Near Linear Time Approximation Schemes for Uncapacitated and Capacitated $b$–Matching Problems in Nonbipartite Graphs$^*$

Kook Jin Ahn$¹$  Sudipto Guha$‡$

Abstract

We present the first fully polynomial approximation schemes for the maximum weighted (uncapacitated or capacitated) $b$–Matching problem for nonbipartite graphs that run in time (near) linear in the number of edges, that is, given any $\delta > 0$ the algorithm produces a $(1 - \delta)$ approximation in $O(m \text{poly}(\delta^{-1}, \log n))$ time. We provide fractional solutions for the standard linear programming formulations for these problems and subsequently also provide fully polynomial (near) linear time approximation schemes for rounding the fractional solutions.

Through these problems as a vehicle, we also present several ideas in the context of solving linear programs approximately using fast primal-dual algorithms. First, we show that approximation algorithms can be used to reduce the width of the formulation, and as a consequence we induce faster convergence. Second, even though the dual of these problems have exponentially many variables and an efficient exact computation of dual weights is infeasible, we can efficiently compute and use a sparse approximation of the dual weights using a combination of (i) adding perturbation to the constraints of the polytope and (ii) amplification followed by thresholding of the dual weights.

These algorithms also have the advantage that they use $O(n \text{poly}(\delta^{-1}, \log n))$ storage space and only make $O(\delta^{-4} \log^2 (1/\delta) \log n)$ (or better) passes over a read only list of edges. These algorithms therefore can be run in the semi-streaming model and serve as exemplars where algorithms and ideas developed for the streaming model gives us algorithms for combinatorial optimization problems that were not known in absence of the streaming constraints.

$^*$An extended abstract of the paper will appear in SODA 2014.

$¹$Google, 1600 Amphitheatre Parkway Mountain View, CA 94043. Email kookjin@google.com. This work was done while the author was at University of Pennsylvania.

$‡$Department of Computer and Information Sciences, University of Pennsylvania, Philadelphia, PA. Email: sudipto@cis.upenn.edu Research supported in part by NSF Award CCF-1117216.
1 Introduction

In this paper we provide the first near linear time fully polynomial \((1 - \delta)\)-approximation schemes (for any \(\delta > 0\)) for finding fractional as well as integral solutions for the weighted maximum \(b\)-Matching problem in nonbipartite graphs. We address both uncapacitated and capacitated versions of the problem.

**Definition 1.** [29, Chapter 31] In the \(b\)-Matching problem we are given a weighted (possibly non-bipartite) graph \(G = \langle V, E, \{w_{ij}\}, \{b_i\}\rangle\) where \(w_{ij}\) is the weight of edge \((i,j)\) and \(b_i\) is the capacity of the vertex \(i\). Let \(|V| = n\) and \(|E| = m\). For simplicity we assume \(b_i, w_{ij}\) are integers in \([0,\text{poly}\,n]\). We can select an edge \((i,j)\) with multiplicity \(y_{ij}\) such that \(\sum_{j: (i,j) \in E} y_{ij} \leq b_i\) for all vertices \(i\) and the goal is to maximize \(\sum_{(i,j) \in E} w_{ij} y_{ij}\). Let \(B = \sum_i b_i\), note that without loss of generality, \(B \geq n\).

**Definition 2.** [29, Chapters 32 & 33] In the Capacitated \(b\)-Matching problem we have an additional restriction for every edge \((i,j) \in E\) that \(y_{ij} \leq c_{ij}\) where \(c_{ij}\) are also given in the input (also assumed to be an integer in \([0,\text{poly}\,n]\)). A problem with \(c_{ij} = 1\) for all \((i,j) \in E\) is also referred to as an “unit capacity” or “simple” \(b\)-Matching problem in the literature.

The \(b\)-Matching problem is a fundamental problem with a long and rich history in combinatorial optimization, see [29, Chapters 31–33]. The \(b\)-matching problems have recently found applications in Machine Learning [21], Müller-Hannemann and Schwartz [26] also provide an excellent survey of different algorithms for variants of \(b\)-Matching.

**Previous results:** The current best algorithms for different variants of \(b\)-Matching are:

- To date, the best exact algorithms for the \(b\)-Matching problem in general graphs are super-linear (see [29, Chapter 31]).
  - Gabow [15] gave an \(O(nm \log n)\) algorithm for the unweighted \((w_{ij} = 1)\) capacitated problem. For \(c_{ij} \leq 1\) this reduces to \(O(\min\{\sqrt{Bm}, nm \log n\})\).
  - For the weighted uncapacitated case Anstee [4] gave an \(O(n^2m)\) algorithm; an \(O(m^2)\) algorithm is in [15].
  - Letchford et al. [24], building on Padberg and Rao [27], gave an \(O(n^2m \log(n^2/m))\) time algorithm for the weighted, uncapacitated/capacitated problem.

- In terms of approximation algorithms there has been progress in different directions but there is no \((1 - \delta)\)-approximation scheme (for any \(\delta > 0\)) faster than computing the optimum solution for the nonbipartite case, the focus of this paper. Distributed algorithms with \(O(1)\) or weaker guarantees have been discussed by Koufogiannakis and Young [23]. Mestre [25] provided a \((\frac{2}{3} - \delta)\) approximation algorithm running in time \(O(m(\max_i b_i) \log \frac{1}{\delta})\) time for weighted unit capacity \(b\)-Matching [25]. For the bipartite case, \((1 - \delta)\) approximation schemes were provided in [2] (see the end of the section for more detailed comparison). It is known that solving the bipartite relaxation for the weighted \(b\)-Matching problem within a \((1 - \delta)\) approximation (for any \(\delta > 0\)) will always produce a \((\frac{2}{3} - \delta)\)-approximation algorithm for general non-bipartite graphs [13, 14]. This approximation is also tight\(^1\) — no approach which only uses bipartite relaxations will breach the \(\frac{2}{3}\) barrier.

---

\(^1\)Consider all \(b_i = 1, w_{ij} = 1\) for a triangle graph. The bipartite relaxation would produce a fractional solution of \(\frac{3}{2}\) whereas the optimum solution is 1.
It is not too hard to see that if we were to copy each node $b_i$ times then the $b$–Matching problems reduce to maximum weighted matching (but the graph may continue to have odd-cycles). However the size of the graph increases drastically under such a transformation. This approach can be used to produce fast (weakly polynomial time) algorithms that depend on $B$, for example as in [16]. A compressed representation was introduced in [15] to avoid this blowup. But this representation is not approximation preserving – see the discussion in [20]. On the other hand, if we do not copy the vertices then standard augmentation path based techniques (including recent elegant results for maximum matching such as [10] [11]) do not immediately apply. The augmentation structure needed for $b$–Matching are not just blossoms (which are standard in matching literature) but also blossoms with “petals/arms” or forests that are attached to the blossom, see the discussion in [20]. Fast, approximate solutions for combinatorial optimization problems have recently seen a lot of activity [7] [22], and in this paper we provide fast approximation schemes for the $b$–Matching problem.

**Statement of Results:** We assume that the edges in the graph $G = (V, E)$ are presented as a read only list $(\ldots, (i, j, w_{ij}), \ldots)$ in arbitrary order. The space complexity will be measured in words and we assume that the integers in the input are bounded from above by poly $n$ to avoid bit-complexity issues.

**Theorem 1** (Fractional $b$–Matching). Given any nonbipartite graph, for any $0 < \delta \leq 1/16$, we find a $(1 - 14\delta)$-approximate (to the standard LP relaxation, $LP_1$ described shortly) fractional weighted $b$–Matching using additional “work” space (space excluding the read-only input) $O(n \text{poly}\{\delta^{-1}, \ln n\})$ and making $T = O(\delta^{-4}(\ln(1/\delta))\ln n)$ passes over the list of edges. The running time$^2$ is $O(mT + n \text{poly}\{\delta^{-1}, \ln n\})$.

Note that the standard LP relaxation has integral optimal solutions for integral $b_i$ [29] Chapter 31.

**Theorem 2** (Integral $b$–Matching). Given a fractional $b$–matching $y$ for a non-bipartite graph which satisfies the constraints in the standard LP formulation, we find an integral $b$–Matching of weight at least $(1 - \delta)$ times the LP objective in $O(\text{poly}\{\delta^{-1}, \ln n\})$ time and $O(m'\delta^2)$ space where $m' = |\{(i, j) | y_{ij} > 0\}|$.

**Theorem 3** (Fractional, Capacitated $b$–Matching). Given any nonbipartite graph, for any $0 < \delta \leq 1/16$, we find a $(1 - 14\delta)$-approximate (to the standard LP relaxation for the capacitated case, $LP_8$ which is different from $LP_1$) fractional weighted capacitated $b$–Matching using $O(mR/\delta + \text{min}\{B, m\} \text{poly}\{\delta^{-1}, \ln n\})$ time, $O(\text{min}\{m, B\} \text{poly}\{\delta^{-1}, \ln n\})$ additional “work” space, making $R = O(\delta^{-4}(\ln(1/\delta))\ln n)$ passes over the list of edges where $B = \sum_b b_i$. Moreover the feasible fractional solution $\hat{y}_{ij}$ has the property that $\sum_{(i, j) \in E} \hat{y}_{ij} \leq 2R\beta^*$ where $\hat{E} = \{(i, j) | (i, j) \in E, \hat{y}_{ij} > 0\}$ and $\beta^*$ is the weight of the optimum capacitated $b$–Matching.

The restriction $\sum_{(i, j) \in \hat{E}} \hat{y}_{ij} \leq T\beta^*$ is explicitly used in the next theorem.

**Theorem 4** (Integral, Capacitated $b$–Matching). Given a feasible fractional (as described in Theorem 3) capacitated $b$–Matching $y$ for a non-bipartite graph, we find an integral $b$–Matching of weight at least $(1 - \delta)\sum_{(i, j)} w_{ij}y_{ij} - \delta\beta^*$ in $O(m'R\delta^{-3}\ln(R/\delta))$ time and $O(m'/\delta^2)$ space where $m' = |\hat{E}|$. If the fractional solution is $(1 - 14\delta)$-approximate then we have a $(1 - 16\delta)$-approximate integral solution.

---

$^2$Consider a star graph where the central node has $b_i = n$ and the leaf nodes have $b_i = 1$. The replication of the central node will make the number of edges $n^2$.

$^3$The exact exponent of $\delta, \log n$ in the poly() term depends on [19] [5] and we omit discussion of such in this paper.
1.1 Ideas, Comparisons and Roadmap

We discuss the overall approach, the roadblocks faced and how they are mitigated. We begin with the standard exact linear programming formulation for the $b$–Matching problems.

**Definition 3 (Volume of Sets & Odd-Sets).** Given a graph $G = (V,E)$, with $|V| = n$ and $|E| = m$, and non-negative values $b_i$ for each $i \in V$, define the volume of a set $U \subseteq V$ to be $||U||_b = \sum_{i \in U} b_i$. Define $\mathcal{O} = \{U \mid ||U||_b \text{ is odd} \}$ and $\mathcal{O}_\delta = \{U \mid U \in \mathcal{O}; ||U||_b \leq 1/\delta \}$.

Consider the formulation $\text{LP1}$ (see [29, Chapter 31]) parametrized by the vector $b$. The variable $y_{ij}$ (which is the same as $y_{ji}$) corresponds to the fractional relaxation of the “multiplicity” of the edge $(i,j)$ in the uncapacitated $b$–Matching. Throughout this paper we assume that if $(i,j) \notin E$ then $y_{ij} = 0$ as well as $(i,i) \notin E$; no self loops are allowed.

$$\beta^* = \text{LP1}(b) = \max \sum_{(i,j)} w_{ij}y_{ij}$$

$$\sum_j y_{ij} \leq b_i \quad \forall i \in V$$

$$\sum_{(i,j); i,j \in U} y_{ij} \leq \lfloor ||U||_b/2 \rfloor \quad \forall U \in \mathcal{O}$$

$$y_{ij} \geq 0$$

We have $m$ variables and exponentially many constraints. The constraints in the primal $\text{LP1}$ correspond to the vertices and odd sets. It is well known that the formulation $\text{LP1}$ has an integral optimum solution when $b_i$ are integers. It is not difficult to see that if we only retain the constraints for odd sets $U$ where $U \in \mathcal{O}_\delta$ then a fractional solution of the modified system; when multiplied by $(1 - \delta)$; satisfies $\text{LP1}$. That relaxed formulation, which captures $1 - \delta$ approximate fractional solutions, still has $n^{1/\delta}$ constraints which is exponential in $1/\delta$. We need fast primal-dual frameworks to solve such a system efficiently. Most primal dual algorithms, see the survey of Arora, Hazan and Kale [5], maintain an infeasible primal solution and seek to improve it iteratively. This is achieved by maintaining or constructing dual weights that indicate an “improvement direction”. The multiplicative weight meta-method in its standard form, uses the sign of the violation to decide between increasing or decreasing the dual weights. Alternate approaches, such as the Fractional Packing framework [28], use linear operators but most applications of the standard meta-method will require that we compute the $n^{1/\delta}$ violations.

**The main ideas:** We introduce several ideas;

(i) We add carefully calibrated **perturbations** to RHS of the constraints such that the constraints which have violation close to the maximum violation (in ratio, the LHS divided by the RHS) define a laminar family. We show that these constraints can be found efficiently.

(ii) We **amplify** the effect of the violations and use **thresholding** to show that a modified primal-dual framework which uses only the violations defined in the previous step converges fast.

(iii) We show that we can use efficient approximation algorithms to bound the **width** or the rate of convergence. We use a $O(1)$ or $O(\log 1/\delta)$ approximation algorithms as a substep of the overall framework. The approximation ratio only alters the rate of convergence and does not affect the guarantee that the final solution is a $(1 - \delta)$ approximation. This is an unusual use of approximation algorithms.

Observe that the question of bypassing the evaluation of exponentially many constraints is reminiscent of the maximum violation approach in the context of Ellipsoid algorithms (see [18]).
where a separation oracle suffices for polynomial time solvability. Often that separation oracle needs to provide the the maximum violated constraint, assuming that the number of variables \((m)\) is polynomial. However unlike that approach, we need to find a collection of such constraints efficiently.

On a more technical note, we also modify parts of the proof of the fractional packing framework to handle the case that even though there are \(M = O(n^{1/\delta})\) constants and only a few of them matter at any step. However, we have no apriori information about which constraints matter, and a “standard” application of approximate primal-dual framework will require the amplification term to depend on \(\log M\). We avoid this and show that the comparable term is \(O(\delta \log M)\) taking into account the specific way the dual weights are computed – that the integral/sum of the weights does not grow as the worst case analysis suggests. While it is feasible to modularize such an argument as a general method, modularization also comes at a cost of losing sight of how the entire framework functions for a specific task. For \(b\)-Matching, such a global view is necessary for extending the proof to the capacitated case. Therefore we have chosen to not emphasize this technical point.

**Perturbations:** We discuss the idea of perturbation first. The approach of Padberg-Rao [27], which is often referred to as the “minimum odd-cut” approach, follows the maximum violated constraint approach in the context of matching. Many proofs using the minimum odd-cut approach rely on fascinating combinatorial techniques such as Total Dual Integrality (TDI), laminariti, etc., (see Giles and Pulleyblank [17], Cook [8], Cunningham and Marsh [9], and also Schrijver [29]). However all these techniques rely on **exact optimality** of the primal and dual solutions. In fact such relationships do not exist for arbitrary feasible primal or dual solutions. It is then natural to ask the follow up question – **If notions such as laminarity do not exist for arbitrary feasible primal or dual solutions – then can we modify the polytope to achieve such properties?** The answer to the last question is surprising – if we **perturb** the polytope slightly, then two interesting theorems can be proven:

**Theorem 5.** Suppose that we are given a graph \(G\) with \(n\) vertices and any non-negative edge weights \(\{y_{ij}\}\) satisfying (i) \(\hat{y}_{ij} = \hat{y}_{ji} \geq 0\) (ii) \(\hat{y}_{ii} = 0\) for all \(i\), and (iii) \(\sum_j \hat{y}_{ij} \leq b_i\) for all \(i\). Given \(\delta \in (0, \frac{1}{16}]\), define:

\[
\lambda_U = \sum_{(i,j)\delta U} \hat{y}_{ij} \quad \text{where} \quad \hat{b}_U = \left[\frac{||U||_b}{2} - \frac{\delta^2 ||U||_b^2}{4}\right] \quad \text{and} \quad \hat{\lambda} = \max_{U \in O_\delta} \lambda_U
\]

If \(\hat{\lambda} \geq 1 + 3\delta\), the set \(L_1 = \{U : \hat{\lambda}_U \geq \hat{\lambda} - \delta^3; U \in O_\delta\}\) defines a laminar family. Moreover for any \(x \geq 2\) we have \(||\{U : \hat{\lambda}_U \geq \hat{\lambda} - \delta^x; U \in O_\delta\}|| \leq n^3 + (n/\delta)^{1+\delta(x-1)/2}\).

In other words, Theorem 3 states that if we were to focus on the constraints that are almost as violated as the maximum violated constraint of the perturbed polytope, then those constraints correspond to a laminar family for any plausible (such as nonnegative, obeying the vertex constraints, etc.) primal solution candidate \(\{\hat{y}_{ij}\}\). This can be viewed as a strengthening of the laminarity properties which were observed at the optimum dual solution in [9, 29]. In the primal space, the laminarity of the optimum dual solution, can be viewed as the following property: Consider an infeasible primal solution in a small (infinitesimal) neighborhood of the optimum primal solution, and generate dual weights based on the exponentials of the violation – the large dual values will correspond to a laminar family. Theorem 5 extends that characterization to all primal candidates where the neighborhood is defined by the perturbations. The intuitive reason is simple — if we were

\(L\) is defined as laminar if for any two sets \(U, U' \in L\), \(U \cap U'\) is either \(U\), \(U'\) or \(\emptyset\).
to ignore the floor and ceil functions then for a fixed $\hat{\lambda}_U$, the function $\sum_{(i,j),i,j \in U} \hat{y}_{ij}$ is a concave function of $||U||_b$. As a result if two such $U_1,U_2$ intersect at a non-singleton odd set $U_3 \neq U_1,U_2$ (the union $U_4 \neq U_1,U_2$ is also an odd set) then $\max\{\hat{\lambda}_{U_3},\hat{\lambda}_{U_4}\}$ will exceed $\min\{\hat{\lambda}_{U_1},\hat{\lambda}_{U_2}\}$ by $\delta^3$. Of course, the floor and ceil functions, singleton or even set intersections cannot be ignored and more details are required, but the idea behind the proof remains the same. Theorem 5 is a standalone combinatorial characterization and is proved in Section 3. However Theorem 5 does not (yet) give us an algorithm. This leads us to the next main theorem:

**Theorem 6.** For a graph $G$ with $n$ vertices and $\{\hat{y}_{ij}\}$ and the definitions of $\{\hat{\lambda}_U\}$ exactly as in the statement of Theorem 3 and $\delta \in (0, \frac{1}{16}]$, if $\hat{\lambda} \geq 1 + 3\delta$ we can find the set $L_2 = \{U : \hat{\lambda}_U \geq \hat{\lambda} - \frac{\delta^3}{10}; U \in O_\delta\}$ in $O(m' + n \text{poly}(\delta^{-1}, \log n))$ time using $O(n\delta^{-5})$ space where $m' = \{|(i,j)|\hat{y}_{ij} > 0\}$.

The proof of Theorem 6 relies on the fact that $L_2 \subseteq L_1$ is a laminar family as proved in Theorem 5. These two theorems are intended to be used in tandem along with the observation that if $\hat{\lambda}$ is small then we are in an easy case. The proof of Theorem 6 is presented in Section 4. We provide a sketch of a proof and algorithmic ideas here. There are two hurdles to overcome in Theorem 6: (i) How do we even know $\hat{\lambda}$ efficiently, i.e., in near linear time? and (ii) How do we find all sets in $L_2$ efficiently? An inefficient answer to (i) follows from the minimum odd-cut approach of Padberg and Rao 27, because large $\hat{\lambda}_U$ implies a small cut. But that approach uses exact Gomory-Hu trees and the computation of such is not known to be in near linear time. As regards (ii) we show that using the specific way $\hat{\lambda}$ is being found, we can find all sets in $L_2$ simultaneously. We discuss both in the following.

Define $L_1(\ell) = \{U|U \in L_1, ||U||_b = \ell\}$ and $L_2(\ell) = \{U|U \in L_2, ||U||_b = \ell\}$ for $\ell \in [3, 1/\delta]$. Note that $L_1(\ell) \supseteq L_2(\ell)$. Observe that it suffices to identify $L_2(\ell)$ for some fixed $\ell$ – we can repeat this for different $\ell$. Now, if we can reduce the problem of finding $L_2(\ell)$ to some problem of finding low cuts in an unweighted graph then there exists near linear time algorithms for finding a representation of all small cuts 19 10 (the equivalent of Gomory-Hu trees for small cuts). Note, that the algorithm of 27 still guarantees finding one cut. Our solution (see Section 6) is to construct an unweighted graph $G_\varphi$ with $p_{ij} = [\varphi \hat{y}_{ij}]$ parallel edges between $i$ and $j$ where $\varphi = 50/\delta^4$. We can merge all pairs of nodes which have more than $2\varphi$ edges between them, and delete nodes with degree larger than $2\varphi/\delta$ giving us a bounded degree graph which can be stored in small space. Subsequently, we add a special node $s$ and construct unweighted graphs $G_\varphi(\ell, \hat{\lambda})$ with the following two properties:

**Property 1.** If $\hat{\lambda} - \frac{\delta^3}{100} < \hat{\lambda} \leq \hat{\lambda}$, then (i) all sets in $L_2(\ell)$ have a cut which is at most $\kappa(\ell)$ and (ii) all odd cuts of $G_\varphi(\ell, \hat{\lambda})$ which do not contain $s$ and have cut at most $\kappa(\ell)$ belong to $L_1(\ell)$. Here $\kappa(\ell) = \lfloor \varphi \hat{\lambda}(1 - \delta^2\ell^2/2)\rfloor + \frac{12\varphi}{\delta} + 1 < 2\varphi$.

**Property 2.** We show in Lemma 7 that we can extend the algorithm in 27 to efficiently extract a collection $L$ of maximal odd-sets in $G_\varphi(\ell, \hat{\lambda})$, not containing $s$ and cut at most $\kappa(\ell)$ – such that any such set which is not chosen must intersect with some set in the collection.

If we have a maximal collection $L$ then $L \subseteq L_1(\ell)$ by condition (ii) of Property 1. Due to Theorem 5, the intersection of two such sets $U_1,U_2 \in L_1(\ell)$ will be either empty or of size $\ell$ by laminarity – the latter implies $U_1 = U_2$. Therefore the sets in $L_1(\ell)$ are disjoint. Any $U \in L_2(\ell) - L$ has a cut of at most $\kappa(\ell)$ using condition (i) of Property 1 and therefore must intersect with some set in $L$. This is impossible because $U \in L_2(\ell)$ implies $U \in L_1(\ell)$ and $L \subseteq L_1(\ell)$ and we just argued that the sets in $L_1(\ell)$ are disjoint! Therefore no such $U$ exists and $L_2(\ell) \subseteq L$.

We now have a complete algorithm: we perform a binary search over the estimate $\hat{\lambda} \in [1 + 3\delta, \frac{3}{2} + \delta^2]$, and we can decide if there exists a set $U \in L_2(\ell)$ in time $O(n \text{poly}(\delta^{-1}, \log n))$ as we
vary $\ell, \tilde{\lambda}$. This gives us $\tilde{\lambda}$. We now find the collections $\mathcal{L}$ for each $\ell$ and compute all $\tilde{\lambda}_U$ exactly (either remembering the $\tilde{y}_{ij}$ of the the edges stored in $G_\varphi$ or by another pass over $G$). We can now return $\cup_\ell L_2(\ell)$. The complete proof of Theorem 6 is in Section 4.

**Amplification and Thresholding:** So far we have described how the constraints that are close to the maximum (ratio) violation can be found efficiently (Theorems 5 and 6). We show that computing just these constraints suffice for a fast convergence of a primal-dual algorithm. In particular we modify the primal-dual framework (the second part of Theorem 5 is useful here) – such that the weight of the constraints closer to the maximum (ratio) violation are amplified, and we use only those constraints for computing the update step. As a net result, the nonbipartite problem reduces to a bipartite matching problem (See Algorithm 1, system $\text{LP}^4$) on suitably defined effective weights which are dynamic.

**Width and Approximations:** We use a final idea that it suffices to solve the oracle required by the primal-dual algorithm approximately, increasing the vertex capacities $b_i$ by a $O(1)$ factor. The approximation guarantee simply changes the “width” of the polytope or the rate of convergence by 2 times the same $O(1)$ factor. For example if we were seeking a solution to $\{c^T x \geq \beta; Ax \leq b; x \geq 0\}$ and we have a $\rho$-approximate solution $x'$ satisfying $\{c^T x' \geq \beta/\rho; Ax' \leq b; x' \geq 0\}$; then by simple scaling we have a solution for $\{c^T x \geq \beta; Ax \leq \rho b; x \geq 0\}$ which implies that the width of the polytope is $\rho$. This same idea is used in the context of the capacitated $b$–Matching problem where we exceed the vertex capacities by a factor $O(\log \frac{1}{\delta})$ but do not increase the edge capacities at all. Thus the computation for the capacitated $b$–Matching problem maintains the invariant that edge capacities are never violated at any stage of the algorithm.

**Comparison to Other Results:** Efficient approximation schemes for bipartite graphs were considered in [2] – that paper also provided inefficient ($\Omega(n^{1/\delta}m)$ time) algorithms for the non-bipartite case. In a companion paper [3] we consider uncapacitated $b$-matching problems, specifically the weighted bipartite case and the unweighted non-bipartite case. That paper introduces a “dual-primal” approach where we start from the dual of the matching polytope and explicitly add TDI constraints. For nonbipartite graphs, which is the subject under discussion in this paper, [3] provides a $(1-\delta)$-approximation algorithm for the unweighted uncapacitated $b$–Matching problem in $O(m\delta^{-2}(\log^2(n/\delta))\alpha_m n + n^{1+1/p})$ time where $p \geq 1$ is any integer and $\alpha_m$ is the inverse Ackermann function (using the analysis of union-find). That running time has a better dependence on $\delta$ than the running time of $O(mT)$ (Theorem 7 in the regime $m \gg n$), because $T = \delta^{-4}(\log(1/\delta)) \log n$, but the dependence on $\log n$ is worse. We do not think that this manuscript and [3] are comparable though they illustrate those goals using the same combinatorial problems – and matching is one of the most celebrated combinatorial optimization problems.

However an interesting theme does emerge from the two manuscripts. Primal formulations (for any problem) can accommodate weights and constraints such as capacities, using the same solution technique. At the same time, reducing the number of iterations is difficult in the primal setting since the bottleneck is a simple concentration of measure argument.

**Roadmap:** Theorems 5 and 6 are proved in Sections 3 and 4 respectively. The overall algorithm is presented in Section 2 along with pointers to other parts of the proof. The details of the proof are modularized and are in Sections 3 and 4 which handle finding the initial solution, and computing the updates efficiently, respectively. We discuss the capacitated problem alongside the uncapacitated problem while finding the initial solution (Section 5). Section 7 proves the rounding algorithm of Theorem 2. Section 8 discusses capacitated $b$–Matching. It proves Theorems 3 and 4 and refers to the other sections as appropriate.
2 (1 − δ) Approximate Fractional b-Matching

In this section we prove Theorem 1 using a primal-dual algorithm. Algorithm 1 is the main algorithm of this paper. A naive application of primal-dual techniques, such as at the fractional packing or multiplicative weight update, will increase the number of convergence steps which are required because the number of constraints will be $n^{1/δ}$. We will use the two main theorems 5 and 6 to reduce the effective number of constraints – however note that these keep changing throughout the algorithm. We first prove:

Algorithm 1 Near Linear Time Approximation Scheme for b-Matching

1: Let $||U||_b = \sum_{i \in U} b_i$ and $O = \{U \mid ||U||_b \text{ is odd } \}$. Fix $δ \in (0, \frac{1}{16}]$. Let $O_δ = \{U \mid U \in O; ||U||_b \leq 1/δ \}$.
2: Initialize $ε = 1/8$. Find an initial solution (See Section 5) with $β = β_0$ (and $β^* ≤ β_0 ≤ 6β^*$), $λ ≤ λ_0 = 12$ such that $\text{LP}2$ holds. Note $y_{ij} = y_{j'i}$ throughout.

\[ \text{A : } \begin{cases} \sum_i y_{ij} ≤ λ_b_i & \forall i \in V \text{ where } b_i = (1−4δ)b_i \\ \sum_{(i,j) \in U} y_{ij} ≤ λ_{b_U} & \forall U \in O_δ \text{ where } b_U = \left[\frac{||U||_b}{2} \right] - \frac{δ^2||U||^2_b}{4} \\ \sum_{(i,j)} w_{ij}y_{ij} ≥ (1−δ)β & \text{ (LP2)} \end{cases} \]

\[ \text{P}[β]: \begin{cases} \sum_i y_{ij} ≤ 6b_i & \forall i \in V \\ y_{ij} ≥ 0 \end{cases} \]

3: Let $α = 50δ^{-3} \ln n$, $λ_i = \sum_j y_{ij}/b_i$, $λ_U = \sum_{(i,j)\in U} y_{ij}/b_U$ and $λ = \max\{\max_i λ_i, \max_U λ_U\}$.
4: The algorithm proceeds in superphases which are further subdivided into phases. A new superphase starts when $\lambda ≤ 1 + 8ε$ (we will be decreasing $ε$ gradually). A new phase starts either at the start of a superphase or when $(1−8δ)λ_t$ where $λ_t$ is the value of $λ$ at start of phase $t$. Note $ε ≥ δ$.
5: while $ε > δ$
6: while in phase $t$
7: Compute $λ$ exactly (for every iteration within the phase) and find a laminar collection of odd sets $L$ such that $U ∉ L ⇒ λ_U ≤ λ − δ/2$. (Lemma 7 proves the validity of lines 6,7.)
8: If $λ < \max\{1 + 8ε, (1−8δ)λ_t\}$ then end phase $t$ (goto line 14).
9: Set $x_i = \exp(αλ_i)/b_i$ for $i \in V$. For $U ∈ L$ set $z_U = \exp(αλ_U)/b_U$ else $z_U = 0$.
10: For any $γ ≥ 0$ let $L(y', γ) = \sum_{(i,j)} w_{ij}y'_{ij} + γ(\sum_{(i,j)} y'_{ij}x_i + x_j + \sum_{U ∈ L, i,j \in U} z_U )$. Let $γ = \sum_i x_i b_i + \sum_{U ∈ O_δ} z_U b_U$. Consider:
11: \[ \begin{cases} \sum_{(i,j)} \tilde{y}_{ij}x_i + x_j + \sum_{U ∈ L, i,j \in U} z_U \leq γ & \text{s.t. } \tilde{y} \in P[β] \text{ (LP3)} \\ \tilde{L}(y', γ) ≥ β − ϵγ & \text{s.t. } y' \in P \text{ (LP4)} \end{cases} \]
12: Use $O(\ln \frac{1}{δ})$ calls (binary search over $γ$) to $\text{LP}4$ to find a $\{\tilde{y}_{ij}\}$ feasible for $\text{LP}3$. (Lemma 9). If any of the substeps fail then set $β ← (1−δ)β$ and repeat till a feasible solution of $\text{LP}3$ is found.
13: end while
14: If ($ε = δ$) output result (line 17).
15: If ($λ ≤ 1 + 8ε$) then start new superphase (and phase) with $ε ← \max\{2ε/3, δ\}$ else start phase $t+1$.
16: end while
17: Output: $\{y^*_{ij} = \frac{y_{ij}}{1+8δ}\}$. Note $(\sum_{(i,j)} w_{ij}y^*_{ij} ≥ (1−14δ)β^* \text{ and } \{y^*_{ij}\} \text{ is feasible for } \text{LP1} \}$. (Theorem 10).

Lemma 7. Fix $δ \in (0, \frac{1}{16}]$. If $λ > 1 + 8δ$ then we can find $L = \{U \mid λ_U ≥ λ − 3/10; U ∈ O_δ\}$ in $O(m + n \text{poly}\{δ^{-1}, \ln n\})$ time. Moreover $L$ defines a laminar family. Finally, for any $x ≥ 2$ we
have \(|\{U : \lambda_U \geq \lambda - \delta^x; U \in \mathcal{O}_\delta\}| \leq n^3 + (n/\delta)^{1+\delta(x-3)/2}\).

Proof. Let \(\Delta = \max\{1, \max_i(1 - 4\delta)\lambda_i\} = \max\{1, \max_i \sum_j y_{ij}/b_i\}\) and \(\hat{y}_{ij} = y_{ij}/\Delta\). Let \(\hat{\lambda}_U = \sum_{i,j \in U} \hat{y}_{ij}/b_U\) and \(\hat{\lambda} = \max_{U \in \mathcal{O}_\Delta} \hat{\lambda}_U\). Observe that \(\lambda_U = \hat{\lambda}_U\) if and only if \(\lambda > \max_i \lambda_i\) then \(\lambda = \hat{\lambda}\). Moreover we satisfy \(\sum_j \hat{y}_{ij} \leq b_i\).

Suppose that \(\hat{\lambda} \leq 1 + 3\delta\) and \(\Delta = 1\). Then for all \(U\) we have \(\lambda_U = \Delta \hat{\lambda}_U \leq \Delta \hat{\lambda} \leq \Delta(1 + 3\delta) < 1 + 8\delta\) and \(\max_i \lambda_i \leq (1 - 4\delta) \leq 1 + 8\delta\) for \(\delta \in (0, \frac{1}{16})\). This contradicts the assumption that \(\lambda > 1 + 8\delta\). Therefore, if \(\hat{\lambda} \leq 1 + 3\delta\) then we must have \(\lambda > 1\). Now consider the vertex \(i\) which defined \(\lambda_i\); then

\[
\lambda \geq \lambda_i = \lambda \geq (1 + 4\delta) \Delta \geq (1 + 3\delta) \Delta + \delta \lambda > \hat{\lambda} \lambda + \delta
\]

which implies \(\lambda - \delta \geq \lambda_U\) for every \(U\). In this case \(L = \emptyset\) and \(|\{U : \lambda_U \geq \lambda - \delta^x; U \in \mathcal{O}_\delta\}| = 0\) for \(x \geq 2\). Therefore the remaining case is \(\hat{\lambda} > 1 + 3\delta\). But in this case Theorems 5 and 6 apply! This is because we satisfy \(\sum_j \hat{y}_{ij} \leq b_i\). To find \(L\), compute \(\Delta, \hat{y}_{ij}\) and run the algorithm in Theorem 6 and check if \(\hat{\lambda} > 1 + 3\delta\) based on the sets returned. If the check is true then we can compute \(\lambda = \{\Delta \hat{\lambda}, \max_i \lambda_i\}\) and return the sets satisfying \(\lambda_U \geq \lambda - \delta^3/10\).

Lemma 8. (Proved in Section 5, see Theorem 7) There exists an algorithm for \(LP_4\) uses \(O(m)\) time, \(O(n/\delta)\) space, a single pass over the edges and outputs a solution with \(O(n)\) non-zero edges. This is obtained by using a factor 6 primal-dual approximation algorithm that obeys all the vertex constraints and multiplying that solution by 6.

The next lemma indicates why we needed a primal-dual approximation algorithm in Lemma 8 — because we did not succeed in finding a solution of size \(\beta\) after the scaling then we know that the parameter \(\beta\) is not feasible and should be lowered.

Lemma 9. (Proven in Section 6) If \((1-4\delta)\beta^* \geq \beta\) then (i) we always solve \(LP_4\) (and do not decrease \(\beta\)) and (ii) we can solve \(LP_3\) using \(O(\ln(1/\delta))\) invocations of \(LP_4\) (for different \(\beta \geq 0\)) and using the convex combination of two solutions.

Theorem 10. Algorithm 7 produces a feasible fractional b–Matching of weight at least \((1-14\delta)\beta^*\) and invokes \(LP_4\) at most \(T = O(\delta^{-4}(\ln(1/\delta)) \ln n)\) times.

Proof. The first observation is that based on Lemma 9 the algorithm never decreases \(\beta\) once \(\beta \leq (1 - 4\delta)\beta^*\). Therefore the final value of \(\beta\) is at least \((1 - \delta)(1 - 4\delta)\beta^*\). Observe that the entire algorithm can be analyzed at the final value of the \(\beta\). Since the constraints \(\mathcal{P}[\beta_1] \Rightarrow \mathcal{P}[\beta_2]\) for \(\beta_1 \geq \beta_2\), we apply induction that any step for \(\beta_1\) continues to be a legitimate step for \(\beta_2\). In effect we are running the algorithm simultaneously for all \(\beta\).

When Algorithm 7 stops, all the constraints \(A\) are violated by at most a factor of \(1 + 8\delta\). Scaling the \(y_{ij}\) to \(y''_{ij} = y_{ij}/(1 + 8\delta)\) we ensure that all constraints are satisfied. Note that \(\sum_j y''_{ij} \leq (1 - 4\delta)b_i\). Therefore for any \(U \notin \mathcal{O}_\delta\) we have

\[
2 \sum_{(i,j), i,j \in U} y''_{ij} \leq \sum_{i \in U} \sum_j y''_{ij} \leq \sum_{i \in U} (1 - 4\delta)b_i \leq (1 - 4\delta)||U||_b \leq ||U||_b - 1
\]

therefore all constraints of \(LP_4\) are satisfied. We are guaranteed \(\sum_{(i,j)} w_{ij} y''_{ij} \geq (1 + 8\delta)^{-1}(1 - \delta)\beta\) (the polytope \(\mathcal{P}\) has the \((1 - \delta)\) approximation built in). When Algorithm 7 stops, \(\sum_{(i,j)} w_{ij} y_{ij} \geq (1 + 8\delta)^{-1}(1 - \delta)^2(1 - 4\delta)\beta^* \geq (1 - 14\delta)\beta^*\). The rest of the proof will be similar in spirit to [28], however the analysis is quite different materially. We analyze the number of rounds within phase \(t\); when \(\lambda > \max\{1 + 8\delta, 1 + 6\epsilon, (1 - \epsilon)\lambda_t\}\) and \(\epsilon\) remains unchanged.
Define $z'_U = e^{\lambda U/\delta} b_U$ for all $U \in \mathcal{O}_\delta$. Note $z_U = z'_U$ for $U \in L$ and 0 otherwise. Denote $\{x_1, \{z'_U\}\}$ by the vector $u(\mathcal{O}_\delta)$ and denote $\{x_1, \{z_U\}\}$ by the vector $u(L)$.

$u(L)^T A y = \sum_{i \in L} \lambda_i e^{\lambda_i c} + \sum_{U \in L} \lambda_U e^{\lambda_U c}$ and $\gamma = \sum_{i \in L} \tilde{b}_i x_i + \sum_{U \in L} z_U b_U = \sum_{i \in L} e^{\lambda_i c} + \sum_{U \in L} e^{\lambda_U c}$. Observe that $e^{\lambda_U} \leq \gamma$ since $\lambda = \max_i \lambda_i$ or $\lambda = \max_{U \in \mathcal{O}_\delta} \lambda_U$ for some $U \in L$. Finally $\gamma \leq 2n e^{\lambda_0}$ since $L$ is laminar and therefore has at most $n$ sets. Obviously, $u(\mathcal{O}_\delta)^T A y \geq u(L)^T A y$.

Define $\Psi = \sum_{i \in \mathcal{O}_\delta} \tilde{b}_i x_i + \sum_{i \in \mathcal{O}_\delta} \lambda_i \lambda_U b_U = \sum_{i \in \mathcal{O}_\delta} e^{\lambda_i c} + \sum_{i \in \mathcal{O}_\delta} e^{\lambda_U c}$ and note $\gamma \leq \Psi$. Now, 

(i) If $\lambda_U \leq (1 - \delta^2) \lambda \leq \lambda - \delta^2$ then the corresponding $e^{\lambda_U c} \leq e^{\lambda_0 c} e^{-50 \delta - 1 \ln n} = e^{\alpha_\lambda c/n^{(50/\delta)}}$. There are at most $n^{1/\delta}$ such sets and therefore $\sum_{U: \lambda_U \leq (1 - \delta^2) \lambda} e^{\lambda_U c} \leq e^{\lambda_0 c/n^{(49/\delta)}}$.

(ii) Likewise (assuming $\delta(x-3)/2 \geq 2$) if $\lambda_U \leq (1 - \delta^x + \Delta(x)) \lambda \leq \lambda - \delta^x + \Delta(x)$ then the corresponding $e^{\lambda_U c} \leq e^{\lambda_0 c/n^{50\delta^x + \Delta(x)}}$. Using Theorem 5 we know that there are at most $n^3 + (n/\delta)^{1 + \delta(x-3)/2} \leq 2(n/\delta)^3$ such sets. We can set $\Delta(x) = \frac{3x-3}{2}$ and the total contribution of $\sum_{U: \lambda_U \leq \lambda - \delta^x + \Delta(x)} e^{\lambda_U c} \leq e^{\lambda_0 c/n^{49\delta(x-3)/2 - 4}} \leq e^{\lambda_0 c/n^{50 - 4}} \leq e^{\lambda_0 c/n^{54}}$.

(iii) We now geometrically divide the interval $(x, 3]$ (for the analysis) and recurse on $(\frac{x+3}{2}, 3]$ till $\delta(x-3)/2 < 2$. At this point the number of remaining constraints is small since $n^3 + (n/\delta)^{1 + \delta(x-3)/2} \leq 2(n/\delta)^3$. We will reach the point within $2 + \log \log(1/\delta)$ iterations. Now for the remaining $U \in \mathcal{O}_\delta$ if $U \notin L$ then we have $\lambda_U \leq \lambda - \delta^3$. Each such $e^{\lambda_U c} \leq e^{\lambda_0 c e^{-\delta^3/10}} = e^{\lambda_0 c/n^5}$. Summing up over such $2(n/\delta)^3$ sets the total contribution is still at most $2e^{\lambda_0 c \delta^3/n^2}$.

Since $\frac{1}{n^{50/\delta}} + \frac{2 + \log \log(1/\delta)}{n^{49}} + \frac{\delta x - 3}{n^2} \leq \frac{1}{n}$, we get:

$$\sum_{U: \lambda_U \notin L} e^{\lambda_U c} \leq \frac{e^{\lambda_0 c}}{n} \leq \frac{\gamma}{n} \implies \Psi = \gamma + \sum_{U \notin L, U \notin \mathcal{O}_\delta} e^{\lambda_U c} \leq \gamma \left(1 + \frac{1}{n}\right) \leq 4n e^{\alpha_\lambda} \quad (2)$$

Since $U \in L \implies \lambda_U \geq (1 - \delta^3/10) \lambda$, $\sum j y_{ij} = \lambda_i b_i$, $\sum_{i,j \in U} y_{ij} = \lambda_U b_U$ we have:

$$u(L)^T A y = \sum_{i \in L} e^{\lambda_i c} \lambda_i + \sum_{U \in L} \lambda_U e^{\lambda_U c} \geq \sum_{i: \lambda_i \geq (1 - \delta^3/10) \lambda} e^{\lambda_i c} \lambda_i + \sum_{U \in L} \lambda_U e^{\lambda_U c}$$

$$= \lambda \left(1 - \frac{\delta^3}{10}\right) \left(\gamma - \sum_{i: \lambda_i < (1 - \delta^3/10) \lambda} e^{\lambda_i c}\right)$$

but $\sum_{i: \lambda_i < (1 - \delta^3/10) \lambda} e^{\lambda_i c}$ can again be bounded as $\gamma/n$ exactly as in step (i)-(iii), because $\lambda > 1$ and there are only $n$ terms. This implies $u(L)^T A y \geq \lambda(1 - \delta^3/10)(1 - 1/n) \gamma$. From Equation (2), with some simplification,

$$u(\mathcal{O}_\delta)^T A y \geq u(L)^T A y > (1 + 4\epsilon) \Psi \quad (3)$$

Now for any $\tilde{y} \in \mathcal{P}$, i.e., $\sum j \tilde{y}_{ij} \leq 6b_i$, we have $\sum_{i,j \in U} \tilde{y}_{ij} \leq 6||U||_b/2$ as well as $\sum j \tilde{y}_{ij} \leq 12b_i$ since $\delta \leq \frac{1}{8}$, $\sum_{i,j \in U} \tilde{y}_{ij} \leq 3||U||_b$ implies $\sum_{i,j \in U} \tilde{y}_{ij} \leq 12b_i$, since $b_U \geq \left\lfloor \frac{||U||_b}{2} \right\rfloor - \frac{1}{4}$ for $U \in \mathcal{O}_\delta$ and $||U||_b / \left(\left\lfloor \frac{||U||_b}{2} \right\rfloor - \frac{1}{4}\right)$ is maximized at $||U||_b = 3$. As a consequence $\tilde{\lambda}_i, \tilde{\lambda}_U \leq \lambda_0$. Since we repeatedly take convex combination of the current candidate solution $y$ with a $\tilde{y} \in \mathcal{P}$, and the initial solution satisfies $\lambda \leq \lambda_0$; we have $\lambda_i, \lambda_U$ upper bounded by $\lambda_0$ throughout the algorithm. Therefore $\tilde{\lambda}_U = \sum_{i,j \in U} \tilde{y}_{ij}/b_U \leq 12$. 

9
Note $u(O_\delta)^T A\hat{y} = \sum_i \lambda_i e^{\lambda_i} + \sum_{U \in O_\delta} \lambda_U e^{\lambda_U}$. $u(L)^T A\hat{y}$ has the second summand restricted to $\sum_{U \in L}$. Using first part of Equation (2):

$$u(L)^T A\hat{y} \geq u(O_\delta)^T A\hat{y} - \sum_{U \notin L} \lambda_U e^{\lambda_U} \geq u(O_\delta)^T A\hat{y} - \frac{12}{n}$$

(4)

After the update, let the new current solution be denoted by $\{y''_i\}$. Let $\lambda''_i = \sum_j y''_{ij}/b_i; \lambda''_U = \sum_{i,j \in U} y''_{ij}/b_i$. Note $\lambda''_i = (1 - \sigma)\lambda_i + \sigma\breve{\lambda}_i$ and $\lambda''_U = (1 - \sigma)\lambda_U + \sigma\breve{\lambda}_U$. Since $\lambda_i, \lambda_U \leq 12$ we have all $|\sigma(\lambda_i - \lambda_i)|$ and $|\sigma(\lambda_U - \lambda_U)| \leq \epsilon/4$. For $|\Delta| \leq \frac{\epsilon}{4}$; we have $e^{\epsilon + \Delta} \leq e^{\epsilon}(1 + \Delta + \epsilon|\Delta|/2)$. Therefore:

$$e^{\alpha\lambda''_i} \leq e^{\alpha\lambda_i} (1 + \sigma(\breve{\lambda}_i - \lambda_i) + \epsilon\sigma(\breve{\lambda}_i + \lambda_i)) \quad \text{and} \quad e^{\alpha\lambda''_U} \leq e^{\alpha\lambda_U} (1 + \sigma(\breve{\lambda}_U - \lambda_U) + \epsilon\sigma(\breve{\lambda}_U + \lambda_U))$$

Rearranging and summing over $i, U$ we get

$$\Psi'' = \sum_i e^{\alpha\lambda''_i} + \sum_{U \in O_\delta} e^{\alpha\lambda''_U} \leq \Psi + \left(1 + \frac{\epsilon}{2}\right) \sigma a u(O_\delta)^T A\hat{y} - \left(1 - \frac{\epsilon}{2}\right) \sigma_t a u(O_\delta)^T A\hat{y} \quad \text{(Using Equation 4)}$$

$$\leq \Psi + \sigma a \left(1 + \frac{\epsilon}{2}\right) \left(1 + \frac{12}{n}\right) - \sigma a \left(1 - \frac{\epsilon}{2}\right) \sigma_t a u(O_\delta)^T A\hat{y}$$

$$\leq \Psi + \sigma a \left(1 + \frac{\epsilon}{2}\right) \left(1 + \frac{12}{n}\right) - \sigma a \left(1 - \frac{\epsilon}{2}\right) (\frac{1}{4} \Psi) \quad \text{(Using Equation 3)}$$

$$\leq \left(1 - \sigma a \right) \Psi \quad \text{(Using \(\Psi \leq \Psi'' \)) \text{ and } \frac{\epsilon}{4} \leq \delta \leq \frac{1}{6}}$$

Therefore the potential $\Psi$ decreases. Note that between two phases $e^{(1-8\delta)\lambda_t\alpha} \leq \Psi \leq 4ne^{\lambda_t\alpha}$ from Equation (2). This implies that each phase will end within $O(\frac{\ln n}{\sigma a} + \frac{\delta \lambda_t \alpha}{\sigma a})$ updates to $y$; which is $O(\frac{n\ln n}{\epsilon} + \frac{\lambda_t \lambda \delta^2 \ln n}{\epsilon^2})$ which is $O(\frac{\delta^2 \ln n}{\epsilon})$ since $1 + 8\epsilon \leq \lambda_t \leq 1 + 12\epsilon$ (the previous value of $\epsilon$ being larger by a factor $3/2$ and $\lambda_0 = O(1)$ for the superphase containing the phase $t$. Therefore the number of updates in a phase is bounded by $O(\frac{\delta^2 \ln n}{\epsilon})$. The number of phases in a superphase is at most $\ln \frac{1}{1+12\epsilon} = O(\frac{1}{\delta})$. Therefore the number of updates in a superphase is $O(\frac{\delta^3 \ln n}{\epsilon})$. Now $\epsilon$ is decreased by a factor of $3/2$ and therefore the last two terms dominate (corresponding to $\epsilon = \delta$ and $\epsilon \in [\delta, 3\delta/2]$), giving us $R = O(\delta^{-4} \ln n)$. Note each iteration uses $O(\frac{1}{\delta})$ passes. This proves the bound on $T$ once we have $\beta^*(1 - 4\delta) \geq \beta$. The number of extra steps in decreasing $\beta$ is at most $O(\frac{1}{\delta})$ since $6\beta^* \geq \beta_0 \geq \beta^*$ and is absorbed in $O(\frac{\delta^3 \ln n}{\epsilon})$. 

The running time of Algorithm [1] follows from two parts: (i) Step 7 which contributes the term $O(n \text{poly}\{\delta^{-1}, \ln n\})$ and (ii) Step 10 where the effective edge weights are computed. The effective edge weights can be computed in $O(1)$ time if we precompute the $\sum_{U \in E, j \notin U} z_U$ for each pair of edges $(i, j)$. Note there can be at most at most $\sum_{U \in E, j \notin U} z_U = O(n \text{poly}\{\delta^{-2}\})$ such edges for any $L$. Therefore part (ii) contributes $O(mT)$. We can now conclude Theorem 1.

**Theorem 1 (Fractional b-Matching).** Given any nonbipartite graph, for any $0 < \delta \leq 1/16$, we find a $(1 - 14\delta)$-approximate (to the standard LP relaxation, LP[1]) fractional weighted b-Matching using additional “work” space (space excluding the read-only input) $O(n \text{poly}\{\delta^{-1}, \ln n\})$ and making $T = O(\delta^{-4}(\ln(1/\delta)) \ln n)$ passes over the list of edges. The running time is $O(mT + n \text{poly}\{\delta^{-1}, \ln n\})$. 

10
3 Proof of Theorem 5

We observe a simple fact before we prove Theorem 5.

Fact 11. Recall $\hat{b}_U = \left[ \frac{||U||_b}{2} \right] - f(||U||_b)$ where $f(\ell) = \frac{\delta^2\ell^2}{4}$ and $\delta \in (0, \frac{1}{10}]$. We can verify that $f(\ell)$ is convex, monotonic for $0 \leq \ell \leq 2/\delta$ and:

(\$F1\$): For $3 \leq ||U||_b \leq 2/\delta$ (irrespective of odd or even) we have $\hat{b}_U \geq (1 - \delta) \left[ \frac{||U||_b}{2} \right]$.

(\$F2\$): For any $\ell_1, \ell_2; f(\ell_1) + f(\ell_2) = f(\ell_1 + \ell_2 - 1) - (2\ell_1\ell_2 - 2\ell_1 - 2\ell_2 + 1)\frac{\delta^2}{4}$.

(\$F3\$): For integers $\ell_1, \ell_2, \ell_3, \ell_4 \in [3, 2/\delta]$ such that $\ell_1 + 2t \leq \ell_2 \leq \ell_3 \leq \ell_4 - 2t$ and $\ell_1 + \ell_4 = \ell_2 + \ell_3$, we have $f(\ell_2) + f(\ell_3) \leq f(\ell_1) + f(\ell_4) - 2\ell^2\delta^2$.

Theorem 5 Suppose that we are given a graph $G$ with $n$ vertices and any non-negative edge weights $\{y_{ij}\}$ satisfying (i) $\hat{y}_{ij} = \hat{y}_{ji} \geq 0$ (ii) $\hat{y}_{ii} = 0$ for all $i$, and (iii) $\sum_j \hat{y}_{ij} \leq b_i$ for all $i$. Given $\delta \in (0, \frac{1}{10}]$, define:

$$\hat{\lambda}_U = \frac{\sum_{(i,j):i,j \in U} \hat{y}_{ij}}{b_U} \quad \text{where} \quad b_U = \left[ \frac{||U||_b}{2} \right] - \frac{\delta^2||U||_b^2}{4} \quad \text{and} \quad \hat{\lambda} = \max_{i \in G} \hat{\lambda}_U$$

If $\hat{\lambda} \geq 1 + 3\delta$, the set $L_1 = \{U: \hat{\lambda}_U \geq \hat{\lambda} - \delta^3; U \in \mathcal{O}_\delta\}$ defines a laminar family. Moreover for any $x \geq 2$ we have $|\{U: \hat{\lambda}_U \geq \hat{\lambda} - \delta^3; U \in \mathcal{O}_\delta\}| \leq n^3 + (n/\delta)^{1+\delta(\epsilon-3)/2}$.

Proof. Consider two sets $A_1, A_2 \in \mathcal{O}_\delta$ such that $\hat{\lambda}_{A_1}, \hat{\lambda}_{A_2} \geq \hat{\lambda} - \delta^3 > 1 + 2\delta$ (since $x \geq 2$) and neither $A_1 \cap A_2, A_2 \cap A_1 \neq \emptyset$. For any set $U$ (with $||U||_b \geq 1$, even or odd, large or small) define $\hat{Y}_U = \sum_{(i,j):i,j \in U} \hat{y}_{ij}$ and $\hat{b}_U$. For $||U||_b = 1$ we have $\hat{Y}_U = 0$. Let $\hat{\lambda}_U = \hat{Y}_U/\hat{b}_U$. There are now two cases.

Case 1: $||A_1 \cap A_2||_b$ is even. Let $||A_1 \cap A_2||_b = ||D||_b = 2t$. Let $Q_1 = \sum_{i \in D} \sum_{j \in A_1 - A_2} \hat{y}_{ij}$ (the cut between $D$ and $A_1 - A_2$ using the edge weights $\hat{y}_{ij}$) and $Q_2 = \sum_{i \in D} \sum_{j \in A_2 - A_1} \hat{y}_{ij}$. Without loss of generality, assume that $Q_1 \leq Q_2$ (otherwise we can switch $A_1, A_2$). Let $C = A_1 - A_2$ and $A = A_1$. Let $2\ell - 1 = ||C||_b$ which is odd. Then, using $Q_1 \leq Q_2$ and definitions of $\hat{Y}_C, \hat{Y}_D$ we have $\hat{Y}_C = \hat{Y}_A - Q_1 - \hat{Y}_D$ and $\hat{Y}_D \leq \frac{3}{2}(\sum_{i \in D} \sum_{j} \hat{y}_{ij} - Q_1 - Q_2) \leq \frac{||D||_b}{2} - \frac{Q_1 + Q_2}{2}$.

This implies:

$$\hat{Y}_C \geq \hat{Y}_A - \frac{||D||_b}{2} - \frac{Q_1}{2} + \frac{Q_2}{2} \geq \hat{Y}_A - \frac{||D||_b}{2} = \hat{Y}_A - t$$

Now $\hat{Y}_A = \hat{\lambda}_A \hat{b}_A > (1 + 3\delta)(1 - \delta) \left[ \frac{||A||_b}{2} \right] \geq \frac{||A||_b}{2}$ using Condition $F1$, Fact 11 for $\delta \leq \frac{1}{8}$, and the lower bound on $\hat{\lambda}$. Therefore $\hat{Y}_A > t$ and $\hat{Y}_C > 0$ which means $||C||_b \geq 3$. Therefore we can refer to $\hat{b}_C, \hat{\lambda}_C$. Since $||D||_b = ||A||_b - ||C||_b$,

$$\hat{b}_A - \hat{b}_C = \left[ \frac{||A||_b}{2} \right] - f(||A||_b) - \left[ \frac{||C||_b}{2} \right] + f(||C||_b)$$

$$= \frac{||D||_b}{2} - (f(||A||_b) - f(||C||_b)) = t - t\delta(t + 2\ell - 1)\delta \geq (1 - \delta)t$$

where the last line uses $\frac{1}{2} \geq ||A||_b \geq (2t + 2\ell - 1)$ because $A \in \mathcal{O}_\delta$. From Equations (5) and (6), and $\hat{Y}_C = \hat{\lambda}_C \hat{b}_C, \hat{Y}_A = \hat{\lambda}_A \hat{b}_A$ we get:

$$\hat{\lambda}_C \hat{b}_C \geq \hat{\lambda}_A \hat{b}_A - t = \hat{\lambda}_A \hat{b}_A - t \geq (\hat{\lambda} - \delta^3)\hat{b}_A - t = \hat{\lambda}_A \hat{b}_A - \delta^3\hat{b}_A - t$$

$$\geq \hat{\lambda}_A (\hat{b}_C + (1 - \delta)t) - \delta^3\hat{b}_A - t > \hat{\lambda}_C \hat{b}_C + (1 + 3\delta)(1 - \delta)t - \delta^3\hat{b}_A - t \geq \hat{\lambda}_C \hat{b}_C + \delta t - \delta^3\hat{b}_A$$
Since $\tilde{b}_A \leq 1/\delta$ this implies that $t < \delta^{x-1} \tilde{b}_A \leq \delta^{x-2}$ which contradicts $A_1 \cap A_2 \neq \emptyset$ for $x \geq 2$.

**Case II:** $|A_1 \cap A_2|_b$ is odd. Let $C = A_1 \cup A_2$, and $D = A_1 \cap A_2$. Let $||C||_b = \ell_1 ||A_2||_b = \ell_2$. Even if $||C||_b \geq 1/\delta$ extend the definitions $\tilde{b}_C = \left[ \frac{||C||_b}{2} \right] - f(||C||_b)$ and $\lambda_C = \tilde{Y}_C/\tilde{b}_C$. Now $\tilde{Y}_A \leq \frac{||C||_b}{2}$ since $\sum_j \tilde{y}_{ij} \leq b_i$. Note that if $||C||_b \geq 1/\delta$ then using Condition $F1$, Fact 11

$$\tilde{b}_C \geq (1-\delta) \left[ \frac{||C||_b}{2} \right] = (1-\delta) \left[ \frac{||C||_b}{2} \right] \left( 1 - \frac{1}{||C||_b} \right) \geq (1-\delta)^2 \frac{||C||_b}{2}$$

which implies that $\lambda_C \leq (1-\delta)^{-2} \leq 1 + 3\delta < \lambda$. Now, we always have: $\tilde{Y}_C + \tilde{Y}_D = \tilde{Y}_A_1 + \tilde{Y}_A_2$ and $||C||_b = ||A_1||_b + ||A_2||_b$. Therefore:

$$\tilde{Y}_C + \tilde{Y}_D = \tilde{Y}_A_1 + \tilde{Y}_A_2 = \lambda_A \tilde{b}_A_1 + \lambda_A \tilde{b}_A_2 \geq (\lambda - \delta^x) (\tilde{b}_A_1 + \tilde{b}_A_2)$$

If $||D||_b = 1$, then by Condition $F3$ in Fact 11 $\tilde{b}_C = \tilde{b}_A_1 + \tilde{b}_A_2 - \frac{\delta^2}{4} (2\ell_1 \ell_2 - 2\ell_1 - 2\ell_2 + 1)$:

$$\lambda \tilde{b}_C \geq \lambda_C \tilde{b}_C \geq (\lambda - \delta^x) (\tilde{b}_A_1 + \tilde{b}_A_2) \geq \lambda (\tilde{b}_A_1 + \tilde{b}_A_2) - \delta^x (\tilde{b}_A_1 + \tilde{b}_A_2)$$

$$\geq \lambda \tilde{b}_C + \frac{\delta^2}{4} (2\ell_1 \ell_2 - 2\ell_1 - 2\ell_2 + 1) - \delta^x \left( \frac{\ell_1 + \ell_2 - 2}{2} \right)$$

since $\tilde{b}_A_1 + \tilde{b}_A_2 \leq (\ell_1 + \ell_2 - 2)/2$. Therefore we would have a contradiction if

$$\lambda (2\ell_1 \ell_2 - 2\ell_1 - 2\ell_2 + 1) - 2\delta^{-2} (\ell_1 + \ell_2 - 2) > 0$$

(8)

Observe that for $x \geq 3$ the term $2\delta^{x-2} (\ell_1 + \ell_2 - 2)$ is at most 2 whereas $(2\ell_1 \ell_2 - 2\ell_1 - 2\ell_2 + 1) \geq 7$ since $3 \leq \ell_1, \ell_2 \leq \frac{1}{\delta}$. Since $\lambda > 1$ we have a contradiction for $||D||_b = 1, x \geq 3$.

Now consider $||D||_b \geq 3$. Without loss of generality, $||A_2 - D||_b \geq ||A_1 - D||_b$. Let $||A_1 - D||_b = 2t$. Using Condition $F3$ in Fact 11 $\tilde{b}_C + \tilde{b}_D \leq \tilde{b}_A_1 + \tilde{b}_A_2 - 2t^2 \delta^2$. Note $\lambda_D \leq \lambda$. From Equation 7:

$$\lambda (\tilde{b}_C + \tilde{b}_D) \geq \lambda_C \tilde{b}_C + \lambda_D \tilde{b}_D \geq \tilde{Y}_C + \tilde{Y}_D \geq \lambda (\tilde{b}_A_1 + \tilde{b}_A_2) - \delta^* (\tilde{b}_A_1 + \tilde{b}_A_2)$$

$$\geq \lambda (\tilde{b}_C + \tilde{b}_D) + 2t^2 \delta^2 \lambda - \delta^* (\tilde{b}_A_1 + \tilde{b}_A_2)$$

(9)

Again, this is infeasible if $x \geq 3$ since $\tilde{b}_A_1 + \tilde{b}_A_2 \leq 2/\delta$ and $\lambda \geq 1$. Therefore for $x \geq 3$, in all cases we arrived at a contradiction to $A_1 \cap A_2 \neq \emptyset$. Thus we have proved that $\{U : \lambda_U \geq \lambda - \delta^x; U \in \mathcal{O}_\delta\}$ is a laminar family. We now prove the second part.

Consider $L'_f = \{U : \lambda_U \geq \lambda - \delta^x; U \in \mathcal{O}_\delta; ||U||_b = \ell\}$. From Case I, no two distinct sets $A_1, A_2 \in L'_f$ intersect when $||A_1 \cap A_2||_b$ is even. From Case II for $\ell \geq 5$, they cannot have $||D||_b = 1$ because $(2\ell^2 - 4\ell + 1) > 2(\ell - 2)$ for $\ell \geq 5$. Note $||A_1 - D||_b = ||A_2 - D||_b$ because $||A_1||_b = ||A_2||_b = \ell$. Moreover for $t \geq \delta^{(x-3)/2}$ we would have $2t^2 \delta^2 \lambda > \delta^* (\tilde{b}_A_1 + \tilde{b}_A_2)$ in Equation 7. Therefore two distinct sets $A_1, A_2 \in L'_f$ which intersect, cannot differ by more than $\delta^{(x-3)/2}$ elements. This means that $|L'_f| \leq n (n/\delta)^{\delta^{(x-3)/2}}$ for $\ell \geq 5$ — to see this choose a maximal collection of disjoint sets and every other set has to intersect one of these sets. If we fix a set we can throw out $\delta^{(x-3)/2}$ elements in $\ell \delta^{(x-3)/2}$ ways and include new elements in $n \delta^{(x-3)/2}$ ways. Note $|L'_3| \leq n^3$ and $\ell \leq 1/\delta$. Thus the total number of sets is $n^3 + (n/\delta)(n/\delta)^{\delta^{(x-3)/2}}$. The lemma follows. 

4 Proof of Theorem 6

We first state Lemma 12 which would be used in the proof of Theorem 6. We then state and prove the theorem and finish the section with the proof of Lemma 12.
Lemma 12. Given an unweighted graph $G$ with parameter $\kappa$ and a special node $s$, in time $O(n \text{poly}(\kappa, \log n))$ we can identify a collection $\mathcal{L}$ of odd-sets which (i) each $U \in \mathcal{L}$ does not contain $s$ (ii) each $U \in \mathcal{L}$ defines a cut of at most $\kappa$ in $G$ and (iii) every other odd set not containing $s$ and with a cut less than $\kappa$ intersects with a set in $\mathcal{L}$.

Theorem 6. For a graph $G$ with $n$ vertices and any non-negative edge weights $\hat{y}_{ij} = \hat{y}_{ji}$ such that $\hat{y}_{ii} = 0$ and $\sum_j \hat{y}_{ij} \leq b_i$ for all $i$; and $\delta \in (0, \frac{1}{10}]$, define:

$$\hat{\lambda}_U = \frac{\sum_{(i,j) : i < j \in U} \hat{y}_{ij}}{b_U} \text{ where } b_U = \left\lceil \left\lceil \frac{|U|}{b} \right\rceil - \frac{\delta^2|U|^2}{4} \right\rceil \text{ and } \hat{\lambda} = \max_{U \in \mathcal{O}_\delta} \hat{\lambda}_U$$

If $\hat{\lambda} \geq 1 + 3\delta$ we can find the set $L_2 = \{U : \hat{\lambda}_U \geq \hat{\lambda} - \delta^3/10; U \in \mathcal{O}_\delta\}$ in $O(m' + n \text{poly}(\delta^{-1}, \log n))$ time using $O(n \delta^{-5})$ space where $m' = |\{(i,j)|\hat{y}_{ij} > 0\}|$.

Proof. We first observe that $L_2$ is a laminar family using Theorem 5 and $L_2 \subseteq L_1$. Second, observe that for any $U$ we have $\sum_{(i,j) : i < j \in U} \hat{y}_{ij} \leq \frac{1}{2} \sum_{i \in U} \sum_j \hat{y}_{ij} \leq \frac{1}{2} \sum_{i \in U} b_i = |U|/b/2$. Therefore $\hat{\lambda} \leq \frac{3}{2} / (1 - \frac{\delta^2}{12}) \leq \frac{3}{2} + \delta^2$; the worst case gap between the vertex constraints and odd-set constraints of size up to $1/\delta$ still happen at size $3$ for the said range of $\delta$.

We maintain an estimate $\hat{\lambda}$ of such that $\hat{\lambda} - \delta^3 / 100 < \hat{\lambda}_U \leq \hat{\lambda} \leq \frac{3}{2} + \delta^2$. This estimate can be found using binary search (as described below).

Create a graph $G_\varphi$ with $p_{ij} = \lfloor \varphi \hat{y}_{ij} \rfloor$ parallel edges between $i$ and $j$ where $\varphi = 50/\delta^4$ (this parameter can be optimized but we omit that in the interest of simplicity). This is an unweighted graph. This graph can be constructed in a single pass over $\{(i,j)\}$. We also “merge” all pairs of vertices $i$ and $j$ if $p_{ij}$ exceeds $2\varphi$. Moreover delete vertices $i$ with $2\varphi/\delta$ edges – note that these vertices must have $b_i \geq \sum_j \hat{y}_{ij} > 1/\delta$ and cannot participate in any odd set in $\mathcal{O}_\delta$. This gives us a graph $G_\varphi$ with at most $O(n \delta^{-5})$ edges.

Now for an odd $\ell \in [3, 1/\delta]$ and $\hat{\lambda}$, create $G_\varphi(\ell, \hat{\lambda})$ as follows: Let $q_i = \lfloor \varphi \hat{\lambda}(1 - \delta^2 \ell / 2) \rfloor$ for all $i$. Since $q_i > (1 + \delta)\varphi b_i > \sum_j p_{ij}$ (because $\hat{\lambda}$ is large) we can add a new node $s$ and add $q_i - \sum_j p_{ij}$ edges between $s$ and $i$ (for all$i$). This gives us a graph $G_\varphi(\ell, \hat{\lambda})$ of size $O(n \delta^{-5})$ edges for all $\ell$. Let $\kappa(\ell) = \lfloor \varphi \hat{\lambda}(1 - \delta^2 \ell^2/2) \rfloor + \frac{12\ell}{\delta} + 1 < 2\varphi$. Now:

$$q_i - \kappa(\ell) \geq \varphi \hat{\lambda}(1 - \delta^2 - 1) - \varphi \hat{\lambda}(1 - \delta^2 \ell^2/2) - \frac{12\ell}{\delta} - 1 = \frac{\varphi \hat{\lambda} \delta^2 \ell - 2}{\delta} - \frac{12\ell}{\delta} - 2$$

which is positive for $\varphi = 50/\delta^4$ and $\ell \geq 3$. Therefore $q_i > \kappa(\ell)$.

We now show that for $\text{Cut}(U)$ to be small for any odd set $U$ we must have $\|U\|_b \leq 1/\delta$. Using the definitions of $q_i, p_{ij}$ and $\kappa(\ell)$ we have:

$$\text{Cut}(U) - \kappa(\ell) = \sum_{i \in U} q_i - 2 \sum_{(i,j) : i < j \in U} p_{ij} - \kappa(\ell) \geq \sum_{i \in U} (\varphi \hat{\lambda}(1 - \delta^2 \ell) b_i - 1) - 2\varphi \sum_{(i,j) : i < j \in U} \hat{y}_{ij} - \varphi \hat{\lambda}(1 - \delta^2 \ell^2/2) - \frac{12\ell}{\delta} - 1$$

Using Equation 10 and that $\forall U, \sum_{(i,j) : i < j \in U} \hat{y}_{ij} \leq \|U\|_b/2$, we have:
\[ \text{Cut}(U) - \kappa(\ell) \geq \varphi \lambda (1 - \delta^2 \ell) ||U||_b - |U| - \varphi ||U||_b - \varphi \lambda (1 - \delta^2 \ell^2 / 2) - \frac{12\ell}{\delta} - 1 \]

\[ \geq \varphi \lambda (1 - \delta^2 \ell) ||U||_b - |U| - \varphi ||U||_b - \varphi \lambda - \frac{12\ell}{\delta} - 1 \]

\[ \geq \varphi \lambda (1 - \delta) ||U||_b - \varphi ||U||_b - \varphi \lambda - \delta^2 \varphi ||U||_b \]  
(Since \( \ell \leq 1/\delta, \delta^2 \varphi ||U||_b > |U| + \frac{12\ell}{\delta} + 1 \))

\[ = \varphi \left( \lambda (1 - \delta) ||U||_b - \lambda - (1 + \delta^3) ||U||_b \right) \]

\[ \geq \varphi \left( (1 + 3\delta) (1 - \delta) ||U||_b - \frac{3}{2} - \delta^2 - (1 + \delta^2) ||U||_b \right) \]  
(Since \( 1 + 3\delta \leq \lambda \leq \frac{3}{2} + \delta^2 \))

\[ \geq \varphi \left( 2\delta (1 - 2\delta) ||U||_b - \frac{3}{2} - \delta^2 \right) > \varphi \left( 2(1 - 2\delta) - \frac{3}{2} - \delta^2 \right) > 0 \]

where the last inequality follows from \( ||U||_b > 1/\delta \) and \( \delta \in (0, \frac{1}{16}] \). Therefore no odd-set with \( ||U||_b > 1/\delta \) satisfies \( \text{Cut}(U) \leq \kappa(\ell) \).

We now show Property [1], namely: If \( \tilde{\lambda} - \frac{\delta^3}{100} < \hat{\lambda} \leq \tilde{\lambda} \), then (i) all sets in \( L_2(\ell) \) have a cut which is at most \( \kappa(\ell) \) and (ii) all odd sets of \( G_{\varphi}(\ell, \lambda) \) which do not contain \( s \) and have cut at most \( \kappa(\ell) \) belong to \( L_1(\ell) \). For part (i) for a set \( U \in L_2(\ell) \) with \( ||U||_b = \ell \), note \( |U| \leq ||U||_b = \ell \) and:

\[ \text{Cut}(U) = \sum_{i \in U} q_i - 2 \sum_{(i,j) : i,j \in U} p_{ij} \leq \sum_{i \in U} \varphi \lambda (1 - \delta^2 \ell) b_i - 2 \varphi \sum_{(i,j) : i,j \in U} \tilde{g}_{ij} + |U|^2 \leq \varphi \lambda (1 - \delta^2 \ell) ||U||_b - 2 \varphi \lambda_U \tilde{b}_U + \ell^2 \leq \varphi \lambda (1 - \delta^2 \ell) ||U||_b - 2 \varphi \left( \tilde{\lambda} - \frac{\delta^3}{100} - \frac{\delta^3}{10} \right) \tilde{b}_U + \ell^2 = \varphi \lambda (1 - \delta^2 \ell^2 / 2) + \frac{11\delta^3 \varphi \tilde{b}_U}{50} + \ell^2 = \varphi \lambda (1 - \delta^2 \ell^2 / 2) + \frac{11\tilde{b}_U}{\delta} + \ell^2 \leq \varphi \lambda (1 - \delta^2 \ell^2 / 2) + \frac{11\delta^3}{10} \leq \kappa(\ell) \]  
(since \( b_U < ||U||_b = \ell \leq 1/\delta \))

To prove part (ii) if \( \text{Cut}(U) \leq \kappa(\ell) \) then:

\[ \sum_{(i,j) : i,j \in U} p_{ij} = \frac{1}{2} \left( \sum_{i \in U} q_i - \text{Cut}(U') \right) \geq \frac{1}{2} \left( \sum_{i \in U} (\varphi \lambda (1 - \delta^2 \ell) b_i - 1) - \kappa(\ell) \right) \]

\[ \geq \frac{1}{2} \left( \sum_{i \in U} (\varphi \lambda (1 - \delta^2 \ell) b_i - 1) - \varphi \lambda (1 - \delta^2 \ell^2 / 2) - \frac{12\ell}{\delta} - 1 \right) \]

\[ \geq \varphi \lambda \left( \frac{||U||_b}{2} - \frac{\delta^2 ||U||_b^2}{4} \right) + \frac{\varphi \lambda \delta^2}{4} (||U||_b - \ell)^2 - \frac{|U|}{2} - \frac{12\ell}{\delta} - 1 \]

\[ = \varphi \lambda \tilde{b}_U + \frac{\varphi \lambda \delta^2}{4} (||U||_b - \ell)^2 - \frac{|U|}{2} - \frac{12\ell}{\delta} - 1 \]

But since \( \lambda \geq \lambda_U \) and \( \varphi b_U \lambda_U = \varphi \sum_{(i,j) : i,j \in U} \tilde{g}_{ij} \geq \sum_{(i,j) : i,j \in U} p_{ij} \) we have

\[ \varphi \lambda \tilde{b}_U \geq \varphi \lambda_U \tilde{b}_U \geq \varphi \lambda \tilde{b}_U + \frac{\varphi \lambda \delta^2}{4} (||U||_b - \ell)^2 - \frac{|U|}{2} - \frac{12\ell}{\delta} - 1 \]  
(11)
But that is a contradiction unless $|U|_b = \ell$, otherwise the quadratic term, $\frac{3\delta^3}{100(1 - \frac{3\delta}{4})} – \frac{12\delta^3}{50(1 - \frac{3\delta}{4})} = \frac{12\delta^3}{50(1 - \frac{3\delta}{4})} > \frac{3\delta^3}{100(1 - \frac{3\delta}{4})}$ is larger than the negative terms which are at most $\frac{1}{25} + \frac{12}{25} + 1$ in the RHS of Equation 11. Therefore $\text{cut}(U) \leq \kappa(\ell)$ for an odd-set implies $|U|_b = \ell$. But then Equation 11 implies (again using $|U| \leq |U|_b = \ell$):

$$\varphi \lambda_U b_U \geq \varphi \lambda b_U - \frac{\ell}{2} - \frac{12\ell}{\delta} - 1$$

Now $\tilde{b}_U \geq \frac{\ell}{2}(1 - \frac{3\delta}{4})$ when $|U|_b = \ell \geq 3$; thus we have:

$$\tilde{\lambda}_U \geq \tilde{\lambda} - \frac{\ell}{2\varphi b_U} - \frac{12\ell}{\delta \varphi b_U} - \frac{1}{\varphi b_U} \geq \tilde{\lambda} - \frac{3\delta^3}{100(1 - \frac{3\delta}{4})} - \frac{36\delta^3}{50(1 - \frac{3\delta}{4})} - \delta^4 > \tilde{\lambda} - \delta^3 \geq \lambda - \delta^3$$

in other words, $\text{cut}(U) \leq \kappa(\ell)$ for an odd-set implies $U \in L_1(\ell)$, as claimed in part (ii).

We now apply Lemma 12 to extract a collection $L$ of odd-sets in $G_{\varphi}(\ell, \tilde{\lambda})$, not containing $s$ and cut at most $\kappa(\ell)$ – such that any such set which is not chosen must intersect with some set in the collection $L$.

If we have a maximal collection $L$ then $L \subseteq L_1(\ell)$ by part (ii) of Property 1. Due to Theorem 3, the intersection of two such sets $U_1, U_2 \in L_1(\ell)$ will be either empty or of size $\ell$ by laminarity – the latter implies $U_1 = U_2$. Therefore the sets in $L_1(\ell)$ are disjoint. Any $U \in L_2(\ell) - L$ has a cut of at most $\kappa(\ell)$ using part (i) of Property 1 and therefore must intersect with some set in $L$. This is impossible because $U \in L_2(\ell)$ implies $U \in L_1(\ell)$ and $L \subseteq L_1(\ell)$ and we just argued that the sets in $L_1(\ell)$ are disjoint! Therefore no such $U$ exists and $L_2(\ell) \subseteq L$.

We now have a complete algorithm: we perform a binary search over the estimate $\tilde{\lambda} \in [1 + 3\delta, \frac{3}{2} + \delta^2]$, and we can decide if there exists a set $U \in L_2(\ell)$ in time $O(n \text{ poly}(\delta^{-1}, \log n))$ as we vary $\ell, \tilde{\lambda}$. This gives us $\tilde{\lambda}$. We now find the collections $L$ for each $\ell$ and compute all $\tilde{\lambda}_U$ exactly (either remembering the $\tilde{y}_{ij}$ of the the edges stored in $G_{\varphi}$ or by another pass over $G$). We can now return $\cup_L L_2(\ell)$. Observe that $G_{\varphi}$ does not need to be constructed more than once; it can be stored and reused. The running time follows from simple accounting.

In the remainder of the section we prove Lemma 12.

4.1 Proof of Lemma 12

**Lemma 12.** Given an unweighted graph $G$ with parameter $\kappa$ and a special node $s$, in time $O(n \text{ poly}(\kappa, \log n))$ we can identify a collection $L$ of odd-sets which (i) each $U \in L$ does not contain $s$ (ii) each $U \in L$ defines a cut of at most $\kappa$ in $G$ and (iii) every other odd set not containing $s$ and with a cut less than $\kappa$ intersects with a set in $L$.

**Proof.** The algorithm is given in Figure 2. First, consider the following:

**Theorem 13 (16, 19).** Given a graph with $n$ nodes and $m$ edges (possibly with parallel edges), in time $O(m) + \tilde{O}(mk^2)$ we can construct a weighted tree $T$ that represents all min $s$–$t$ cuts in $G$ of value at most $\kappa$. The nodes of this tree are subsets of vertices. The mincut of any pair of vertices that belong to the same subset (the same node in the tree $T$) is larger than $\kappa$ and for any pair of vertices $i, j$ belonging to different subsets (nodes in the tree $T$) the mincut is specified by the partition corresponding to the least weighted edge in the tree $T$ between the two nodes that contain $i$ and $j$ respectively.
Algorithm 2: Finding a maximal collection of odd-sets

1: $\mathcal{L} \leftarrow \emptyset$. Initially $G' = G$. The node $s \in V(G)$.
2: repeat
3: Assign the $s$ duplicity $b_s = 1$ if $\sum_{i \in V(G')} b_i$ is odd. Otherwise let $b_s = 2$.
4: Construct a tree $T$ that represents all low $s$-$t$ cuts in $G'$ using Theorem 13. The nodes of this tree $T$ correspond to subsets of vertices of $V(G')$.
5: Make the vertex set containing $s$ the root of $T$ and orient all edges towards the root. The oriented edges represent an edge from a child to a parent. Let $D(e)$ indicate the set of descendant subsets of an edge $e$ (including the child subset which is the tail of the edge, but not including the parent subset which is the head of the edge).
6: Using dynamic programming starting at the leaf, mark every edge as admissible/inadmissible based on the $\sum_{S \in D(e)} \sum_{i \in S} b_i$ over the descendant subsets of that edge being odd/even respectively.
7: Starting from the root $s$ downwards, pick the edges $e$ in parallel such that (c1) the weight of $e$ (corresponding to a cut) is at most $\kappa$, (c2) $\sum_{S \in D(e)} \sum_{i \in S} b_i$ is odd and (c3) no edge $e'$ on the path from $e$ to $s$ satisfies (i) and (ii). Let the odd-set $U_e$ corresponding to this edge $e \in T$ be $U_e = \cup_{S \in D(e)} S$.
8: If the odd-sets found are $U_{e_1}, \ldots, U_{e_g}$ then $\mathcal{L} \leftarrow \mathcal{L} \cup \{U_{e_1}, \ldots, U_{e_t}\}$. Observe that the sets $U_{e_r}$ are disjoint and do not contain $s$.
9: Merge all vertices in $\cup_{r=1}^g U_{e_r}$ with $s$. Observe that for any set $U$ that does not contain $s$ and does not intersect any $U_{e_r}$, the cut $\text{cut}(U)$ is unchanged.
10: until no new odd set has been found in $G'$. 
11: return $\mathcal{L}$.

Lemma 14 (Implicit in [27]). For any odd-set $U$ in $G$ with cut $\kappa$, there exists an edge $(u, v)$ in the low min $s$-$t$ cut tree $T$ such that $u \in U$, $v \notin U$ and removing $(u, v)$ from the tree results in two connected components of odd sizes. In addition, one of the components define an odd set with cut $\kappa$ in the original graph $G$.

Proof. (Of Lemma 14) Consider all the edges in the tree $T$ that crosses the boundary of $U$. Removing each such edge from the tree results in two connected components where sizes of two components are both even or both odd. $U$ can be represented as inclusion and exclusion of these sets. Therefore, if the size of these sets are all even, then $|U|_b$ has to be even. So there exists at least one edge $(u, v)$ in the tree $T$ such that its removal results in two connected components of odd sizes. $U$ is an $u - v$ cut and therefore, the weight of $(u, v)$ in the tree $T$ is at most $\kappa$. Choosing the side that does not contain $s$, we obtain an odd set in the original graph. In addition, the corresponding cut size is less than $\kappa$. 

(Continuing with Proof of Lemma 12) All that remains to be proven is that the loop in Algorithm 2 needs to be run only a few times. Suppose after $t'$ repetitions $Q_{t'}$ is the maximum collection of disjoint odd-sets which are $\kappa$-attached and we choose $U_{e_1}, \ldots, U_{e_g}$ to be added to $\mathcal{L}$ in the $t' + 1$st iteration. We first claim that $|Q_{t'+1}| \leq g$. To see this we first map every odd-set in $Q_{t'+1}$ to an edge in the tree as specified by the existence proof in Lemma 14. This map need not be constructive – the map is only used for this proof. Observe that this can be a many to one map; i.e., several sets mapping to the same edge.

Now every edges $e_1, \ldots, e_g$ chosen in Algorithm 2 satisfy the property for all $j$: no edge $e'$ on the path from the head of $e_j$ (recall that the edges are oriented towards the root $s$) to $s$ is one of the edges in our map. Because in that case we would have chosen that edge $e'$ instead of $e_j$.

Now the sets in $Q_{t'+1}$ could not have mapped to any edges in the path towards $s$. Now, if a set in $Q_{t'+1}$ mapped to an edge $e'$ which is a descendant of the tail of some $e_j$ (again, the edges are oriented towards $s$) then this set intersects with our chosen $U_{e_j}$ which is not possible.
Therefore any set in \( Q_{t+1} \) must have mapped to the same edges in the tree; i.e., \( e_1, \ldots, e_g \). But then the vertex at the head of the edge belongs to the set in \( Q_{t+1} \). Therefore there can be at most \( g \) such sets. This proves \( |Q_{t+1}| \leq g \).

We next claim that \( |Q_{t+1}| \leq |Q_t| - g \). Consider \( Q' = Q_{t+1} \cup \{U_{e_1}, \ldots, U_{e_g}\} \). \( Q' \) is a collection of disjoint odd-sets which define a cut of size \( \kappa \) in \( G \) after \( t' \) repetitions. Obviously \( |Q'| = |Q_{t+1}| + g \) and by the definition of \( Q' \), \( |Q'| \leq |Q_t| \). Therefore, \( |Q_{t+1}| \leq |Q_t| - g \).

Therefore, in the worst case, \( |Q_t| \) decreases by a factor \( 1/2 \) and therefore in \( O(\log n) \) iterations over this loop we would eliminate all odd-sets that define a cut of size \( \kappa \) in \( G' \).

\[ \square \]

5 Initial Solutions and Approximations

In this section we provide primal-dual approximation algorithms for both uncapacitated and capacitated \( b \)-Matching. The capacities \( b_i', c_{ij}' \), for vertices and edges respectively, need not be integral for this section. Each edge \((i, j)\) has weight \( w_{ij}' \). In the uncapacitated case \( c_{ij}' = \infty \). The formulation \([\text{LP5}]\) captures the basic constraints which are sufficient for the purposes of this section – we are explicitly writing down a relaxation which omits non-bipartite constraints. The system \([\text{LP6}]\) is the dual of \([\text{LP5}]\). These formulations are undirected and \((i, j) = (j, i)\).

\[
\tilde{\lambda} = \max \sum_{i,j} w_{ij}' y_{ij}
\]
\[
\frac{1}{c_{ij}'} y_{ij} \leq 1 \quad \forall (i, j) \in E
\]
\[
y_{ij} \geq 0
\]
\[\text{LP5}\]

\[
\tilde{\lambda} = \min \sum_i p_i + \sum_{i,j} q_{ij}
\]
\[
\frac{1}{b_i} + \frac{1}{b_j} \leq w_{ij}' \quad \forall (i, j) \in E
\]
\[
p_i, q_{ij} \geq 0
\]
\[\text{LP6}\]

A simple primal-dual algorithm is provided in Algorithm 3. Observe that we maintain a feasible primal and a feasible dual solution. Observe that after the update, for any deleted edge we have \( \frac{b_i}{b_i} + \frac{b_j}{b_j} \geq w_{ij}' \).

**Algorithm 3** Linear time single pass algorithm for capacitated \( b \)-Matching

1. We start with all \( p_i = 0 \). Throughout the algorithm we will maintain the invariant \( p_i \geq \sum_j w_{ij} y_{ij} \).

2. Assume that we have some hypothetical vertex \( v \) which has \( 0 \) weight edges to every other node with \( b_v = \infty \) and \( y_{v,j} = b_j \) for all \( j \).

3. For each new edge \( e = (i, j) \) do
   a. If \( \frac{b_i}{b_i} + \frac{b_j}{b_j} \geq w_{ij}' \) then do nothing, otherwise:
   b. Let \( x = \min \{c_{ij}', b_i', b_j'\} \).
   c. Delete the cheapest \( x \) (fractionally) edges incident to \( i \) (and same for \( j \)). In more detail: Order the edges \( \{(i', j) | y_{i',j} \geq 0\} \) in increasing order of \( w_{i'j}' \). Find \( i(j) \) such that \( \sum_{i' < i(j)} y_{i'(j)} < x \) and \( \sum_{i' < i(j)} y_{i'(j)} \geq x \). Set \( y_{i'(j)} \leftarrow \sum_{i' < i(j)} y_{i'(j)} - x \). For \( i' > i(j) \) keep \( y_{i'j} \) unchanged. For \( i' < i(j) \) set \( y_{i'j} = 0 \).
   d. Set \( y_{i,j} = x \). Increase \( p_i, p_j \) to be at least \( 2 \sum_j w_{ij}' y_{ij}, 2 \sum_i w_{ij}' y_{ij} \) respectively.

**Definition 4.** Let \( \Upsilon = \sum_i p_i \). Define the increase in \( p_i, p_j \) due to the edge \((i, j)\) to be the **direct** contribution of edge \((i, j)\). If the edge \((i, j)\) replaces \( e_1, \ldots, e_\ell \) (possibly the last edge is replaced fractionally) then make two copies of the edge \( e_\ell \), one copy got deleted and the other copy stayed in the solution. Therefore without loss of generality define the **indirect** contribution of the edge \((i, j)\) to be the sum of the direct and indirect contributions of the edges \( e_1, \ldots, e_\ell \).
The direct contribution of any edge \((i, j)\) is at most \(4w_{ij}y_{ij}\) (two vertices, each of whose \(p_i\) value increases by at most \(2w_{ij}y_{ij}\)). We increased \(p_i, p_j\) only when \(\frac{p_i}{b_i} + \frac{p_j}{b_j} < w_{ij}\). Since \(p_i \geq 2 \sum_j w_{ij}y_{ij}\) (and likewise for \(p_j\)) before the edges incident on \(i\) (and some of the edges incident to \(j\)) were deleted; the direct contribution of the edges deleted when \((i, j)\) was inserted is at most \(\frac{1}{2}(2w_{ij}y_{ij})\). To see this, divide \((i, j)\) and the deleted edges infinitesimally; for each infinitesimal copy of \((i, j)\) with \(y_{ij} = \Delta\). If an infinitesimal copy of \((i, j)\) causes the deletion of \(e_1, e_2\) (incident at \(i, j\) respectively, each with the same infinitesimal \(\Delta_{ij}\) amount as \((i, j)\)) then \(w'_{ij} \geq 2(w'(e_1) + w'(e_2))\) because we deleted the cheapest edges. The direct contribution of these edges is \(\Delta(2w'(e_1) + 2w'(e_2))\). Therefore the direct contribution of the edges deleted by the insertion of \((i, j)\) is at most \(\frac{1}{2}(2w'_{ij}y_{ij}) = w'_{ij}y_{ij}\).

Inductively, the indirect contribution of edge \((i, j)\) is also at most \(2w'_{ij}y_{ij}\) using the facts that the weights of the (sets of) edges in a chain of deletions decrease geometrically by factor 2. Therefore \(\sum_i p_i = \Upsilon \leq 6 \sum_{(i,j)} w'_{ij}y_{ij}\). The above accounting of the charge is the “trail of the dead” analysis in [12], and can also be found in the analysis of call-admission algorithms [1]. Therefore if at the end we are left with a set \(S\) of edges; for the capacitated problem we set \(q_{ij} = w'_{ij}y_{ij}\) for \((i, j) \in S\) and 0 otherwise. This is a feasible dual solution and observe that \(\sum_i p_i + \sum_{i,j} q_{ij} \leq 7 \sum_{(i,j) \in S} w'_{ij}y_{ij}\). But this is a feasible dual solution and therefore \(\hat{\beta} \leq 7 \sum_{(i,j) \in S} w'_{ij}y_{ij}\). Furthermore, observe that the solution either saturates and edge or a vertex at each step. Thus:

**Theorem 15.** We can solve the capacitated \(b\)-Matching problem to an approximation factor 7 within the optimum fractional solution. Moreover the number of edges in the solution is \(\min\{m, B\}\).

For the uncapacitated case, the variables \(q_{ij}\) do not exist. Therefore \(\sum_i p_i \leq 6 \sum_{(i,j) \in S} w'_{ij}y_{ij}\). This gives a 6 approximation and we have at most \(O(n)\) edges.

**Theorem 16.** We can solve the uncapacitated \(b\)-Matching problem to an approximation factor 6 of the optimum fractional solution. Moreover the number of edges in the solution is \(O(n)\). Furthermore setting \(y_{ij} = 6y_{ij}\) will give us a solution where \(\sum_{(i,j)} y_{ij}^\dagger w'_{ij} \geq \beta^*\) and \(\sum_j y_{ij}^\dagger \leq 6b_i\) which gives us the initial solution for Algorithm 2 as well as the solution sought after in Lemma 3.

6 Lagrangians and Proof of Lemma 9

In this section we prove Lemma 9. Recall \(\gamma = \sum_i x_i b_i = \sum_{U \in L} z_U b_U\) where \(b_i = (1 - 4\delta)b_i\) and \(\bar{b}_U = \left[\frac{\|U\|_2}{2}\right] - \delta^2\|U\|_2^2/4\).

**Lemma 9.** If \((1 - 4\delta)\beta^* \geq \beta\) then (i) we always solve \([LP3]\) (and do not decrease \(\beta\)) and (ii) we can solve \([LP3]\) using \(O(\ln(1/\delta))\) invocations of \([LP4]\) (for different \(\rho \geq 0\)) and using the convex combination of two solutions. The solution for \([LP3]\) uses \(O(m)\) time, \(O(n/\delta)\) space, a single pass over the edges and outputs a solution with \(O(n)\) non-zero edges. \(\beta^*\) is defined in \([LP1]\) (repeated for convenience).

\[
\mathcal{L}(\gamma, \beta) = \sum_{(i,j)} w_{ij}y_{ij} - \beta\left(\sum_{(i,j)} y_{ij}(x_i + x_j + \sum_{U \in L, j \in U} z_U)\right) \geq \beta - \gamma \\

\mathcal{P} : \left\{ \sum_j y_{ij} \leq 6b_i \quad \forall i \in V, \quad y_{ij} \geq 0 \right\}
\]
\[ \sum_{(i,j)} \tilde{y}_{ij}(x_i + x_j + \sum_{u \in L, i,j \in U} z_u) \leq \gamma \quad \text{LP3} \]

Proof. Consider the optimum solution of \text{LP1} given by \{y_{ij}^*\} and define \[ y_{ij}' = \frac{1}{1 - 4\delta} y_{ij}^* \]. This solution satisfies \[ \sum_{(i,j)} w_{ij} y_{ij}' = \frac{1}{1 - 4\delta} \beta^* \] and \[ \sum_i y_{ij}' \leq \frac{1}{1 - 4\delta} b_i \] and \[ \sum_{(i,j): i,j \in U} y_{ij}' \leq \frac{1}{1 - 4\delta} \left[ ||U||_2 \right] \] for any \( U \). In particular we satisfy \[ \sum_{(i,j): i,j \in U} y_{ij}' \leq \frac{||U||_2}{2} - \frac{\delta^2 ||U||_2}{4} = \tilde{b}_U \].

Since \( x_i, z_U \) are non-negative, \{y_{ij}'\} satisfies the linear combination that

\[ \sum_{i} x_i \sum_{(i,j)} y_{ij}' + \sum_{U \in L} \sum_{(i,j): i,j \in U} z_U y_{ij}' \leq \sum_{i} x_i \tilde{b}_i + \sum_{U \in L} z_U \tilde{b}_U \leq \sum_{i} x_i \tilde{b}_i + \sum_{U \in O} z_U \tilde{b}_U = \gamma \]

The left hand side rearranges to: \[ \sum_{(i,j)} y_{ij}'(x_i + x_j + \sum_{U \in L, i,j \in U} z_U) \leq \gamma \] which implies that \text{LP3} is feasible as long as \( \beta \geq (1 - 4\delta) \beta^* \). And since \( \delta \geq 0 \) we have: \[ \mathcal{L}(y', \delta) \geq \beta^* - \delta \gamma \] which implies that \text{LP4} is feasible with the stronger constraint that \[ \sum_j y_{ij}' \leq b_i \].

Given an \( L \) we can precompute and store the quantities \( (x_i + x_j + \sum_{U \in L, i,j \in U} z_U) \) for all \( (i,j) \) irrespective of \( i \) in \( E \) or otherwise. This can be done in \( n/\delta \) time and space since each set of \( L \) is at most of size \( O(1/\delta) \) and therefore we affect at most \( O(n/\delta) \) edges. The problem \text{LP4} reduces to finding a \( b \)-Matching (ignoring the odd set constraints) using “effective weights” instead of \( w_{ij} \).

Suppose we had a single pass \( O(m) \) time 6 approximation algorithm using \( O(n) \) edges, then we can find that solution \{y_{ij}'\} and simply set \[ y_{ij}' = \frac{6}{1 - 4\delta} y_{ij}^* \]. The approximation factor guarantees that the contribution of \[ y_{ij}' \] is more than \( (1 - 4\delta) \beta^* - \delta \gamma \).

Note that the solution \{y_{ij}'\} only needs to satisfy \[ \sum_j y_{ij} \leq 6b_i \]. We show in Theorem 16 in Section 5 how to find such a 6 approximation (along with how to compute an initial solution), but any \( O(m) \) time \( c \)-approximation will work (making \( \lambda_0 = 2c \) in Algorithm 1). To solve \text{LP3} we need one final ingredient:

**Lemma 17.** Suppose \( \mathcal{P} \) is a polytope such that \( \mathcal{P} \subset \{y; A_2 y \geq 0; y \geq 0\} \) and \( \beta_1, \beta_2 \geq 0 \). Consider the system \text{LP5}:

\[ \{y; A_1 y \geq \beta_1; A_2 y \leq \beta_2; y \in \mathcal{P}\} \quad \text{LP7} \]

For any \( r > 0 \), \( 1 \geq q > 0 \), given \text{LP7} and a subroutine that finds \( y \in \mathcal{P} \) such that \( (A_1 - q A_2) y \geq q(\beta_1 - \beta_2) \) for any \( q \geq 0 \), we can find \( y \in \mathcal{P} \) such that \( A_1 y \geq q(1 - r) \beta_1 \) and \( A_2 y \leq \beta_2 \) with \( O(\log \frac{1}{r}) \) invocations of the subroutine.

Proof. Let \( \mathcal{L}(y, q) = (A_1 - q A_2) y \) and let \( y^0 \) be the solution returned by the subroutine for \( q \). If \( y^0 \) satisfy \( A_2 y^0 \leq \beta_2 \), \( y^0 \) is our solution. For \( q = \frac{\beta_1}{\beta_2} \), we use \( y^{\text{max}} = 0 \) instead of the subroutine which satisfies \( A_2 y^0 \leq \beta_2 \). Now we use binary search to find \( q^+, q^- \) such that \( 0 \leq q^+ - q^- \leq \frac{\beta_1}{\beta_2} \) with \( A_2 y^{q^+} \geq \beta_2 \) and \( A_2 y^{q^-} \leq \beta_2 \). This step takes \( O(\log \frac{1}{r}) \) invocations of the subroutine. We take a linear combination \( y = a y^{q^+} + (1 - a) y^{q^-}, a \in [0, 1] \) such that \( A_2 y = \beta_2 \). Since \( y^{q^+}, y^{q^-} \in \mathcal{P} \), their linear combination \( y \) is also in \( \mathcal{P} \). Note that

\[ a \mathcal{L}(y^{q^+}, q^+) + (1 - a) \mathcal{L}(y^{q^-}, q^-) \geq a q \beta_1 - q \beta_2 - a(q^+ - q^-) q \beta_2 \geq q(1 - r) \beta_1 - q^- \beta_2 \]
Lemma 19. (Second Phase) If $in$ which satisfies $LP_1$ Theorem 2, then

$$A_1y = aL(y^{o^+}, e^+) + (1-a)L(y^{o^-}, e^-) + a\phi^+ A_2y^{o^+} + (1-a)\phi^- A_2y^{e^-}$$

$$\geq (1-r)q\beta_1 - \phi^- \beta_2 + a(\phi^+ - \phi^-)A_2y^{\phi^+} + A_2(a\phi^- y^{\phi^+} + (1-a)\phi^+ y^{e^-})$$

$$\geq (1-r)q\beta_1 - \phi^- \beta_2 + A_2 (a\phi^- y^{\phi^+} + (1-a)\phi^+ y^{e^-}) \quad \text{(Using $A_2 y^{\phi^+}$)}$$

$$\geq (1-r)q\beta_1 - \phi^- \beta_2 + A_2 (a\phi^- y^{\phi^+} + (1-a)\phi^+ y^{e^-}) \quad \text{(Using $y = ay^{\phi^+} + (1-a)y^{e^-}$)}$$

$$\geq (1-r)q\beta_1 - \phi^- \beta_2 + (1 - A_2 y) = (1-r)q\beta_1 \quad \text{(Using $\beta_2 = A_2 y$)}$$

Lemma 17 follows.

Lemma 9 follows from applying Lemma 17 with $q = 1$ and $r = \delta$.

7 Rounding Fractional Uncapacitated $b$-Matchings

In this section, we discuss a space and time efficient rounding algorithm.

**Theorem 2 (Integral $b$-Matching).** Given a fractional $b$-matching $y$ for a non-bipartite graph which satisfies $LP_1 b$ (parametrized over $b$) where $|\{(i,j) | y_{ij} > 0\}| = m'$, we find an integral $b$-Matching of weight at least $(1- 2\delta) \sum_{(i,j) \in E} w_{ij} y_{ij}$ in $O(m'\delta^{-3}\log(1/\delta))$ time and $O(m'/\delta^2)$ space.

The algorithm is given in Algorithm 4. We prove the following lemma:

**Lemma 18. (First Phase and the Output Phase)** Suppose that all vertex constraints are satisfied and $\sum_j y_{ij} \leq b_i - 1$. Then, for any odd set $U$ that contains $i$, the corresponding odd set constraint is satisfied. The fractional solution $\{y_{ij}^{(1)}\}$ obtained in the first phase of Algorithm 4 is feasible for $LP_1 b^{(4)}$ — and a an integral $M^{(1)}$ which is a $(1- 2\delta)$-approximation of $LP_1 b^{(1)}$ can be output along with $M^{(0)}$ to satisfy Theorem 4.

Proof. For any $U \in O, i \in U, \sum_{i' \in U} \sum_{j} y_{i'j} \leq 0 \sum_{i' \in U} \sum_{j} y_{i'j} \leq 1/2 \sum_{i' \in U} (\sum_{j} b_{ij'}) - 1 = \frac{\|U\|_b - 1}{2} = \frac{\|U\|_b}{2}$. Thus it follows that any vertex which has an edge incident to it in $M^{(0)}$ cannot be in any violated odd-set in $LP_1 b^{(4)}$. Then any violated odd-set in $LP_1 b^{(4)}$ with respect to $\{y_{ij}^{(1)}\}$ must also be a violated odd-set in $LP_1 b$; contradicting the fact that we started with a $\{y_{ij}^{(0)}\}$ feasible for $LP_1 b$. Now $M^{(0)} \bigcup M^{(1)}$ is feasible since both are integral and we know that $b_i^{(1)}$ must be $b_i - \sum_j y_{ij}^{(0)}$. Observe that $w(M^{(0)}) \geq (1- \delta) \sum_{(i,j) \in E} w_{ij} (y_{ij} - y_{ij}^{(1)})$ where $w(M^{(0)}) = \sum_{j \in E} y_{ij}^{(0)} w_{ij}$. Therefore if $w(M^{(1)}) \geq (1- \delta) \sum_{(i,j) \in E} w_{ij} y_{ij}^{(1)}$ then $w(M^{(0)}) + w(M^{(1)})$ is at least $(1- 2\delta) \sum_{(i,j) \in E} w_{ij} y_{ij}$ as desired.

**Lemma 19. (Second Phase)** If $\{y_{ij}^{(1)}\}$ satisfies $LP_1 b^{(4)}$ over $V$, then $\{y_{ij}^{(2)}\}$ satisfies $LP_1 b^{(2)}$ over $G^{(2)}$. Moreover $\sum_{i,j} w_{ij} y_{ij}^{(2)} = (1- \delta) \sum_{i,j} w_{ij} y_{ij}^{(1)}$

Proof. Observe that any vertex which participates in any split produces vertices which have (fractionally) at least $t$ edges. After scaling we have $1- \delta) \sum_{j} y_{ij}^{(1)} \leq \sum_{j} y_{ij}^{(1)} - \delta \leq \sum_{j} y_{ij}^{(1)} - 2 \leq b_i^{(2)} - 1$ from the definition of $b^{(2)}$ in line (3b) of Algorithm 4. Therefore the new vertices cannot be in any violated vertex or set constraint; from the first part of Lemma 18 (now applied to $LP_1 b^{(4)}$ instead of $LP_1 b$). Therefore the Lemma follows.
Algorithm 4 Rounding a fractional $b$-Matching

1: First Phase: Removing edges with large multiplicities $t = \lceil 2/\delta \rceil$.
   (a) If $y_{ij} \geq t$ add $\hat{y}^{(0)}_{ij} = \lfloor y_{ij} \rfloor - 1$ copies of $(i, j)$ to $\mathcal{M}^{(0)}$.
   (b) Set $y^{(1)}_{ij} = 0$ if $y_{ij} \geq t$ and $y^{(1)}_{ij} = y_{ij}$ otherwise.
   (c) Let $b^{(1)}_i = \min \{ b_i - \sum_j \hat{y}^{(0)}_{ij}, \lfloor \sum_j y^{(1)}_{ij} \rfloor + 1 \}$.

2: Second Phase: Subdividing large capacity vertices.
   (a) While there exists a vertex $i$ s.t. $\sum_j y^{(1)}_{ij} \geq 3t$ do
      (i) Observe that given a set of numbers $q_1, \ldots, q_k$ such that each $q_j \leq 1$ and $\sum_j q_j = Y \geq 3$; we
          can easily partition the set of numbers such that each partition $S$ satisfies $1 \leq \sum_{j \in S} q_j \leq 3$.
      (ii) Order the vertices adjacent to $i$ arbitrarily. Select the prefix $S$ in that order such that the sum
          is between $t$ and $2t$ (each edge is at most $t$ from Step 1b). Create a new copy $i'$ of $i$ with this
          prefix and $y^{(1)}_{ij'} = y^{(1)}_{ij}$ for $j \in S$ and delete the edges from $S$ incident to $i$.
   (b) If no copies of $i$ were created then $b^{(2)}_i = b^{(1)}_i$. For every new $i'$ (corresponding to $i$) created from
       the partition $S$ (which may have now become $S'$ with subsequent splits), assign $b^{(2)}_{i'} = \lfloor \sum_{j \in S'} y^{(1)}_{ij} \rfloor$.
       Note $b^{(2)}_i \leq 3t$ for all vertices. We now have a vertex set $V^{(2)}$. Set $y^{(2)}_{ij} = (1 - \delta y^{(1)}_{ij})$ for $i, j \in V^{(2)}$.

3: Third Phase: Reduction to weighted matching.
   (a) For each $i \in V^{(2)}$ with $b^{(2)}_i$, create $i(1), i(2), \ldots, i(b^{(2)}_i)$.
   (b) For each edge $(i, j)$, create a complete bipartite graph between $i(1), i(2), \ldots$ and $j(1), j(2), \ldots$ with
       every edge having weight $w_{ij}$. Let this new graph be $G^{(3)}$. As an example:

   \[ \begin{array}{ccc}
       \bullet & \bullet & \bullet \\
       \bullet & \bullet & \cdot \\
       \bullet & \cdot & \cdot \\
   \end{array} \]

   (c) Run any fast approximation for finding a $(1 - \epsilon)$-approximate maximum weighted matching in $G^{(3)}$
       let this matching be $\mathcal{M}^{(3)}$.

4: Output: $\mathcal{M}^{(0)} \cup \mathcal{M}^{(1)}$. Observe (i) Matching $\mathcal{M}^{(3)}$ implies a $b$-matching $\mathcal{M}^{(2)}$ in $G^{(2)}$ of same weight
       (merge edges). (ii) Matching $\mathcal{M}^{(2)}$ implies a $b$-matching $\mathcal{M}^{(1)}$ in $G^{(1)}$ of same weight (merge vertices).

Finally it is easy to see that any integral $b$-Matching in $G^{(2)}$ has an integral matching in $G^{(3)}$
of the same weight and vice versa — moreover given a matching for $G^{(3)}$ the integral $b$-Matching
for $G^{(2)}$ can be constructed trivially. Also, the number of edges in $G^{(3)}$ is at most $O(\delta^{-2} m')$
since each vertex in $G^{(2)}$ is split into $O(\delta^{-1})$ vertices in $G^{(3)}$. We are guaranteed a maximum
$b$-Matching in $G^{(2)}$ of weight at least $\sum_{(i,j) \in E^{(2)}} w_{ij} y^{(2)}_{ij}$ since $\{y^{(2)}_{ij}\}$ satisfies $\text{LP}[b^{(2)}]$ over $G^{(2)}$.
Therefore we are guaranteed a matching of the same weight in $G^{(3)}$. Now, we use the approximation
algorithm in [10, 11] which returns a $(1 - \delta)$-approximate maximum weighted matching in $G^{(3)}$ in
$O(m' \delta^{-3} \log(1/\delta))$ time and space. From the $(1 - \delta)$-approximate maximum matching we can
construct a $b$-Matching in $G^{(2)}$ of the same weight (and therefore a $b$-Matching $\mathcal{M}^{(1)}$ in $G^{(2)}$ of
the same weight). Theorem 2 follows.
8 The Capacitated b–Matching Problem

In this section, we present algorithms for the capacitated b–Matching problems in general graphs. We reduce the capacitated b–Matching problem to the uncapacitated b–Matching problem and apply the algorithm in Section 2. However, a naive reduction does not provide a near linear time algorithm as desired. The capacitated b–Matching problem is defined as follows:

Definition 2. [22, Chapters 32 & 33] In the Capacitated b–Matching problem, we have an additional restriction for every edge \((i, j) \in E\) that \(y_{ij} \leq c_{ij}\) where \(c_{ij}\) are also given in the input (also assumed to be an integer in \([0, \text{poly} n]\)). A problem with \(c_{ij} = 1\) for all \((i, j) \in E\) is also referred to as an “unit capacity” or “simple” b–Matching problem in the literature.

Definition 5 (Volume of Sets & Odd-Sets for Capacitated b–Matching). Given a graph \(G = (V, E, c)\), with \(|V| = n\) and \(|E| = m\), and non-negative values \(b_i\) for each \(i \in V\), define the volume of a set \(U \subseteq V\) to be \(||U||_{b,c} = \sum_{i \in U} b_i + \sum_{j \in V \setminus U} c_{ij}\). Define \(\mathcal{O} = \{|U| \mid |U||_{b,c} \text{ is odd}\}\) and \(\mathcal{O}_\delta = \{|U| \mid U \in \mathcal{O}; |U||_{b,c} \leq 1/\delta\}\).

The standard exact linear programming formulation for capacitated b–Matching is similar to the LP for uncapacitated b–Matching (LP8). LP8 is the standard LP, LP8 also has an integral optimum solution when \(b_i\) and \(c_{ij}\) are integers (See also [29, Chapter 32]).

\[
\beta^* = \text{LP8}(b, c) = \max \sum_{(i,j)} w_{ij} y_{ij} \quad \text{s.t.} \quad \begin{array}{l}
\sum_j y_{ij} \leq b_i \quad \forall i \in V \\
\sum_{j : (i,j) \in U} y_{ij} \leq |U||_{b,c}/2 \quad \forall U \in \mathcal{O} \\
0 \leq y_{ij} \leq c_{ij} 
\end{array}
\] (LP8)

Statement of Results and Roadmap: Theorems 3 and 4 summarize the results for capacitated b–Matching problem. Note that the restriction \(\sum_{(i,j) \in E} w_{ij} \leq T^* b^*\) of Theorem 3 is explicitly used in Theorem 4. Section 8.1 presents the overview of the algorithm and how the ideas of Section 2 are used. Sections 8.3 and 8.2 prove modularized parts of the proof of Theorem 3. Section 8.4 proves Theorem 4.

Theorem 3 (Fractional Capacitated b–Matching). Given any nonbipartite graph, for any \(0 < \delta \leq 1/16\), we find a \((1 - 14\delta)\)-approximate fractional weighted capacitated b-Matching using \(O(mR/\delta + \min\{B, m\} \text{poly}\{\delta^{-1}, \ln n\})\) time, \(O(\min\{m, B\} \text{poly}\{\delta^{-1}, \ln n\})\) additional “work” space, making \(R = O(\delta^{-4}(\ln^2(1/\delta)) \ln n)\) passes over the list of edges where \(B = \sum b_i\). Moreover, the feasible fractional solution \(\{\hat{y}_{ij}\}\) has the property that \(\sum_{(i,j) \in E} w_{ij} \leq 2R\beta^*\) where \(E = \{(i, j) \mid (i, j) \in E, \hat{y}_{ij} > 0\}\) and \(\beta^*\) is the weight of the optimum capacitated b–Matching.

Theorem 4 (Integral Capacitated b–Matching). Given a feasible fractional (as described in Theorem 3) capacitated b–Matching \(y\) for a non-bipartite graph, we can find an integral b-matching of weight at least \((1 - \delta)\sum_{(i,j)} w_{ij} y_{ij} - \delta\beta^*\) in \(O(m'T\delta^{-3}\ln(T/\delta))\) time and \(O(m'\delta^2)\) space where \(m' = |E|\). If the fractional solution is \((1 - 14\delta)\)-approximate then we have a \((1 - 16\delta)\)-approximate integral solution.

Algorithm Overview: Assume that the graph is presented as a read only list \((..., (i, j, w_{ij}, c_{ij}), ...)\) in arbitrary order. We are given a graph \(G = (V, E)\) with \(|V| = n, |E| = m\) and \(B = \sum b_i\). We also assume that the working space is \(O(B)\) rather than \(O(n)\) because it is possible for an approximation b–Matching to have \(B\) edges. For example, suppose that \(G\) is a complete unweighted graph.
with \( b_i = n \) for all \( i \) and \( c_{ij} = 1 \) for all \((i, j)\). Then, the optimal solution or any constant-factor approximation solution requires \( O(B) = O(n^2) \) edges. Our approach will be to use a reduction to the uncapacitated \( b \)-Matching problem while maintaining the following invariant:

**Invariant 1.** During the execution of the primal dual algorithm we will maintain that \( y_{ij} \leq c_{ij} \) (for any current solution as well as any updates).

Assuming that the invariant holds, we can reduce the capacitated \( b \)-Matching problem into an equivalent \( b \)-Matching problem – some of this connection is known in the literature (see [29, Chapter 32]). However we need slightly different reduction parameters since we want approximation preserving reductions, which were not considered before. Intuitively, the graph obtained from this reduction can be viewed as a “long code” of the capacitated graph, and we will run Algorithm 1 on that encoding. We then indicate which substeps of Algorithm 1 are modified to ensure the invariant. Note that a primal candidate is infeasible and thus, it can violate the invariant even if the invariant is written as a constraint. We begin with the following:

### 8.1 Long and Short Representations

**Definition 6.** Given a graph \( G = (V, E) \) with vertex and edge capacities. Consider subdividing each edge \( e = (i, j) \) into \((i, p_{ij,i}), (p_{ij,i}, p_{ij,j}), (p_{ij,j}, j)\) where \( p_{ij,i}, p_{ij,j} \) are new additional vertices with capacity \( b_{p_{ij,i}} = b_{p_{ij,j}} = c_{ij} \). For \( i \in V \) set \( b_i = b \). We use the weights \( \frac{1}{2}w_{ij}, 0, \frac{1}{2}w_{ij} \) for \((i, p_{ij,i}), (p_{ij,i}, p_{ij,j}), (p_{ij,j}, j)\) respectively. Denote this graph as \( \text{LONG}(G) \). Let the vertex set of \( \text{LONG}(G) \) be \( V_L \) and the edge set be \( E_L \). We use \( w_{ij} \) for the edge weights and \(|U|_{b,c} = \sum_{i \in U} b_i \) for \( U \subseteq V_L \) to indicate the volume of a set \( U \) in \( \text{LONG}(G) \). The odd-sets in \( \text{LONG}(G) \) are \( O_L \). Recall that \( O_\delta = \{ U \in O, ||U||_b \leq 1/\delta \} \), where \( O \) was the set of odd sets over \( V \). Likewise define \( O_{\delta_L} = \{ U \in O_L, |U|_{b,c} \leq 1/\delta \} \).

The notation distinguishes \( G \) and \( \text{LONG}(G) \) since they will need to be referred simultaneously. Figure 1 provides an example of the transformation.

![Figure 1: Example of G and LONG(G) – the edges are labeled with their respective capacities in G. These become vertex capacities in the transformed instance LONG(G).](image)

**Definition 7.** Given an assignment \( y = \{y_{ij}\} \) over \( G \) such that \( y_{ij} \leq c_{ij} \) for all \((i, j) \in E \) we define \( \text{LONG}(y) = \{y'_{ij,j'}\} \) over \( \text{LONG}(G) \) as follows: we set \( y_{i,p_{ij,i}} = y'_{p_{ij,i},j} = y_{ij} \) and \( y_{p_{ij,i}, p_{ij,j}} = c_{ij} - y_{ij} \).

As above, given an assignment \( y^c = \{y'_{ij,j'}\} \) over \( \text{LONG}(G) \) such that \( y_{i,p_{ij,i}} = y'_{p_{ij,i},j} \) and \( y_{p_{ij,i}, p_{ij,j}} = c_{ij} \) we define \( \text{SHORT}(y^c) \) over \( G \) as follows: we set \( y_{ij} = y'_{i,p_{ij,i}} \). As long as the capacity constraints are met initially by \( y, y^c \) note that \( \text{SHORT}(\text{LONG}(y)) = y \) and \( \text{LONG}(\text{SHORT}(y^c)) = y^c \).

Assume \( y_{ij} = 0 \) for \((i, j) \notin E \) and likewise \( y'_{ij} = 0 \) for \((i, j) \notin E_L \) for ease of notation.

Note that Invariant 1 implies that \( y \) is non-negative vector if and only if \( \text{LONG}(y) \) is non-negative. It is not difficult to see that the capacitated \( b \)-Matching in \( G \) is equivalent to (uncapacitated) \( b \)-Matching in \( \text{LONG}(G) \). This is encoded in the next theorem.

**Theorem 20.** [29, Implicit in proof of Theorem 32.2, Vol A, pages 564–565] For any \( \lambda^c_0 \geq 2 \), \( \text{LONG}(y) \) is feasible for \( \text{LP10} \) on \( \text{LONG}(G) \) iff \( y \) is feasible for \( \text{LP9} \) on \( G \).
If we start from LP11 instead of LP2 then the subproblem corresponding to LP3 which we need to solve is given in LP12.

\[
\sum_{(i,j) \in E} y_{ij} \leq c_{ij} \quad \forall (i,j) \in E
\]

Moreover under the restrictions to LP9 and LP10:

\[
\begin{align*}
\sum_{(i,j) \in E;i,j \in U} y_{ij} & \leq \left\lfloor \frac{|U| b_{c}}{2} \right\rfloor \quad \forall U \in O \iff \\
\sum_{(i,j) \in E;i,j \in U} w_{ij} y_{ij} & \geq (1 - \delta)\beta
\end{align*}
\]

Note that for any \( i \in V_L \setminus V \) we have \( \sum_{j} y_{ij} = b_{ij}^c \); which explains the \( \lambda_0^c \geq 2 \) precondition. Moreover the property \( \sum_{(i,j)} w_{ij} y_{ij} \geq \beta \) is preserved as \( \sum_{(i',j')} w_{i'j'}^c y_{i'j'}^c \geq \beta \) is a consequence of the special weight sequence \( \frac{1}{2} w_{ij}, 0, \frac{1}{2} w_{ij} \). Furthermore if \( \beta^* = \max \sum_{(i,j)} y_{ij} w_{ij} \) and \( \beta' = \max \sum_{(i',j')} y_{i'j'}^c w_{i'j'}^c \) then \( \beta^* = \beta' \). Thus the problem of maximum weighted capacitated \( b \)-Matching can be viewed as max \( \beta \) subject to LP10 or LP9 when \( \lambda_0 = 2 \).

An Algorithm that almost suffices: Suppose that we run Algorithm 1 on page 7 on LONG(G) starting from the formulation LP11 (instead of LP2 using \( m \) instead of \( n \) in determining \( \alpha_i \); note \( m \leq n^2 \)) and solving appropriate subproblems in lines 11-13 of Algorithm 1. Note \( \lambda_0 \geq 2 \).

If we start from LP11 instead of LP2 then the subproblem corresponding to LP3 which we need to solve is given in LP12.

\[
\sum_{(i,j) \in E} \tilde{y}_{ij}(x_i^c + x_j^c + \sum_{U \in L^c,i,j \in U} z_{ij}^c) \leq \gamma^c = \sum_{i \in V_L} x_i^c \tilde{b}_i^c + \sum_{U \in L^c} z_{ij}^c \tilde{b}_U \quad \text{s.t.} \quad \tilde{y}^c \in Q^c[\beta, \lambda_0^c] \quad (LP12)
\]

where the laminar family is \( L^c \) in this case. We use the following definition and claim before proceeding further.

---

\(^5\) This is slightly different from the weights \( w_{ij}, w_{ij}, w_{ij} \) used in [29] Theorem 32.4, Vol A, page 567], which showed that capacitated \( b \)-Matching reduced to uncapacitated \( b \)-Matching with a “constant shift” in the objective function of \( \sum_{(i,j) \in E} z_{ij} w_{ij} \). A (constant) shift at the top level of an approximate primal-dual algorithm will force the substeps of that algorithm to be more accurate — this is not an issue for exact solutions.
Definition 8. Denote \( \eta_{ij'} = x_{ij'}^c + x_{ij'}^r + \sum_{U \in \mathcal{L}_c : j' \in U} z_{ij'}^r \) for an edge \((i', j') \in E_c\). For an edge \((i, j) \in E\) (in \(G\)) define \(g_{ij} = \eta_{ip_{ij},i} + \eta_{pj_{ij},j} - \eta_{pj_{ij},p_{ij,j}}\). Define \(K = \sum_{(i,j) \in E} \eta_{pj_{ij},p_{ij,j}} c_{ij}\). 

Claim 1. The system \(LP_{12}\) is equivalent to \(LP_{13}\) using Theorem 20. Note \(\lambda_0^c \geq 2\).

\[
\begin{align*}
\sum_{(i,j) \in E} g_{ij} y_{ij} &\leq \gamma^c - K \\
\sum_{(i,j) \in E} w_{ij} y_{ij} &\geq (1 - \delta) \beta \\
\sum_{j} y_{ij} &\leq \lambda_0^c h_i/2 \quad \forall i \in V \\
y_{ij} &\leq c_{ij} \quad \forall (i,j) \in E \\
y_{ij} &\geq 0 \\
\end{align*}
\]

(LP13)

The subproblem equivalent to \(LP_3\) in the current context can be formulated in two different ways. For any \(q \geq 0\) define \(LP_{14}\) as follows:

\[
\begin{align*}
\sum_{(i,j) \in E} (w_{ij} - q g_{ij}) y_{ij} &\geq \beta - q(\gamma^c - K) \\
\sum_{j} y_{ij} &\leq \lambda_0^c h_i/2 \quad \forall i \in V \\
y_{ij} &\leq c_{ij} \quad \forall (i,j) \in E \\
y_{ij} &\geq 0 \\
\end{align*}
\]

(LP14)

\(LP_{14}\) subject to \(y^c \in Q^c[\lambda_0^c]\)

The Main Obstacle: If we apply the algorithm directly on \(LP_{11}\) the question remains: How do we maintain Invariant 7? This corresponds to the \(y_{ij} \leq c_{ij}\) constraint in \(LP_{13}\). Observe that the algorithm suggested in the proof of Lemma 8 (the algorithm for Theorem 16 in page 17) first computes a 6 approximation and then multiplies that solution by a factor of 6 to meet (or overwhelm) the objective value target which is required for a solution of the system \(LP_4\). This does not work here because (after multiplying by 6) we may have \(y_{ij} \approx 6c_{ij}\) violating \(y_{ij} \leq c_{ij}\) in \(LP_{14}\). Therefore invoking this type of a solution any number of times (as is done in the proof of Lemma 9 which uses Lemma 8 for different values of \(q\) at most \(O(\ln(1/\delta))\) times) will not solve \(LP_{12}\) (or equivalently \(LP_{13}\)). Note that \(y_{ij} = y_{ip_{ij},i}^c\) — this obstacle is independent of using the representation \(G\) or \(LONG(G)\). This obstacle is relevant for the initial solution as well. On the other hand, if we did not multiply by 6 then there need not have been a feasible solution to begin with (even for the initial solution)! This part needs a new idea different from computing a \(r\)-approximation and multiplying the solution by \(r\).

Such an idea is provided by Theorem 22 but a technical obstacle arises from the definition \(g_{ij}\) in the proof of any such algorithm. Note that the proof outline of Lemma 9 does not (immediately) work if we cannot guarantee that \(A_2 y = \sum_{(i,j) \in E} g_{ij} y_{ij} \geq 0\) in the precondition of Lemma 17. To ease this we prove the following:

Lemma 21. (Proved in Section 8.2) Suppose that \(\lambda > 1 + 8 \delta\) (otherwise the algorithm has stopped) and the current primal solution is \(y^c\). Suppose \(U \in \mathcal{O}_\delta(V')\) contains \(p_{ij,j}\) and either of the following conditions are met: (i) both \(i,j \notin U\) or (ii) \(y_{ip_{ij},i}^c = y_{pj_{ij},j}^c = 0\), then \(\lambda_U < \lambda - \delta^2\) and such an \(U \notin \mathcal{L}_c\).

Therefore we can compute the \(z_U\) in time \(O(m' \text{poly}\{\delta^{-1}, \log n\})\) where \(m' = |\{(i,j) | \text{SHORT}(y^c)_{ij} \neq 0\}|\) because the other edges cannot define any odd set in \(\mathcal{L}_c\). Further as a consequence of the lemma, \(g_{ij} \geq 0\) for all \((i,j) \in E\).

Observe that Lemma 21 removes the technical difficulty that was raised earlier. We can now prove the following theorem (similar to the role of Lemma 9 and Theorem 16 in Algorithm 1):

\[\text{25}\]

Theorem 22. (Proved in Section 8.3) If \( \beta \leq (1 - 4\delta)\beta^* \) a solution to LP14 always exists for any \( \lambda_0^* \geq 2 \). Moreover for \( \lambda_0^* = O(\ln \frac{1}{\delta}) \) we can solve LP13 in \( O\left(\frac{m}{\beta^*} \ln \left(\frac{1}{\delta}\right)\right) \) time, \( O(\min\{m, B\} \cdot \frac{1}{\beta} \ln \left(\frac{1}{\delta}\right)) \) space and \( q = 14 \ln \left(\frac{2}{\delta}\right) = O\left(\ln \left(\frac{1}{\delta}\right)\right) \) passes over the list of edges of \( G \). Moreover if we define \( S = \{(i, j) \in E | y_{ij} > 0\} \) then \( \sum_{(i, j) \in S} w_{ij} \leq 2q\beta^* \) and \( |S| \leq O(\min\{m, B\} \ln \frac{1}{\delta}) \).

The same algorithm also provides an initial (infeasible) solution of value \( \beta_0 \) where \( \beta_0 \leq \lambda_0^*\beta^* \).

It is feasible to reduce the running time by a factor of \( 1/\delta \) at the cost of increasing the number of passes and the parameter \( q \) by a factor of \( \ln \frac{1}{\delta} \). However we chose the smaller value of \( q \). We can now follow the identical outline of the proof of Theorem 10 applied to LP11. The running time is dominated by \( T = O(\delta^{-4}(\ln(1/\delta)) \ln n) \) solutions of LP12. The number of passes is \( R = qT \) where \( q \) is defined as in Theorem 22. The space requirement is \( O(m'\text{poly}\{\delta^{-1}, \ln n\}) \) times \( T \) where \( m' = |\{(i, j) | \text{SHORT}(\gamma^c)_{ij} \neq 0\}| \). Since we use a convex combination of \( T \) matchings, each of which uses \( \min\{m, B\} \ln \frac{1}{\delta} \) space and \( T \) itself is \( \text{poly}\{\delta^{-1}, \ln n\} \), the total space requirement is \( O(\min\{m, B\} \text{poly}\{\delta^{-1}, \ln n\}) \).

Summarizing:

Theorem 3 (Fractional Capacitated b-Matching). Given any nonbipartite graph, for any \( 0 < \delta \leq 1/16 \), we find a \( (1 - 14\delta) \)-approximate fractional weighted capacitated b-Matching using \( O(R/\delta + \min\{B, m\} \text{poly}\{\delta^{-1}, \ln n\}) \) time, \( O(\min\{m, B\} \text{poly}\{\delta^{-1}, \ln n\}) \) additional “work” space, making \( R = O(\delta^{-4}(\ln^2(1/\delta)) \ln n) \) passes over the list of edges where \( B = \sum b_i \). The algorithm returns a solution \( \hat{y}_{ij} \) such that the subgraph \( \hat{E} = \{(i, j) | (i, j) \in E, y_{ij} > 0\} \) satisfies \( \sum_{(i, j) \in \hat{E}} w_{ij} \leq 2R\beta^* \) where \( \beta^* \) is the weight of the optimum capacitated b-Matching.

The condition \( \sum_{(i, j) \in \hat{E}} w_{ij} \leq 2R\beta^* \) follows from application of Theorem 22 at most \( T \) times.

8.2 Proof of Lemma 21

Lemma 21. Suppose that \( \lambda > 1 + 8\delta \) (otherwise the algorithm has stopped) and the current primal solution is \( \gamma^c \). Suppose \( U \in \mathcal{O}_B(V') \) contains \( p_{ij,i} \) and either of the following conditions are met (i) both \( i, j \notin U \) or (ii) \( y^c_{p_{ij,i}} = y^c_{p_{ij,j}} = 0 \), then \( \lambda_U < \lambda - \delta^2 \) and such an \( U \notin \mathcal{L}_c \). Therefore we can compute the \( z_U \) in time \( O(m' \text{poly}\{\delta^{-1}, \ln n\}) \) where \( m' = |\{(i, j) | \text{SHORT}(\gamma^c)_{ij} \neq 0\}| \) because the other edges cannot define any odd set in \( \mathcal{L}_c \). Further as a consequence of the lemma, \( g_{ij} \geq 0 \) for all \( (i, j) \in E \).

Proof. Define \( Y^c_U = \sum_{(i', j') : i', j' \in U} y^c_{ij} \) for all sets \( U \) (even or odd). For an odd set \( U \) we have \( \lambda_U \tilde{b}^c_U = Y^c_U \). We focus on the first part of the lemma. Assume (for contradiction) \( \lambda_U \geq \lambda - \delta^2 \geq 1 + 7\delta \) for such a set \( U \) as in the statement of the lemma. Note \( b^c_{p_{ij,i}} = c_{ij} \). Also note that for any \( U \in \mathcal{O}_B \) we have \( (1 - \delta) \left\lfloor \frac{||U||_{b,c}}{2} \right\rfloor \leq \tilde{b}^c_U \) (based on \( F1 \) from Fact 11 in page 11).

Suppose that \( p_{ij,j} \notin U \). In this event, define \( U' = U - \{p_{ij,j}\} \) and \( Y^c_U = Y^c_{U'} \) in both cases (i) and (ii). Now we have two possibilities \( c_{ij} \) is odd or \( c_{ij} \) is even.

If \( c_{ij} \) is odd then \( ||U'||_{b,c} \) is even but in that case:

\[
\lambda_U \tilde{b}^c_U = Y^c_U = Y^c_{U'} \leq \frac{1}{2} \sum_{i' \in U'} \sum_{j' = 1}^{y^c_{i'j'}} \leq \frac{1}{2} \lambda \sum_{i' \in U'} \tilde{b}^c_{i'j'} = \frac{1}{2} \lambda(1 - 4\delta)||U'||_{b,c}
\]

\[
= \lambda(1 - 4\delta) \left\lfloor \frac{||U||_{b,c}}{2} \right\rfloor \leq \lambda(1 - 4\delta) \left\lfloor \frac{\tilde{b}^c_U}{1 - \delta} \right\rfloor \leq \lambda(1 - \delta)\tilde{b}^c_U \quad (\text{since} \ \delta \leq \frac{1}{8})
\]

which implies \( \lambda_U \leq (1 - \delta/4)\lambda \), thereby proving the lemma. On the other hand if \( c_{ij} \) is even (and therefore \( \geq 2 \)) then \( U' \) cannot be a single element because then \( 0 = Y^c_U = Y^c_{U'} = \lambda_U \tilde{b}^c_U > (1 + 7\delta)\tilde{b}^c_U \).
which is impossible. Therefore $U'$ is an odd-set and $\lambda_{U'}$, $\bar{b}_{U'}$ is defined. Now $||U'||_{b,c} = ||U||_{b,c} - c_{ij}$, and therefore $\bar{b}_{U'} - \bar{b}_{U'}' = \frac{c_{ij}}{2} - \delta^2 c_{ij} ||U||_{b,c} \geq (1 - 2\delta) c_{ij}/2 \leq c_{ij}/4$ since $U \in O_{\delta L}$ and $\delta \leq 1/8$, therefore $||U||_{b,c} \leq \frac{1}{8}$. Now, $\lambda b_{U'} \geq \lambda_{U'} \bar{b}_{U'} = Y_c^U = Y_c^U = \lambda_{U'}$ implying:

$$\lambda \geq \lambda_{U'} \bar{b}_{U'} \geq \lambda_{U'} \left(1 + \frac{c_{ij}}{4\bar{b}_{U'}}\right) \geq \lambda_{U'} \left(1 + \frac{\delta}{4}\right)$$

implying $\lambda(1 - \delta/8) \geq \lambda_{U'}$. Therefore if $p_{ij,j} \not\in U$ then $\lambda(1 - \delta^2) \geq \lambda_{U'}$ irrespective of $c_{ij}$ being odd or even.

Now suppose that $p_{ij,j} \in U'$. In this event, define $U' = U - \{p_{ij,i}, p_{ij,j}\}$. Observe that $||U'||_{b,c} = ||U||_{b,c} - 2c_{ij}$ is always odd and $Y_c^U = Y_c^U - c_{ij}$ if either of the conditions (i) $i, j \not\in U$ or (ii) $y^c_{p_{ij,i}} = y^c_{p_{ij,j}} = 0$ are met.

Since $Y_c^U \geq (1 + 7\delta) \bar{b}_{U'}$ and $\bar{b}_{U'} \geq (1 - \delta) \frac{||U||_{b,c}}{2}$ we have $Y_c^U \geq (1 + 7\delta)(1 - \delta)c_{ij} - c_{ij} > 0$. Therefore $U'$ is not a singleton set and $\lambda_{U'}$, $\bar{b}_{U'}$ is defined. But then, $\bar{b}_{U'} = c_{ij} - 2\delta^2 \bar{b}_{U'} ||U||_{b,c} \geq (1 - 2\delta)c_{ij}$. Using $1 + 7\delta \leq 2, 14\delta \leq 2, c_{ij} \geq 1, \bar{b}_{U'} \leq \frac{1}{8}$, we have:

$$\lambda \bar{b}_{U'} \geq \lambda_{U'} \bar{b}_{U'} = Y_c^U - c_{ij} = \lambda_{U'} \left(\bar{b}_{U'} - \frac{c_{ij}}{\lambda_{U'}}\right) \geq \lambda_{U'} \left(\bar{b}_{U'} - \frac{c_{ij}}{1 + 7\delta}\right)$$

$$\geq \lambda_{U'} \bar{b}_{U'} \left(\frac{\bar{b}_{U'} - \frac{c_{ij}}{1 + 7\delta}}{\bar{b}_{U'}}\right) \geq \lambda_{U'} \bar{b}_{U'} \left(1 + \frac{1 - 2\delta}{\bar{b}_{U'}} - \frac{c_{ij}}{(1 + 7\delta)\bar{b}_{U'}}\right)$$

$$\lambda_{U'} \bar{b}_{U'} \left(1 + \frac{(5 - 14\delta)\delta c_{ij}}{(1 + 7\delta)\bar{b}_{U'}}\right) = \lambda_{U'} \bar{b}_{U'} \left(1 + \frac{3\delta^2}{2}\right) \geq \lambda_{U'} \bar{b}_{U'} \left(\frac{1}{1 - \delta^2}\right)$$

and therefore if $p_{ij,j} \in U$ then $\lambda(1 - \delta^2) \geq \lambda_{U'}$.

Therefore the first part of the lemma follows, that is, if either of the conditions (i) $i, j \not\in U$ or (ii) $y^c_{p_{ij,i}} = y^c_{p_{ij,j}} = 0$, are met then we do not need to consider any $U \in O_{\delta L}$ such that $p_{ij,i} \in U$ for inclusion in $\mathcal{L}_c$. At the same time any $U \in O_{\delta L}$ must have some node $p_{ij,i}$ or $p_{ij,j}$, because all the edges are incident to one of these two types of nodes. Condition (ii) rules out computing odd sets containing $p_{ij,i}$ or $p_{ij,j}$ such that $y^c_{p_{ij,i}} = 0$ or that $\text{Short}(y^c)_{ij} = 0$. It is also worth noting that $x^c_{p_{ij,i}}$ can be stored implicitly. Therefore $\mathcal{L}_c$ can be defined on the graph with $m'$ nodes where $m'$ is as described in the statement of the theorem, and the time to compute $\mathcal{L}_c$ is $O(m' \text{poly}\{\delta^{-1}, \log n\})$ as stated. Finally, observe that based on Definition 8

$$a_{ij} = \left(x_i^c + x_{p_{ij,i}}^c + \sum_{U : i, p_{ij,i} \in U} z_{iU}^c\right) + \left(x_j^c + x_{p_{ij,j}}^c + \sum_{U : j, p_{ij,j} \in U} z_{jU}^c\right) - \left(x_{p_{ij,i}}^c + x_{p_{ij,j}}^c + \sum_{U : p_{ij,i}, p_{ij,j} \in U} z_{iU}^c\right)$$

but then $a_{ij}$ can only be negative if there exists a set $U \in L_c$ such that $p_{ij,i}, p_{ij,j} \in U$ and $i, j \not\in U$. The first part of the lemma, condition (i), just ruled out that possibility.

\[\square\]

### 8.3 Proof of Theorem 22

**Theorem 22.** If $\beta \leq (1 - 4\delta)\beta^*$ a solution to LP14 always exists for any $\lambda_0 \geq 2$. Moreover for $\beta \leq (1 - 4\delta)\beta^*$ and $\lambda_0 = O(\ln \frac{1}{\delta})$ we can solve LP13 in $O(\frac{\ln(1/\delta)}{\delta})$ time, $O(\min\{m, B\} \cdot \frac{1}{\delta} \ln(\frac{1}{\delta}))$
space and \( q = 14 \ln(\frac{2}{\delta}) = O(\ln(\frac{1}{\delta})) \) passes over the list of edges of \( G \). Moreover if we define \( S = \{(i,j) \in E | y_{ij} > 0\} \) then \( \sum_{(i,j) \in S} w_{ij} \leq 2q\beta^* \) and \( |S| \leq O(\min\{m,B\} \ln \frac{1}{\delta}) \).

\[
\sum_{(i,j) \in E} (w_{ij} - qg_{ij})y_{ij} \geq \beta - q(\gamma^c - K)
\]

\[
\sum_{j} y_{ij} \leq \lambda_0 b_i/2 \quad \forall i \in V \quad \text{LP14}
\]

\[
\sum_{j} y_{ij} \leq \lambda_0 b_i/2 \quad \forall i \in V \quad \text{LP13}
\]

\[
y_{ij} \leq c_{ij} \quad \forall (i,j) \in E
\]

\[
y_{ij} \geq 0
\]

The difference with \( \text{LP11} \) is that we can guarantee \( \sum_{(i,j) \in E} \tilde{y}_{ij}w_{ij} \geq \beta \) whereas \( Q^c[\beta, \lambda_0^c] \) in \( \text{LP11} \) only requires \( \sum_{(i,j) \in E} \tilde{y}_{ij}w_{ij}^c \geq (1 - \delta)\beta \) Note that \( \tilde{b}_{ci} \geq (1 - \delta) \left[ \frac{\|U\|_b}{2} \right] \geq (1 - 4\delta) \left[ \frac{\|U\|_b}{2} \right] \).

Multiplying rows of \( A_L^c \) with the non-negative numbers \( \{x_i^c\}, \{z_U^c\} \) we get that

\[
\sum_{(i',j') \in E_{L}} \text{LONG}(\tilde{y})_{i',j'} \left( x_i^c + x_j^c + \sum_{U \in L_{i',j'}} z_U^c \right) = \sum_{(i',j') \in E_{L}} \text{LONG}(\tilde{y})_{i',j'} \eta_{i',j'} \leq \gamma^c
\]

\[
\Longleftrightarrow \sum_{(i,j) \in E} g_{ij} \tilde{y}_{ij} \leq \gamma^c - K
\]

Multiplying the equation with any \( q \geq 0 \) and subtracting from \( \sum_{(i,j) \in E} \tilde{y}_{ij}w_{ij} \geq \beta \) we get

\[
\sum_{(i,j) \in E} (w_{ij} - qg_{ij})\tilde{y}_{ij} \geq \beta - q(\gamma^c - K)
\]

The remainder of the constraints of \( \text{LP14} \) are satisfied as well, namely:

\[
\tilde{y}_{ij} \leq c_{ij} \quad \forall (i,j) \in E
\]

\[
\sum_{j} \tilde{y}_{ij} \leq \lambda_0^c b_i/2 \quad \forall i \in V
\]

\[
\tilde{y}_{ij} \geq 0
\]

which proves the first part of the theorem for any \( \lambda_0^c \geq 2 \).
**The Second Part:** We express \( \text{LP14} \) as a maximization problem for \( \lambda_0 = 2 \) which is provided in \( \text{LP15} \) using \( w^c_{ij} = \max\{0, w_{ij} - qg_{ij}\} \) for \( (i, j) \in E \) and \( w^c_{ij} = 0 \) otherwise. Also we define \( c_{ij} = \min\{c_{ij}, b_i, b_j\} \). We also formulate the (slightly modified) dual \( \text{LP16} \)

\[
\begin{align*}
\tau^* &= \max \sum_{(i,j) \in E} w^c_{ij} y_{ij} \\ y_{ij} &\leq c'_{ij} \\ \sum_j y_{ij} &\leq b_i \\ y_{ij} &\geq 0
\end{align*}
\]

Based on the first part of the theorem \( \mathbf{y} \) satisfied \( \sum_{(i,j)} w_{ij} \frac{\gamma}{\nu} \geq \beta - \varrho(\gamma^c - K) \) and the remaining constraints of \( \text{LP15} \). Therefore the optimum solution \( \hat{y}_{ij} \) of \( \text{LP15} \) satisfies \( \sum_{(i,j)} w^c_{ij} \hat{y}_{ij} = \tau^* \geq \beta - \varrho(\gamma^c - K) \). Note that \( \hat{y}_{ij} \) can be fractional. Moreover \( \tau^* \leq \beta^* \) since \( w^c_{ij} \leq w_{ij} \) and \( c'_{ij} \leq c_{ij} \).

Now for any \( b_i, c'_{ij} \geq 0 \) we can find a solution \( \mathbf{y}^{(1)} \) such that \( \sum_{(i,j)} w_{ij} \mathbf{y}^{(1)} \geq \tau_1 \geq \tau^*/7 \) using Theorem 15 in Section 5. Define \( \tau^*(1) = \tau^* \). We run an iterative procedure where we set the constraints of \( \text{LP15} \) to \( y_{ij} \leq \max\{0, c'_{ij} - \mathbf{y}^{(1)}(1)\} \) then the optimum solution of this modified \( \text{LP15} \) denoted by \( \tau^*(2) \) is at least \( \tau^*(1) - \tau_1 \). Otherwise it is easy to see that the optimum solution of unmodified \( \text{LP15} \) cannot be \( \tau^* \). Therefore we can obtain a solution \( \mathbf{y}^{(2)} \) such that \( \sum_{i,j} w^c_{ij} \mathbf{y}^{(2)} = \tau_2 \geq \tau^*(2) \). We now repeat the process by modifying \( \text{LP15} \) to \( y_{ij} \leq \max\{0, c'_{ij} - \mathbf{y}^{(1)}(1) - \mathbf{y}^{(2)}(2)\} \). Proceeding in this fashion we obtain solutions \( \{\mathbf{y}^{(l)}\}_{l=1}^{\tau^*} \). Observe, that by construction \( \sum_{l=1}^{\tau^*} \mathbf{y}^{(l)} \leq c_{ij} \) for all \( (i, j) \) and therefore the union of these \( \tau^* \) solutions satisfies Invariant 1. We now claim:

\[
\sum_{l=1}^{r} \sum_{i,j} w^c_{ij} \mathbf{y}^{(l)} \geq \left(1 - \left(\frac{6}{7}\right)^r\right) \tau^*
\]

We prove Equation (12) by induction for all \( \tau^* \) for a fixed upper bound on \( r \); we just proved the base case for \( r = 1 \) since \( \tau_1 \geq \tau^*/7 \). In the inductive case, applying the hypothesis on 2, \ldots, \( r \) we get \( \sum_{l=2}^{r} \sum_{i,j} w^c_{ij} \mathbf{y}^{(l)} \geq \left(1 - \left(\frac{6}{7}\right)^{r-1}\right) \tau^*(2) \). Now if \( \tau^*(2) \geq \frac{6}{7} \tau^* \) then:

\[
\sum_{l=1}^{r} \sum_{i,j} w^c_{ij} \mathbf{y}^{(l)} \geq \frac{1}{7} \tau^* + \left(1 - \left(\frac{6}{7}\right)^{r-1}\right) \frac{6}{7} \tau^*
\]

and Equation (12) holds. Otherwise if \( \tau^*(2), 6 \tau^*/7 \) then:

\[
\sum_{l=1}^{r} \sum_{i,j} w^c_{ij} \mathbf{y}^{(l)} \geq \left(\tau^* - \tau^*(2)\right) + \left(1 - \left(\frac{6}{7}\right)^{r-1}\right) \tau^*(2) \geq \tau^* - \left(\frac{6}{7}\right)^{r-1} \tau^*(2) \geq \left(1 - \left(\frac{6}{7}\right)^{r}\right) \tau^*
\]

Therefore in all cases we will have \( \sum_{l=1}^{\tau^*} w^c_{ij} \mathbf{y}^{(l)} \geq \left(1 - \frac{6}{7}\right) \tau^* \). This implies that if we collect \( q = 7 \ln \frac{2}{3} \) solutions given by the 7 approximation algorithm then their union will be more than \( \left(1 - \frac{6}{7}\right)^{\tau^*} \) provided \( \beta \leq (1 - 4q)\beta^* \).

For the initial solution, set \( q = 0 \). This proves the bound on \( \lambda_0, \beta_0 \) and the number of edges. Applying Lemma 17 with \( r = \delta/2 \) and \( q = (1 - \frac{6}{7}) \), after we compute the solutions for the \( r \) nonnegative values of \( q \) in parallel we have a solution as desired by the Theorem. Note that we can increase the number of passes to \( O(\ln \frac{1}{\lambda}) \) and reduce the running time to \( O(m \ln \frac{1}{\lambda}) \). But
that choice would have increased the value of $q$. The current choice of computing the solutions in parallel proves the bound on $q = O(\ln \frac{1}{\delta})$.

Finally note that in each pass of the algorithm we compute a feasible, integral, capacitated $b$–Matching (with capacities less than that of the original graph) and therefore $\sum_{(i,j) \in E} w_{ij} \leq 2q\beta^*$ — the factor 2 arises from the convex combination of the solutions constructed by Lemma 17. □

8.4 Rounding Fractional Capacitated $b$-Matchings

In this section, we prove the following:

**Theorem 4 (Integral Capacitated $b$–Matching).** Given a fractional capacitated $b$-matching $y$ such that the optimum solution is at most $\beta^*$ and $\sum_{(i,j) \in E} w_{ij} \leq R\beta^*$ where $E = \{(i, j)|y_{ij} > 0\}$, we can find an integral $b$-matching of weight at least $(1 - \delta)\sum_{i,j} w_{ij}y_{ij} - \delta\beta^*$ in $O(m'R\delta^{-3} \log(R/\delta))$ time and $O(m'/\delta^2)$ space where $m' = |E|$. Note that if the fractional solution is at least $(1 - 14\delta)$-approximate then we have a $(1 - 16\delta)$-approximate integral solution.

The rounding is achieved by Algorithm 5. The general outline of the algorithm and its proof is similar to the uncapacitated case discussed in Section 7.

**Lemma 23.** $y_{ij}^{(1)}$ is a feasible fractional capacitated $b$–Matching in $G_c^{(1)}$.

**Proof.** Consider $\text{LONG}(y^{(1)})$ and $\text{LONG}(G)$. The only vertices whose capacities were affected in $\text{LONG}(G)$ are the following vertices: (i) the corresponding vertex in $G$ has an edge incident to it in $M_c^{(1)}$ and (ii) the corresponding edge $(i, j) \in G$ had $c_{ij} > \lceil y_{ij}^{(1)} \rceil + 1$.

In both cases the difference between the sum of the new edge multiplicities and the new capacities (the slack) is at least 1 and the first part of Lemma 18 tells us that these vertices in $\text{LONG}(G)$ cannot be part of a violated odd-set in $\text{LONG}(G)$. Therefore $\text{LONG}(y^{(1)})$ is feasible for $\text{LONG}(G)$. The lemma follows from Theorem 20. □

Therefore the only task that remains is to find a $(1 - \delta)$ approximate rounding of the fractional solution $y_{ij}^{(1)}$ on $G_c^{(1)} = (V, E^{(1)})$ with vertex and capacities $\{b_{ij}^{(1)}\}$ and $\{c_{ij}^{(1)}\}$ respectively.

**Lemma 24.** Let $\text{OPT}^{(1)}$ be the maximum capacitated $b$–Matching of $G_c^{(1)} = (V, E^{(1)})$. Let $W = \sum_{(i,j) \in E^{(1)}} c_{ij}^{(1)}w_{ij}$. Then $W \leq 2R\beta^* + \text{OPT}^{(1)} \leq (2R + 1)\beta^*$.

**Proof.** $\text{OPT}^{(1)} \leq \beta^*$ because we are only decreasing vertex and edge capacities. By construction, $c_{ij}^{(1)} \leq \lceil y_{ij}^{(1)} \rceil + 1 \leq y_{ij}^{(1)} + 2$. Note $\sum_{(i,j) \in E^{(1)}} y_{ij}^{(1)}w_{ij} \leq \text{OPT}^{(1)}$. We have already seen that $\sum_{(i,j) \in E} w_{ij} \leq R\beta^*$. Observe that $\sum_{(i,j) \in E^{(1)}} w_{ij} \leq \sum_{(i,j) \in E} w_{ij}$. The lemma follows. □

**Lemma 25.** Algorithm 5 outputs a capacitated $b$–Matching of weight at least $(1 - \delta)\sum_{(i,j) \in E} w_{ij}y_{ij} - \delta\beta^*$.

**Proof.** Since second phase is exactly the same as in the uncapacitated case in Section 7, we have $\sum_{(i,j) \in E^{(2)}} w_{ij}y_{ij}^{(2)} \geq (1 - \delta)\sum_{(i,j) \in E^{(1)}} w_{ij}y_{ij}^{(1)}$. Moreover since we did not introduce any new edge $W = \sum_{(i,j) \in E^{(2)}} w_{ij}c_{ij}^{(2)} = \sum_{(i,j) \in E^{(1)}} c_{ij}^{(1)}w_{ij} \leq (2R + 1)\beta^*$.

Now consider the step 4 (third phase) of Algorithm 5, we first construct $\text{LONG}(G_c^{(2)})$ where the edge $(i, j)$ gets subdivided to $(i, p_{ijj}, (p_{ijj}, p_{ijj}), (p_{ijj}, j)$ with weights $w_{ij}, w_{ij}, w_{ij}$. Based on Theorem 32.4, Vol A, page 567, there exists a feasible uncapacitated $b$–Matching in $\text{LONG}(G_c^{(2)})$.}
Algorithm 5 Rounding a fractional capacitated $b$-Matching in small space

1: First Phase: **Removing edges with large multiplicities** (no change from Algorithm 4 except tracking edge capacities). $t = \lceil 2/3 \rceil$.

   (a) If $y_{ij} \geq t$ add $y_{ij}^{(0)} = \lfloor y_{ij} \rfloor - 1$ copies of $(i,j)$ to $\mathcal{M}_{c}^{(0)}$.

   (b) Set $\hat{y}_{ij}^{(1)} = \begin{cases} \hat{y}_{ij}^{(0)} & \text{if } y_{ij} \geq t \\ y_{ij} & \text{otherwise} \end{cases}$. Set $b_{i}^{(1)} = \min \left\{ b_{i} - \sum_{j} y_{ij}^{(0)}, \lceil \sum_{j} \hat{y}_{ij}^{(1)} \rceil + 1 \right\}$ and $c_{ij}^{(1)} = \min \{ c_{ij}, \lceil \hat{y}_{ij}^{(1)} \rceil + 1 \}$. This describes the graph $G_{c}^{(1)}$. Note $c_{ij}^{(1)} \leq t + 1$.

2: Second Phase: **Subdividing vertices with large multiplicities.** (no change from Algorithm 4 except tracking edge capacities). We set $c_{ij}^{(2)} = c_{ij}^{(1)}$ where the edge $(i,j)$ got assigned to $i'$ and $j'$ which are copies of $i$ and $j$ respectively. This defines $G_{c}^{(2)}$. Note the edges are not split, only vertices are split.

3: Third Phase: **Reducing the problem to a weighted matching on small graph.** (different from Algorithm 4). The following steps can be viewed as first constructing $\text{LONG}(G_{c}^{(2)})$ and then applying the third phase of Algorithm 4. In more details:

   (a) For each $i \in V^{(2)}$ with $b_{i}^{(2)}$, create $i(1), i(2), \ldots, i(b_{i}^{(2)})$. For each edge $e = (i,j)$, we create $2v_{ij}$ vertices $\{p_{e,1}, p_{e,2}, \ldots, p_{e,2v_{ij}}, q_{e,1}, q_{e,2}, \ldots, q_{e,2v_{ij}}\}$. The graph $G^{(3)}$ is obtained by merging all the vertices in $\cup M_{c}^{(3)}$. Let this matching be $G_{c}^{(3)}$ which corresponds to a capacitated $b$-Matching $\mathcal{M}_{c}^{(2)}$ in $G_{c}^{(2)}$ of same weight (merge edges) which in turn implies a $b$-Matching $\mathcal{M}_{c}^{(1)}$ in $G_{c}^{(1)}$ of same weight (merge vertices). $\mathcal{M}_{c}^{(0)} \cup \mathcal{M}_{c}^{(1)}$ is the final solution.

(c) Run any fast approximation for finding a $(1 - \frac{\delta}{4(R+1)})$-approximate maximum weighted matching in $G^{(3)}$ let this matching be $\mathcal{M}_{c}^{(3)}$.

4: Final Phase: **Output** (same as Algorithm 4). Matching $\mathcal{M}_{c}^{(3)}$ implies a $b$-Matching $\mathcal{M}_{c}^{(2)}$ in $G_{c}^{(2)}$ of same weight (merge edges) which in turn implies a $b$-Matching $\mathcal{M}_{c}^{(1)}$ in $G_{c}^{(1)}$ of same weight (merge vertices). $\mathcal{M}_{c}^{(0)} \cup \mathcal{M}_{c}^{(1)}$ is the final solution.

whose weight is at least $\sum_{(i,j) \in E^{(2)}} w_{ij} y_{ij}^{(2)} + \mathcal{W}$. Moreover if we find an uncapacitated $b$-Matching in $\text{LONG}(G_{c}^{(2)})$ of weight $W$ then there exists a capacitated $b$-Matching in $G_{c}^{(2)}$ of weight $W - \mathcal{W}$.

Step 4 of Algorithm 5 then reduces finding an uncapacitated $b$-Matching in $\text{LONG}(G_{c}^{(2)})$ to a matching in $G^{(3)}$ where we replace every edge by a complete bipartite graph corresponding to the capacities of the two endpoints. Let the weight of the maximum matching of this graph $G^{(3)}$ be $w(\mathcal{M}^{*})$. Then $w(\mathcal{M}^{*}) \geq \sum_{(i,j) \in E^{(2)}} w_{ij} y_{ij}^{(2)} + \mathcal{W}$. Suppose that we find a $(\frac{\delta}{4R+4})$-approximate maximum matching in $G^{(3)}$, using the algorithm in [10] [11] which takes time $|E(G^{(3)})|$ times $O(\frac{m^{2}}{\delta} \log(R/\delta))$ which is $O(m^{2}R^{\delta-3} \log(R/\delta))$. This gives us a matching of weight at least $w(\mathcal{M}^{*}) - \delta w(\mathcal{M}^{*})$ which corresponds to a capacitated $b$-Matching in $G_{c}^{(2)}$ with weight $w(\mathcal{M}^{*}) - \delta w(\mathcal{M}^{*}) - \mathcal{W}$.
We get a matching $\mathcal{M}_c^{(1)}$ in $G_c^{(1)}$ of weight $w(\mathcal{M}_c^{(1)}) \geq (1 - \delta) \sum_{(i,j) \in E^{(1)}} w_{ij}y_{ij}^{(1)} - \delta \beta^*$. Observe that $w(\mathcal{M}_c^{(0)}) \geq (1 - \delta) \sum_{(i,j) \in E} w_{ij} \left( x_{ij} - y_{ij}^{(1)} \right)$ where $w(\mathcal{M}_c^{(0)}) = \sum_{(i,j) \in E} x_{ij}^{(0)} w_{ij}$. Then $w(\mathcal{M}_c^{(0)}) + w(\mathcal{M}_c^{(1)})$ is at least $(1 - \delta) \sum_{(i,j) \in E} w_{ij} x_{ij} - \delta \beta^*$ as desired. This proves Lemma 25, which in turn proves Theorem 4. □

References

[1] R. Adler and Y. Azar. Beating the logarithmic lower bound: Randomized preemptive disjoint paths and call control algorithms. *SODA*, pages 1–10, 1999.

[2] K. Ahn and S. Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *Inf. Comput.*, 222:59–79, 2013.

[3] K. J. Ahn and S. Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. *CORR, arXiv 1307.4359*, 2013.

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

[5] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta algorithm and applications. *Theory of Comput.*, 8(6):121–164, 2012.

[6] A. Bhalgat, R. Hariharan, T. Kavitha, and D. Panigrahi. An $\tilde{O}(mn)$ Gomory-Hu tree construction algorithm for unweighted graphs. *Proc. STOC*, 2007.

[7] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. *STOC*, pages 273–282, 2011.

[8] W. Cook. On box totally dual integral polyhedra. In *Mathematical Programming*, pages 48–61, 1986.

[9] W. H. Cunningham and A. B. Marsh. A primal algorithm for optimum matching. *Polyhedral Combinatorics*, 8:50–72, 1978.

[10] R. Duan and S. Pettie. Approximating maximum weight matching in near-linear time. In *Proc. FOCS*, pages 673–682, 2010.

[11] R. Duan, S. Pettie, and H.-H. Su. Scaling algorithms for approximate and exact maximum weight matching. In *Arxiv* [http://arxiv.org/abs/1112.0790](http://arxiv.org/abs/1112.0790), 2011.

[12] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005.

[13] Z. Füredi. Maximum degree and fractional matchings in uniform hypergraphs. *Combinatorica*, 1(2):155–162, 1981.

[14] Z. Füredi, J. Kahn, and P. D. Seymour. On the fractional matching polytope of a hypergraph. *Combinatorica*, 13(2):167–180, 1993.
[15] H. N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. *Proc. STOC*, pages 448–456, 1983.

[16] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for general graph-matching problems. *J. ACM*, 38(4):815–853, 1991.

[17] F. R. Giles and W. R. Pulleyblank. Total dual integrality and integer polyhedra. *Linear Algebra and Applications*, 25:191–196, 1979.

[18] M. Grotschel, L. Lovasz, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 1993.

[19] R. Hariharan, T. Kavitha, and D. Panigrahi. Efficient algorithms for computing all low s-t edge connectivities and related problems. *Proc. SODA*, 2007.

[20] S. Hougardy. Linear time approximation algorithms for degree constrained subgraph problems. *Research Trends in Comb. Opt., Springer*, pages 185–200, 2008.

[21] T. Jebara, J. Wang, and S.-F. Chang. Graph construction and b-matching for semi-supervised learning. In *Proc. ICML*, 2009.

[22] J. A. Kelner, G. L. Miller, and R. Peng. Faster approximate multicommodity flow using quadratically coupled flows. *STOC*, pages 1–18, 2012.

[23] C. Koufogiannakis and N. E. Young. Distributed fractional packing and maximum weighted b-matching via tail-recursive duality. *DISC*, pages 221–238, 2009.

[24] A. N. Letchford, G. Reinelt, and D. O. Theis. A faster exact separation algorithm for blossom inequalities. *Proceedings of IPCO, LNCS 3064*, pages 196–205, 2004.

[25] J. Mestre. Greedy in approximation algorithms. *ESA*, pages 528–539, 2006.

[26] M. Müller-Hannemann and A. Schwartz. Implementing weighted b-matching algorithms: towards a flexible software design. *J. Exp. Algorithmics*, 4, 1999.

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

[28] S. A. Plotkin, D. B. Shmoys, and É. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Math. of OR*, 20:257–301, 1995.

[29] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.