1 + a_2)\epsilon)$ -approximate solution to PRIMAL using τ rounds of deferred $\mathbf{u}$ -sparsifier construction, $\tau_o = O(\rho_o(\frac{1}{\epsilon} + \epsilon \log \frac{1}{1-\epsilon_0})^{\frac{\log(m/\epsilon)}{\log \gamma}} + \frac{\log a_\epsilon}{\log \gamma})$. In each round, we construct $O(\epsilon^{-1} \log \gamma)$ deferred $\mathbf{u}$ -sparsifiers.

For a fixed deferred sparsifier, MICROORACLE is invoked at most $\tau_i = O(\rho_i(\log \rho_i)(\log \tilde{n}_o) \log \frac{1}{\epsilon})$ times and we use a simple exponential of a linear combination of the returned solutions (of part (ii)) to define the weights $\mathbf{u}$. See Algorithm 2. Each nonzero $\mathbf{u}_\ell \in [(2m/\epsilon)^{-O(\frac{\rho_o}{\epsilon(1-\epsilon_0)})}, 1]$ and $\log \mathcal{L}_0 = O(\frac{\rho_o}{\epsilon(1-\epsilon_0)} \log(m/\epsilon) + \log \gamma)$. Moreover, the algorithm computes (approximately) $\max \lambda; \mathbf{A}\tilde{\mathbf{x}} \geq \lambda \mathbf{c}$ for a $\tilde{\mathbf{x}}$, which is a weighted average of the $\mathbf{x}$ returned by successive applications of part (ii)—the algorithm can stop earlier than the stated number of τ rounds if it observes $\lambda \geq (1 - 3\epsilon)$.

ALGORITHM 2: An Algorithm for Theorem 1.3 (Compare Algorithm 1)

- 1: Start with $\mathbf{x} = \mathbf{x}_0$ and $\beta = \beta_0$, where $\mathbf{x}_0, \beta_0$ refer to the initial solution.
- 2: Consider the following **family** of decision problems:

$$\begin{aligned} \{\mathbf{A}\mathbf{x} \geq \mathbf{c}, \mathbf{x} \in \mathcal{P}(\beta)\} \quad \text{where} \\ \mathcal{P}(\beta) = \begin{cases} \mathbf{P}_0 \mathbf{x} \leq 2\mathbf{q}_0 \\ \tilde{\mathbf{Q}}(\beta) = \{\mathbf{b}^T \mathbf{x} \leq \beta; \mathbf{P}_i \mathbf{x} \leq \mathbf{q}_i; \mathbf{x} \in \mathbf{Q}; \mathbf{x} \geq \mathbf{0}\} \end{cases} \end{aligned} \quad (\text{DUAL})$$

- 3: **while** $\lambda < 1 - 3\epsilon$ (where $\lambda = \min_\ell (\mathbf{A}\mathbf{x})_\ell / \mathbf{c}_\ell$) **do**
 - 4: Define exponential weights $\mathbf{u}$ (Section 6.1). Compute $\frac{\ln \gamma}{\epsilon}$ deferred $\mathbf{u}$ -sparsifiers denoted by $\{\mathcal{D}_q\}$.
 - 5: Consider the union of the constraints sampled in the previous step and compute a $(1 - a_3\epsilon)$ approximation to PRIMAL restricted to these constraints. Say that value is β' .
 - 6: If $\beta' > \beta(1 - a_3\epsilon)/(1 + \epsilon)$, remember the new solution and set $\beta = \beta'(1 + \epsilon)/(1 - a_3\epsilon)$.
 - 7: Update $\mathbf{x}, \mathbf{u}(q)$ etc. as required by the proof of Theorem 1.3; we are guaranteed to not invoke condition (i) of MICROORACLE due to Step (5). This provides a new sampling weight function for Step (4).
 - 8: **end while**
 - 9: Output the $\mathbf{y}$ corresponding to largest $\beta' \geq (1 - a_3\epsilon)\beta/(1 + \epsilon)$. We prove that such a β' exists.
-

The running time of Algorithm 1 for Theorem 1.3 is dominated by the time to construct the deferred sparsifiers (which includes the evaluation of $\mathbf{u}$) plus the invocations of MICROORACLE. The space used to maintain the current average $\mathbf{x}$ is $O(\tilde{n}_x \tau_o \tau_i \frac{1}{\epsilon} \log \gamma)$. The space to store a single deferred sparsifier is typically $O(n\gamma^2 \text{poly}(\epsilon^{-1}, \log n, \log \gamma))$ (this is application specific and depends on $\mathbf{u}, \mathcal{L}_0$). Therefore, if ρ_o, ρ_i are $\text{poly}(\epsilon^{-1})$, $\gamma = n^{1/(2p)}$, and $\tilde{n}_x = O(n\epsilon^{-1} \log n)$, the overall asymptotic space complexity will typically be at most $n^{1+1/p}$ (absorbing the $\text{poly}(\epsilon^{-1}, \log n)$ terms), which is $o(m)$ if $m \gg n^{1+1/p}$. For b -matching, $\tilde{n}_x = O(n\epsilon^{-1} \log B)$, where $B = \sum_i b_i$.

Roadmap: We relegate the proof of Theorem 1.3 to Section 6. We provide a proof of the main structural characterization (Theorem 1.2) in Section 2 next. We discuss the construction of deferred cut-sparsifiers in Section 3, specially in resource-constrained models. We then show how these definitions and theorems are applied to bipartite matching in Section 4 as a warmup, which also handles the initial solution. The main theorem for weighted nonbipartite b -matching, Theorem 1.1, is proved in Section 5. The bipartite result demonstrates how the edge weights can be handled, and therefore the nonbipartite result in Section 5 starts from that point and focuses on the odd set constraints.

2 NEW RELAXATIONS FOR MATCHINGS

In this section, we prove a new relaxation for maximum matching in nonbipartite graphs, as provided by Theorem 1.2. The theorem is an approximate characterization of the dual of the b -matching polytope and an essential part of the results for nonbipartite graphs in this article.

THEOREM 1.2. *For $0 < \epsilon \leq \frac{1}{16}$ suppose we have $G = (V, \hat{E})$ such that the weight $\hat{w}_{ij}$ of every edge $(i, j) \in \hat{E}$ is of the form $\hat{w}_k = (1 + \epsilon)^k$ for $k \geq 0$, and thus $\hat{E}_k = \{(i, j) | \hat{w}_{ij} = \hat{w}_k\}$ and $\hat{E} = \cup_k \hat{E}_k$. Let $\hat{w}_L$ be the largest weight. Then $\hat{\beta} \leq \tilde{\beta} \leq (1 + \epsilon)\hat{\beta}$, where $\tilde{\beta}, \hat{\beta}$ are defined by [LP6](#) and [LP7](#), respectively.*

ACM Transactions on Parallel Computing, Vol. 4, No. 4, Article 17. Publication date: January 2018.

ALGORITHM 3: Producing a Feasible Solution for LP6

-
- 1: (REMOVE LARGE SETS) For every $U \notin \mathcal{O}_s$, for every $i \in U$ we set $x_i^* \leftarrow \min\{x_i^* + z_U^*/2, \hat{w}_L\}$ and $z_U^* \leftarrow 0$. This preserves the feasibility of LP7 and increases the objective by at most $(1 + \epsilon)$. Let this transformed set of variables be $\hat{x}_i, \hat{z}_U$. Recall that the largest-weight class is $\hat{w}_L$.
 - 2: (PRODUCE $x'_{i(k)}$ VALUES) Set $x_i = \hat{x}_i$. For each k , if $x_i > \hat{w}_k$, set $x_{i(k)} = \hat{w}_k$; otherwise (for $x_i \leq \hat{w}_k$), set $x_{i(k)} = x_i$. For **bipartite** graphs the algorithm skips to Step (17).
 - 3: (ORDER SETS) Order the U with $\hat{z}_U > 0$ in **decreasing** order of $\|U\|_b$ —note that this is a laminar family. We will assign the $z_{U,\ell}$ corresponding to this order—and we will assign all $z_{U,\ell}$ before proceeding to the next set U' in this order.
 - 4: Initialize all $z_{U,\ell} = 0$. Initialize $\text{Sat}(i) = -1$ for all $i \in V$.
 A value of $\text{Sat}(i) \geq 0$ corresponds to the largest $k \geq 0$ such that $\sum_{\ell \leq k} (\sum_{U:i \in U} z'_{U,\ell}) \geq \hat{w}_k$. Notice that when we consider U according to Step (3), $\sum_{\ell \leq k} (\sum_{U':i \in U'} z_{U',\ell})$ is the same for all $i \in U$ and the $\text{Sat}(i)$ values are the same for all $i \in U'$. Define that value as $\text{Sat}(U)$.
 - 5: **for** the next set U according to the order in Step (3) **do**
 - 6: $k = \text{Sat}(U) + 1$.
 - 7: **while** $(k \leq L)$ and $(\hat{z}_U > 0)$ **do**
 - 8: **if** $\hat{z}_U + \sum_{\ell \leq k} (\sum_{U:i \in U} z_{U,\ell}) \geq \hat{w}_k$ **then**
 - 9: Set $z'_{U,k} \leftarrow \hat{w}_k - \sum_{\ell \leq k} (\sum_{U:i \in U} z_{U,\ell})$.
 - 10: $\hat{z}_U \leftarrow \hat{z}_U - z_{U,k}$.
 - 11: $\text{Sat}(i) \leftarrow k$ for all $i \in U$ and $k \leftarrow k + 1$.
 - 12: **else**
 - 13: Set $z_{U,k} = \hat{z}_U, \hat{z}_U \leftarrow 0$.
 - 14: **end if**
 - 15: **end while**
 - 16: **end for**
 - 17: Return $\{x_{i(\ell)}\}, \{x_i\}, \{z_{U,\ell}\}$.
-

for $i \in A \cap B$ by z . These changes preserve the feasibility and the objective value of the solution. On the other hand, they decrease $\sum_{z_U \neq 0} z_U \|U\|_b$ because we replace A and B by $A - B$ and $B - A$ (obviously, $\|A - B\|_b < \|A\|_b$ and $\|B - A\|_b < \|B\|_b$). Therefore, it contradicts the fact that the chosen solution belongs to $\mathcal{S}_2$.

- (2) $\|A \cap B\|_b$ **is odd**: Let $z = \min\{z_A, z_B\}$. We reduce z_A and z_B by z and increase $z_{A \cup B}, z_{A \cap B}$ by z . Again, these changes preserve the feasibility and the objective value of the solution. Since $\|A \cup B\|_b + \|A \cap B\|_b = \|A\|_b + \|B\|_b$ and $\|A \cup B\|_b > \|A\|_b, \|B\|_b$, $\|A \cup B\|_b^2 + \|A \cap B\|_b^2 \geq \|A\|_b^2 + \|B\|_b^2$. So $\sum_{z_U \neq 0} z_U \|U\|_b^2$ increases, which contradicts the fact that the solution maximizes $\sum_{z_U \neq 0} z_U \|U\|_b^2$.

Therefore, L is a laminar family. □

PROOF. We begin with an optimum solution $\{x_i^*\}, \{z_U^*\}$ of LP6 with the weights $\hat{w}_{ij}$, where the sets U such that $z_U > 0$ define a laminar family—such an optimum solution is guaranteed by Theorem 2.1. Note that it is important that Theorem 2.1 did not require integral edge weights. We produce a feasible solution for LP6 using the sequence of transformations given by Algorithm 3.

We have $\hat{\beta} = \sum_i b_i x_i^* + \sum_{U \in \mathcal{O}} \lfloor \|U\|_b / 2 \rfloor z_U^*$. Moreover, as a consequence of that optimality, note that $\sum_{U:i \in U} z_U^* \leq \hat{w}_L$ —because otherwise we can decrease the z_U^* corresponding to the smallest U with $i \in U, z_U^* > 0$ without affecting the feasibility. Observe that as a consequence of Step (1) of

Algorithm 3, we have

$$\hat{x}_i + \hat{x}_j + \sum_{U \in O_s, i, j \in U} \hat{z}_U \geq \hat{w}_{ij}, \quad \hat{x}_i \leq \hat{w}_L \quad \text{and}, \quad \sum_{U: i \in U} \hat{z}_U \leq \hat{w}_L. \quad (2)$$

A consequence of Step 2 is that $x_{i(k)} = \min\{\hat{w}_k, \hat{x}_i\}$ for all i . Therefore, for bipartite graphs, suppose we have an edge $(i, j) \in \hat{E}_k$; then

$$x_{i(k)} + x_{j(k)} \geq \min\{\hat{w}_k, \hat{x}_i + \hat{x}_j\} \geq \hat{w}_{ij},$$

which implies that the scaling in Step (17) satisfies $x_{i(k)} + x_{j(k)} \geq \hat{w}_{ij}$. Moreover, note that $x_{i(k)} \leq \hat{w}_k$. Now $\sum_i x_i b_i = \sum_i \hat{x}_i b_i$ and for bipartite graphs the z_U variables are absent, and hence $\sum_i b_i x_i \leq \hat{\beta}$. Therefore, for bipartite graphs the lemma is true. We now focus on nonbipartite graphs and make the following observations about the assignment to $z_{U, \ell}$:

$$\sum_{\ell \leq k} \left(\sum_{U: i \in U} z'_{U, \ell} \right) = \min \left\{ \hat{w}_k, \sum_{U: i \in U} \hat{z}_U \right\}.$$

Therefore, for an edge $(i, j) \in E_k$:

$$x_{i(k)} + x_{j(k)} + \sum_{\ell \leq k} \left(\sum_{U \in O_s, i, j \in U} z'_{U, \ell} \right) \geq \min \left\{ \hat{w}_k, \hat{x}_i + \hat{x}_j + \sum_{U \in O_s, i, j \in U} \hat{z}_U \right\} \geq \hat{w}_k.$$

Now from Step (1):

$$\sum_i x_i b_i + \sum_{U \in O_s} \left\lfloor \frac{\|U\|_b}{2} \right\rfloor \sum_{\ell} z_{U, \ell} \leq \sum_i \hat{x}_i b_i + \sum_{U \in O_s} \left\lfloor \frac{\|U\|_b}{2} \right\rfloor \hat{z}_U \leq (1 + \epsilon) \hat{\beta}.$$

The theorem follows. $\square$

3 DEFERRED CUT-SPARSIFICATION

In this section, we focus on deferred cut-sparsification, which is a key ingredient in reducing the number of iterations in the dual-primal framework provided in Theorem 1.3. Recall that Definition 3 requires sparsification data structures that can be constructed in parallel. However, instead of directly producing a sparsifier from the j th data structure, we wait till the simulation of the $j - 1$ previous iterations have completed and then we have the knowledge of the exact edge weights required to produce the j th cut-sparsifier. We then refine the j th data structure to produce the required cut-sparsifier, as shown in Lemma 3.1. We repeat the definition of deferred cut-sparsifiers for readability.

Definition 2 (The Deferred Cut-Sparsifier Problem). Consider the problem: We are given a weighted graph $G = (V, E, u)$ but the weights are **not revealed** to us. Instead we are given $\{\varsigma_{ij}\}$ with the promise that $\varsigma_{ij}/\chi \leq u_{ij} \leq \varsigma_{ij}\chi$. We have to produce a smaller summary data structure $\mathcal{D}$ storing only some of the edges. Once $\mathcal{D}$ is constructed, **then** the exact weights u_{ij} of only those edges stored in our structure $\mathcal{D}$ are revealed to us. We then output a sparsifier $H = (V, E', u^s)$.

LEMMA 3.1. Given $\{\varsigma_{ij}\}$ and edge weights $\in [1/\mathcal{L}_0, \mathcal{L}_0]$, we create an $O(n\chi^2\xi^{-2}(\log \mathcal{L}_0) \log^4 n)$ -size deferred sparsifier. We can construct a $(1 \pm \xi)$ -sparsification from the deferred sparsification when the edge weights of the stored edges are revealed.

The algorithm can be implemented in a single round of sketching, which implies that it can be implemented in $O(1)$ rounds in the Map-Reduce model or a single pass in the semistreaming model. In the semistreaming model, the algorithm runs in time $O(m(\log(n/\xi))\alpha_{m,n})$, where $\alpha_{m,n}$ is the inverse Ackermann function.

PROOF. The proof will follow from the fact that existing algorithms for sparsifier construction will allow us to implement the deferred version. We refer the reader to the excellent survey of Fung et al. (2011) for different algorithms for construction of sparsifiers. The main intuition (which in some form dates back to Nagamochi and Ibaraki (1992) and used in Benczúr and Karger (1996)) is that weighted sparsifiers are constructed using an *unweighted graph* by (i) first determining the probability p_e of sampling the edge e and then (ii) if e is sampled, then assigning it a value/importance w_e/p_e . For a weighted graph, part (i) is performed for each weight class in $[(1 + \varepsilon)^\ell, (1 + \varepsilon)^{\ell+1})$. The space requirement is $O(n\varepsilon^{-2}(\log \mathcal{L}_0) \log^4 n)$ when the weights are in the range $[1/\mathcal{L}_0, \mathcal{L}_0]$. The probability is the inverse of the connectivity across the edge e —and this can be achieved by a layered subsampling of edges $G_1, G_2, \dots$, where G_i contains the edges in G_{i-1} sampled with probability $1/2$.

We make the following observation: *If we have the promise weight ς_{ij} satisfying $u_{ij} \leq \varsigma_{ij}\chi \leq u_{ij}\chi^2$, then if we compute p'_e based on ς_{ij} (applying Benczúr and Karger (1996) sampling on the weights ς_{ij}) and multiply p'_e by $O(\chi^2)$, then we are guaranteeing that edge e is sampled with probability at least p_e .*

We repeat the process for weights ς_{ij} in the range $[2^\ell, 2^{\ell+1})$ for different values of ℓ and increase the sampling probability by $O(\chi^2)$. This concludes the construction of the data structure $\mathcal{D}$. Observe that the sum (allowing multiple edges between two nodes) of sparsifiers of a set of graphs is a sparsifier of the sum of the graphs. We can split the graph using ς_{ij} values, construct the deferred sparsifier of each graph, and then take the sum of the sparsifiers (keeping one edge between the vertices). The space requirement increases by $\log \mathcal{L}_0$ factor. The lemma follows. $\square$

3.1 (Deferred) Sparsification in Specific Models

The Streaming Model. We revisit the algorithm in Ahn et al. (2012b); see also Fung et al. (2011). Those results consider a single weight class. Note that if the edge set is partitioned across different weight classes, then the union of the sparsifiers corresponds to a sparsifier of the union. Algorithm 4 illustrates the overall algorithm. The running time follows from inspection, with the added twist that the number of edges in Step 13 of Algorithm 4 decreases geometrically. The overall running time is $O(m(\log(n/\xi))\alpha_{m,n})$ across all edges.

ALGORITHM 4: A Streaming Construction of Graph Sparsification

- 1: Let $G_0 = G$. Let G_i be obtained from sampling edges from G_{i-1} with prob. $1/2$.
 - 2: Let $k = O(\frac{1}{\varepsilon^2} \log^2 n)$.
 - 3: **for** $i = 1$ to $\log_2 m$ in parallel **do**
 - 4: Let $F_1^i = F_2^i = \dots = F_k^i = \emptyset$. Initialize k union find structures $UF_1^i, UF_2^i, \dots, UF_k^i$.
 - 5: **for** each $e = (u, v) \in G_i$ **do**
 - 6: Find the smallest j such that $UF_j^i.find(u) \neq UF_j^i.find(v)$.
 - 7: Let $F_j^i \leftarrow F_j^i \cup \{e\}$ and perform $UF_j^i.union(u, v)$.
 - 8: **end for**
 - 9: **end for**
 - 10: Let $H = (V, \emptyset)$.
 - 11: **for** $j = 1$ to $k - 1$ **do**
 - 12: **for** each $e = (u, v) \in F_j$ **do**
 - 13: If i' is the smallest value of $i \in [0, \log_2 m]$ s.t. $UF_k^i.find(u) \neq UF_k^i.find(v)$, insert e in H with weight $w_e/2^{i'}$. (Note that we get a weighted graph.)
 - 14: **end for**
 - 15: **end for**
 - 16: **return** H .
-

The MapReduce Model. We show how the algorithm mentioned in Lemma 3.1 can be implemented in the MapReduce model using $O(1)$ rounds. We use the fact that the algorithm in Ahn et al. (2012b) uses linear sketches of the vertex-edge adjacency matrix (after assigning arbitrary, lexicographic assignment of directions). We also assume that the central server has $\tilde{O}(n^{1+1/p})$ space. The algorithm proceeds as follows:

- (1) *First-Round Mapper:* For each edge (u, v) , the mapper generates $O(\text{polylog } n)$ bits of randomness, which will be used later for sketching. Let R be these random bits. The mapper outputs $(u, (u, v, R))$ and $(v, (u, v, R))$.
- (2) *First-Round Reducer:* Each reducer obtains a list of edges incident on a single vertex u and random bits corresponding to those edges. Using this list and random bits, construct ℓ_0 -sampling sketches for a vector $\mathbf{x}_u$. Let $S\mathbf{x}_u$ be the sketches. Output $(u, S\mathbf{x}_u)$.
- (3) *Second-Round Mapper:* Given $(u, S\mathbf{x}_u)$, output $(1, (u, S\mathbf{x}_u))$ so that all values are collected into one reducer.
- (4) *Second-Round Reducer:* Now we have all sketches $S\mathbf{x}_u$ in one machine. The rest of the algorithm is identical to the postprocessing of the sparsification algorithm in the context of dynamic graph streams (Ahn et al. 2012b).

4 WARMUP: WEIGHTED BIPARTITE MATCHING

In this section, we apply Theorem 1.3 and Theorem 1.2 to weighted bipartite b -matching. The bipartite b -matching illustrates how the framework and the new relaxation are used in combination and serves as a warmup for the general nonbipartite case. In particular, we prove Theorem 4.1.

THEOREM 4.1. *For any constant $0 < \epsilon \leq \frac{1}{16}$ and $p > 1$, we can find a $(1 - O(\epsilon))$ approximate integral weighted bipartite b -matching in $O(m \text{poly}(\epsilon^{-1}, \log n))$ time, $O(p/\epsilon)$ rounds, and $O(n^{1+1/p} \log B)$ centralized space.*

We begin with an alternate LP relaxation. We then provide a construction of the MICROORACLE used by the framework in Section 4.1. We provide the required initial solution in Section 4.2. Note that while it may appear that Section 4.2 is an approximation algorithm, the goal is to construct a feasible dual solution and the approximation factor is only a secondary consideration. Recall the system LP1 that defines the maximum weighted nonbipartite b -matching. The constraints corresponding to odd sets would automatically be satisfied in the context of bipartite graphs and we ignore those constraints in the remainder of this section. The resulting system is provided in LP8. We will consider the alternate rescaled system LP9. We can then consider the dual of LP9 and apply Theorem 1.2 to get LP11 and Lemma 4.2. We apply Theorem 1.3 on the system LP11. However, since the odd set constraints are absent, we can provide a simple direct proof of Lemma 4.2, and this section can be read independent of Section 2. We start with the following two definitions.

Definition 4. Let $W^* = \max_{(i,j) \in E} w_{ij}$. Using $O(p)$ rounds and $n^{1+1/p}$ space, we can easily find an edge with the maximum weight W^* (using ℓ_0 sampling, which can be implemented using sketches).

Definition 5. For $k \geq 0$, let $\hat{w}_k = (1 + \epsilon)^k$. Recall $B = \sum_i b_i$. Given an edge $(i, j) \in E$, we can define a level $k \geq 0$ such that $\frac{\epsilon W^*}{B} \hat{w}_k \leq w_{ij} < \frac{\epsilon W^*}{B} \hat{w}_{k+1}$ and let $\hat{w}_{ij}$ be that $\hat{w}_k$. Note that the mapping from $(i, j, w_{ij}) \rightarrow k$ is unique and only needs knowledge of W^* , B and the transformation of $w_{ij} \rightarrow \hat{w}_k$ can be achieved efficiently. Let $\hat{E}_k = \{(i, j) | (i, j) \in E, \hat{w}_{ij} = \hat{w}_k = (1 + \epsilon)^k\}$ and let $\hat{E} = \bigcup_k \hat{E}_k$. Let $\hat{w}_L$ correspond to the largest weight class in $\hat{E}$. Observe that $L = O(\frac{1}{\epsilon} \ln B)$ and the number of levels is $L + 1$.

OBSERVATION 1. $\hat{\beta}_b \leq \frac{B}{\epsilon W^*} \beta^* \leq \frac{(1+\epsilon)}{(1-\epsilon)} \hat{\beta}_b$, where $\hat{\beta}_b$ is defined in LP9. Moreover, any b -matching in $\hat{E}$ corresponds to a b -matching in E under rescaling of edge weights.

$$\left. \begin{aligned} \beta^* = \max \sum_{(i,j) \in E} w_{ij} y_{ij} \\ \sum_{j:(i,j) \in E} y_{ij} \leq b_i \quad \forall i \\ y_{ij} \geq 0 \quad \forall (i,j) \in E \end{aligned} \right| \quad \text{(LP8)} \quad \left. \begin{aligned} \hat{\beta}_b = \max \sum_{(i,j) \in \hat{E}} \hat{w}_{ij} y_{ij} \\ \sum_{j:(i,j) \in \hat{E}} y_{ij} \leq b_i \quad \forall i \\ y_{ij} \geq 0 \quad \forall (i,j) \in \hat{E} \end{aligned} \right| \quad \text{(LP9)}$$

Observation 1 follows from $W^* \leq \beta^*$, $\sum_{(i,j) \in E} w_{ij} y_{ij} \leq BW^*$ and that the constraints of LP9 imply the vertex constraints of LP1. Further, starting from a feasible solution of LP1, if we remove the edges in $E - \hat{E}$, we get a feasible solution for LP9 with the desired bound. In the remainder of the discussion, we find an integral $(1 - O(\epsilon))$ -approximate solution of LP9 and use the observation to show that the same solution is an integral $(1 - O(\epsilon))$ -approximation to the original problem.

LEMMA 4.2. $\hat{\beta}_b = \hat{\beta}'_b$ where

$$\left. \begin{aligned} \hat{\beta}_b = \max \sum_i b_i x_i \quad \text{(LP10)} \\ x_i + x_j \geq \hat{w}(k) \quad \forall (i,j) \in \hat{E}_k \\ x_i \geq 0 \quad \forall i, \end{aligned} \right| \quad \begin{aligned} \hat{\beta}'_b = \max \sum_i b_i x_i \quad \text{(LP11)} \\ x_{i(k)} + x_{j(k)} \geq \hat{w}(k) \quad \forall (i,j) \in \hat{E}_k \\ x_{i(k)} \leq x_i \quad \forall i, k \\ 2x_{i(k)} \leq 3\hat{w}_k \quad \forall i, k \\ x_{i(k)}, x_i \geq 0 \quad \forall i, k. \end{aligned}$$

PROOF. Note that the dual of LP9 is LP10. We will be applying the dual-primal method and focus on the modified dual polytope LP11 instead of LP10. Observe that LP10 is a relaxation of LP11. Given a solution of LP10, we can produce a solution of LP11 of the same value by defining $x_{i(k)} = \max\{0, \max_{(i,j) \in \hat{E}_k} \hat{w}_k - x_i\}$. Observe that $x_{i(k)} \leq \hat{w}_k$. $\square$

Applying Theorem 1.3: We define $\mathbf{Q}, \mathbf{A}, \mathbf{c}, \mathbf{P}_o, \mathbf{P}_i, \mathbf{q}_o, \mathbf{q}_i$ for the weighted bipartite case starting from LP11 as follows:

$$\begin{aligned} \mathbf{Q} : \{x_i - x_{i(\ell)} \geq 0, \forall i, \ell\} \\ \{\mathbf{Ax} \geq \mathbf{c}\} : \{x_{i(k)} + x_{j(k)} \geq \hat{w}_k, \forall (i,j) \in \hat{E}_k\} \\ \{\mathbf{P}_o \mathbf{x} \leq \mathbf{q}_o\} : \{2x_{i(k)} \leq 3\hat{w}_k, \forall i, k\} \\ \{\mathbf{P}_i \mathbf{x} \leq \mathbf{q}_i\} : \{2x_{i(k)} \leq 24\hat{w}_k/\epsilon, \forall i, k\}. \end{aligned}$$

And finally, we define $\mathbb{G}(\mathbf{u}^s, \mathbf{x})$ as null. We show that LP9 is $(6, O(1/\epsilon))$ -dual-primal amenable with $a_1 = 3$, since a solution of $\mathbf{Ax} \geq (1 - 3\epsilon)\mathbf{c}$, setting $x_i = \frac{1}{1-3\epsilon} \max_{\ell} x_{i(\ell)}$, will satisfy the dual of LP9. Therefore, the conditions (d1)–(d3) in Definition 3 hold with $a_1 = 3$. Note that for $\mathbf{P}_o \mathbf{x} \leq \mathbf{q}_o$ we do not need to have constraints for i, k if the vertex i has no edges of level k incident to it. The same holds for $\mathbf{P}_i \mathbf{x} \leq \mathbf{q}_i$. Therefore, $\tilde{n}_o = \tilde{n}_i = \min\{m, \frac{n}{\epsilon} \log B\}$. Note $u_{ijk} = \exp(-\alpha(x_{i(k)} + x_{j(k)})/\hat{w}_k)/\hat{w}_k$. The next lemma follows from sparsifiers and addresses Condition (d4) of Definition 3.

LEMMA 4.3. Let $\mathbf{u} = \{u_{ijk}\}$, $\mathbf{x} = \{x_{i(k)}, x_i\}$. Suppose that for each weight class k we construct a $(1 \pm \epsilon/16)$ -Cut-Sparsifier sparsifier where the initial importance³ of an edge (i, j) is given by u_{ijk} and the sparsified importance is u_{ijk}^s . Then,

$$(\mathbf{u}^s)^T \mathbf{A} \mathbf{x} \geq \sum_k \left(\sum_{(i,j) \in \hat{E}_k} \hat{w}_k u_{ijk}^s \right), \mathbf{x} \in \mathcal{Q} \Rightarrow \mathbf{u}^T \mathbf{A} \mathbf{x} \geq \left(1 - \frac{\epsilon}{2}\right) \sum_k \left(\sum_{(i,j) \in \hat{E}_k} \hat{w}_k u_{ijk} \right),$$

which is the desired equation SWITCH in Condition (d4) of Definition 3.

PROOF. Note that each edge belongs to exactly one class and $\mathbf{u}^T \mathbf{A} \mathbf{x} = \sum_{i,k} x_{i(k)} (\sum_{j:(i,j) \in \hat{E}_k} u_{ijk})$. And the $\sum_{j:(i,j) \in \hat{E}_k}$ corresponds to a cut defined by separating the vertex i from the remainder of the graph. The lemma now follows from definition of sparsifiers and the fact that every $x_{i(k)} \geq 0$. $\square$

We now focus on Condition (d5) of Definition 3, which defines the initial (infeasible) solution to the dual of the bipartite matching problem. We relegate its proof to Section 4.2 to focus on the algorithm and applying the dual-primal framework. Note that some of the parameters are intentionally suboptimal, to ensure that the exact same initial solution can be used in the nonbipartite case.

THEOREM 4.4 (PROVED IN SECTION 4.2). For each weight class k and the corresponding edge set $\hat{E}_k$, construct a maximal b -matching⁴ denoted as M_k . Each $\{M_k\}$ is computed using $O(n^{1+1/(2p)})$ space and $O(p)$ rounds of sketching, in parallel. If for a vertex i is saturated in M_k (i.e., has degree of b_i), then we set $x_{i(k)} = r\hat{w}_k = \epsilon\hat{w}_k/256$; otherwise, $x_{i(k)} = 0$. Finally, set $x_i = \max_k x_{i(k)}$. This defines $\mathbf{x}_0$. The constructed solution satisfies the constraint corresponding to $\mathcal{Q}, \mathbf{P}_o, \mathbf{A}\mathbf{x}_0 \geq rc$ and $\epsilon r \hat{\beta}_b/4 \leq \beta_0 = \mathbf{b}^T \mathbf{x}_0 = \sum_i b_i x_i \leq \hat{\beta}_b/4$.

The MICROORACLE is provided in Algorithm 5 and proved in Lemma 4.5 in Section 4.1. We can now apply Theorem 1.3 and conclude Theorem 4.1.

4.1 The MICROORACLE for Bipartite Matching

The MICROORACLE is provided in Algorithm 5. The main intuition behind the algorithm is as follows: the dual-primal method tries to solve the dual of the weighted bipartite b -matching. In each iteration, we seek to assign values to the constraints of the new weighted bipartite b -matching such that the multiplicative weight update framework (applied to the dual) succeeds.

We focus on the sum of the violations of the constraints that are unsatisfied. If that sum is small, we show that we can have a feasible solution to the weighted bipartite b -matching with a corresponding small loss in the objective function. Otherwise, if that sum of the violations is not small, then we can isolate a few constraints such that setting dual variables proportional to the violation of the constraints would provide the required oracle for the dual-primal framework.

LEMMA 4.5. Algorithm 5 provides us a MICROORACLE such that given $\{u_{ijk}^s, \zeta_{ik}\}$, either we provide a solution of LP12 or a large integral matching of weight at least $(1 - O(\epsilon))\hat{\beta}_b$ restricted to

³The sparsifier literature refers to this as the edge weight. But the edge weight refers to w_{ij} in our current context. We are applying sparsifiers to the importance of an edge (i, j) with respect to the primal-dual solution of linear programs.

⁴Note that this allows edges to be present with multiplicity higher than 1, but the multiplicity of (i, j) is at most $\min\{b_i, b_j\}$.

ALGORITHM 5: The MICROORACLE for Weighted Bipartite Matching

- 1: Let $\gamma = \sum_k \hat{w}_k \left(\sum_{(i,j) \in \hat{E}_k} u_{ijk}^s - 3\rho \sum_i \zeta_{ik} \right)$. Note $\rho > 0, 0 < \epsilon \leq \frac{1}{16}$. Also $\gamma > 0$ below.
- 2: Define for all $i, \text{Pos}(i) = \left\{ k \left| \sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\rho \zeta_{ik} > 0 \right. \right\}$ and for all i, ℓ

$$\Delta(i, \ell) = \sum_{k \in \text{Pos}(i), \ell \geq k} \hat{w}_k \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\rho \zeta_{ik} \right) + \sum_{k \in \text{Pos}(i), k > \ell} \hat{w}_\ell \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\rho \zeta_{ik} \right).$$
- 3: Let $k_i^* = \arg\max_\ell \Delta(i, \ell) > \frac{\gamma b_i \hat{w}_\ell}{\beta}$ and $k_i^* = -1$ if no ℓ exists.
- 4: $\mathcal{V}iol(V) = \{i | k_i^* \geq 0\}$ and $\Gamma(V) = \sum_{i \in \mathcal{V}iol(V)} \Delta(i, k_i^*)$.
- 5: **if** $\Gamma(V) \geq \epsilon\gamma/24$ **then**
- 6: **for** all $i \in \mathcal{V}iol(V)$, (implies $k_i^* \geq 0$) set $x_{i(\ell)} = \begin{cases} \gamma \hat{w}_{k_i^*} / \Gamma(V) & \ell \in \text{Pos}(i), \ell > k_i^* \\ \gamma \hat{w}_\ell / \Gamma(V) & \ell \in \text{Pos}(i), \ell \leq k_i^* \end{cases}$.
- 7: Set $x_{i(\ell)} = 0$ for $i \notin \mathcal{V}iol(V)$ or $\ell \notin \text{Pos}(i)$, $x_i \max_k x_{i(k)}$ and **return**.
- 8: **end if**
- 9: $\tilde{\zeta}_{ik} \leftarrow \begin{cases} \frac{1}{2\rho} \sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s & \text{if } i \in \mathcal{V}iol(V), k \leq k_i^*, k \in \text{Pos}(i) \\ \zeta_{ik} & \text{otherwise} \end{cases}$.
- 10: Let $\gamma' = \sum_k \hat{w}_k \left(\sum_{(i,j) \in \hat{E}_k} u_{ijk}^s - 3\rho \sum_i \tilde{\zeta}_{ik} \right)$.
- 11: Set $y_{ij} \leftarrow \frac{(1-\epsilon/4)\beta}{(1+\epsilon/2)\gamma} u_{ijk}^s$. Set $\mu_{ik} = \frac{(1-\epsilon/4)\beta\rho}{(1+\epsilon/2)\gamma} \tilde{\zeta}_{ik}$. **return** $\{y_{ij}\}, \{\mu_{ik}\}$.

$\{(i, j) | u_{ijk}^s > 0\}$.

$$\begin{aligned} \sum_{i,k} x_{i(k)} \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\rho \zeta_{ik} \right) &\geq (1 - O(\epsilon)) \sum_k \hat{w}_k \left(\sum_{(i,j) \in \hat{E}'_k} u_{ijk}^s - 3\rho \sum_i \zeta_{ik} \right) & (\text{LP12}) \\ \sum_i b_i x_i &\leq \beta \\ x_{i(k)} &\leq \frac{24\hat{w}_k}{\epsilon} & \forall i, k \\ x_i - x_{i(k)} & & \forall i, k \\ x_{i(k)}, x_i &\geq 0 & \forall i, k \end{aligned}$$

PROOF. For the **return** step in Step 7 in the algorithm, observe that if $i \notin \mathcal{V}iol(V)$, then $x_{i(k)} = 0$. For $i \in \mathcal{V}iol(V)$,

$$\begin{aligned} &\sum_k x_{i(k)} \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\rho \zeta_{ik} \right) \\ &= \left(\sum_{k \leq k_i^*, k \in \text{Pos}(i)} \frac{\gamma \hat{w}_k}{\Gamma(V)} \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\rho \zeta_{ik} \right) + \sum_{k > k_i^*, k \in \text{Pos}(i)} \frac{\gamma \hat{w}_{k_i^*}}{\Gamma(V)} \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\rho \zeta_{ik} \right) \right) \\ &\quad (\text{using definitions of } x_{i(\ell)}, \Delta(i, \ell), \mathcal{V}iol(V) \text{ in Steps (6), (2), and (4), respectively}) \\ &= \frac{\gamma}{\Gamma(V)} \Delta(i, k_i^*). \end{aligned}$$

Therefore, using the definition of $\Gamma(V)$ in Step (4) of Algorithm 5,

$$\sum_{i,k} x_{i(k)} \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\varrho \zeta_{ik} \right) = \frac{\gamma}{\Gamma(V)} \sum_{i \in \mathcal{V}^{Viol}(V)} \Delta(i, k_i^*) = \gamma.$$

At the same time,

$$\begin{aligned} \sum_i x_i b_i &= \sum_{i \in \mathcal{V}^{Viol}(V)} \frac{\gamma \hat{w}_{k_i^*}}{\Gamma(V)} b_i = \frac{\beta}{\Gamma(V)} \sum_{i \in \mathcal{V}^{Viol}(V)} \frac{\gamma b_i \hat{w}_{k_i^*}}{\beta} \leq \sum_{i \in \mathcal{V}^{Viol}(V)} \Delta(i, k_i^*) \quad (\text{Definition of } k_i^*) \\ &= \frac{\beta}{\Gamma(V)} \Gamma(V) = \beta \quad (\text{Definition of } \Gamma(V)). \end{aligned}$$

Observe that each $x_{i(\ell)} \leq \frac{24}{\epsilon} \hat{w}_\ell$. Therefore, we satisfy Lemma 5.3 in this case. In the remainder of the proof, we assume $\Gamma(V) \leq \epsilon \gamma / 24$. Observe that

$$\begin{aligned} \gamma' &= \sum_k \hat{w}_k \left(\sum_{(i,j) \in \hat{E}_k} u_{ijk}^s - 3\varrho \sum_i \bar{\zeta}_{ik} \right) = \gamma - 3 \sum_{i \in \mathcal{V}^{Viol}(i)} \left(\sum_{k \in \mathcal{P}os(i), k \leq k_i^*} \hat{w}_k (\bar{\zeta}_{ik} - \zeta_{ik}) \right) \\ &= \gamma - \frac{3}{2} \sum_{i \in \mathcal{V}^{Viol}(i)} \sum_{k \in \mathcal{P}os(i), k \leq k_i^*} \hat{w}_k \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\varrho \zeta_{ik} \right) \geq \gamma - \frac{3}{2} \sum_{i \in \mathcal{V}^{Viol}(i)} \Delta(i, k_i^*) \\ &\geq \gamma - \frac{3}{2} \Gamma(V) \geq \gamma - \epsilon \gamma / 16, \end{aligned}$$

and as a consequence, $\gamma' \geq (1 - \epsilon/16)\gamma$. Therefore,

$$\sum_k \hat{w}_k \left(\sum_{(i,j) \in \hat{E}_k} y_{ij} - 3 \sum_i \mu_{ik} \right) = \frac{\beta \gamma'}{\gamma} \left(\frac{1 - \epsilon/4}{1 + \epsilon/2} \right) \geq (1 - O(\epsilon))\beta.$$

Further, for every i ,

$$\begin{aligned} \frac{\gamma b_i \hat{w}_{k_i^*+1}}{\beta} &\geq \sum_{k \in \mathcal{P}os(i), k > k_i^*} \hat{w}_{k_i^*+1} \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\varrho \zeta_{ik} \right) \\ \Rightarrow \frac{\gamma b_i}{\beta} &\geq \sum_{k \in \mathcal{P}os(i), k > k_i^*} \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\varrho \zeta_{ik} \right) \geq \sum_{k \in \mathcal{P}os(i), k > k_i^*} \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\varrho \bar{\zeta}_{ik} \right) \\ \Rightarrow b_i &\geq \sum_{k \in \mathcal{P}os(i), k > k_i^*} \left(\sum_{j:(i,j) \in \hat{E}_k} y_{ij} - 2\mu_{ik} \right). \end{aligned}$$

Note that for $k \leq k_i^*$, either $k \notin \mathcal{P}os(i)$ or, by construction, $\sum_{j:(i,j) \in \hat{E}_k} y_{ij} - 2\mu_{ik} \leq 0$. Therefore, if we define $y_{i(k)}$ as

$$y_{i(k)} = \max \left\{ \sum_{j:(i,j) \in \hat{E}_k} y_{ij} - 2\mu_{ik}, 0 \right\},$$

then we have $\{y_{ij}, y_{i(k)}, \mu_{ik}\}$ which satisfies

$$\left. \begin{array}{l} \sum_k \hat{w}_k \left(\sum_{(i,j) \in \hat{E}_k} y_{ij} - 3 \sum_i \mu_{ik} \right) \\ \geq (1 - O(\epsilon))\beta \\ \sum_{j:(i,j) \in \hat{E}_k} y_{ij} - 2\mu_{ik} \leq y_{i(k)} \quad \forall i, k \\ \sum_k y_{i(k)} \leq b_i \quad \forall i \\ y_{i(k)}, y_{ij} \geq 0 \quad \forall i, k, j \end{array} \right| \begin{array}{l} \hat{\beta}_b = \max \sum_i b_i x_i \quad (\text{LP11}) \\ x_{i(k)} + x_{j(k)} \geq \hat{w}(k) \quad \forall (i,j) \in \hat{E}_k \quad (\text{Dual of } y_{ij}) \\ x_{i(k)} \leq x_i \quad \forall i, k \quad (\text{Dual of } y_{i(k)}) \\ 2x_{i(k)} \leq 3\hat{w}_k \quad \forall i, k \quad (\text{Dual of } \mu_{ik}) \\ x_{i(k)}, x_i \geq 0 \quad \forall i, k, \end{array}$$

but that implies that the dual optimization problem expressed in LP11 must have $\hat{\beta}_b \leq (1 + O(\epsilon))\beta$. Therefore, we have a large matching in the subgraph defined by $\{(i,j) | y_{ij} > 0\}$. We can extract an integral solution in near-linear time using algorithms such as Duan and Pettie (2010) for matching and Ahn and Guha (2014) for b -matching. The lemma follows. $\square$

4.2 Initial Dual Solution: Proof of Theorem 4.4

In this section, we prove the following theorem; recall that the weight class k corresponded to edges with weight $(1 + \epsilon)^k$.

THEOREM 4.4. *For each weight class k and the corresponding edge set $\hat{E}_k$, construct a maximal b -matching denoted as M_k . Edge (i, j) is allowed to be present with multiplicity higher than 1, but at most $\min\{b_i, b_j\}$. Each $\{M_k\}$ is computed using $O(n^{1+1/(2p)})$ space and $O(p)$ rounds of sketching, in parallel. If a vertex i is saturated in M_k (i.e., has degree of b_i), then we set $x_{i(k)} = r\hat{w}_k = \epsilon\hat{w}_k/256$; otherwise, $x_{i(k)} = 0$. Finally, set $x_i = \max_k x_{i(k)}$. This defines $\mathbf{x}_0$. The constructed solution satisfies the constraint corresponding to $\mathbf{Q}, \mathbf{P}_0, \mathbf{A}\mathbf{x}_0 \geq \mathbf{r}\mathbf{c}$, and $\epsilon r \hat{\beta}_b/4 \leq \beta_0 = \mathbf{b}^T \mathbf{x}_0 = \sum_i b_i x_i \leq \hat{\beta}_b/4$.*

The running time corresponds to constructing L different maximal b -matchings, one for each weight class. We reuse the following result to prove the theorem.

LEMMA 4.6 (LATTANZI ET AL. 2011, LEMMA 3.1). *If we sample edges in a graph uniformly at random with probability q , then with probability at least $1 - e^{-n}$, for every subgraph with at least $2n/q$ edges we have chosen at least one edge.*

LEMMA 4.7. *We can construct a maximal b -matching for any set of edges using $O(n^{1+1/p})$ space and $O(p)$ rounds of sketching.*

PROOF. Lemma 4.6 was used in Lattanzi et al. (2011, Lemma 3.2) to produce a maximal matching in $O(p)$ phases when $q = O(n^{1+1/p}/|E_t|)$ for each phase t —there E_t was the number of edges remaining before the start of phase t . There it was proven that for maximal matching, $|E_t|$ decreases by a factor of $n^{1/p}$ at each step. Now observe that for the uncapacitated b -matching, whenever we choose an edge (i, j) , we can increase its multiplicity to $\min\{b_i, b_j\}$ and therefore saturate one or both of the endpoints. Therefore, for every edge chosen we can ensure that one of the endpoints is not in consideration anymore. As a consequence, the analysis of Lattanzi et al. (2011, Lemma 3.2), instead of being limited to the set of unmatched vertices, holds for the set of vertices i that do not yet have degree equal to b_i . By the exact same analysis, the number of edges decreases by a factor $n^{1/p}$ at each step and after $O(1/p)$ steps we can guarantee that we have a maximal b -matching to which we cannot add any more edges. Therefore, the algorithm in Section 3 of Lattanzi et al. (2011) works, with the change that in step 3 we construct a maximal b -matching (and saturate one of the vertices when we choose an edge), and step 4 considers the set of vertices i that do not yet have

degree equal to b_i . We conclude with the observation that the primitives used in this algorithm can be implemented using sketches (Ahn et al. 2012a). $\square$

We now focus on the quality of the initial solution. It is straightforward to verify that the conditions of $\mathbf{Q}, \mathbf{P}_o, \mathbf{A}\mathbf{x}_0 \geq rc$ are satisfied. If we naively try to bound $\beta_0 = \mathbf{b}^T \mathbf{x}_0$, then that bound will be $2rL\hat{\beta}_b$ since we have two endpoints and L levels and use each maximum b -matching up to a fraction r . We seek to remove the dependence on L because it is a nonconstant term.

CLAIM 1. *Let the edges in M_k be $E(M_k)$. Let $|E(M_k)|$ denote the number of edges (counting each copy of an edge separately). Then $\hat{\beta}_b \leq 2 \sum_k w_k |E(M_k)|$.*

PROOF. Consider the edges in the optimum solution that are in level k , denoted by OPT_k . Counting each copy separately, let the number of such edges be $|OPT_k|$. Given the maximality of M_k for level k , we must have at least $|OPT_k|/2$ edges in M_k because a single copy of a single edge (i, j) can dislodge at most two edges from OPT_k , one each from i and j . For multiplicities, choose a copy of an edge adjacent to i to dislodge arbitrarily. The lemma follows from summing over all k . $\square$

CLAIM 2. $2r|E(M_k)|\hat{w}_k \geq \sum_i b_i x_{i(k)} \geq r|E(M_k)|\hat{w}_k$ for each k .

PROOF. Consider an edge by edge accounting where each (copy of an) edge in M_k pays $r\hat{w}_k$ to an endpoint i , which satisfies degree equal to b_i in the maximal matching for the level k . By maximality, one endpoint must satisfy the condition and there are at most two endpoints. The total charge collected by vertex i is $b_i x_{i(k)}$; note $x_{i(k)} = r\hat{w}_k$ if i has b_i adjacent edges and is 0 otherwise. The claim follows. $\square$

Since the $\hat{w}_k$ are powers of $(1 + \epsilon)$ and $x_{i(k)} = r\hat{w}_k$ or 0, we have $\frac{\epsilon}{1+\epsilon} \sum_k x_{i(k)} \leq \max_k x_{i(k)} \leq \sum_k x_{i(k)}$. Using Claim 2, $1 + \epsilon \leq 1$, and $x_i = \max_k x_{i(k)}$, we have

$$\sum_i b_i x_i \geq \frac{\epsilon}{1 + \epsilon} \sum_k b_i x_{i(k)} \geq \sum_k \frac{\epsilon}{2} r |E(M_k)| \hat{w}_k \geq \frac{\epsilon r}{4} \hat{\beta}_b$$

For the other direction, consider the following algorithm: we add all the edges corresponding to the feasible matching at the highest level. This defines a current matching. We then proceed to the next lower level and if we can add any edge from this edge to the current matching, then we do so (in arbitrary order) while maintaining feasibility. Let the final matching be $\mathcal{M}$.

CLAIM 3. $\frac{4}{\epsilon} \sum_k \hat{w}_k |E(M_k)| \leq \text{weight}(\mathcal{M}) \leq \hat{\beta}_b$.

PROOF. Observe that the weights of the edges in the levels go down by a factor of 2. If we cannot add an edge from M_k due to the fact that its endpoint i already satisfies its degree to be b_i , then we say that the edge at level k is blocked. In particular, if y edges were blocked, then there must be y existing edges (including multiplicities) that block these edges. We choose an arbitrary permutation that charges each blocked edge to a single blocking edge. If both endpoints of an edge are blocked, we choose one of the endpoints arbitrarily. Since the edge weights in the levels go down by powers of $(1 + \epsilon)$, the total weight of edges (recursively) blocked by a single edge is at most $(2 + \frac{2}{(1+\epsilon)} + \frac{2}{(1+\epsilon)^2} + \dots) \leq 2 \frac{1+\epsilon}{\epsilon}$ times the weight of the edge attached to i . Each such edge is charged $2 \frac{1+\epsilon}{\epsilon}$ times its weight. Since $\epsilon \leq 1$ and $\mathcal{M}$ is a feasible matching, the claim follows. $\square$

For $r = \frac{\epsilon}{32}$, $\sum_i b_i x_i \leq \sum_k b_i x_{i(k)} \leq 2r \sum_k |E(M_k)| \hat{w}_k \leq \frac{8r}{\epsilon} \hat{\beta}_b \leq \hat{\beta}_b/4$. The theorem follows.

5 WEIGHTED NONBIPARTITE B -MATCHING

In this section, we show the application of Theorems 1.2 and 1.3 in the context of nonbipartite b -matching and prove Theorem 1.1.

THEOREM 1.1. *For any constant $0 < \epsilon \leq \frac{1}{16}$ and $p > 1$, we can find a $(1 - O(\epsilon))$ approximate integral weighted b -matching for nonbipartite graphs in $O(m \text{poly}(\epsilon^{-1}, \log n))$ time, $O(p/\epsilon)$ rounds, and $O(n^{1+1/p} \log B)$ centralized space.*

Roadmap: We will be reusing almost all the results for the bipartite case discussed in Section 4. In particular, we will use the discretization (Definition 4), levels (Definition 5), and the rescaling (Observation 1) for constructing the appropriate relaxation before we can apply the framework (Theorem 1.3). Note that the rescaling applies irrespective of the bipartiteness of the graph; however, we now have extra constraints as shown in LP13. We apply Theorem 1.2 (which corresponds to the same discretization) and know that $\hat{\beta} \leq \tilde{\beta} \leq (1 + O(\epsilon))\hat{\beta}$. We will also be reusing the initial solution, since the dual feasible solution for the bipartite relaxation can easily provide a dual feasible solution for the nonbipartite case. We will also reuse the main intuition in constructing the new MICROORACLE: we consider the violated constraints and the sum of the violations. Since the number of constraints are large, we choose a maximal set of disjoint constraints. If the sum of the violations is small, then we prove that we have a large b -matching. Otherwise, when the sum of the violations is large, we show that we can assign dual values that are proportional to the violations and preserve the required properties of the oracle required for the dual-primal framework.

We are interested in the solution of the dual LP6 or finding a subgraph with a matching of size $\beta(1 - O(\epsilon))$ such that $\hat{\beta} \leq (1 + O(\epsilon))\beta$.

$$\begin{array}{lcl}
 \hat{\beta} = \max \sum_{(i,j) \in \hat{E}} \hat{w}_{ij} y_{ij} & & \\
 \sum_{j:(i,j) \in \hat{E}} y_{ij} \leq b_i \forall i & \text{(LP13)} & \\
 \sum_{(i,j) \in \hat{E}: i, j \in U} y_{ij} \leq \lfloor \|U\|_b / 2 \rfloor \forall U \in \mathcal{O} & & \\
 y_{ij} \geq 0 \quad \forall (i,j) \in \hat{E} & &
 \end{array}
 \quad
 \begin{array}{l}
 \tilde{\beta} = \min \sum_i b_i x_i + \sum_{U \in \mathcal{O}_s} \left\lfloor \frac{\|U\|_b}{2} \right\rfloor \sum_{\ell} z_{U,\ell} \\
 x_{i(k)} + x_{j(k)} + \sum_{\ell \leq k} \left(\sum_{U \in \mathcal{O}_s: i, j \in U} z_{U,\ell} \right) \geq \hat{w}_k \quad \forall (i,j) \in E_k \\
 2x_{i(k)} + \sum_{\ell \leq k} \left(\sum_{U \in \mathcal{O}_s: i \in U} z_{U,\ell} \right) \leq 3\hat{w}_k \quad \forall i, k \\
 x_i - x_{i(k)} \geq 0 \quad \forall i, k \\
 x_{i(k)}, x_i, z_{U,\ell} \geq 0
 \end{array}
 \quad \text{(LP6)}$$

We define $\mathbf{Q}, \mathbf{A}, \mathbf{c}, \mathbf{P}_o, \mathbf{P}_i, \mathbf{q}_o, \mathbf{q}_i$ as follows:

$$\begin{aligned}
 \mathbf{Q} &: \{x_i - x_{i(\ell)} \geq 0, \forall i, \ell\} \\
 \{\mathbf{A}\mathbf{x} \geq \mathbf{c}\} &: \left\{ x_{i(k)} + x_{j(k)} + \sum_{\ell \leq k} \left(\sum_{U \in \mathcal{O}_s: i, j \in U} z_{U,\ell} \right) \geq \hat{w}_k \quad \forall (i,j) \in \hat{E}_k \right\} \\
 \{\mathbf{P}_o \mathbf{x} \leq \mathbf{q}_o\} &: \left\{ 2x_{i(k)} + \sum_{\ell \leq k} \left(\sum_{U \in \mathcal{O}_s: i \in U} z_{U,\ell} \right) \leq 3\hat{w}_k \quad \forall i, k \right\} \\
 \{\mathbf{P}_i \mathbf{x} \leq \mathbf{q}_i\} &: \left\{ 2x_{i(k)} + \sum_{\ell \leq k} \left(\sum_{U \in \mathcal{O}_s: i \in U} z_{U,\ell} \right) \leq \frac{24\hat{w}_k}{\epsilon^2} \quad \forall i, k \right\}.
 \end{aligned}$$

The constraint $G(\mathbf{u}^s, \mathbf{x})$ is now necessary and somewhat involved:

$$G(\mathbf{u}^s, \mathbf{x}) : \left\{ z_{U,\ell} \left(\sum_{k \geq \ell} \left(\sum_{(i,j) \in \hat{E}_k, i, j \in U} u_{ijk}^s - \sum_{i \in U} \left(\sum_{j \notin U, (i,j) \in \hat{E}_k} u_{ijk}^s \right) \right) \right) \geq 0 \forall U \in \mathcal{O}_s, \ell \right\}.$$

We show that LP13 is $(6, O(\epsilon^{-2}))$ -dual-primal amenable with $a_1 = 3$, since a solution of $\mathbf{Ax} \geq (1 - 3\epsilon)\mathbf{c}$, setting $x_i = \frac{1}{1-3\epsilon} \max_{\ell} x_{i(\ell)}$, $z_U = \frac{1}{1-3\epsilon} \sum_{\ell} z_{U,\ell}$, will satisfy the dual of LP13. Therefore, the conditions (d1) to (d3) in Definition 3 hold with $a_1 = 3$. Note that for $\mathbf{P}_o \mathbf{x} \leq \mathbf{q}_o$ we do not need to have constraints for i, k if the vertex i has no edges of level k incident to it. The same holds for $\mathbf{P}_i \mathbf{x} \leq \mathbf{q}_i$. Therefore, $\tilde{n}_o = \tilde{n}_i = \min\{m, \frac{n}{\epsilon} \log B\}$. We focus on condition (d4) in Definition 3. Let $\mathbf{x} = \{x_{i(k)}, x_i, z_{U,\ell}\}$. Note u_{ijk} is

$$\frac{1}{\hat{w}_k} \exp \left(-\alpha \left(x_{i(k)} + x_{j(k)} + \sum_{\ell \leq k} \left(\sum_{U \in O_s, i, j \in U} z_{U,\ell} \right) \right) / \hat{w}_k \right),$$

and with $\mathbf{u} = \{u_{ijk}\}$, we have

$$\mathbf{u}^T \mathbf{Ax} = \sum_{i,k} x_{i(k)} \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk} \right) + \sum_{U \in O_s, \ell} z_{U,\ell} \left(\sum_{k \geq \ell} \left(\sum_{(i,j) \in \hat{E}_k, i, j \in U} u_{ijk} \right) \right).$$

The next lemma follows from sparsifiers. The lemma is analogous to Lemma 4.3; however, we now have extra $z_{U,\ell}$ variables. We relegate the proof of the lemma to Section 5.1 to not interrupt the discussion.

LEMMA 5.1. *If for each $k \geq 0$ we have $H_k = (V, E'_k, \{u_{ijk}^s\})$ as a $(1 \pm \epsilon/16)$ -Cut-Sparsifier for $G_k^w = (V, \hat{E}_k, \{u_{ijk}\})$, then let their union be $\mathbf{u}^s = \{u_{ijk}^s\}$. Then, if $\mathbf{x} \in \mathbf{Q}$,*

$$(\mathbf{u}^s)^T \mathbf{Ax} \geq \sum_k \left(\sum_{(i,j) \in \hat{E}_k} \hat{w}_k u_{ijk}^s \right), \mathbb{G}(\mathbf{u}^s, \mathbf{x}), \quad \Rightarrow \quad \mathbf{u}^T \mathbf{Ax} \geq \left(1 - \frac{\epsilon}{2}\right) \sum_k \left(\sum_{(i,j) \in \hat{E}_k} \hat{w}_k u_{ijk} \right),$$

which is the desired equation SWITCH in Condition (d4) of Definition 3.

For condition (d5), observe that setting all $z_{U,\ell} = 0$ corresponds to a bipartite relaxation. Let β^b be the optimum value of the bipartite relaxation, $\hat{\beta} \leq \beta^b \leq \frac{3}{2}\hat{\beta}$. Therefore, we can use the initial for the bipartite case (Theorem 4.4) without any change! The MICROORACLE is provided in Algorithm 6 and proved in Lemma 5.3. Note that the current multipliers on the constraints are updated using convex combinations of solutions. Note that since we have changed the roles of the primal of the dual, the constraints now correspond to edges. The multiplier u_{ijk} for an edge (i, k) of weight $\hat{w}_k$ (in level k) depends on $x_{i(k)}, x_{j(k)}$ and the sets U with $z_{U,\ell} > 0$ such that $i, j \in U$ and $k \geq \ell$. Based on the disjointedness property in Lemma 5.3 and the proof of Theorem 1.3, the maximum number of variables relevant to any edge (at any point of time in the algorithm) is at most $O(\epsilon^{-5}(\log B)(\log^2 n) \log^2 \frac{1}{\epsilon})$ as stated earlier. Thus, we can conclude Theorem 1.1 once we prove Lemma 5.3.

We now focus on the MICROORACLE, which uses the following lemma whose proof is relegated to Section 5.2.

LEMMA 5.2 (PROVED IN SECTION 5.2). *In time $O(n \text{poly}(\log n, \frac{1}{\epsilon}))$ we can find a collection $\mathcal{K}(\ell)$ such that every pair of sets are mutually disjoint and Equations (3) and (4) hold in Step 12.*

Observe that Lemma 5.2 implies that Algorithm 6 runs in time $O(nL \text{poly}(\log n, 1/\epsilon))$ for $L+1$ levels of the discretized weights since we invoke Lemma 5.2 at most $L+1$ times.

ALGORITHM 6: The Part (ii) of MICROORACLE for Matching

- 1: Let $\gamma = \sum_k \hat{w}_k \left(\sum_{(i,j) \in \hat{E}_k} u_{ijk}^s - 3\varrho \sum_i \zeta_{ik} \right)$. Note $\varrho > 0, 0 < \epsilon \leq \frac{1}{16}$. Also $\gamma > 0$ below.
- 2: Define for all $i, \mathcal{Pos}(i) = \left\{ k \left| \sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\varrho \zeta_{ik} > 0 \right. \right\}$ and for all i, ℓ

$$\Delta(i, \ell) = \sum_{k \in \mathcal{Pos}(i), \ell \geq k} \hat{w}_k \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\varrho \zeta_{ik} \right) + \sum_{k \in \mathcal{Pos}(i), k > \ell} \hat{w}_\ell \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\varrho \zeta_{ik} \right).$$
- 3: Let $k_i^* = \operatorname{argmax}_\ell \Delta(i, \ell) > \frac{\gamma b_i \hat{w}_\ell}{\beta}$ and $k_i^* = -1$ if no ℓ exists.
- 4: $\mathcal{Viol}(V) = \{i | k_i^* \geq 0\}$ and $\Gamma(V) = \sum_{i \in \mathcal{Viol}(V)} \Delta(i, k_i^*)$.
- 5: **if** $\Gamma(V) \geq \epsilon \gamma / 24$ **then**
 - 6: **for** all $i \in \mathcal{Viol}(V)$, (implies $k_i^* \geq 0$) set $x_{i(\ell)} = \begin{cases} \gamma \hat{w}_{k_i^*} / \Gamma(V) & \ell \in \mathcal{Pos}(i), \ell > k_i^* \\ \gamma \hat{w}_\ell / \Gamma(V) & \ell \in \mathcal{Pos}(i), \ell \leq k_i^* \end{cases}$.
 - 7: Set $x_{i(\ell)} = 0$ for $i \notin \mathcal{Viol}(V)$ or $\ell \notin \mathcal{Pos}(i)$, all $z_{U, \ell} = 0$ and **return**.
 - 8: **end if**
 - 9: $\tilde{\zeta}_{ik} \leftarrow \begin{cases} \frac{1}{2\varrho} \sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s & \text{if } i \in \mathcal{Viol}(V), k \leq k_i^*, k \in \mathcal{Pos}(i) \\ \zeta_{ik} & \text{otherwise} \end{cases}$.
 - 10: Let $\gamma' = \sum_k \hat{w}_k \left(\sum_{(i,j) \in \hat{E}_k} u_{ijk}^s - 3\varrho \sum_i \tilde{\zeta}_{ik} \right)$.
 - 11: **for** level $\ell = L$ **downto** 0 **do**
 - 12: Find a collection of sets $\mathcal{K}(\ell)$ using Lemma 5.2 such that any two sets in $\mathcal{K}(\ell)$ are disjoint and for all $U \in \mathcal{K}(\ell)$:
$$\sum_{k \geq \ell} \left(\sum_{(i,j) \in \hat{E}_k: i, j \in U} u_{ijk}^s - \sum_{i \in U} \varrho \tilde{\zeta}_{ik} \right) \geq \frac{\gamma \lfloor \|U\|_b / 2 \rfloor}{(1 - \epsilon/4)\beta}, \quad (3)$$

and for all $U' \cap \left(\cup_{U \in \mathcal{K}(\ell)} U \right) = \emptyset$,

$$\sum_{k \geq \ell} \left(\sum_{(i,j) \in \hat{E}_k: i, j \in U'} u_{ijk}^s - \sum_{i \in U'} \varrho \tilde{\zeta}_{ik} \right) \leq \gamma \frac{\lfloor \|U'\|_b / 2 \rfloor + \frac{\epsilon}{2}}{(1 - \epsilon/4)\beta}. \quad (4)$$
 - 13: For $U \in \mathcal{K}(\ell)$ define $\Delta(U, \ell) = \sum_{k \geq \ell} \left(\sum_{(i,j) \in \hat{E}_k, i, j \in U} u_{ijk}^s - \varrho \sum_{i \in U} \tilde{\zeta}_{ik} \right)$.
 - 14: **end for**
 - 15: Define $\Gamma(\mathcal{O}_s) = \sum_\ell \sum_{U \in \mathcal{K}(\ell)} \Delta(U, \ell) \hat{w}_\ell$.
 - 16: **if** $\Gamma(\mathcal{O}_s) \geq \epsilon \gamma' / 24$ (Note use of γ' .) **then**
 - 17: For $U \in \mathcal{K}(\ell)$ set $z_{U, \ell} = \gamma' \hat{w}_\ell / \Gamma(\mathcal{O}_s)$ otherwise $z_{U, \ell} = 0$. (Note use of γ' .)
 - 18: Set $z_U = \max_\ell z_{U, \ell}$ for all U . All $x_{i(\ell)}$ are set to 0 and **return**.
 - 19: **end if**
 - 20: For $i \in U \in \mathcal{K}(\ell)$, set $\hat{\zeta}_{i\ell} \leftarrow \tilde{\zeta}_{i\ell} + \frac{b_i \gamma}{2\varrho \beta} \Delta(U, \ell)$, otherwise $\hat{\zeta}_{i\ell} \leftarrow \tilde{\zeta}_{i\ell}$.
 - 21: Set $y_{ij} \leftarrow \frac{(1-\epsilon/4)\beta}{(1+\epsilon/2)\gamma} u_{ijk}^s$. Set $\mu_{ik} \leftarrow \frac{(1-\epsilon/4)\beta \varrho}{(1+\epsilon/2)\gamma} \hat{\zeta}_{ik}$. **return** $\{y_{ij}\}, \{\mu_{ik}\}$.

LEMMA 5.3 (THE MICROORACLE). Suppose we are given nonnegative $\{u_{ijk}^s\}, \{\zeta_{ik}\}, \beta$, and $\epsilon \in (0, \frac{1}{16})$, $\varrho > 0$, such that u_{ijk}^s corresponds to an edge (i, j) and $k \geq 0$. Suppose further that there exists at most one k such that $u_{ijk}^s \neq 0$. Let $E'_k = \{(i, j) | u_{ijk}^s \neq 0\}$ and $E' = \cup_k E'_k$. Then, using time $O(|E'| \text{poly}(\epsilon^{-1}, \log n))$, Algorithm 6 provides (i) a dual feasible solution to the system LP13 that demonstrates a subgraph containing a large matching or (ii) a solution to the system LP14 along with

$\mathbb{G}(\mathbf{u}^s, \mathbf{x})$, where $\hat{w}_k = (1 + \epsilon)^k$.

$$\begin{aligned}
& \sum_{i,k} x_{i(k)} \left(\sum_{j:(i,j) \in E'_k} u_{ijk}^s - 2\varrho \zeta_{ik} \right) + \sum_{U \in \mathcal{O}_s, \ell} z_{U, \ell} \left(\sum_{k \geq \ell} \left(\sum_{(i,j) \in E'_k, i, j \in U} u_{ijk}^s - \varrho \sum_{i \in U} \zeta_{ik} \right) \right) \\
& \geq \left(1 - \frac{\epsilon}{16}\right) \sum_k \hat{w}_k \left(\sum_{(i,j) \in E'_k} u_{ijk}^s - 3\varrho \sum_i \zeta_{ik} \right) \\
& z_U \sum_{k \geq \ell} \left(\sum_{(i,j) \in \hat{E}_k, i, j \in U} u_{ijk}^s - \sum_{i \in U} \left(\sum_{j \notin U, (i,j) \in \hat{E}_k} u_{ijk}^s \right) \right) \geq 0 \quad \forall U \in \mathcal{O}_s, \ell \\
& \sum_i b_i x_i + \sum_{\ell, U \in \mathcal{O}_s} z_{U, \ell} \lfloor \|U\|_b / 2 \rfloor \leq \beta \quad (\text{LP14}) \\
& x_{i(k)} + \sum_{U \in \mathcal{O}_s: i \in U} \sum_{\ell \leq k} z_{U, \ell} \leq \left(\frac{24}{\epsilon} + \frac{24}{\epsilon^2} \right) \hat{w}_k \quad \forall i, k \\
& x_i - x_{i(\ell)} \quad \forall i, \ell \\
& x_{i(k)}, x_i, z_{U, k} \geq 0 \quad \forall i, U, k
\end{aligned}$$

Furthermore, $\{U | z_{U, \ell} > 0\}$ are disjoint for any fixed ℓ , which shows $\tilde{n}_x = O(nL)$.

PROOF. Observe that the first few steps of Algorithm 6 are identical to Algorithm 5. Therefore, we can use the entire proof of Lemma 4.5 and observe that at Line 10, for any i and any set S of indices,

$$\frac{\gamma b_i}{\beta} \geq \sum_{k \in S} \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\varrho \tilde{\zeta}_{ik} \right). \quad (5)$$

The key idea is that we consider introducing the edges corresponding to the largest weight class first. If the total violation was larger than ϵ times the target solution, then at class ℓ we would assign a value of at most $\hat{w}_\ell / \epsilon$ to that constraint in the multiplicative weight update. Suppose that we ensure that each edge participates in at most one violated odd set per weight class. Then, even if an edge participated in multiple violated constraints, we can upper bound the total value assigned to a constraint in the multiplicative update corresponding to an edge to be its weight times $O(1/\epsilon^2)$ using a standard geometric series argument. This bounds the width parameter. On the other hand, if the sum of the violations were small and we wished to provide a feasible solution, we can adjust the variables ζ and charge the adjustment to the amount of violation. Since the amount of violation was small, our final solution changes only by a small amount in the objective function.

Observe that the **return** statement in Step (18) in Algorithm 6 satisfies

$$\begin{aligned}
& \sum_{\ell} \sum_{U \in \mathcal{O}_s} z_{U, \ell} \left(\sum_{k \geq \ell} \left(\sum_{(i,j) \in \hat{E}_k, i, j \in U} u_{ijk}^s - \varrho \sum_{i \in U} \zeta_{ik} \right) \right) \\
& \geq \sum_{\ell} \sum_{U \in \mathcal{O}_s} z_{U, \ell} \left(\sum_{k \geq \ell} \left(\sum_{(i,j) \in \hat{E}_k, i, j \in U} u_{ijk}^s - \varrho \sum_{i \in U} \tilde{\zeta}_{ik} \right) \right) \quad (\text{since } \zeta_{ik} \leq \tilde{\zeta}_{ik} \text{ for all } i, k.) \\
& = \sum_{\ell} \sum_{U \in \mathcal{K}(\ell)} \frac{\hat{w}_\ell \gamma'}{\Gamma(\mathcal{O}_s)} \left(\sum_{k \geq \ell} \left(\sum_{(i,j) \in \hat{E}_k, i, j \in U} u_{ijk}^s - \varrho \sum_{i \in U} \tilde{\zeta}_{ik} \right) \right) \quad (\text{Definition of } z_{U, \ell}, \text{ Step (17)}) \\
& = \frac{\gamma'}{\Gamma(\mathcal{O}_s)} \sum_{\ell} \hat{w}_\ell \sum_{U \in \mathcal{K}(\ell)} \Delta(U, \ell) = \frac{\gamma'}{\Gamma(\mathcal{O}_s)} \Gamma(\mathcal{O}_s) \geq \left(1 - \frac{\epsilon}{16}\right) \gamma \quad (\text{Using Steps (10), (15)})
\end{aligned}$$

$$\begin{aligned}
\sum_{\ell, U \in \mathcal{O}_s} z_{U, \ell} \lfloor \|U\|_b / 2 \rfloor &= \sum_{\ell} \sum_{U \in \mathcal{K}(\ell)} \frac{\hat{w}_{\ell} \gamma'}{\Gamma} \lfloor \|U\|_b / 2 \rfloor \\
&\leq \sum_{\ell} \sum_{U \in \mathcal{K}(\ell)} \frac{\hat{w}_{\ell} \gamma'}{\Gamma(\mathcal{O}_s)} \frac{\beta}{\gamma} \sum_{k \geq \ell} \left(\sum_{(i, j) \in \hat{E}_k: i, j \in U} u_{ijk}^s - \varrho \bar{\zeta}_{ik} \right) \quad (\text{Due to Equation (3)}) \\
&\leq \frac{\beta}{\Gamma(\mathcal{O}_s)} \sum_{\ell} \hat{w}_{\ell} \sum_{U \in \mathcal{K}(\ell)} \Delta(U, \ell) = \frac{\beta}{\Gamma(\mathcal{O}_s)} \Gamma(\mathcal{O}_s) = \beta \quad (\gamma' \leq \gamma, \text{Definition of } \Delta(U, \ell), \Gamma(\mathcal{O}_s)).
\end{aligned}$$

Again, observe that $z_{U, \ell} \leq 24\hat{w}_{\ell}/\epsilon$ and that the $\mathcal{K}(\ell)$ are mutually disjoint. Therefore, for any fixed i , $\sum_{U: i \in U} z_{U, \ell} \leq 24\hat{w}_{\ell}/\epsilon$ as well. Thus, for any i ,

$$\sum_{\ell \leq k} \left(\sum_{U: i \in U} z_{U, \ell} \right) \leq \frac{24}{\epsilon} \hat{w}_k \left(1 + \frac{1}{1 + \epsilon} + \frac{1}{(1 + \epsilon)^2} \cdots \right) \leq \frac{24}{\epsilon^2} \hat{w}_k.$$

Finally, based on simple accounting of the edges in a cut,

$$2 \sum_{(i, j) \in \hat{E}_k: i, j \in U} u_{ijk}^s + \sum_{i \in U} \left(\sum_{j \notin U, (i, j) \in \hat{E}_k} u_{ijk}^s \right) = \sum_{i \in U} \left(\sum_{j: (i, j) \in \hat{E}_k} u_{ijk}^s \right) \quad \forall U, k,$$

which implies that

$$\sum_{k \geq \ell} \left(2 \sum_{(i, j) \in \hat{E}_k: i, j \in U} u_{ijk}^s + \sum_{i \in U} \left(\sum_{j \notin U, (i, j) \in \hat{E}_k} u_{ijk}^s \right) \right) = \sum_{k \geq \ell} \left(\sum_{i \in U} \left(\sum_{j: (i, j) \in \hat{E}_k} u_{ijk}^s \right) \right). \quad (6)$$

Therefore, if U, ℓ were to violate $\mathbb{G}(\mathbf{u}^s, \mathbf{x})$, then

$$\sum_{k \geq \ell} \left(\sum_{(i, j) \in \hat{E}_k: i, j \in U} u_{ijk}^s - \sum_{i \in U} \left(\sum_{j \notin U, (i, j) \in \hat{E}_k} u_{ijk}^s \right) \right) < 0,$$

which along with Equation (6) implies

$$3 \sum_{k \geq \ell} \left(\sum_{(i, j) \in \hat{E}_k: i, j \in U} u_{ijk}^s \right) < \sum_{k \geq \ell} \left(\sum_{i \in U} \left(\sum_{j: (i, j) \in \hat{E}_k} u_{ijk}^s \right) \right). \quad (7)$$

But from Equation (3), if $U \in \mathcal{K}(\ell)$ (which must happen if $z_{U, \ell} > 0$), then, since $\|U\|_b \geq 3$,

$$\begin{aligned}
\sum_{k \geq \ell} \left(\sum_{(i, j) \in \hat{E}_k: i, j \in U} u_{ijk}^s - \sum_{i \in U} \varrho \bar{\zeta}_{ik} \right) &\geq \frac{\gamma}{\beta} \left\lfloor \frac{\|U\|_b}{2} \right\rfloor \geq \frac{\gamma}{\beta} \frac{\|U\|_b}{3} = \frac{1}{3} \sum_{i \in U} \frac{\gamma b_i}{\beta} \\
&\geq \frac{1}{3} \sum_{i \in U} \sum_{k \geq \ell} \left(\sum_{j: (i, j) \in \hat{E}_k} u_{ijk}^s - 2\varrho \bar{\zeta}_{ik} \right) \quad (\text{Using Equation (5)}).
\end{aligned} \quad (8)$$

But since $\bar{\zeta}_{ik} \geq 0$, we have

$$\begin{aligned}
3 \sum_{k \geq \ell} \left(\sum_{(i, j) \in \hat{E}_k: i, j \in U} u_{ijk}^s \right) - 2 \sum_{k \geq \ell} \left(\sum_{i \in U} \varrho \bar{\zeta}_{ik} \right) &\geq 3 \sum_{k \geq \ell} \left(\sum_{(i, j) \in \hat{E}_k: i, j \in U} u_{ijk}^s - \sum_{i \in U} \varrho \bar{\zeta}_{ik} \right) \\
&\geq \sum_{i \in U} \sum_{k \geq \ell} \left(\sum_{j: (i, j) \in \hat{E}_k} u_{ijk}^s - 2\varrho \bar{\zeta}_{ik} \right) \quad (\text{Using Equation (8)}),
\end{aligned}$$

which contradicts Equation (7). Therefore, for all $z_{U,\ell} > 0$, the constraints $\mathbb{G}(\mathbf{u}^s, \mathbf{x})$ hold. We satisfy Lemma 5.3 in this case as well.

In the remainder of the proof, we assume $\Gamma(O_s) \leq \epsilon^2 \gamma / 64$ and show that the constraints of LP15 are satisfied.

$$\begin{aligned}
 \sum_k \hat{w}_k \left(\sum_{(i,j) \in E'_k} y_{ij} - 3 \sum_i \mu_{ik} \right) &\geq (1 - \epsilon) \beta & (\text{LP15}) \\
 \sum_{j:(i,j) \in \hat{E}'_k} (y_{ij} - 2\mu_{ik}) &\leq y_{i(k)} & \forall i, k \\
 \sum_k y_{i(k)} &\leq b_i & \forall i \\
 \sum_{k \geq \ell} \left(\sum_{(i,j) \in \hat{E}_k: i, j \in U} y_{ij} - \sum_{i \in U} \mu_{ik} \right) &\leq \left\lfloor \frac{\|U\|_b}{2} \right\rfloor & \forall U \in O_s, \forall \ell \\
 y_{ij}, y_{i(k)}, \mu_{ik} &\geq 0 & \forall (i,j) \in \hat{E}, i, k
 \end{aligned}$$

Observe that due to Equation (5), even if we increase $\tilde{\zeta}_{ik}$ to $\hat{\zeta}_{ik}$, we continue to satisfy

$$\sum_{k \in S} \left(\sum_{j:(i,j) \in \hat{E}_k} y_{ij} - 2\mu_{ik} \right) \leq b_i. \quad (9)$$

We set $y_{(k)} = \max\{0, \sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\varrho \hat{\zeta}_{ik}\}$. Setting $S = \{k : y_{i(k)} > 0\}$ in Equation (9) satisfies the constraint $\sum_k y_{i(k)} \leq b_i$ for vertex i in LP15. Moreover, for $i \in (\cup_{U \in \mathcal{K}(\ell)} U)$, $\hat{\zeta}_{i\ell}$ increased from $\tilde{\zeta}_{i\ell}$ by $\frac{\gamma b_i}{2\varrho\beta}$ —using the observation in Step (9) of Algorithm 6 for $i \in (\cup_{U \in \mathcal{K}(\ell)} U)$ we have

$$\sum_{k \geq \ell} \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s - 2\varrho \hat{\zeta}_{ik} \right) \leq 0 \quad \Rightarrow \quad \sum_{k \geq \ell} \left(\sum_{j:(i,j) \in \hat{E}_k} y_{ij} - 2\mu_{ik} \right) \leq 0, \quad (10)$$

which as we will shortly see corresponds to the vertices having no effect on feasibility of LP15.

For an $U' \in O_s$ and ℓ , if $U' \cap (\cup_{U \in \mathcal{K}(\ell)} U) = \emptyset$, then, from Equation (4):

$$(1 + \epsilon/2) \left(\sum_{k \geq \ell} \left(\sum_{(i,j) \in \hat{E}_k: i, j \in U'} y_{ij} - \sum_{i \in U'} \mu_{ik} \right) \right) \leq \lfloor \|U'\|_b / 2 \rfloor + \frac{\epsilon}{2},$$

which implies that the constraint corresponding to U', ℓ is satisfied in LP15. If, on the other hand, $U' \cap (\cup_{U \in \mathcal{K}(\ell)} U) = U'' \neq \emptyset$, then

$$\begin{aligned}
 \sum_{k \geq \ell} \left(\sum_{(i,j) \in \hat{E}_k: i, j \in U'} y_{ij} - \sum_{i \in U'} \mu_{ik} \right) &= \sum_{k \geq \ell} \left(\frac{1}{2} \sum_{i \in U'} \left(\sum_{j \in U', (i,j) \in \hat{E}_k} y_{ij} \right) - \sum_{i \in U'} \mu_{ik} \right) \\
 &= \frac{1}{2} \sum_{i \in U'} \left(\sum_{k \geq \ell} \left(\sum_{j \in U', (i,j) \in \hat{E}_k} y_{ij} - 2\mu_{ik} \right) \right) \leq \frac{1}{2} \sum_{i \in U'} \left(\sum_{k \geq \ell} \left(\sum_{j:(i,j) \in \hat{E}_k} y_{ij} - 2\mu_{ik} \right) \right),
 \end{aligned}$$

which, using Equations (10) and (9), implies that

$$\begin{aligned} \sum_{k \geq \ell} \left(\sum_{(i,j) \in \hat{E}_k: i,j \in U'} y_{ij} - \sum_{i \in U'} \mu_{ik} \right) &\leq \frac{1}{2} \sum_{i \in U' - U''} \left(\sum_{k \geq \ell} \left(\sum_{j: (i,j) \in \hat{E}_k} y_{ij} - 2\mu_{ik} \right) \right) \\ &\leq \frac{1}{2} \sum_{i \in U' - U''} b_i = \frac{\|U\|_b - \|U''\|_b}{2} \leq \frac{\|U\|_b - 1}{2} = \left\lfloor \frac{\|U\|_b}{2} \right\rfloor, \end{aligned}$$

which implies that the constraint corresponding to U', ℓ is also satisfied in LP15. We now focus on the last remaining constraint of LP15:

$$\begin{aligned} \sum_k \hat{w}_k \left(\sum_{(i,j) \in \hat{E}_k} u_{ijk}^s - 3\varrho \sum_i \hat{\zeta}_{ik} \right) &= \gamma' - 3 \sum_{\ell} \hat{w}_{\ell} \sum_{U \in \mathcal{K}(\ell)} \sum_{i \in U} \frac{\gamma b_i}{2\beta} = \gamma' - 3 \sum_{\ell} \hat{w}_{\ell} \sum_{U \in \mathcal{K}(\ell)} \frac{\gamma \|U\|_b}{2\beta} \\ &\geq \gamma' - 3 \sum_{\ell} \hat{w}_{\ell} \sum_{U \in \mathcal{K}(\ell)} \frac{3\gamma \lfloor \|U\|_b/2 \rfloor}{2\beta} \quad (\text{Using } \lfloor \|U\|_b/2 \rfloor \geq \frac{1}{3} \|U\|_b) \\ &\geq \gamma' - 3 \sum_{\ell} \hat{w}_{\ell} \sum_{U \in \mathcal{K}(\ell)} \frac{3}{2} \left(\sum_{k \geq \ell} \left(\sum_{(i,j) \in \hat{E}_k: i,j \in U} u_{ijk}^s - \sum_{i \in U} \varrho \hat{\zeta}_{ik} \right) \right) \quad (\text{Using Equation (3)}) \\ &= \gamma' - 3 \sum_{\ell} \hat{w}_{\ell} \sum_{U \in \mathcal{K}(\ell)} \frac{3}{2} \Delta(U, \ell) \quad (\text{Using Definition of } \Delta(U, \ell)) \\ &\geq \gamma' - \frac{9}{2} \Gamma(O_s) \geq \gamma' - 3\epsilon\gamma/16 \geq (1 - \epsilon/4)\gamma. \end{aligned}$$

Therefore,

$$\sum_k \hat{w}_k \left(\sum_{(i,j) \in \hat{E}_k} y_{ij} - 3 \sum_i \mu_{ik} \right) \geq (1 - \epsilon/4)\gamma \frac{(1 - \epsilon/4)\beta}{(1 + \epsilon/2)\gamma} \geq (1 - \epsilon)\beta.$$

The lemma follows. $\square$

5.1 Proof of Lemma 5.1

The goal of the lemma is to prove that computing with sparsifiers was sufficient from the perspective of the multiplicative weights update algorithm. We provide a self-contained description that implies Lemma 5.1.

LEMMA 5.4. *Suppose we are given a graph $G = (V, E)$ where the weight of every edge (i, j) is discretized as $\hat{w}_{ij} = \hat{w}_k = (1 + \epsilon)^k$ for some $k \geq 0$. Let $E_k = \{(i, j) | \hat{w}_{ij} = \hat{w}_k\}$. Define $\text{Set}(\ell) = \{k | k \geq \ell\}$. Given nonnegative $\mathbf{x} = \{x_{i(k)}\}_{\{z_{U, \ell}\}}$, $\mathbf{u} = \{u_{ijk}\}$, define*

$$F(\mathbf{x}, \mathbf{u}) = \sum_{i, \ell} x_{i(\ell)} \left(\sum_{k: k \in \text{Set}(\ell)} \left(\sum_{j: (i,j) \in E_k} u_{ijk} \right) \right) + \sum_{U \in O_s, \ell} z_{U, \ell} \left(\sum_{k: k \in \text{Set}(\ell)} \left(\sum_{(i,j) \in E_k, i, j \in U} u_{ijk} \right) \right).$$

Further, suppose the following:

- (a) Given any (i, j) , the variables u_{ijk}, u_{ijk}^s are only nonzero for k such that $\hat{w}_k = \hat{w}_{ij}$.
- (b) For each $k \geq 0$, we have $H_k = (V, E'_k, u^s)$ as a $(\epsilon/16)$ -Cut-Sparsifier for $G_k = (V, \hat{E}_k, u)$, where $(i, j) \in E'_k$ only if $(i, j) \in \hat{E}_k$. Assume $u_{ijk}^s = 0$ for $(i, j) \in \hat{E}_k - E'_k$.

(c) Let the constraint $\mathbb{G}(\mathbf{u}^s, \mathbf{x})$ indicate the property that for any $z_{U,\ell} > 0$, the $\mathbf{u}^s$ satisfies

$$\sum_{k \in \text{Set}(\ell)} \left(\sum_{(i,j) \in \hat{E}_k, i,j \in U} u_{ijk}^s \right) \geq \sum_{k \in \text{Set}(\ell)} \left(\sum_{i \in U} \sum_{j \notin U, (i,j) \in \hat{E}_k} u_{ijk}^s \right).$$

Then, for all nonnegative $\mathbf{x} = \{x_{i(k)}\}, \{z_{U,\ell}\}$,

$$F(\mathbf{x}, \mathbf{u}^s) \geq \left(1 - \frac{\epsilon}{8}\right) \sum_k \sum_{(i,j) \in \hat{E}_k} \hat{w}_k u_{ijk}^s \Rightarrow F(\mathbf{x}, \mathbf{u}) \geq \left(1 - \frac{\epsilon}{2}\right) \sum_k \sum_{(i,j) \in \hat{E}_k} w_k u_{ijk}.$$

PROOF. For a cut defined by the single vertex i using the sparsifier H_k , we have

$$\left(1 - \frac{\epsilon}{16}\right) \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s \right) \leq \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk} \right) \leq \left(1 + \frac{\epsilon}{16}\right) \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s \right). \quad (11)$$

Multiplying this by $\hat{w}_k > 0$ and summing over k we have

$$\left(1 - \frac{\epsilon}{16}\right) \sum_k \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s \hat{w}_k \right) \leq \sum_k \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk} \hat{w}_k \right) \leq \left(1 + \frac{\epsilon}{16}\right) \sum_k \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s \hat{w}_k \right). \quad (12)$$

Therefore, for $\epsilon \leq 1$, using Equation (12),

$$\begin{aligned} \sum_k \left(\sum_{(i,j) \in \hat{E}_k} \hat{w}_k u_{ijk}^s \right) &= \frac{1}{2} \sum_k \sum_i \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s \hat{w}_k \right) \geq \frac{1}{2} \sum_i \frac{1}{1 + \frac{\epsilon}{16}} \sum_k \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s \hat{w}_k \right) \\ &\geq \left(1 - \frac{\epsilon}{8}\right) \frac{1}{2} \sum_i \sum_k \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk} \hat{w}_k \right) = \left(1 - \frac{\epsilon}{8}\right) \sum_k \left(\sum_{(i,j) \in \hat{E}_k} \hat{w}_k u_{ijk} \right). \end{aligned} \quad (13)$$

Define $\text{Cut}(U, u^s, \text{Set}(\ell)) = \sum_{k \in \text{Set}(\ell)} \sum_{i \in U} \sum_{j \notin U, (i,j) \in \hat{E}_k} u_{ijk}^s$. Now observe that for any U, ℓ with $z_{U,\ell} > 0$,

$$\sum_{k \in \text{Set}(\ell)} \left(\sum_{(i,j) \in \hat{E}_k, i,j \in U} u_{ijk} \right) = \frac{1}{2} \left[\sum_{i \in U} \sum_k \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk} \right) - \text{Cut}(U, u, \text{Set}(\ell)) \right] \quad (14)$$

$$\sum_{k \in \text{Set}(\ell)} \left(\sum_{(i,j) \in \hat{E}_k, i,j \in U} u_{ijk}^s \right) = \frac{1}{2} \left[\sum_{i \in U} \sum_k \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s \right) - \text{Cut}(U, u^s, \text{Set}(\ell)) \right]. \quad (15)$$

Moreover, $(1 - \frac{\epsilon}{16})\text{Cut}(U, u^s, \text{Set}(\ell)) \leq \text{Cut}(U, u, \text{Set}(\ell)) \leq (1 + \frac{\epsilon}{16})\text{Cut}(U, u^s, \text{Set}(\ell))$; thus, from Equations (14) and (second part of) (11),

$$\begin{aligned}
\sum_{k \in \text{Set}(\ell)} \left(\sum_{(i,j) \in \hat{E}_k, i,j \in U} u_{ijk} \right) &= \frac{1}{2} \left[\sum_{i \in U} \sum_k \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk} \right) - \text{Cut}(U, u, \text{Set}(\ell)) \right] \\
&\geq \frac{1}{2} \sum_{i \in U} \left(1 - \frac{\epsilon}{16} \right) \sum_k \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s \right) - \frac{1}{2} \left(1 + \frac{\epsilon}{16} \right) \text{Cut}(U, u^s, \text{Set}(\ell)) \\
&\geq \left(1 - \frac{\epsilon}{16} \right) \left[\sum_{i \in U} \sum_k \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s \right) - \text{Cut}(U, u^s, \text{Set}(\ell)) \right] - \frac{\epsilon}{16} \text{Cut}(U, u^s, \text{Set}(\ell)) \\
&= \left(1 - \frac{\epsilon}{16} \right) \sum_{k \in \text{Set}(\ell)} \left(\sum_{(i,j) \in \hat{E}_k, i,j \in U} u_{ijk}^s \right) - \frac{\epsilon}{16} \text{Cut}(U, u^s, \text{Set}(\ell)) \quad (\text{Equation (15)}) \\
&\geq \left(1 - \frac{\epsilon}{8} \right) \sum_{k \in \text{Set}(\ell)} \left(\sum_{(i,j) \in \hat{E}_k, i,j \in U} u_{ijk}^s \right) \quad (\text{Given condition (c)}).
\end{aligned}$$

Therefore, we have

$$\sum_{U \in \mathcal{O}_s, \ell} z_{U, \ell} \left(\sum_{k \in \text{Set}(\ell)} \left(\sum_{(i,j) \in \hat{E}_k, i,j \in U} u_{ijk} \right) \right) \geq \left(1 - \frac{\epsilon}{8} \right) \sum_{U \in \mathcal{O}_s, \ell} z_{U, \ell} \left(\sum_{k \in \text{Set}(\ell)} \left(\sum_{(i,j) \in \hat{E}_k, i,j \in U} u_{ijk}^s \right) \right). \quad (16)$$

Using the sparsifier H_k for each k at cuts defined by the single vertices, for $k \in \text{Set}(\ell)$ and $x_{i(\ell)} \geq 0$,

$$\sum_{k \in \text{Set}(\ell)} \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk} \right) \geq \left(1 - \frac{\epsilon}{16} \right) \sum_{k \in \text{Set}(\ell)} \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s \right) \geq \left(1 - \frac{\epsilon}{8} \right) \sum_{k \in \text{Set}(\ell)} \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s \right). \quad (17)$$

Equation (17) multiplied by $x_{i(\ell)} \geq 0$ and summed over different i, ℓ gives

$$\sum_{i, \ell} x_{i(\ell)} \sum_{k \in \text{Set}(\ell)} \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk} \right) \geq \left(1 - \frac{\epsilon}{8} \right) \sum_{i, \ell} x_{i(\ell)} \sum_{k \in \text{Set}(\ell)} \left(\sum_{j:(i,j) \in \hat{E}_k} u_{ijk}^s \right),$$

which, when added to Equation (16), gives $F(\mathbf{x}, \mathbf{u}) \geq (1 - \frac{\epsilon}{8})F(\mathbf{x}, \mathbf{u}^s)$. But then,

$$\begin{aligned}
F(\mathbf{x}, \mathbf{u}) &\geq \left(1 - \frac{\epsilon}{8} \right) F(\mathbf{x}, \mathbf{u}^s) \geq \left(1 - \frac{\epsilon}{8} \right)^2 \sum_k \left(\sum_{(i,j) \in \hat{E}_k} \hat{w}_k u_{ijk}^s \right) \quad (\text{From statement of lemma}) \\
&\geq \left(1 - \frac{\epsilon}{8} \right)^3 \sum_k \left(\sum_{(i,j) \in \hat{E}_k} \hat{w}_k u_{ijk} \right) \quad (\text{Using Equation (13)}).
\end{aligned}$$

The lemma follows. $\square$

5.2 Proof of Lemma 5.2

The goal of the lemma is to find a disjoint collection of odd sets such that if we restrict the attention to the weight classes ℓ or higher and use the multipliers provided by the dual-primal framework as a fractional assignment of edges, we would have violated that odd-set constraint.

LEMMA 5.2. *In time $O(n \text{poly}(\log n, \frac{1}{\epsilon}))$, we can find a collection $\mathcal{K}(\ell)$ such that every pair of sets is mutually disjoint and Equations (3) and (4) hold in Step 12.*

PROOF. We first consider a graph problem where we collect a maximal collection of dense small odd sets. This abstraction will also be used later for the case when $w_{ij} \neq 1$. Note we will be upper bounding $n \text{poly}(\kappa, \log n)$ by $n^{1+1/p}$ asymptotically. We first prove:

LEMMA 5.5. *Given $G = (V, E)$, nonnegative numbers q_{ij} on edges and $\hat{q}_i$ on nodes, such that*

(A1) *If $b_i = 1$, then $\hat{q}_i \geq C$.*

(A2) *$\sum_j q_{ij} \leq \hat{q}_i$ for all i .*

(A3) *Any odd set U with $\|U\|_b > 4/\epsilon$ satisfies $\sum_{i \in U} \left(\hat{q}_i - \sum_j q_{ij} \right) > 1$*

using space $O(n\epsilon^{-2})$ and $O(m) + O(n \text{poly}(\epsilon^{-1}, \log n))$ time and a single pass over the list of edges we can find a collection $\mathcal{L}$ of mutually disjoint sets such that

(i) *Every odd set $U \in \mathcal{L}$ satisfies $\sum_{(i,j): i,j \in U} q_{ij} \geq \frac{1}{2} \left(\sum_{i \in U} \hat{q}_i - 1 \right)$.*

(ii) *Every odd set $U \notin \mathcal{L}$ either intersects with a set in $\mathcal{L}$ or satisfies $\sum_{(i,j): i,j \in U} q_{ij} \leq \frac{1}{2} \left(\sum_{i \in U} \hat{q}_i - (1 - \epsilon) \right)$.*

PROOF. We create a graph H on the vertex set $V \cup \{s\}$, where we have $\lfloor q_{ij} 8\epsilon^{-3} \rfloor$ parallel edges between i and j . After the edges have been added, we add edges between i and s till the degree of s is $\lceil \hat{q}_i 8\epsilon^{-3} \rceil$ —this is feasible due to (A2). We use a lemma from Ahn and Guha (2014), which combines the minimum odd-cut approach in Padberg and Rao (1982) with the construction of approximate Gomory-Hu Trees (Hariharan et al. 2007; Bhalgat et al. 2007).

LEMMA 5.6 (AHN AND GUHA 2014, LEMMA 12). *Given an unweighted graph G with parameter κ and a special node s , in time $O(n \text{poly}(\kappa, \log n))$, we can identify a collection $\mathcal{L}$ of mutually disjoint odd-sets that (i) do not contain s , (ii) define cut of at most κ in G , and (iii) every other odd set not containing s and with a cut less than κ intersects with a set in $\mathcal{L}$.*

We now apply Lemma 5.6 with $\kappa = \lceil 8\epsilon^{-3} \rceil$. Note that any set that is returned in $\mathcal{L}$ satisfies condition (i) easily since

$$\sum_{(i,j): i,j \in U} \lfloor q_{ij} 8\epsilon^{-3} \rfloor \geq \frac{1}{2} \left(\sum_{i \in U} \lceil \hat{q}_i 8\epsilon^{-3} \rceil - \lceil 8\epsilon^{-3} \rceil \right) \Rightarrow \sum_{(i,j): i,j \in U} q_{ij} \geq \frac{1}{2} \left(\sum_{i \in U} \hat{q}_i - 1 \right).$$

For any odd set that is not returned and does not intersect any of the sets returned, the cut after discretization is at least κ :

$$\begin{aligned} \sum_{(i,j):i,j \in U} \lfloor q_{ij} 8\epsilon^{-3} \rfloor &\leq \frac{1}{2} \left(\sum_{i \in U} \lceil \hat{q}_i 8\epsilon^{-3} \rceil - \lfloor 8\epsilon^{-3} \rfloor \right) \\ &\Rightarrow \sum_{(i,j):i,j \in U} q_{ij} \leq \frac{\binom{4/\epsilon}{2}}{8\epsilon^{-3}} + \frac{1}{2} \left(\sum_{i \in U} \hat{q}_i + \frac{4/\epsilon}{8\epsilon^{-3}} - 1 \right), \end{aligned}$$

which gives us condition (ii). $\square$

Observe that (A1) and (A3) imply that singleton vertices or very large sets cannot be present in $\mathcal{L}$. We are now ready to prove Lemma 5.2. The algorithm is simple:

- (1) Set $\hat{q}_i(\ell) = b_i + \frac{2(1-\epsilon/4)\varrho\beta}{\gamma} \sum_{k \geq \ell} \tilde{\zeta}_{ik}$.
- (2) For i, j consider the k such that $u_{ijk}^s \neq 0$. Set $q_{ij}(\ell) = \frac{(1-\epsilon/4)\beta}{\gamma} u_{ijk}^s$ if $k \geq \ell$ and 0 otherwise.
- (3) Let $C = 1$ and apply Lemma 5.5 and get a collection of mutually disjoint sets, which we denote by $\mathcal{K}(\ell)$.

Equation (5), proved just after the description of Algorithm 6, applied to $S = \{k | k \geq \ell\}$ (multiplied by $(1 - \epsilon/4)\beta/\gamma$), gives us

$$(1 - \epsilon/4)b_i \geq \sum_j q_{ij}(\ell) - \sum_{k \geq \ell} \frac{2(1 - \epsilon/4)\varrho\beta}{\gamma} \tilde{\zeta}_{ik} \Rightarrow \hat{q}_i(\ell) \geq \sum_j q_{ij}(\ell).$$

And for $b_i = 1$, $\hat{q}_i(\ell) \geq b_i = C$. Moreover, for any set U ,

$$\sum_{i \in U} \left(\hat{q}_i(\ell) - \sum_j \hat{q}_i(\ell) \right) = \sum_{i \in U} \left(b_i - \frac{(1 - \epsilon/4)\beta}{\gamma} \sum_{k \geq \ell} \left(\sum_{(i,j) \in \hat{E}_k} u_{ijk}^s - 2\varrho\zeta_{ik} \right) \right) \geq \sum_{i \in U} \frac{\epsilon b_i}{4} = \epsilon \|U\|_b.$$

Therefore, if $\|U\|_b > 4/\epsilon$, then $\sum_{i \in U} (\hat{q}_i(\ell) - \sum_j \hat{q}_i(\ell)) > 1$. This implies that the conditions (A1) through (A3) of Lemma 5.5 are valid and we can apply Lemma 5.5. The lemma follows.

6 PROOF OF DUAL PRIMAL FRAMEWORK: THEOREM 1.3

In this section, we prove Theorem 1.3, based on Definition 3, which we repeat for readability. We use parts of Definition 3 to cast the problem in the statement of Theorem 1.3 as a multiplicative weights update problem (Arora et al. 2012). We use the fractional packing and covering variant in Plotkin et al. (1995). The deferred sparsifiers allow us to reduce the number of iterations, as shown in Theorem 6.1. The multiplicative weight update requires a solution of a subroutine, and we solve that subroutine using another recursive application of the multiplicative weight method, as in Theorem 6.2. These two theorems, in combination, provide us Theorem 1.3.

Definition 3. The problem $\beta^* = \max \mathbf{c}^T \mathbf{y}; \mathbf{y} \in \{\mathcal{A}^T \mathbf{y} \leq \mathbf{b}, \mathbf{y} \geq \mathbf{0}\}$ is defined to be (ρ_0, ρ_i) -**Dual-Primal amenable** if there exist $\mathbf{A} \in \mathbb{R}_+^{m \times N}$, $\mathbf{b}, \mathbf{x} \in \mathbb{R}_+^N$, $\mathbf{P}_0 \in \mathbb{R}_+^{\tilde{n}_0 \times N}$, $\mathbf{q}_0 \in \mathbb{R}_+^{\tilde{n}_0}$, $\mathbf{P}_i \in \mathbb{R}_+^{\tilde{n}_i \times N}$, $\mathbf{q}_i \in \mathbb{R}_+^{\tilde{n}_i}$, convex polytope $\mathbf{Q}$ with $\mathbf{0} \in \mathbf{Q}$, and an absolute constant a_1 such that the following hold simultaneously:

- (d1) (DUAL FEASIBILITY.) A feasible solution to $\{\mathbf{b}^T \mathbf{x} \leq \beta, \mathbf{A}\mathbf{x} \geq (1 - 3\epsilon)\mathbf{c}, \mathbf{x} \in \mathbf{Q}, \mathbf{x} \geq \mathbf{0}\}$ implies $\beta^* \leq \beta/(1 - a_1\epsilon)$.
- (d2) (OUTER WIDTH.) $\{\mathbf{P}_0 \mathbf{x} \leq 2\mathbf{q}_0, \mathbf{x} \in \mathbf{Q}, \mathbf{x} \geq \mathbf{0}\}$ implies $\mathbf{A}\mathbf{x} \leq \rho_0 \mathbf{c}$.
- (d3) (INNER WIDTH.) $\{\mathbf{P}_i \mathbf{x} \leq \mathbf{q}_i, \mathbf{x} \in \mathbf{Q}, \mathbf{x} \geq \mathbf{0}\}$ implies $\mathbf{P}_0 \mathbf{x} \leq \rho_i \mathbf{q}_0$.

- (d4) (A DEFERRED $\mathbf{u}$ -SPARSIFIER.) Given $\mathbf{u}, \mathbf{v} \in \mathbb{R}_+^m$, and the promise that all nonzero $\mathbf{u}_\ell$ satisfy $1/\mathcal{L}_0 \leq \mathbf{v}_\ell/\gamma \leq \mathbf{u}_\ell \leq \gamma \mathbf{v}_\ell \leq \mathcal{L}_0$ for each $\ell \in [m]$ given fixed $\gamma, \mathcal{L}_0 \geq 1$, we can construct a data structure $\mathcal{D}$ that samples a subset of indices in $[m]$ of size $\tilde{n}_s$ based on the $\mathbf{v}$ values and stores the indices. After $\mathcal{D}$ has been constructed, the exact values of those stored entries of $\mathbf{u}$ are revealed and the data structure constructs a nonnegative vector $\mathbf{u}^s$ such that for some property $\mathbb{G}(\mathbf{u}^s, \mathbf{x})$ that is convex in $\mathbf{x}$ and $\mathbf{x} = \mathbf{0}$ satisfies $\mathbb{G}(\mathbf{u}^s, \mathbf{x})$, we have:

$$(\mathbf{u}^s)^T \mathbf{A} \mathbf{x} \geq \left(1 - \frac{\epsilon}{8}\right) (\mathbf{u}^s)^T \mathbf{c}, \quad \mathbb{G}(\mathbf{u}^s, \mathbf{x}), \mathbf{x} \geq 0 \Rightarrow \mathbf{u}^T \mathbf{A} \mathbf{x} \geq \left(1 - \frac{\epsilon}{2}\right) \mathbf{u}^T \mathbf{c} \quad (\text{SWITCH})$$

- (d5) (INITIAL SOLUTION.) We can efficiently find a solution to INITIAL for some $a_\epsilon \geq 2$:

$$\begin{cases} \mathbf{A} \mathbf{x}_0 \geq (1 - \epsilon_0) \mathbf{c} \\ \frac{\beta^*}{a_\epsilon} \leq \beta_0 = \mathbf{b}^T \mathbf{x}_0 < \frac{\beta^*}{2} \\ \mathbf{P}_o \mathbf{x}_0 \leq 2 \mathbf{q}_o \\ \mathbf{P}_i \mathbf{x}_0 \leq \mathbf{q}_i \\ \mathbf{x}_0 \in \mathbf{Q}, \mathbf{x}_0 \geq 0. \end{cases} \quad (\text{INITIAL})$$

THEOREM 1.3. Suppose for a (ρ_o, ρ_i) -dual-primal amenable system PRIMAL given any constant ϵ , $0 < \epsilon \leq 1$, $\mathbf{u}^s \in \mathbb{R}_+^m$, $\zeta \in \mathbb{R}_+^{\tilde{n}_o}$, $\rho > 0$, a MICROORACLE provides either:

- (i) A feasible solution for PRIMAL with $\mathbf{c}^T \mathbf{y} \geq (1 - a_2 \epsilon) \beta$ such that for all $\ell \in [m]$, $\mathbf{y}_\ell > 0$ implies $\mathbf{u}_\ell^s > 0$ —that is, a solution only involving the primal variables corresponding to the sampled constraints in the deferred sparsifier.
- (ii) A solution $\mathbf{x}$ of LAGINNER, where the number of nonzero entries of $\mathbf{x}$ is at most $\tilde{n}_x$.

$$\begin{aligned} (\mathbf{u}^s)^T \mathbf{A} \mathbf{x} - \rho \zeta^T \mathbf{P}_o \mathbf{x} &\geq \left(1 - \frac{1}{16}\right) (\mathbf{u}^s)^T \mathbf{c} - \rho \zeta^T \mathbf{q}_o \\ \mathbb{G}(\mathbf{u}^s, \mathbf{x}) & \\ \tilde{\mathbf{Q}}(\beta) = \{\mathbf{b}^T \mathbf{x} \leq \beta, \mathbf{P}_i \mathbf{x} \leq \mathbf{q}_i, \mathbf{x} \in \mathbf{Q}, \mathbf{x} \geq 0\} & \end{aligned} \quad (\text{LAGINNER})$$

Then we can find a $(1 - (1 + a_1 + a_2)\epsilon)$ -approximate solution to PRIMAL using τ rounds of deferred $\mathbf{u}$ -sparsifier construction, $\tau = O(\rho_o(\frac{1}{\epsilon} + \epsilon \log \frac{1}{1-\epsilon_0}) \frac{\log(m/\epsilon)}{\log \gamma} + \frac{\log a_\epsilon}{\log \gamma})$.

In each round, we construct $O(\epsilon^{-1} \log \gamma)$ deferred $\mathbf{u}$ -sparsifiers. The algorithm invokes MICROORACLE for a fixed deferred sparsifier for at most $O(\rho_i(\log \rho_i)(\log \tilde{n}_o) \log \frac{1}{\epsilon})$ times. Each nonzero $\mathbf{u}_\ell \in [(2m/\epsilon)^{-O(\frac{\rho_o}{\epsilon(1-\epsilon_0)})}, 1]$, and finally $\log \mathcal{L}_0 = O(\frac{\rho_o}{\epsilon(1-\epsilon_0)} \log(m/\epsilon) + \log \gamma)$. Moreover, the algorithm computes (approximately) $\max \lambda; \mathbf{A} \tilde{\mathbf{x}} \geq \lambda \mathbf{c}$ for a $\tilde{\mathbf{x}}$, which is a weighted average of the $\mathbf{x}$ returned by successive applications of part (ii)—the algorithm can stop earlier than the stated number of τ rounds if it observes $\lambda \geq (1 - 3\epsilon)$.

Roadmap: We prove Theorem 1.3 using Theorems 6.1 and 6.2. Intuitively, we cast the problem targeted in the statement of Theorem 1.3 as a fractional covering problem (Plotkin et al. 1995), which uses a variant of the multiplicative weight update method (see the survey of Arora et al. (2012)). Theorem 6.1 starts from a solution based on Plotkin et al. (1995) and reduces the number of rounds based on the existence of the deferred sparsifiers. Theorem 6.1 is interesting in its own right and can be used when $\mathbf{P}_o = \mathbf{P}_i, \mathbf{q}_o = \mathbf{q}_i$, which implies $\rho_i = 2$. However, computing an efficient solution of the subproblem SPARSE is nonobvious for small values of ρ_o .

THEOREM 6.1 (PROVED IN SECTION 6.1). Suppose for a (ρ_o, ρ_i) -dual-primal amenable system PRIMAL given any $\mathbf{u}^s \in \mathbb{R}_+^m$, $\mathbf{u}^s \geq 0$, a MINIORACLE provides either condition (i) as in Theorem 1.3 or a

solution $\mathbf{x}$ of SPARSE, where the number of nonzero entries of $\mathbf{x}$ is at most $\tilde{n}_x$:

$$\begin{aligned} (\mathbf{u}^s)^T \mathbf{A} \mathbf{x} &\geq (1 - \epsilon/8)(\mathbf{u}^s)^T \mathbf{c} \\ \mathbf{P}_o \mathbf{x} &\leq 2\mathbf{q}_o \\ \mathbb{G}(\mathbf{u}^s, \mathbf{x}) & \\ \tilde{\mathbf{Q}}(\beta) &= \{\mathbf{b}^T \mathbf{x} \leq \beta, \mathbf{P}_i \mathbf{x} \leq \mathbf{q}_i, \mathbf{x} \in \mathbf{Q}, \mathbf{x} \geq 0\}. \end{aligned} \tag{SPARSE}$$

Then the conclusions of Theorem 1.3 (number of rounds, deferred sparsifiers in each round, existence of early stopping certificate, and the range of values for the multipliers) hold.

Theorem 6.2 provides a surprising use of the sparsifiers, namely, that we can iterate over the small-size sparsifier without requiring fresh access to input. The algorithm that proves Theorem 6.2 recasts the subproblem SPARSE as a fractional packing problem. It applies Plotkin et al. (1995) and demonstrates that multiple solutions of LAGINNER suffice to find an efficient solution of SPARSE. The overall intuition corresponds to reducing ρ_o at the expense of increasing ρ_i . Note that ρ_i does not affect the overall number of iterations, whereas the number of rounds depends on ρ_o .

THEOREM 6.2 (PROVED IN SECTION 6.2). *We use $\tau_i = O(\rho_i(\log \rho_i)(\log \tilde{n}_o) \log \frac{1}{\epsilon})$ invocations of MICROORACLE to implement an invocation of SPARSE.*

Theorems 6.1 and 6.2 together imply Theorem 1.3.

6.1 Proof of Theorem 6.1

In this subsection, we cast the problem outlined in Definition 3 as a fractional covering problem and show that the number of iterations can be reduced using deferred sparsifiers provided we can solve the system SPARSE as a substep. We use the following result from Plotkin et al. (1995); we change the notation to fit our context—note that we start with a different initial condition than stated in Lemma 3.6 in Plotkin et al. (1995). That lemma in Plotkin et al. (1995) relied on $O(M)$ rounds of computation to produce a $\epsilon_0 = 1 - 1/M$ solution, that is, $\mathbf{A}\mathbf{x}_0 \geq \mathbf{c}/M$, where M is the number of constraints. The altered initial condition gives us Theorem 6.3.

THEOREM 6.3 (PLOTKIN ET AL. 1995). *Suppose we are given a decision problem $\mathbf{A}\mathbf{x} \geq \mathbf{c}$ such that $\mathbf{x} \in \mathcal{P}$, where $\mathbf{A} \in \mathbb{R}^{M \times N}$, $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{c} \in \mathbb{R}^M$, where $\mathcal{P}$ is some polytope such that $\mathbf{0} \leq \mathbf{A}\mathbf{x} \leq \rho\mathbf{c}$ for all $\mathbf{x} \in \mathcal{P}$. Suppose we have an initial $\mathbf{x}_0 \in \mathcal{P}$ satisfying $\mathbf{A}\mathbf{x}_0 \geq (1 - \epsilon_0)\mathbf{c}$. The algorithm sets $\mathbf{x} = \mathbf{x}_0$ and proceeds in phases. In phase t , it determines $\lambda_t = \min_{\ell} (\mathbf{A}\mathbf{x})_{\ell} / \mathbf{c}_{\ell}$. It then repeatedly queries an ORACLE-C for $\arg\max_{\tilde{\mathbf{x}} \in \mathcal{P}} \mathbf{u}^T \mathbf{A} \tilde{\mathbf{x}}$, where*

$$\mathbf{u}_{\ell} = \exp(-\alpha(\mathbf{A}\mathbf{x})_{\ell} / \mathbf{c}_{\ell}) / \mathbf{c}_{\ell} \quad \alpha = O(\lambda_t^{-1} \epsilon^{-1} \ln(M/\epsilon)),$$

and performs an update step $\mathbf{x} \leftarrow (1 - \sigma)\mathbf{x} + \sigma\tilde{\mathbf{x}}$, where $\sigma = \epsilon/(4\alpha\rho)$ (note that the failure to produce an $\tilde{\mathbf{x}}$ implies that the decision problem is infeasible). The phase continues till $\lambda = \min_{\ell} (\mathbf{A}\mathbf{x})_{\ell} / \mathbf{c}_{\ell} \geq \max\{2\lambda_t, 1 - 3\epsilon\}$. If $\lambda < 1 - 3\epsilon$, then a new phase is started till $\lambda \geq (1 - 3\epsilon)$. Then after at most $T = O(\rho(\epsilon^{-2} + \log \frac{1}{1-\epsilon_0}) \log \frac{M}{\epsilon})$ invocations of ORACLE-C, the fractional covering framework either provides (i) a solution $\mathbf{A}\mathbf{x} \geq (1 - 3\epsilon)\mathbf{c}$, $\mathbf{x} \in \mathcal{P}$ or (ii) a nonnegative vector $\mathbf{y} \in \mathbb{R}^M$ such that $\mathbf{y}^T \mathbf{A}\mathbf{x} < \mathbf{y}^T \mathbf{c}$ for all $\mathbf{x} \in \mathcal{P}$, thereby proving the infeasibility of $\{\mathbf{A}\mathbf{x} \geq \mathbf{c}, \mathbf{x} \in \mathcal{P}\}$.

The framework explicitly maintains $\lambda = \min_{\ell} (\mathbf{A}\mathbf{x})_{\ell} / \mathbf{c}_{\ell}$ and the $\mathbf{u}_{\ell}$ change by a factor of at most $e^{\pm\epsilon}$ from one invocation of ORACLE-C to the next. Each nonzero $\mathbf{u}_{\ell} \in [(2M/\epsilon)^{-O(\frac{\rho_o}{\epsilon(1-\epsilon_0)})}, 1]$.

We use a small modification that is not present in Plotkin et al. (1995).

COROLLARY 6.4. *Theorem 6.3 holds if ORACLE-C finds $\mathbf{x} \in \mathcal{P}$ satisfying $\mathbf{u}^T \mathbf{A} \tilde{\mathbf{x}} \geq (1 - \epsilon/2)\mathbf{u}^T \mathbf{c}$ or reports that no such solutions exist.*

PROOF. Note that $(\mathbf{u}^T \mathbf{A} \tilde{\mathbf{x}} \geq (1 - \epsilon/2) \mathbf{u}^T \mathbf{c})$ implies

$$\left((1 - \epsilon/2) \mathbf{u}^T \mathbf{A} \tilde{\mathbf{x}} \geq (1 + \epsilon/2) \mathbf{u}^T \mathbf{A} \mathbf{x} + \epsilon \mathbf{u}^T \mathbf{c}/2 \right).$$

Lemma 3.3 in Plotkin et al. (1995) requires the RHS of the implication to hold with $\epsilon \mathbf{u}^T \mathbf{c}$ (it uses notation $\mathbf{y}, \mathbf{b}$ instead of $\mathbf{u}, \mathbf{c}$); however, ensuring $\epsilon \mathbf{u}^T \mathbf{c}/2$ only increases the number of iterations in Theorem 6.3 by a factor of 2. Lemma 3.1 in Plotkin et al. (1995) shows $\mathbf{u}^T \mathbf{A} \mathbf{x} \leq (1 + \epsilon) \lambda \mathbf{u}^T \mathbf{c}$ and therefore $(1 + \epsilon/2) \mathbf{u}^T \mathbf{A} \mathbf{x} \leq (1 + \epsilon/2)(1 + \epsilon)(1 - 3\epsilon) \mathbf{u}^T \mathbf{b} \leq (1 - 3\epsilon/2) \mathbf{u}^T \mathbf{c}$. The implication follows using simple calculations. $\square$

PROOF (OF THEOREM 6.1). As stated in the theorem, we have an initial solution $\mathbf{x}_0$ such that $\beta^*/a_\epsilon \leq \beta_0 = \mathbf{b}^T \mathbf{x}_0 < \beta^*/2$ and $\mathbf{x} \in \mathcal{P}(\beta_0)$ along with $\mathbf{A} \mathbf{x}_0 \geq (1 - \epsilon_0) \mathbf{c}$. We consider Algorithm 7—this is **not** the final algorithm.

Based on Corollary 6.4, using $M = m$ and $m \geq 1/\epsilon$, if for some β we have $T = O(\rho_o(\epsilon^{-2} + \log \frac{1}{(1-\epsilon_0)}) \log m)$ such solutions $\mathbf{x}$, then we would have $\mathbf{A} \mathbf{x} \geq (1 - 3\epsilon) \mathbf{c}$ with $\mathbf{b}^T \mathbf{x} \leq \beta$. Since $\beta_0 \leq \beta^*/2$, it implies that at least one call to MINIORACLE will provide a $\mathbf{y}$ as in condition (i) since we are starting from $\beta = \beta^0$. Now consider the *largest* value of β , say, β' , for which any call to MINIORACLE has provided a $\mathbf{y}$ as in (i)—such a value exists because we cannot have primal feasible solutions with value more than β^* .

Since an invocation to MINIORACLE has provided *primal feasible solution* $\mathbf{y}$ for PRIMAL satisfying $\mathbf{c}^T \mathbf{y} \geq (1 - a_2 \epsilon) \beta'$. Now for the value of $(1 + \epsilon) \beta'$ we have T solutions of OUTER (irrespective of when and where we raised the β values since a solution of OUTER with $\beta = \beta''$ continues to hold for larger values of $\beta > \beta''$), and therefore $\mathbf{A} \mathbf{x} \geq (1 - 3\epsilon) \mathbf{c}$ with $\mathbf{x} \in Q, \mathbf{c}^T \mathbf{x} \leq (1 + \epsilon) \beta'$. By condition (d1), we have a proof that $(1 + \epsilon) \beta' \geq \beta^*/(1 - a_2 \epsilon)$. Therefore, we have a $\mathbf{y}$ (provided by the last invocation of MINIORACLE to return (i)) such that $\mathbf{b}^T \mathbf{y} \geq (1 - (1 + a_1 + a_2) \epsilon) \beta^*$, proving the quality of approximation of Theorem 6.1 (same as in Theorem 1.3). Note that the number of time we increased β is at most $O(\frac{1}{\epsilon} \log a_\epsilon)$. Note that $\tau = (T + O(\frac{1}{\epsilon} \log a_\epsilon))/(\epsilon^{-1} \log \gamma)$ is the number

ALGORITHM 7: An Intermediate Algorithm for Theorem 6.1

- 1: Start with $\mathbf{x} = \mathbf{x}_0$ and $\beta = \beta_0$, where $\mathbf{x}_0, \beta_0$ refer to the initial solution INITIAL.
- 2: Consider the following **family** of decision problems:

$$\begin{aligned} & \{\mathbf{A} \mathbf{x} \geq \mathbf{c}, \mathbf{x} \in \mathcal{P}(\beta)\}, \quad \text{where} \\ & \mathcal{P}(\beta) = \left\{ \begin{array}{l} \mathbf{P}_o \mathbf{x} \leq 2 \mathbf{q}_o \\ \tilde{\mathbf{Q}}(\beta) = \{\mathbf{b}^T \mathbf{x} \leq \beta; \mathbf{P}_i \mathbf{x} \leq \mathbf{q}_i; \mathbf{x} \in Q; \mathbf{x} \geq \mathbf{0}\} \end{array} \right\}. \end{aligned} \quad (\text{DUAL})$$

We will be solving the entire family simultaneously. Note that $\tilde{\mathbf{Q}}(\beta') \subseteq \tilde{\mathbf{Q}}(\beta)$ for $\beta' \leq \beta$. This is why we start from a low value of $\beta_0 \leq \beta^*/2$.

- 3: **while** $\lambda < 1 - 3\epsilon$ (where $\lambda = \min_\ell (\mathbf{A} \mathbf{x})_\ell / \mathbf{c}_\ell$) **do**
- 4: Define $\mathbf{u}$ as in Corollary 6.4 and consider the system OUTER:

$$\mathbf{u}^T \mathbf{A} \tilde{\mathbf{x}} \geq (1 - \epsilon/2) \mathbf{u}^T \mathbf{c}, \quad \tilde{\mathbf{x}} \in \mathcal{P}(\beta). \quad (\text{OUTER})$$

- 5: Set $\mathbf{u}^s = \mathbf{u}$ and invoke MINIORACLE to solve OUTER
 - (a) If the MINIORACLE returns $\tilde{\mathbf{x}}$ then update $\mathbf{x}, \mathbf{u}$ as in Corollary 6.4.
 - (b) Otherwise (if MINIORACLE returns $\mathbf{y}$) then set $\beta \leftarrow (1 + \epsilon)\beta$; remember only the last such $\mathbf{y}$; and repeat MINIORACLE with the new β .

6: **end while**

- 7: Output the last remembered $\mathbf{y}$ corresponding to $\beta' = \beta/(1 + \epsilon)$. We prove that such a β' exists.
-

of times we invoke `MINIORACLE`. We modify Algorithm 7 to Algorithm 8—observe that the new algorithm simplifies to the description of Algorithm 2 discussed in Section 1.

ALGORITHM 8: The Algorithm for Theorem 6.1

- 1: Start with $\mathbf{x} = \mathbf{x}_0$ and $\beta = \beta_0$, where $\mathbf{x}_0, \beta_0$ refer to the initial solution `INITIAL`.
- 2: Consider the following family of decision problems:

$$\begin{aligned} &\{\mathbf{Ax} \geq \mathbf{c}, \mathbf{x} \in \mathcal{P}(\beta)\}, \quad \text{where} \\ &\mathcal{P}(\beta) = \left\{ \begin{array}{l} \mathbf{P}_o \mathbf{x} \leq 2\mathbf{q}_o \\ \tilde{\mathbf{Q}}(\beta) = \{\mathbf{b}^T \mathbf{x} \leq \beta; \mathbf{P}_i \mathbf{x} \leq \mathbf{q}_i; \mathbf{x} \in \mathbf{Q}; \mathbf{x} \geq \mathbf{0}\} \end{array} \right\}. \end{aligned} \quad (\text{DUAL})$$

- 3: **while** $\lambda < 1 - 3\epsilon$ (where $\lambda = \min_{\ell} (\mathbf{Ax})_{\ell} / c_{\ell}$) **do**
 - 4: Define weights $\mathbf{u}$ as in Corollary 6.4. Compute $\frac{\ln \gamma}{\epsilon}$ deferred $\mathbf{u}$ -sparsifiers denoted by $\{\mathcal{D}_q\}$.
 - 5: **for** $q = 1$ to $\frac{\ln \gamma}{\epsilon}$ **do**
 - 6: Define `OUTER` (based on the current $\mathbf{x}, \mathbf{u}(q)$) as

$$\mathbf{u}^T(q) \mathbf{Ax} \geq (1 - \epsilon/2) \mathbf{u}^T(q) \mathbf{b} \quad \tilde{\mathbf{x}} \in \mathbf{Q}_o(\beta). \quad (\text{OUTER})$$
 - 7: Refine $\mathcal{D}_q$ based on the current $\mathbf{x}$ to produce $\mathbf{u}^s(q)$.
 - 8: Invoke `MINIORACLE` to solve `SPARSE`
 - (a) If the `MINIORACLE` returns $\tilde{\mathbf{x}}$, update $\mathbf{x}$ as in Corollary 6.4. Note `SPARSE` $\Rightarrow$ `OUTER`.
 - (b) Otherwise (if `MINIORACLE` returns $\mathbf{y}$) then set $\beta \leftarrow (1 + \epsilon)\beta$; remember only the last such $\mathbf{y}$; and repeat `MINIORACLE` with the new β .
 - 9: **end for**
 - 10: **end while**
 - 11: Output the last remembered $\mathbf{y}$ corresponding to $\beta' = \beta / (1 + \epsilon)$. We prove that such a β' exists.
-

We observe that in each of the T invocations of `MINIORACLE` that provided a $\mathbf{x}$, the $\mathbf{u}_{\ell}$ changed by a factor of $e^{\pm\epsilon}$. Therefore, if we perform $\epsilon^{-1} \log \gamma$ invocations, the values of $\mathbf{u}_{\ell}$ (for every ℓ) will change by a factor in the range $[1/\gamma, \gamma]$. We apply deferred $\mathbf{u}$ -sparsifiers. Condition (4) in Definition 3 asserts that `SPARSE` (applied to $\mathbf{u}^s$) still implies `OUTER` (applied to $\mathbf{u}$), and the previous proof of quality of approximation and number of invocations remains valid. The theorem now follows from observing that we can construct $e^{\pm\epsilon}$ deferred $\mathbf{u}$ -sparsifiers independently and in parallel. We use the results of the q^{th} invocation (which contains the results of all invocations $1, \dots, q$) to refine the $(q+1)^{st}$ deferred sparsifier and then use that refinement in the $(q+1)^{st}$ invocation. The theorem follows—observe that the guarantee on early stopping is provided by Corollary 6.4. Note that the bound of $\mathbf{u}_{\ell}$ (statement of Theorem 1.3, also holds in Theorem 6.1) follows from simple inspection. Note $\mathcal{L}_0$ is within a factor of $1/\gamma$ of $\min_{\ell} \mathbf{u}_{\ell}$ and the result follows. $\square$

6.2 Proof of Theorem 6.2

In this section, we prove that the system `SPARSE` can be interpreted as a fractional packing problem and solved efficiently using the `LAGINNER`. We rewrite `SPARSE` as follows:

$$\begin{aligned} &\mathbf{P}_o \mathbf{x} \leq \Lambda \mathbf{q}_o \quad \Lambda = 2 \\ &\mathbf{Q}(\mathbf{u}^s, \beta) : \left\{ \begin{array}{l} (\mathbf{u}^s)^T \mathbf{Ax} \geq (1 - \epsilon/8) (\mathbf{u}^s)^T \mathbf{c} \\ \mathbb{G}(\mathbf{u}^s, \mathbf{x}) \\ \tilde{\mathbf{Q}}(\beta) = \{\mathbf{b}^T \mathbf{x} \leq \beta, \mathbf{P}_i \mathbf{x} \leq \mathbf{q}_i, \mathbf{x} \in \mathbf{Q}, \mathbf{x} \geq \mathbf{0}\} \end{array} \right. \end{aligned} \quad (\text{SPARSE})$$

`SPARSE` defines a packing problem. We now consider the following theorem in Plotkin et al. (1995)—that paper used the notation $\mathbf{A}, \lambda, \mathbf{b}, \mathcal{P}, \mathbf{u}, \epsilon, \rho, \alpha, \sigma$ instead of $\mathbf{A}^p, \lambda^p, \mathbf{d}, \mathcal{P}^p, \mathbf{z}, \delta, \rho', \alpha', \sigma'$,

respectively; we use different notation since several of those symbols will be in use when this theorem is applied.

THEOREM 6.5 (PLOTKIN ET AL. 1995). *Suppose we are given a decision problem $A^p \mathbf{x} \leq \mathbf{d}$ such that $\mathbf{x} \in \mathcal{P}^p$, where $A^p \in \mathbb{R}^{M' \times N'}$, $\mathbf{x} \in \mathbb{R}^{N'}$, $\mathbf{d} \in \mathbb{R}^{M'}$, where $\mathcal{P}^p$ is some polytope such that $\mathbf{0} \leq A^p \mathbf{x} \leq \rho' \mathbf{d}$ for all $\mathbf{x} \in \mathcal{P}^p$. Suppose we have an initial $\mathbf{x}_0 \in \mathcal{P}^p$ satisfying $A^p \mathbf{x}_0 \leq \delta_0 \mathbf{d}$. The algorithm sets $\mathbf{x} = \mathbf{x}_0$ and proceeds in phases. In phase t , it determines $\lambda_t^p = \max_r (A^p \mathbf{x})_r / \mathbf{d}_r$. It then repeatedly queries an ORACLE-P for $\arg \min_{\tilde{\mathbf{x}} \in \mathcal{P}^p} \mathbf{z}^T A^p \tilde{\mathbf{x}}$, where*

$$\mathbf{z}_r = \exp(\alpha' (A^p \mathbf{x})_r / \mathbf{d}_r) / \mathbf{d}_r \quad \alpha' = O((\lambda_t^p)^{-1} \delta^{-1} \ln(M'/\delta)),$$

and performs an update step $\mathbf{x} \leftarrow (1 - \sigma')\mathbf{x} + \sigma'\tilde{\mathbf{x}}$, where $\sigma' = \delta/(4\alpha'\rho')$. The phase continues till $\lambda^p = \max_r (A^p \mathbf{x})_r / \mathbf{d}_r \leq \min\{\lambda_t^p/2, 1 + 6\delta\}$. If $\lambda^p > 1 + 6\delta$, and then a new phase is started till $\lambda^p \leq (1 + 6\delta)$. Then after at most $T = O(\rho'(\delta^{-2} + \log \delta_0) \log M')$ successful invocations of ORACLE-P, the fractional packing framework provides a solution $A^p \mathbf{x} \leq (1 + 6\delta)\mathbf{d}$, $\mathbf{x} \in \mathcal{P}^p$.

We use a modification that is not present in Plotkin et al. (1995). The proof of Corollary 6.6 follows from observing that $\mathbf{z}^T A^p \tilde{\mathbf{x}} \leq (1 + \delta/2)\mathbf{z}^T \mathbf{d}$ implies $(1 + \delta/2)\mathbf{z}^T A^p \tilde{\mathbf{x}} \leq (1 - \delta/2)\mathbf{z}^T A^p \mathbf{x} - \delta \mathbf{z}^T \mathbf{d}/2$ and then using exactly the same arguments as were used in the proof of Corollary 6.4 modifying Theorem 6.3.

COROLLARY 6.6. *Theorem 6.3 holds if ORACLE-P finds $\mathbf{x} \in \mathcal{P}^p$ satisfying $\mathbf{z}^T A^p \tilde{\mathbf{x}} \leq (1 + \delta/2)\mathbf{z}^T \mathbf{b}$.*

We use Corollary 6.6 with $\delta = \frac{1}{6}$ and $\mathcal{P}^p = \mathbf{Q}(\mathbf{u}^s, \beta)$ on the packing problem Modified-SPARSE. The solution desired by ORACLE-P is given by INNER, and $\rho' = \rho_i$ from Definition 3.

<p>(Modified-SPARSE)</p> <p>$\mathbf{P}_o \mathbf{x} \leq \mathbf{q}_o$</p> <p>$\mathbf{Q}(\mathbf{u}^s, \beta) : \begin{cases} (\mathbf{u}^s)^T \mathbf{A} \mathbf{x} \geq (1 - \epsilon/8)(\mathbf{u}^s)^T \mathbf{c} \\ \mathbb{G}(\mathbf{u}^s, \mathbf{x}) \\ \tilde{\mathbf{Q}}(\beta) \end{cases}$</p>	<p>(INNER)</p> <p>$\mathbf{z}^T \mathbf{P}_o \mathbf{x} \leq (13/12)\mathbf{z}^T \mathbf{q}_o$</p> <p>$\mathbf{Q}(\mathbf{u}^s, \beta) : \begin{cases} (\mathbf{u}^s)^T \mathbf{A} \mathbf{x} \geq \left(1 - \frac{\epsilon}{8}\right)(\mathbf{u}^s)^T \mathbf{c} \\ \mathbb{G}(\mathbf{u}^s, \mathbf{x}) \\ \tilde{\mathbf{Q}}(\beta) \end{cases}$</p>
---	---

COROLLARY 6.7. *If we have a TRANSIENTORACLE that either provides a $\mathbf{y}$ as desired by condition (i) in Theorem 1.3 or provides a solution of INNER for $O(\rho_i(\log \rho_i) \log \tilde{n}_o)$ steps for the $\mathbf{z}$ specified by Corollary 6.6, Theorem 1.3 holds.*

PROOF. Set $\delta = \frac{1}{6}$. Observe that if any of the invocations to TRANSIENTORACLE provided a $\mathbf{y}$, we implement part (i) of the MINIORACLE. If all the invocations succeed, then we have a solution of SPARSE. The result follows from observing $\delta_0 = \rho'$ and $M' = \tilde{n}_o$. Note that any solution of INNER provides an initial solution with $\delta_0 = \rho_i$ for the application of Corollary 6.6. $\square$

Note that the final Lagrangian used in the desiderata of MICROORACLE is stronger by a factor of 2; that is, even though we want to satisfy $\mathbf{P}_o \mathbf{x} \leq 2\mathbf{q}_o$, we have constraint $LHS \geq (\mathbf{u}^s)^T \mathbf{c} - \rho \mathbf{z}^T \mathbf{q}_o$, which is much stronger than $LHS \geq (\mathbf{u}^s)^T \mathbf{c} - \rho 2\mathbf{z}^T \mathbf{q}_o$. Theorem 6.2 follows from Corollary 6.7 and the following lemma.

LEMMA 6.8. *If we have a MICROORACLE that provides either a $\mathbf{y}$ as desired by condition (i) in Theorem 1.3 or a solution of LAGINNER for any $0 < \varrho$, then Theorem 6.1 holds.*

$$\mathcal{P} : \left\{ \begin{array}{l} \mathbf{z}^T \mathbf{P}_o \mathbf{x} \leq (13/12) \mathbf{z}^T \mathbf{q}_o \quad (\text{INNER}) \\ (\mathbf{u}^s)^T \mathbf{A} \mathbf{x} \geq \left(1 - \frac{\epsilon}{8}\right) (\mathbf{u}^s)^T \mathbf{c} \\ \mathbb{G}(\mathbf{u}^s, \mathbf{x}) \\ \tilde{\mathbf{Q}}(\beta) \end{array} \right. \left| \begin{array}{l} (\text{LAGINNER}) \\ (\mathbf{u}^s)^T \mathbf{A} \mathbf{x} - \varrho \zeta^T \mathbf{P}_o \mathbf{x} \geq \left(1 - \frac{\epsilon}{16}\right) \left[(\mathbf{u}^s)^T \mathbf{c} - \varrho \zeta^T \mathbf{q}_o\right] \\ \mathbb{G}(\mathbf{u}^s, \mathbf{x}) \\ \tilde{\mathbf{Q}}(\beta) = \{\mathbf{b}^T \mathbf{x} \leq \beta, \mathbf{P}_i \mathbf{x} \leq \mathbf{q}_i, \mathbf{x} \in \mathbf{Q}, \mathbf{x} \geq 0\} \end{array} \right.$$

PROOF. Note that if any invocation returns a $\mathbf{y}$, then we have nothing to prove. Therefore, we focus on the solutions to LAGINNER. Since $\mathbf{z}^T \mathbf{q}_o \geq 0$, a solution of LAGINNER implies

$$(\mathbf{u}^s)^T \mathbf{A} \mathbf{x} - \varrho \mathbf{z}^T \mathbf{P}_o \mathbf{x} \geq \left(1 - \frac{\epsilon}{16}\right) \left[(\mathbf{u}^s)^T \mathbf{c} - \varrho \mathbf{z}^T \mathbf{q}_o\right] \geq \left(1 - \frac{\epsilon}{16}\right) \left[(\mathbf{u}^s)^T \mathbf{c} - \frac{13}{12} \varrho \mathbf{z}^T \mathbf{q}_o\right]. \quad (18)$$

We invoke the MICROORACLE with $\varrho = \epsilon(\mathbf{u}^s)^T \mathbf{c} / (16 \zeta^T \mathbf{q}_o)$. If the returned $\mathbf{x}$ satisfies

$$\mathbf{z}^T \mathbf{P}_o \mathbf{x} \leq (13/12) \mathbf{z}^T \mathbf{q}_o, \quad (19)$$

then we have immediately a solution for INNER since

$$(\mathbf{u}^s)^T \mathbf{A} \mathbf{x} - \varrho \mathbf{z}^T \mathbf{P}_o \mathbf{x} \geq \left(1 - \frac{\epsilon}{16}\right) \left[(\mathbf{u}^s)^T \mathbf{c} - \varrho \mathbf{z}^T \mathbf{q}_o\right] = \left(1 - \frac{\epsilon}{16}\right)^2 (\mathbf{u}^s)^T \mathbf{c} > \left(1 - \frac{\epsilon}{8}\right) (\mathbf{u}^s)^T \mathbf{c}.$$

Now if $\varrho \geq \varrho_0 = \frac{12 \mathbf{u}_s^T \mathbf{c}}{13 \mathbf{z}^T \mathbf{q}_o}$, then $\mathbf{x} = \mathbf{0}$ is a feasible solution for Equation (18) since the right-hand side is 0. Note $\mathbf{x} = \mathbf{0}$ also satisfies $\mathbb{G}(\mathbf{u}^s, \mathbf{x})$, $\mathbf{Q}$ by Definition 3 and definitely satisfies Equation (19). Therefore, we can perform a binary search over ϱ and finally achieve an interval $[\varrho_1, \varrho_2]$ where in the corresponding solutions, $\tilde{\mathbf{x}}_1$ does not satisfy Equation (19) and $\tilde{\mathbf{x}}_2$ does (but both satisfy Equation (18), $\mathbb{G}(\mathbf{u}^s, \mathbf{x})$, and $\tilde{\mathbf{Q}}(\beta)$). Let $\Upsilon = \frac{13}{12} \mathbf{z}^T \mathbf{q}_o$. Eventually we can narrow the interval $\varrho_2 - \varrho_1 \leq \epsilon \varrho_0 / 16$, where $\varrho_2 > \varrho_1$ and

$$\mathbf{z}^T \mathbf{P}_o \tilde{\mathbf{x}}_1 = \Upsilon_1 > \Upsilon = \frac{13}{12} \mathbf{z}^T \mathbf{q}_o \text{ and } \mathbf{z}^T \mathbf{P}_o \tilde{\mathbf{x}}_2 = \Upsilon_2 \leq \Upsilon.$$

We then find two numbers s_1, s_2 such that $s_1 + s_2 = 1$, $s_1 \Upsilon_1 + s_2 \Upsilon_2 = \Upsilon = \frac{13}{12} \mathbf{z}^T \mathbf{q}_o$. Let $\mathbf{x} = s_1 \tilde{\mathbf{x}}_1 + s_2 \tilde{\mathbf{x}}_2$. Observe that $\mathbf{x}$ satisfies Equation (19), $\mathbb{G}$, and $\tilde{\mathbf{Q}}(\beta)$. Since $\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2$ both satisfy Equation (18),

$$\begin{aligned} (\mathbf{u}^s)^T \mathbf{A} \mathbf{x} &= s_1 (\mathbf{u}^s)^T \mathbf{A} \tilde{\mathbf{x}}_1 + s_2 (\mathbf{u}^s)^T \mathbf{A} \tilde{\mathbf{x}}_2 \\ &\geq s_1 \left(\left(1 - \frac{\epsilon}{16}\right) \left[(\mathbf{u}^s)^T \mathbf{c} - \varrho_1 \frac{13}{12} \mathbf{z}^T \mathbf{q}_o\right] + \varrho_1 \mathbf{z}^T \mathbf{P}_o \tilde{\mathbf{x}}_1 \right) \\ &\quad + s_2 \left(\left(1 - \frac{\epsilon}{16}\right) \left[(\mathbf{u}^s)^T \mathbf{c} - \varrho_2 \frac{13}{12} \mathbf{z}^T \mathbf{q}_o\right] + \varrho_2 \mathbf{z}^T \mathbf{P}_o \tilde{\mathbf{x}}_2 \right) \\ &= \left(1 - \frac{\epsilon}{16}\right) \left[(\mathbf{u}^s)^T \mathbf{c} - s_1 \varrho_1 \Upsilon - s_2 \varrho_2 \Upsilon\right] + s_1 \varrho_1 \Upsilon_1 + s_2 \varrho_2 \Upsilon_2 \\ &= \left(1 - \frac{\epsilon}{16}\right) \left[(\mathbf{u}^s)^T \mathbf{c} - \varrho_1 \Upsilon\right] - \left(1 - \frac{\epsilon}{16}\right) (\varrho_2 - \varrho_1) s_2 + \varrho_1 \Upsilon + s_2 (\varrho_2 - \varrho_1) \Upsilon_2 \\ &\geq \left(1 - \frac{\epsilon}{16}\right) (\mathbf{u}^s)^T \mathbf{c} - s_2 (\varrho_2 - \varrho_1) \Upsilon \geq \left(1 - \frac{\epsilon}{16}\right) (\mathbf{u}^s)^T \mathbf{c} - \frac{\epsilon \varrho_0}{16} \Upsilon \\ &\geq \left(1 - \frac{\epsilon}{16}\right) (\mathbf{u}^s)^T \mathbf{c} - \frac{\epsilon}{16} (\mathbf{u}^s)^T \mathbf{c}. \end{aligned}$$

This completes the proof of the lemma and Theorem 6.2. $\square$

REFERENCES

- K. J. Ahn and S. Guha. 2013. Linear programming in the semi-streaming model with application to the maximum matching problem. *Inf. Comput., IICALP 2011 Special Issue* 222 (2013), 59–79.
- K. J. Ahn and S. Guha. 2014. Near linear time approximation schemes for uncapacitated and capacitated b -Matching problems in nonbipartite graphs. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, also at *Arxiv 1307.4355* (2014), 239–258.
- K. J. Ahn, S. Guha, and A. McGregor. 2012a. Analyzing graph structure via linear measurements. In *Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA'12)*. 459–467.
- K. J. Ahn, S. Guha, and A. McGregor. 2012b. Graph sketches: Sparsification, spanners and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'12)*. 5–14.
- S. Arora, E. Hazan, and S. Kale. 2012. The multiplicative weights update method: A meta algorithm and applications. *Theor. Comput.* 8, 6 (2012), 121–164.
- B. Bahmani, A. Goel, and K. Munagala. 2014. Efficient primal dual algorithms for MapReduce. In *Proceedings of the 11th International Workshop on Algorithms and Models of the Web Graph*. 59–78.
- A. A. Benczúr and D. R. Karger. 1996. Approximating s - t minimum cuts in $\tilde{O}(n^2)$ time. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC'96)*. 47–55.
- A. Bhalgat, R. Hariharan, T. Kavitha, and D. Panigrahi. 2007. An $\tilde{O}(mn)$ Gomory-Hu tree construction algorithm for unweighted graphs. In *Proceedings of the 39th Annual ACM Symposium on the Theory of Computing (STOC'07)*.
- D. Bienstock and G. Iyengar. 2004. Solving fractional packing problems in $O^*(1/\epsilon)$ iterations. In *Proceedings of the 36th Annual ACM Symposium on the Theory of Computing (STOC'04)*. 146–155.
- A. Drucker, F. Kuhn, and R. Oshman. 2014. On the power of the congested clique model. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC'14)*. 367–376.
- R. Duan and S. Pettie. 2010. Approximating maximum weight matching in near-linear time. In *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science (FOCS'10)*. 673–682.
- S. Eggert, L. Kliemann, and A. Srivastav. 2009. Bipartite graph matchings in the semi-streaming model. In *Proceedings of the 17th Annual European Symposium on Algorithms (ESA'09)*. 492–503.
- L. Epstein, A. Levin, J. Mestre, and D. Segev. 2010. Improved approximation guarantees for weighted matching in the semi-streaming model. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS'10)*. 347–358.
- J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. 2005. On graph problems in a semi-streaming model. *Theor. Comput. Sci.* 348, 2–3 (2005), 207–216.
- D. Foster and R. Vohra. 1999. Regret in the on-line decision problem. *Games Econ. Behav.* 29 (1999), 7–35.
- W. S. Fung, R. Hariharan, N. J. A. Harvey, and D. Panigrahi. 2011. A general framework for graph sparsification. In *Proceedings of the 43th Annual ACM Symposium on the Theory of Computing (STOC'11)*. 71–80.
- M. D. Grigoriadis and L. G. Khachiyan. 1995. A sublinear-time randomized approximation algorithm for matrix games. *Oper. Res. Lett.* 18 (1995), 53–58.
- V. Guruswami and K. Onak. 2013. Superlinear lower bounds for multipass graph processing. *Electronic Colloquium on Computational Complexity (ECCC)* 20, 2 (2013).
- R. Hariharan, T. Kavitha, and D. Panigrahi. 2007. Efficient algorithms for computing all low s - t edge connectivities and related problems. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA'07)*.
- M. Kapralov. 2013. Better bounds for matchings in the streaming model. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA'13)*. 1679–1697.
- L. G. Khachiyan. 1978. Convergence rate of the game processes for solving matrix games. *Zh. Vychisl. Mat. and Mat. Fiz.* 17 (1977): 1421–1431. English translation in *USSR Comput. Math and Math. Phys.* 17 (1978): 78–88.
- P. N. Klein and N. E. Young. 1999. On the number of iterations for Dantzig-Wolfe optimization and packing-covering approximation algorithms. In *Proceedings of the 7th Conference on Integer Programming and Combinatorial Optimization (IPCO'99)*. 320–327.
- S. Lattanzi, B. Moseley, S. Suri, and S. Vassilvitskii. 2011. Filtering: A method for solving graph problems in MapReduce. In *Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'11)*. 85–94.
- N. Littlestone and M. M. Warmuth. 1994. The weighted majority algorithm. *Inf. Comput.* 108 (1994), 212–261.
- M. Luby and N. Nisan. 1993. A parallel approximation algorithm for positive linear programming. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing (STOC'93)*. 448–457.
- Y. Mansour and S. Vardi. 2013. A local computation approximation scheme to maximum matching. In *Proceedings of the 16th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'13)*. 260–273.
- A. McGregor. 2005. Finding graph matchings in data streams. In *Proceedings of the 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'05)*. 170–181.

- H. Nagamochi and T. Ibaraki. 1992. A linear-time algorithm for finding a sparse k -connected spanning subgraph of a k -connected graph. *Algorithmica* 7, 1–6 (1992), 583–596.
- A. Nemirovski. 2005. Prox-method with rate of convergence $O(1/t)$ for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM J. Optim.* 15, 1 (2005), 229–251.
- Y. Nesterov. 2005. Smooth minimization of non-smooth functions. *Math. Program., Ser. A* 103 (2005), 127–152.
- M. W. Padberg and M. R. Rao. 1982. Odd minimum cut-sets and b-matchings. *Math. Oper. Res.* 7, 1 (1982), 67–80.
- S. A. Plotkin, D. B. Shmoys, and É. Tardos. 1995. Fast approximation algorithms for fractional packing and covering problems. *Math. Oper. Res.* 20 (1995), 257–301.
- A. Schrijver. 2003. *Combinatorial Optimization - Polyhedra and Efficiency*. Algorithms and Combinatorics, Vol. 24. Springer.
- M. Zelke. 2008. Weighted matching in the semi-streaming model. In *Proceedings of the 25th International Symposium on Theoretical Aspects of Computer Science (STACS'08)*. 669–680.

Received July 2016; revised August 2017; accepted September 2017