Restricted Max-Min Allocation: Integrality Gap and Approximation Algorithm*

Siu-Wing Cheng† Yuchen Mao‡

October 23, 2021

Abstract

Given a set of players P, a set of indivisible resources R, and a set of non-negative values $\{v_{pr}\}_{p\in P, r\in R}$, an allocation is a partition of R into disjoint subsets $\{C_p\}_{p\in P}$ so that each player p is assigned the resources in C_p. The max-min fair allocation problem is to determine the allocation that maximizes $\min_p \sum_{r\in C_p} v_{pr}$. In the restricted case of this problem, each resource r has an intrinsic value v_r, and $v_{pr} = v_r$ for every player p who desires r and $v_{pr} = 0$ for every player p who does not. We study the *restricted max-min fair allocation problem* in this paper. For this problem, the configuration LP has played an important role in estimating and approximating the optimal solution. Our first result is an upper bound of $3\frac{21}{26}$ on the integrality gap, which is currently the best. It is obtained by a tighter analysis of the local search of Asadpour et al. [TALG'12]. It remains unknown whether this local search runs in polynomial time or not. Our second result is a polynomial-time algorithm that achieves an approximation ratio of $4+\delta$ for any constant $\delta \in (0, 1)$. Our algorithm can be seen as a generalization of the aforementioned local search.

1 Introduction

Allocating resources among players is among the most common optimization problems in our daily life. Depending on the context, resources and players can be interpreted as jobs and machines, workers and tasks, classes and time slots, etc. Objectives and constraints vary. Max-min fairness is a popular objective in the area of networks, distributed systems, and economics. It aims to maximizing the welfare of the least lucky player.

Let P be a set of n players. Let R be a set of m indivisible resources. Let $\{v_{pr}\}_{p\in P, r\in R}$ be a set of non-negative values. For every resource $r \in R$ and every player $p \in P$, the value of r to p is v_{pr}. An allocation is a partition of R into disjoint subsets $\{C_p\}_{p\in P}$ so that each player p is assigned the resources in C_p. The *max-min fair allocation problem* seeks an allocation $\{C_p\}_{p\in P}$ that maximizes $\min_{p\in P} \sum_{r\in C_p} v_{pr}$ [3, 4, 5, 6, 7, 21]. A natural restricted case, the *restricted max-min fair allocation problem*, is that each resource r has an intrinsic value v_r, and $v_{pr} = v_r$ for every player p who desires r and $v_{pr} = 0$ for every player p who does not [1, 2, 4, 9, 13, 14, 15, 17]. Both the general and the restricted max-min allocation problem are NP-hard, and in fact, it is NP-hard to compute a solution better than a 2-approximation [6].

The max-min allocation problem is reminiscent of the classical min-max scheduling problem of computing an assignment of jobs to machines that minimizes the workload of the busiest

*Supported by Research Grants Council, Hong Kong, China (project no. 16207419). An extended abstract appears in Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, 2019, 38:1–38:13.

†Department of Computer Science and Engineering, HKUST, Hong Kong, China. Email : scheng@cse.ust.hk

‡College of Computer Science and Technology, Zhejiang University, China. Email: maoyc@zju.edu.cn

machine. This min-max scheduling problem has a 2-approximation algorithm proposed by Lenstra et al. [19], which is based on rounding the *assignment LP*. Bezáková and Dani [6] tried the following assignment LP on the max-min allocation problem.

$$\begin{array}{rcll} \max & T & & \\ \sum_{r\in R} v_{pr}x_{pr} & \geqslant & T & \forall\, p \in P \\ \sum_{p\in P} x_{pr} & \leqslant & 1 & \forall\, r \in R \\ x_{pr} & \geqslant & 0 & \forall\, p \in P, \forall\, r \in R \end{array}$$

They showed that, by rounding the optimal solution of the assignment LP, one can get an allocation whose objective value is at least $\max\{0, T^* - v_{max}\}$ where T^* is the optimal value of the assignment LP and $v_{max} = \max\{v_{pr} : p \in P, r \in R\}$. When v_{max} approaches T^*, there is no guarantee on the quality of their solution. Indeed, the assignment LP has an unbounded integrality gap, even for the restricted max-min allocation problem [4].

Realizing the weakness of the assignment LP, Bansal and Sviridenko [4] proposed a stronger LP relaxation, called the *configuration LP*, for the max-min allocation problem. Suppose that our goal is an allocation with objective value of T. A *configuration* for a player p is a subset C of the resources such that $v_{pr} > 0$ for all $r \in C$ and $\sum_{r\in C} v_{pr} \geqslant T$. That is, all resources in C are desired by p and their total value for p is at least T. Let $\mathcal{C}_p(T)$ denote the set of all configurations for p. Figure 1 shows the configuration LP, which is denoted by $CLP(T)$. A variable $x_{p,C}$ is associated with each player p and each configuration C in $\mathcal{C}_p(T)$. The first constraint ensures that each player receives at least one unit of configurations. The second constraint guarantees that every resource r is used in at most one unit of configurations. The optimal value of the configuration LP is the largest T for which $CLP(T)$ is feasible. Although the configuration LP may have an exponential number of variables, it can be solved within any constant relative error in polynomial time [4].

$$\begin{array}{rcll} \displaystyle\sum_{C\in\mathcal{C}_p(T)} x_{p,C} & \geqslant & 1, & \forall\, p \in P \\ \displaystyle\sum_{p\in P} \sum_{\substack{C\in\mathcal{C}_p(T) \\ \text{s.t. } r\in C}} x_{p,C} & \leqslant & 1, & \forall\, r \in R \\ x_{p,C} & \geqslant & 0, & \forall\, p \in P \ \forall\, C \in \mathcal{C}_p(T) \end{array}$$

Figure 1: Configuration LP $CLP(T)$.

1.1 Previous Work

For the general max-min allocation problem, Asadpour and Saberi [3] developed a polynomial-time rounding scheme for the configuration LP, and the scheme achieves an approximation ratio of $O(\sqrt{n}\log^3 n)$. Later, Saha and Srinivasan [21] improved it to $O(\sqrt{n\log n})$. These approximation ratios almost match the lower bound of $\Omega(\sqrt{n})$ on the integrality gap of the configuration LP [4]. Bateni et al. [5] and Chakrabarty et al. [7] established a trade-off between the approximation ratio and the running time. For any $\delta > 0$, they achieved an approximation ratio of $O(n^\delta)$ with a running time of $O(n^{1/\delta})$.

As to the restricted max-min allocation problem, Bansal and Sviridenko [4] proposed an $O\big(\frac{\log\log n}{\log\log\log n}\big)$-approximation algorithm by rounding the configuration LP. Feige [13] showed the existence of a constant upper bound, albeit large and unspecified, on the integrality gap of the configuration LP. Feige's proof was made constructive later by Haeupler et al. [14].

Asadpour et al. [2] viewed the restricted max-min allocation problem as a bipartite hyper-graph matching problem. By adapting Haxell's [15] alternating tree technique for bipartite hypergraph matchings, Asadpour et al. proposed a local search algorithm that returns an allocation whose objective value is within a factor of 4 from the optimal value of the configuration LP. Therefore, the integrality gap of the configuration LP is at most 4. Unfortunately, it remains unknown whether their local search runs in polynomial time. Since then, a lot of effort has been devoted to obtaining a polynomial-time local search algorithm. Polacek and Svensson [20] showed that the local search can be done in quasi-polynomial time by building the alternating tree in a more careful way. Annamalai et al. [1] introduced the *greedy* and *lazy update* strategies to the local search so that a $(6+2\sqrt{10}+\delta)$-approximate allocation can be found in $poly(m,n)\cdot n^{poly(1/\delta)}$ time. In an earlier paper, we improved this approximation ratio to $6+\delta$ by employing an adaptive greedy strategy [9]. The same result was also reported by Davies et al. using a different method [11]. Improving the integrality gap of the configuration LP is also of interest. In two independent works, we [8] and Jansen and Rohwedder [17] improved the upper bound for the integrality gap to $3\frac{5}{6}$. The current best lower bound for the integrality gap is 2.

1.2 Our Results

Our first result is that the integrality gap of the configuration LP is at most $3\frac{21}{26}$, an improvement to the previous bound of $3\frac{5}{6}$ [8, 17]. By a tighter analysis of the local search algorithm of Asadpour et al. [2], we show that it constructs an allocation whose objective value is within a factor of $\frac{99}{26}$ from the optimal value of the configuration LP. However, it is unknown whther this algorithm runs in polynomial time.

Theorem 1. *The integrality gap of the configuration LP for the restricted max-min fair allocation problem is at most* $\frac{99}{29}\approx 3.808$.

Our second result addresses the issue of running time. We introduce a new technique of *limited blocking*. With this technique as well as techniques used by previous papers [1, 11], we develop a framework that unifies the approaches in [1, 2]. We use three parameters to control the running time and the approximation ratio. At one extreme setting of these parameters, we obtain the local search algorithm of Asadpour et al. [2], which achieves an approximation ratio better than 4 but a possibly non-polynomial running time. At the other extreme, we obtain a method that resembles the algorithm of Annamalai et al. [1], which achieves a polynomial running time but a worse approximation ratio. By tuning the parameters appropriately, we show that a slight deviation from 4 is sufficient to guarantee a polynomial running time: for any $\delta>0$, a $(4+\delta)$-approximate allocation can be obtained in $poly(m,n)\cdot n^{poly(1/\delta)}$ time.

Theorem 2. *For any constant* $\delta>0$*, there is a* $(4+\delta)$*-approximation algorithm for the restricted max-min fair allocation problem that runs in* $poly(m,n)\cdot n^{poly(1/\delta)}$ *time.*

The conference version [10] of this paper appeared in ICALP 2019. Davies et al. [12] reported a different algorithm that also gives an approximation ratio $4+\delta$ in polynomial time in their paper in SODA 2020.

In Section 2, we give the necessary notations and preliminaries. In Section 3, we describe the algorithm of Asadpour et al. [2] and give a better analysis. In Section 4, we describe our approximation algorithm. Since our algorithm generalizes the local search in Section 3, reading the algorithm in Section 3 will facilitate the understanding of Section 4.

2 Preliminaries

2.1 Allocations and Hypergraph Matchings

Suppose that we aim for an allocation whose max-min value is at least λ. We call a resource r *fat* if $v_r \geqslant \lambda$, and *thin* otherwise. It suffices to assign each player either a single fat resource or some thin resources whose total value for the player is at least λ. In the following, we show that this can be reduced to a matching problem.

Let G be an undirected bipartite graph with the players and fat resources as vertices. For each player p and each fat resource r desired by p, the pair (p, r) form an edge of G. Let $\mathcal{H}$ be a bipartite hypergraph with the players and thin resources as vertices. For each player p and each subset S of thin resources that are desired by p and satisfy $\sum_{r \in S} v_r \geqslant \lambda$, the tuple (p, S) forms a hyper-edge of $\mathcal{H}$. For a hyper-edge e, we use $p(e)$ to denote the player incident to e, and $R(e)$ to denote the set of resources incident to e. For a set $\mathcal{E}$ of hyper-edges, we define $R(\mathcal{E}) = \bigcup_{e \in \mathcal{E}} R(e)$. For a set R of resources, we sometimes write $\sum_{r \in R} v_r$ as $v[R]$.

Finding the target allocation with max-min value at least λ is equivalent to finding a matching M of G and a matching $\mathcal{M}$ of $\mathcal{H}$ such that every player is matched by either M or $\mathcal{M}$. A *matching* of a graph (resp. hypergraph) is a subset of edges (resp. hyper-edges) such that no two edges (resp. hyper-edges) share any common vertex. In both the local search algorithm of Asadpour et al. and our algorithm in Section 4, M is initialized to be an arbitrary maximum matching of G and $\mathcal{M}$ is initialized to be empty. Then we iteratively update $\mathcal{M}$ and M so that one more player is matched in each iteration. When updating $\mathcal{M}$ and M, we maintain an invariant that M is always a maximum matching of G and $\mathcal{M}$ is always a matching of $\mathcal{H}$. The restricted max-min allocation problem is then reduced to the following problem.

> Given a maximum matching M of G, a matching $\mathcal{M}$ of $\mathcal{H}$, and an unmatched player p_0, find a maximum matching M' of G and a matching of $\mathcal{M}'$ of $\mathcal{H}$ that match p_0 as well as all players matched by M and $\mathcal{M}$.

Hence, all the players will be matched after at most n iterations, provided that the target max-min value of λ is achievable.

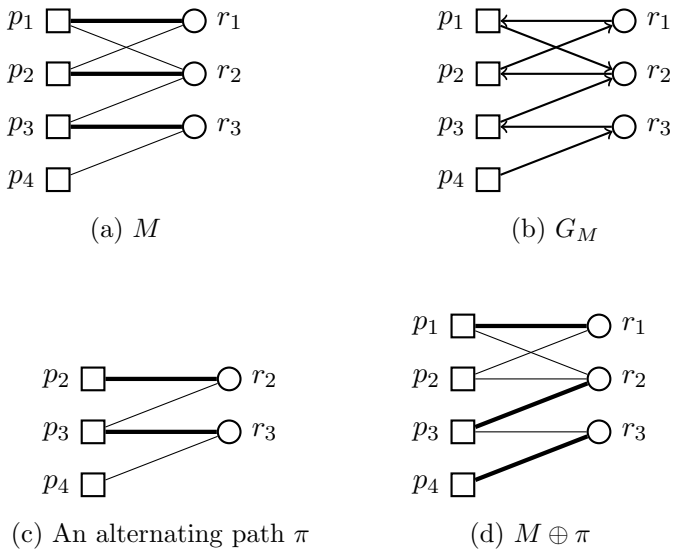
A player does not need to receive more resources than necessary, so we have two additional requirements for M and $\mathcal{M}$: (i) if a player is matched, it is matched by either M or $\mathcal{M}$, but not both; (ii) every hyper-edge in $\mathcal{M}$ is λ-*minimal*. The definition of λ-minimal is given in Definition 3 below. Intuitively, it means that the thin resources in a hyper-edge form a minimal set that has a total value λ or more.

Definition 3. Let $w > 0$ be a real number. A hyper-edge (p, S) is w-*minimal* if $v[S] \geqslant w$ and for every proper subset $S' \subset S$, $v[S'] < w$.

2.2 Alternating Paths and Symmetric Difference

Given a matching M of G, a path in G is an *alternating path* with respect to M if it alternates between edges that are unmatched and matched by M. We allow an alternating path to consist of a single vertex, that is, the path contains no edge at all. For any maximum matching M of G, define a directed graph G_M as follows. G_M has the same vertex set as G. Edges of G_M are obtained by orienting edges of G from p to r if $(p, r) \notin M$ and from r to p if $(p, r) \in M$. Figure 2 gives an example of G_M. Every path in G_M is an alternating path with respect to M in G, and vice versa. In the rest of this paper, we do not distinguish between paths in G_M and alternating paths with respect to M.

Let π be such a path in G_M that starts from a player not matched by M and ends at some player. We call the starting player of π the *source* of π, denoted by $src(\pi)$, and the ending



(a) M (b) G_M

(c) An alternating path π (d) $M \oplus \pi$

Figure 2: G_M, alternating paths, and $\oplus$ operation

player of π the *sink* of π, denoted by $sink(\pi)$. If we view π as a set of edges, we can update M by taking the symmetric difference

$$M \oplus \pi = (M \cup \pi) \setminus (M \cap \pi),$$

which results in a maximum matching $M \oplus \pi$ of G. See Figure 2 for an example. If π has at least one edge, the edge in π incident to $sink(\pi)$ must be a matching edge in M. Therefore, the player $sink(\pi)$ is matched by M before the $\oplus$ operation. After the $\oplus$ operation, $src(\pi)$ becomes matched and $sink(\pi)$ becomes unmatched. If π has no edge, then $sink(\pi) = src(\pi)$ and the $\oplus$ operation does not change the matching.

3 Integrality Gap of the Configuration LP

In this section, we describe the local search of Asadpour et al. [2] and give a tighter analysis. Without loss of generality, we assume that the optimal value of the configuration LP is 1. We show that, for $\lambda = \frac{28}{99}$, the local search is able to find an allocation with max-min value at least λ. Although our description of the algorithm is not the same as that in [2], there is no essential difference. It serves as a prelude to the approximation algorithm in Section 4.

3.1 A High Level Idea

Recall that it suffices to update M and $\mathcal{M}$ to match one more unmatched player p_0. Let q_0 be any player such that there is an alternating path π in G_M from p_0 to q_0.

If there is a λ-minimal hyper-edge e that is incident to q_0 and does not share any resource with the hyper-edges in $\mathcal{M}$, then we can match p_0 by flipping the alternating path π (i.e., $M := M \oplus \pi$) and adding the hyper-edge e to $\mathcal{M}$. See Figure 3 for an illustration. Note that players that are matched beforehand remain matched afterwards.

We are not always that lucky though. It is possible that every hyper-edge incident to q_0 shares a resource with some hyper-edge in $\mathcal{M}$. In this case, we either try another q_0 or release some hyper-edge from $\mathcal{M}$. Suppose that we want to release from $\mathcal{M}$ a hyper-edge e' that matches some player p_1. As mentioned before, we do not want to lose any matched player; therefore, we must match p_1 by another (hyper-)edge before e' is released. Then, p_1 has a similar role as p_0, and we proceed to match p_1 using the same strategy that is used for matching p_0.

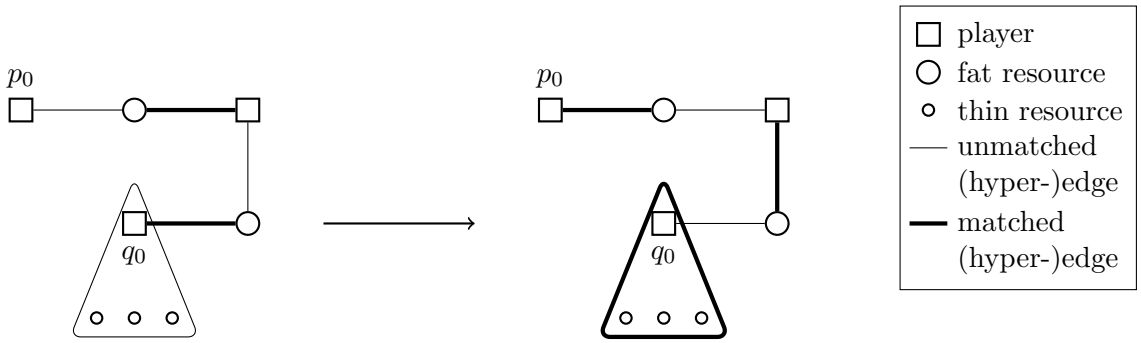


Figure 3: How p_0 is matched in the ideal case.

3.2 The Local Search

The algorithm maintains a stack of tuples $\Sigma = [(a_0, \mathcal{B}_0), (a_1, \mathcal{B}_1), \cdots]$, where a_i is an *addable edge*, and $\mathcal{B}_i$ is the set of *blocking edges* of a_i. Intuitively, a_i is the hyper-edge we want to add to $\mathcal{M}$ and $\mathcal{B}_i$ is the set of hyper-edges in $\mathcal{M}$ that prevent us from doing so. Precise definitions will be given shortly. We use ℓ to denote the index of the last tuple in Σ. The state of the algorithm is characterized by $(M, \mathcal{M}, \Sigma, \ell)$. We use $R(\Sigma)$ to denote the set of thin resources that appear in Σ, that is,

$$R(\Sigma) = \left(\bigcup_{i=0}^{\ell} R(a_i)\right) \cup \left(\bigcup_{i=0}^{\ell} R(\mathcal{B}_i)\right).$$

Let B_i be the set of players incident to the hyper-edges in $\mathcal{B}_i$, i.e., $B_i = \{p(e) : e \in \mathcal{B}_i\}$.

The stack Σ is built inductively. Initially, $a_0 :=$ null, $\mathcal{B}_0 := \{(p_0, \emptyset)\}$, $\Sigma := [(a_0, \mathcal{B}_0)]$, and $\ell := 0$. Given $\Sigma = [(a_0, \mathcal{B}_0), \ldots, (a_\ell, \mathcal{B}_\ell)]$, the algorithm calls a routine Build to construct and push a new tuple $(a_{\ell+1}, \mathcal{B}_{\ell+1})$ onto Σ. We use $B_{\leqslant k}$ to denote $\bigcup_{i=1}^{k} B_i$. We need some definitions before describing Build.

Definition 4. Given a state $(M, \mathcal{M}, \Sigma, \ell)$ of the algorithm, where $\Sigma = [(a_0, \mathcal{B}_0), \ldots, (a_\ell, \mathcal{B}_\ell)]$, a player p is *addable* if there is a path from some player in $B_{\leqslant \ell}$ to p in G_M. A hyper-edge (p, S) is an *addable edge* if p is addable, $S \cap R(\Sigma) = \emptyset$, and (p, S) is λ-minimal.

Definition 5. Given an addable edge a, an edge b in $\mathcal{M}$ is a *blocking* edge of a if b shares some resource with a, i.e., $R(a) \cap R(b) \neq \emptyset$. If an addable edge has no blocking edge, it is *unblocked*; otherwise, it is *blocked*.

The pseudocode of Build is shown below.

Build$(M, \mathcal{M}, \Sigma, \ell)$

1. Arbitrarily pick an addable edge $a_{\ell+1}$.
2. $\mathcal{B}_{\ell+1} := \{e \in \mathcal{M} : e \text{ is a blocking edge of } a_{\ell+1}\}$.
3. Push $(a_{\ell+1}, \mathcal{B}_{\ell+1})$ onto Σ. Set $\ell := \ell + 1$.

Whenever some $\mathcal{B}_i$ in Σ becomes empty, i.e., some addable edge a_i in Σ becomes unblocked, we invoke the routine Contract below to update Σ, M, and $\mathcal{M}$.

CONTRACT$(M, \mathcal{M}, \Sigma, \ell)$

1. Let j be the smallest index such that the addable edge a_j in Σ is unblocked. For clarity, we denote a_j by a^*.
2. Let k be the smallest index such that there exist a blocking edge $b^* \in \mathcal{B}_k$ and a path π^* from $p(b^*)$ to $p(a^*)$ in G_M. (Lemma 8 guarantees the existence of k and that $k < j$.)
3. Delete all tuples $(a_i, \mathcal{B}_i)$'s with $i > k$ from Σ. Set $\ell := k$.
4. $M := M \oplus \pi^*$ and $\mathcal{M} := \mathcal{M} \cup \{a^*\}$.
5. If $k = 0$, then $p(b^*) = p_0$ and the algorithm terminates because step 4 already matches $p(b^*) = p_0$.
6. If $k > 0$, we can release b^* from $\mathcal{M}$ because step 4 matches $p(b^*)$ using a new edge (or hyper-edge), and so we set $\mathcal{M} := \mathcal{M} \setminus \{b^*\}$ and $\mathcal{B}_k := \mathcal{B}_k \setminus \{b^*\}$.

The algorithm keeps calling CONTRACT until all addable edges in Σ are blocked. If p_0 is not matched yet, the algorithm calls BUILD to grow the stack Σ again. It alternates between calling BUILD and CONTRACT until p_0 is matched.

The following observation is clear from the description of BUILD and CONTRACT.

Observation 6. *By Definition 4, no two addable edges in Σ share any resource, that is, $R(a_i) \cap R(a_j) = \emptyset$ for all $i, j \in [1, \ell]$ such that $i \neq j$. By Definitions 4 and 5, for $i \in [1, \ell]$, the addable edge a_i in Σ is not blocked by any edge in $\mathcal{B}_{\leqslant i-1}$, but a_i is blocked by every edge in $\mathcal{B}_i$. This implies that (i) all $\mathcal{B}_i$'s are mutually disjoint, i.e., $\mathcal{B}_i \cap \mathcal{B}_j = \emptyset$ for all $i, j \in [1, \ell]$ such that $i \neq j$, and that (ii) for $i \in [1, \ell]$, all blocking edges of a_i belong to $\mathcal{B}_i$, even after* CONTRACT *modifies $\mathcal{M}$. In other words, a_i is not blocked by any edge in $\mathcal{M} \setminus \mathcal{B}_i$.*

3.3 Running Time

At this stage, let's assume that the algorithm never gets stuck. We show that the algorithm terminates in finite number of steps.

Lemma 7. *Assume that the algorithm never gets stuck. Then, the algorithm terminates after a finite number of calls of* BUILD *and* CONTRACT.

Proof. Define a signature vector $(|B_0|, |B_1|, \ldots, |B_\ell|, \infty)$ with respect to the sequence of tuples in Σ. By Observation 6, $\mathcal{B}_1, \ldots, \mathcal{B}_\ell$ are mutually disjoint subsets of $\mathcal{M}$. Recall that $\mathcal{B}_0 = \{(p_0, \emptyset)\}$. We have $|B_0| + \cdots + |B_\ell| \leqslant n$. Therefore, the number of distinct signature vectors is at most n^n. The signature vector evolves as Σ is updated by the algorithm. After each invocation of BUILD, the signature vector decreases lexicographically because it gains a new second to last entry. After each invocation of CONTRACT, the signature vector also decreases lexicographically because it becomes shorter, and the second to last entry decreases by at least 1. Therefore, no signature vector is repeated. As a result, the algorithm terminates after at most n^n invocations of BUILD and CONTRACT. ◻

$$
\begin{array}{lrcll}
\text{maximize} & \displaystyle\sum_{p\in P} y_p - \sum_{r\in R} z_r & & & \\
\text{subject to} & y_p & \leqslant & \displaystyle\sum_{r\in C} z_r, & \forall\, p \in P, \forall\, C \in \mathcal{C}_p(T) \\
& y_p & \geqslant & 0, & \forall\, p \in P \\
& z_r & \geqslant & 0, & \forall\, r \in R
\end{array}
$$

Figure 4: Dual of the configuration LP.

3.4 Never Getting Stuck

Given Lemma 7, all we need to show is that the algorithm never gets stuck. We first prove that the index k in step 2 of CONTRACT always exists by establishing the following lemma.

Lemma 8. *Let $(M, \mathcal{M}, \Sigma, \ell)$ be a state of the algorithm. For every $i \in [1, \ell]$, there is a path in G_M from some player in $B_{\leqslant i-1}$ to $p(a_i)$.*

Proof. We prove the lemma by induction. Initially, $\ell = 0$, $a_0 = \text{null}$, and $\Sigma = \{(a_0, \mathcal{B}_0)\}$. The lemma trivially holds for $\ell = 0$.

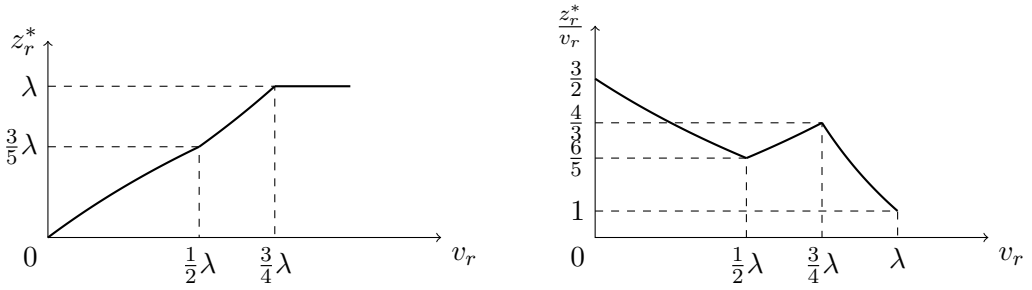
We show that BUILD preserves the lemma. Let $(a_{\ell+1}, \mathcal{B}_{\ell+1})$ be the tuple constructed by BUILD. By Definition 4, $p(a_{\ell+1})$ is an addable player, so there is a path from some player in $B_{\leqslant \ell}$ to $p(a_{\ell+1})$. For any a_i with $i \in [1, \ell]$, since BUILD does not change the old tuples in Σ nor matching M, the path from some players in $B_{\leqslant i-1}$ to $p(a_i)$ exists by the inductive hypothesis.

We show that CONTRACT also preserves the lemma. Let a^*, k, b^*, and π^* be defined as in the description of CONTRACT. Since all the tuples in Σ with indices greater than k are deleted in step 3, we only need to verify the lemma for the remaining $k + 1$ tuples. We claim that for every path π in G_M such that $src(\pi) \in B_{\leqslant k-1}$, π is node-disjoint from π^*. Otherwise, we would be able to find a path from some player in $B_{\leqslant k-1}$ to $p(a^*)$ by first following π, switching at a common node of π and π^*, and then following π^*. This is a contradiction to our choice of k. By our claim, none of the paths with sources in $B_{\leqslant k-1}$ is affected by the operation $M \oplus \pi^*$, including those from some player in $B_{\leqslant i-1}$ to $p(a_i)$ for $i \in [1, k]$. ◻

Next we prove that before p_0 is matched, the algorithm is always able to invoke either BUILD or CONTRACT. This is proved by contradiction. We show that if neither of the two routines can be invoked, then the dual of $CLP(1)$ would be unbounded, contradicting the assumption that 1 is the optimal value of the configuration LP. The dual of the configuration LP is given in Figure 4.

When arguing that the dual is unbounded, we will essentially show that a lower bound on the total "dual value" of the resources in $R(\Sigma)$ is larger than its upper bound. Asadpour et al. [2] implicitly set the dual value of a thin resource r to be v_r, and they used a worst-case upper bound and a worst-case lower bound on the total value of the resources in $R(\Sigma)$. Their proof works only for $\lambda \leqslant \frac{1}{4}$. We observe that the worst-case upper bound and the worst-case lower bound used in [2] cannot occur simultaneously. Specifically, the worst-case upper bound occurs only when the values of all thin resources are close to λ, whereas the worst-case lower bound occurs only when the values of all thin resources are close to zero. Our approach is to magnify the dual value of the thin resources with small values. It helps us to derive a better lower bound without deteriorating the upper bound.

Definition 9. Let $(M, \mathcal{M}, \Sigma, \ell)$ be a state of the algorithm. Let P^+ be the set of players that are reachable in G_M from some player in $B_{\leqslant \ell}$. Let R_f^+ be the set of fat resources that are reachable in G_M from some player in $B_{\leqslant \ell}$. We define the *associated dual solution of the state* to be the following solution $(\{y_p^*\}_{p\in P}, \{z_r^*\}_{r\in R})$ of the dual of $CLP(1)$.



(a) z_r^* versus v_r (b) z_r^*/v_r versus v_r

Figure 5: Dual values for resources in $R(\Sigma)$

$$y_p^* = \begin{cases} 1 - \frac{21}{26}\lambda, & \text{if } p \in P^+, \\ 0, & \text{otherwise.} \end{cases} \qquad z_r^* = \begin{cases} 1 - \frac{21}{26}\lambda, & \text{if } r \in R_f^+, \\ \frac{3\lambda}{2\lambda + v_r} v_r, & \text{if } r \in R(\Sigma) \text{ and } v_r \in (0, \frac{\lambda}{2}), \\ \frac{3\lambda}{3\lambda - v_r} v_r, & \text{if } r \in R(\Sigma) \text{ and } v_r \in [\frac{\lambda}{2}, \frac{3\lambda}{4}), \\ \lambda, & \text{if } r \in R(\Sigma) \text{ and } v_r \in [\frac{3\lambda}{4}, \lambda), \\ 0, & \text{otherwise.} \end{cases}$$

Figure 5 plots z_r^* and z_r^*/v_r against v_r for a thin resource r in $R(\Sigma)$.

Lemma 10. *Let $(M, \mathcal{M}, \Sigma, \ell)$ be a state of the algorithm. For $\lambda = \frac{26}{99}$, if no more addable edge can be added to Σ, then the associated dual solution $(\{y_p^*\}_{p \in P}, \{z_r^*\}_{r \in R})$ of the state is feasible, i.e., $y_p^* \leqslant \sum_{r \in C} z_r^*$ for every $p \in P$ and every $C \in \mathcal{C}_p(1)$.*

Proof. Consider any player $p \in P$ and any configuration $C \in \mathcal{C}_p(1)$. If $p \notin P^+$, then $y_p^* = 0$, and the inequality $y_p^* \leqslant \sum_{r \in C} z_r^*$ holds because z_r^* is non-negative. Assume that $p \in P^+$. So $y_p^* = 1 - \frac{21}{26}\lambda$. We prove that $\sum_{r \in C} z_r^* \geqslant 1 - \frac{21}{26}\lambda$ by a case analysis.

Case 1: C contains a fat resource. Let r_f be a fat resource in C. Since player p desires r_f, G_M contains a (directed) edge either from p to r_f or from r_f to p. Since $p \in P^+$, p is reachable in G_M from some player in $B_{\leqslant \ell}$. If G_M contains an edge from p to r_f, then r_f is obviously also reachable in G_M from some player in $B_{\leqslant \ell}$, i.e., $r_f \in R^+$. If G_M contains an edge from r_f to p, then p must be a player matched to r_f by M. Note that $p \notin B_{\leqslant \ell}$ as no player in $B_{\leqslant \ell}$ is matched by M. Moreover, (r_f, p) is the only edge entering p in G_M. As a consequence, if some player in $B_{\leqslant \ell}$ can reach p in G_M, it must reach r_f first, implying that $r_f \in R^+$. We conclude that $r_f \in R^+$ irrespective of whether (p, r_f) is a matching edge in M. By the construction of the dual solution, we have

$$\sum_{r \in C} z_r^* \geqslant z_{r_f}^* = 1 - \frac{21}{26}\lambda.$$

Case 2: C contains thin resources only. Since $p \in P^+$, p is reachable in G_M from some player in $B_{\leqslant \ell}$. By Definition 4, p is an addable player. Since no more addable edge can be added to Σ, the total value of thin resources in $C \setminus R(\Sigma)$ must be less than λ; otherwise, there would exist some subset $S \subseteq C \setminus R(\Sigma)$ that forms an addable edge (p, S). Recall that $\sum_{r \in C} v_r \geqslant 1$ because $C \in \mathcal{C}_p(1)$. Given $\lambda = \frac{26}{99}$,

$$\sum_{r \in C \cap R(\Sigma)} v_r = \sum_{r \in C} v_r - \sum_{r \in C \setminus R(\Sigma)} v_r > 1 - \lambda = \frac{73}{26}\lambda. \tag{1}$$

We prove that

$$\sum_{r\in C\cap R(\Sigma)} z_r^* \;\geqslant\; 3\lambda \;=\; 1-\frac{21}{26}\lambda \qquad \left(\because \lambda=\frac{26}{99}\right)$$

by examining subcases 2.1 – 2.4 below. Some conclusions are drawn directly from Figure 5, and they can be verified easily.

Case 2.1: At least three resources in $C\cap R(\Sigma)$ have values in $[\frac{3\lambda}{4},\lambda)$. Denote these three resources as r_1, r_2, and r_3. By definition, $z^*_{r_i}=\lambda$ for $i\in\{1,2,3\}$. Then

$$\sum_{r\in C\cap R(\Sigma)} z_r^* \geqslant z^*_{r_1}+z^*_{r_2}+z^*_{r_3}=3\lambda.$$

Case 2.2: Exactly one resource in $C\cap R(\Sigma)$ has a value in $[\frac{3\lambda}{4},\lambda)$. Denote this resource as r_1. By definition, $z^*_{r_1}=\lambda$. Let $R'=(C\cap R(\Sigma))\setminus\{r_1\}$. Then

$$\sum_{r\in R'} v_r \;=\; \sum_{r\in C\cap R(\Sigma)} v_r - v_{r_1} \;\overset{(1)}{>}\; \frac{73}{26}\lambda-\lambda \;=\; \frac{47}{26}\lambda. \tag{2}$$

Every resource in R' has a value in the range $(0,\frac{3}{4}\lambda)$. As illustrated in Figure 5(b), $\frac{z_r^*}{v_r}\geqslant\frac{6}{5}$ for every $r\in R'$. Therefore,

$$\sum_{r\in R'} z_r^* \;\geqslant\; \frac{6}{5}\sum_{r\in R'} v_r \;\overset{(2)}{>}\; \frac{141}{65}\lambda.$$

Then,

$$\sum_{r\in C\cap R(\Sigma)} z_r^* = \sum_{r\in R'} z_r^* + z^*_{r_1} > \frac{141}{65}\lambda+\lambda>3\lambda.$$

Case 2.3: No resource in $C\cap R(\Sigma)$ has a value in $[\frac{3\lambda}{4},\lambda)$. As illustrated in Figure 5(b), $\frac{z_r^*}{v_r}\geqslant\frac{6}{5}$ for every r with $v_r\in(0,\frac{3\lambda}{4})$. Then,

$$\sum_{r\in C\cap R(\Sigma)} z_r^* \;\geqslant\; \frac{6}{5}\sum_{r\in C\cap R(\Sigma)} v_r \;\overset{(1)}{>}\; \frac{6}{5}\cdot\frac{73}{26}\lambda \;>\; 3\lambda.$$

Case 2.4: There are exactly two resources in $C\cap R(\Sigma)$ with values in $[\frac{3\lambda}{4},\lambda)$. This is the last case to be analyzed. Denote these two resources as r_1 and r_2. By definition, $z^*_{r_1}$ and $z^*_{r_2}$ are equal to λ. Let $R'=(C\cap R(\Sigma))\setminus\{r_1,r_2\}$. Then, to prove that

$$\sum_{r\in C\cap R(\Sigma)} z_r^* = \sum_{r\in R'} z_r^* + z^*_{r_1} + z^*_{r_2} \geqslant 3\lambda,$$

it suffices to show that

$$\sum_{r\in R'} z_r^* \geqslant \lambda.$$

Let V_0 denote the multi-set of values of thin resources in R'. Note that $v\in(0,\frac{3}{4}\lambda)$ for all $v\in V_0$. Moreover,

$$\sum_{v\in V_0} v \;=\; \sum_{r\in R'} v_r \;=\; \left(\sum_{r\in C\cap R(\Sigma)} v_r\right) - v_{r_1} - v_{r_2} \;\overset{(1)}{>}\; \frac{73}{26}\lambda-2\lambda \;=\; \frac{21}{26}\lambda.$$

Let $g(v) = \frac{3\lambda}{2\lambda+v}v$. Let $h(v) = \frac{3\lambda}{3\lambda-v}v$. We have

$$\sum_{r\in R'} z_r^* = \sum_{v\in V_0\cap(0,\frac{\lambda}{2})} g(v) + \sum_{v\in V_0\cap[\frac{\lambda}{2},\frac{3\lambda}{4})} h(v).$$

To derive a lower bound for $\sum_{r\in R'} z_r^*$, we will transform V_0 step by step to another multi-set V_1 of values such that

(1) $\sum_{v\in V_1\cap(0,\frac{\lambda}{2})} g(v) + \sum_{v\in V_1\cap[\frac{\lambda}{2},\frac{3\lambda}{4})} h(v) \leqslant \sum_{v\in V_0\cap(0,\frac{\lambda}{2})} g(v) + \sum_{v\in V_0\cap[\frac{\lambda}{2},\frac{3\lambda}{4})} h(v)$,

(2) $\sum_{v\in V_1} v = \frac{21\lambda}{26}$, and

(3) V_1 contains exactly two values, one in $(0, \frac{\lambda}{2})$ and the other one in $[\frac{\lambda}{2}, \frac{3\lambda}{4})$.

Then we give a lower bound for $\sum_{v\in V_1\cap(0,\frac{\lambda}{2})} g(v) + \sum_{v\in V_1\cap[\frac{\lambda}{2},\frac{3\lambda}{4})} h(v)$.

As illustrated in Figure 5(a), both $g(v)$ and $h(v)$ are increasing functions of v. Hence, if we decrease the values in V_0 to some smaller non-negative values, $\sum_{v\in V_0\cap(0,\frac{\lambda}{2})} g(v)+\sum_{v\in V_0\cap[\frac{\lambda}{2},\frac{3\lambda}{4})} h(v)$ does not increase. We keep decreasing the values in V_0 in an arbitrary fashion until $\sum_{v\in V_0} v = \frac{21\lambda}{26}$. Let V_1 be the resulting multi-set. Note that all values in V_1 are in the range $(0, \frac{3\lambda}{4})$.

If $\left|V_1 \cap (0, \frac{\lambda}{2})\right| \geqslant 2$, the following operation is repeated until $\left|V_1 \cap (0, \frac{\lambda}{2})\right| = 1$. Let a and b be two values in $V_1 \cap (0, \frac{\lambda}{2})$. If $a + b \leqslant \frac{\lambda}{2}$, then we replace a and b by $a + b$. If $a + b > \frac{\lambda}{2}$, we replace a and b with $\frac{\lambda}{2}$ and $a + b - \frac{\lambda}{2}$. One can easily verify that the above operation decreases the cardinality of $V_1 \cap (0, \frac{\lambda}{2})$, preserves $\sum_{v\in V_1} v$, and does not increase the value of $\sum_{v\in V_1\cap(0,\frac{\lambda}{2})} g(v) + \sum_{v\in V_1\cap[\frac{\lambda}{2},\frac{3\lambda}{4})} h(v)$.

Now we have $\left|V_1 \cap (0, \frac{\lambda}{2})\right| = 1$. Then $\left|V_1 \cap [\frac{\lambda}{2}, \frac{3\lambda}{4})\right|$ must be equal to 1 because $\sum_{v\in V_1} v = \frac{21\lambda}{26}$. Hence, V_1 meets condition (3), and it is the multi-set we desire.

Now we are ready to derive a lower bound for $\sum_{v\in V_1\cap(0,\frac{\lambda}{2})} g(v) + \sum_{v\in V_1\cap[\frac{\lambda}{2},\frac{3\lambda}{4})} h(v)$. Let $a \in [\frac{\lambda}{2}, \frac{3}{4}\lambda)$ be the larger value in V_1. The smaller value in V_1 is $c - a$ where $c = \frac{21}{26}\lambda$. We have

$$\begin{aligned}
\sum_{v\in V_1\cap(0,\frac{\lambda}{2})} g(v) + \sum_{v\in V_1\cap[\frac{\lambda}{2},\frac{3\lambda}{4})} h(v) &= g(c-a) + h(a) \\
&= \frac{3\lambda(c-a)}{2\lambda+(c-a)} + \frac{3\lambda a}{3\lambda-a} \\
&= 3\lambda\left(1 - \frac{2\lambda}{2\lambda+c-a} + \frac{3\lambda}{3\lambda-a} - 1\right) \\
&= 3\lambda\left(-\frac{2\lambda}{2\lambda+c-a} + \frac{3\lambda}{3\lambda-a}\right).
\end{aligned}$$

One can verify that when $a \in [\frac{\lambda}{2}, \frac{3\lambda}{4})$,

$$\frac{d}{da}\left(g(c-a) + h(a)\right) = 3\lambda\left(-\frac{2\lambda}{(2\lambda+c-a)^2} + \frac{3\lambda}{(3\lambda-a)^2}\right) > 0.$$

So the minimum of $g(c-a) + h(a)$ is attained at $a = \frac{\lambda}{2}$. As a result,

$$\sum_{r\in R'} z_r^* \geqslant g(c-a) + h(a) \geqslant g\left(\frac{21}{26}\lambda - \frac{1}{2}\lambda\right) + h\left(\frac{1}{2}\lambda\right) = \lambda.$$

If follows that $\sum_{r\in C\cap R(\Sigma)} z_r^* = \sum_{r\in R'} z_r^* + 2\lambda \geqslant 3\lambda$. This completes the proof. □

The following lemma is tool for proving Lemma 12. It gives an upper bound for the total dual value of the resources in a hyper-edge, and this upper bound is determined solely by the resource with the least value in that hyper-edge.

Lemma 11. *Let e be an edge that appears in Σ. Let r_0 be the resource with the least value in $R(e)$. Then,*

$$\sum_{r\in R(e)} z_r^* \leqslant \frac{3\lambda}{2} + \frac{z_{r_0}^*}{2}.$$

Proof. Suppose that $v_{r_0} \geqslant \frac{\lambda}{2}$. Then, all resources in R_e have values at least $\frac{\lambda}{2}$. Since e is λ-minimal, $R(e)$ contains exactly two thin resources, including r_0. Let r_1 denote the other resource in $R(e)$. From Figure 5(a), $z_{r_0}^*$ and $z_{r_1}^*$ are at most λ. Thus,

$$\sum_{r\in R(e)} z_r^* = z_{r_0}^* + z_{r_1}^* \leqslant \frac{z_{r_0}^*}{2} + \frac{3\lambda}{2}.$$

Suppose that $v_{r_0} < \frac{\lambda}{2}$. Let r_1 be the resource with the largest value in $R(e)$. If $v_{r_0} \geqslant \lambda - v_{r_1}$, r_0 and r_1 have a total value of at least λ. Thus, $R(e)$ does not contain any other resource because e is λ-minimal. We get $\sum_{r\in R(e)} z_r^* = z_{r_0}^* + z_{r_1}^* \leqslant \frac{z_{r_0}^*}{2} + \frac{3\lambda}{2}$ as before. Suppose that $v_{r_0} < \lambda - v_{r_1}$. Consider an arbitrary resource $r \in R(e)$. Note that $v_{r_0} \leqslant v_r \leqslant v_{r_1}$ by assumption. Therefore, $v_{r_0} < \lambda - v_{r_1} \leqslant \lambda - v_r$. If $v_r \in (0, \frac{\lambda}{2})$, then

$$\frac{z_r^*}{v_r} = \frac{3\lambda}{2\lambda + v_r} \leqslant \frac{3\lambda}{2\lambda + v_{r_0}}.$$

If $v_r \in [\frac{\lambda}{2}, \frac{3}{4}\lambda)$, then

$$\frac{z_r^*}{v_r} = \frac{3\lambda}{3\lambda - v_r} = \frac{3\lambda}{2\lambda + (\lambda - v_r)} < \frac{3\lambda}{2\lambda + v_{r_0}}.$$

If $v_r \in [\frac{3}{4}\lambda, \lambda)$, then $z_r^* = \lambda$. As $v_r \geqslant \frac{3\lambda}{4}$, one can verify that $\frac{\lambda}{v_r} \leqslant \frac{3\lambda}{3\lambda - v_r}$. Thus,

$$\frac{z_r^*}{v_r} = \frac{\lambda}{v_r} \leqslant \frac{3\lambda}{3\lambda - v_r} = \frac{3\lambda}{2\lambda + (\lambda - v_r)} \leqslant \frac{3\lambda}{2\lambda + v_{r_0}}.$$

In summary, for every $r \in R(e)$,

$$\frac{z_r^*}{v_r} \leqslant \frac{3\lambda}{2\lambda + v_{r_0}}.$$

Since e is λ-minimal,

$$\sum_{r\in R(e)} v_r < \lambda + v_{r_0}.$$

Combining these two fact, we obtain

$$\begin{aligned}
\sum_{r\in R_e} z_r^* &= \sum_{r\in R_e} \frac{z_r^*}{v_r} v_r \\
&\leqslant \frac{3\lambda}{2\lambda + v_{r_0}} \sum_{r\in R(e)} v_r \\
&< \frac{3\lambda}{2\lambda + v_{r_0}} (\lambda + v_{r_0}) \\
&= \frac{3\lambda}{2\lambda + v_{r_0}} \left(\lambda + \frac{1}{2} v_{r_0}\right) + \frac{3\lambda}{2\lambda + v_{r_0}} \cdot \frac{1}{2} v_{r_0} \\
&\leqslant \frac{3\lambda}{2} + \frac{z_{r_0}^*}{2}.
\end{aligned}$$

□

Lemma 12. *Let $(M, \mathcal{M}, \Sigma, \ell)$ be a state of the algorithm. Suppose that Σ is not empty. For $\lambda = \frac{26}{99}$, if every addable edge in Σ is blocked, then the objective function value for the associated dual solution $(\{y_p^*\}_{p\in P}, \{z_r^*\}_{r\in R})$ is positive, i.e., $\sum_{p\in P} y_p^* - \sum_{r\in R} z_r^* > 0$.*

Proof. By definition, $y_p^* = 0$ for every $p \notin P^+$ and $z_r^* = 0$ for every $r \notin R_f^+ \cup R(\Sigma)$. Therefore,

$$\sum_{p\in P} y_p^* - \sum_{r\in R} z_r^* = \left(\sum_{p\in P^+} y_p^* - \sum_{r\in R_f^+} z_r^* \right) - \sum_{r\in R(\Sigma)} z_r^*.$$

Consider $\sum_{p\in P^+} y_p^* - \sum_{r\in R_f^+} z_r^*$. For every player $p \in P^+$ and every fat resource $r_f \in R_f^+$, both y_p^* and $z_{r_f}^*$ are equal to $1 - \frac{21}{26}\lambda$. Therefore,

$$\sum_{p\in P^+} y_p^* - \sum_{r\in R_f^+} z_r^* = \left(1 - \frac{21}{26}\lambda\right) \left(|P^+| - |R_f^+| \right).$$

We derive a lower bound for $|P^+| - |R_f^+|$ as follows. By construction, $\mathcal{B}_0 = \{(p_0, \emptyset)\}$ and $B_0 = \{p_0\}$, where p_0 is the unmatched player that we want to match by updating $\mathcal{M}$ and M. Players in the other B_i's are matched by $\mathcal{M}$, so none of them is matched by M. Therefore, no player in $B_{\leqslant \ell}$ is matched by M. Every fat resource that is reachable by a path from some player $B_{\leqslant \ell}$ in G_M must be matched by M; otherwise, such a path would be an augmenting path with respect to M, contradicting the fact that M is a maximum matching of G. Hence, for every fat resource $r_f \in R_f^+$, r_f must have an outgoing edge to some player p in G_M. Since r_f is reachable from $B_{\leqslant \ell}$ in G_M, p is also reachable from $B_{\leqslant \ell}$ in G_M, i.e., $p \in P^+$. We charge r_f to p. Every player in P has in-degree at most 1 in G_M, so it is charged at most once. Every player in $B_{\leqslant \ell}$ is trivially reachable from itself, so $B_{\leqslant \ell} \subseteq P^+$. Since no player in $B_{\leqslant \ell}$ is matched by M, there is no edge entering $B_{\leqslant \ell}$ in G_M, implying that the players in $B_{\leqslant \ell}$ cannot be charged. In conclusion,

$$|P^+| - |R_f^+| \geqslant |B_{\leqslant \ell}|.$$

Putting things together, we get

$$\sum_{p\in P^+} y_p^* - \sum_{r\in R_f^+} z_r^* \geqslant \left(1 - \frac{21}{26}\lambda\right) |B_{\leqslant \ell}|. \tag{3}$$

Consider $\sum_{r\in R(\Sigma)} z_r^*$. As shown in Figure 5(a), z_r^* does not decrease as v_r increases. Hence, Lemma 11 implies that for every edge e in Σ and every resource $r' \in R(e)$,

$$\sum_{r\in R(e)} z_r^* \leqslant \frac{3\lambda}{2} + \frac{z_{r'}^*}{2}. \tag{4}$$

Now consider $(a_i, \mathcal{B}_i)$ for any i. Since every addable edge is blocked, $|\mathcal{B}_i| \geqslant 1$. For simplicity, denote $R(\{a_i\} \cup \mathcal{B}_i)$ as R_i. Consider a resource $r \in R_i$. If r is incident to only one edge in $\{a_i\} \cup \mathcal{B}_i$, then charge z_r^* to that edge. If r is incident to at least two edges, then charge half of z_r^* to one edge and the other half to another edge. Take any edge $e \in \{a_i\} \cup \mathcal{B}_i$. Because $\mathcal{B}_i$ consists of blocking edges of a_i, e must share some resource, say r', with another edge in $\{a_i\} \cup \mathcal{B}_i$. We conclude from (4) that the total charge on e is at most

$$\sum_{r\in R(e)} z_r^* - \frac{z_{r'}^*}{2} \overset{(4)}{\leqslant} \frac{3\lambda}{2}.$$

Summing over all edges in $\{a_i\} \cup \mathcal{B}_i$ gives

$$\sum_{r \in R_i} z_r^* \leqslant |\{a_i\} \cup \mathcal{B}_i| \cdot \frac{3\lambda}{2} = \frac{3\lambda}{2} (|B_i| + 1) \leqslant 3\lambda |B_i|.$$

The last inequality follows from the fact that $|B_i| \geqslant 1$. Recall that $(a_0, \mathcal{B}_0) = (\text{null}, (p_0, \emptyset))$ is incident to no resource. Then, summing over $i \in [1, \ell]$ gives

$$\sum_{r \in R(\Sigma)} z_r^* = \sum_{i=1}^{\ell} \sum_{r \in R_i} z_r^* \leqslant 3\lambda \left(|B_{\leqslant \ell}| - 1 \right). \tag{5}$$

Combining (3) and (5), we obtain

$$\begin{aligned} \sum_{p \in P} y_p^* - \sum_{r \in R} z_r^* &= \sum_{p \in P^+} y_p^* - \sum_{r \in R_f^+} z_r^* - \sum_{r \in R(\Sigma)} z_r^* \\ &\geqslant \left(1 - \frac{21}{26}\lambda\right) |B_{\leqslant \ell}| - 3\lambda \left(|B_{\leqslant \ell}| - 1 \right) \\ &= 3\lambda + \left(1 - \frac{99}{26}\lambda\right) |B_{\leqslant \ell}|. \end{aligned}$$

Given $\lambda = \frac{26}{99}$,

$$\sum_{p \in P} y_p^* - \sum_{r \in R} z_r^* \geqslant 3\lambda > 0.$$

This completes the proof. ◻

Lemma 13. *Let $(M, \mathcal{M}, \Sigma, \ell)$ be a state of the algorithm. Suppose that Σ is not empty. For $\lambda = \frac{26}{99}$, either some addable edge in Σ is unblocked or there is an addable edge to be added to Σ.*

Proof. Assume, for the sake of contradiction, that all addable edges in Σ are blocked, and that no more addable edge can be added to Σ. Consider the associated dual solution $(\{y_p^*\}_{p \in P}, \{z_r^*\}_{r \in R})$ of the state. By Lemmas 10 and 12, for any $\alpha > 0$, $(\{\alpha y_p^*\}_{p \in P}, \{\alpha z_r^*\}_{r \in R})$ is a feasible dual solution with positive objective function value. As α goes to infinity, the objective function value goes to infinity; the dual is unbounded. But this implies the contradiction that the primal LP, $CLP(1)$, is infeasible. ◻

Theorem 1. *The integrality gap of the configuration LP for the restricted max-min fair allocation problem is at most $\frac{99}{29} \approx 3.808$.*

Proof. By Lemmas 8, 7, and 13, for $\lambda = \frac{26}{99}$, the local search can match the unmatched player p_0 in a finite number of steps. By repeating the local search at most n times, we obtain an allocation whose max-min value is at least $\lambda = \frac{26}{99}$. Therefore, the integrality gap of the configuration LP is at most $\frac{99}{26}$. ◻

4 A Generalized Local Search

We present an algorithmic framework that generalizes the local search in Section 3. Again, we assume that the optimal value of the configuration LP is 1. We prove that for $\lambda = \frac{1}{4+\delta}$ where δ is an arbitrary positive constant in $(0, 1)$, our algorithm can take an arbitrary unmatched player p_0 and match p_0 by updating M and $\mathcal{M}$ in $poly(m, n) \cdot n^{poly(1/\delta)}$ time.

4.1 Ideas for Acceleration

We first briefly overview the ideas we use to accelerate the local search in Section 3.2. Some of the ideas are used in previous papers [1, 11]. We introduce a new technique of *limited blocking* which is essential to obtaining the approximation ratio of $4 + \delta$.

- *Layers.* Addable edges and blocking edges are organized in layers. Each layer consists of a set $\mathcal{A}_i$ of addable edges and a set $\mathcal{B}_i$ of $\mathcal{A}_i$'s blocking edges. A key to achieving polynomial running time is to guarantee a geometric growth of $|\mathcal{B}_i|$ with respect to i.

- *Lazy update.* In the local search in Section 3.2, when there is an unblocked addable edge, we immediately use it to update M and $\mathcal{M}$ and release some blocking edge. In order to accelerate the searching procedure, as in [1], we use a set $\mathcal{I}$ to accumulate unblocked addable edges, and take actions only when $\mathcal{I}$ is large enough to release a fraction μ of blocking edges in some layer. Laziness is adjusted using the constant μ.

- *Node-disjoint alternating paths.* We require the alternating paths in G_M that end at players in $A_i = \{p(e) : e \in \mathcal{A}_i\}$ to be node-disjoint. The consequence is that, when a lot of addable edges in $\mathcal{A}_i$ become unblocked, they can be used to release multiple blocking edges simultaneously, and hence a update will be triggered. This idea was used in [11], and was inspired by a similar idea in [1].

- *Greedy and limited blocking.* When proving a geometric growth of $|\mathcal{B}_i|$ with respect to i, it is important to show that, for each layer, the number of its blocking edges is close to that of its addable edges. The closer they are, the better the approximation ratio becomes. In the local search in Section 3.2, each blocking edge blocks exactly one addable edge. Taking this approach, one may get a layer that has one addable edge blocked by many blocking edges, which is bad for our purpose. In the other extreme, if a blocking edge is allowed to block as many addable edges as possible (we call this *unlimited blocking* for short), one may get a layer that has many addable edges blocked by one blocking edge, which is also bad.

  To resolve this issue, Annamalai et al. [1] introduce a *greedy strategy*. The addable edges they use are no longer λ-minimal, but contain more resources than needed. Combined with the unlimited blocking strategy, the greedy strategy gives a lower bound for $|\mathcal{B}_i|$ in terms of $|\mathcal{A}_i|$. When addable edges are $\frac{1}{2}$-minimal, it achieves the best approximation ratio $6 + 2\sqrt{10} + \delta$. However, this greedy strategy is not flexible enough for the approximation ratio to be improved significantly further.

  We introduce a new technique of *limited blocking*. We allow a blocking edge b to block more than one addable edge, but once b shares more than $\beta\lambda$ worth of resources with the addable edges blocked by it, we stop b from blocking more edges in the future. We use a greedy strategy too — addable edges are $(1 + \gamma)\lambda$-minimal. The parameter β reflects the degree of blocking, and γ controls the greediness.

The performance of our algorithm is determined by the three aforementioned parameters μ, β, and γ. If μ, β, and γ are set to 0, the resulting algorithm is essentially the same as the

local search in Section 3.2. It achieves an approximation ratio less than 4, but its running time is not known to be polynomial. If β is a fixed constant greater than or equal to 2, we can set μ and γ accordingly to achieve a polynomial running time. The resulting algorithm resembles the one in [1], and the approximation ratio is a constant larger than and bounded away from 4. We show that by setting μ, β, and γ to be some values that satisfy some simple relations depending on δ, one can guarantee a polynomial running time, while keeping the approximation ratio at $4 + \delta$ for any $\delta \in (0, 1)$.

4.2 Node-disjoint Alternating Paths

Because our algorithm intends to update M and $\mathcal{M}$ in a way that multiple blocking edges can be released simultaneously, it heavily relies on the concept of node-disjoint alternating paths.

Let Π be a set of paths in G_M. Recall that these paths are alternating paths in G with respect to M. We denote the sources of Π as $src(\Pi) = \{src(\pi) : \pi \in \Pi\}$ and the sinks of Π as $sink(\Pi) = \{sink(\pi) : \pi \in \Pi\}$. We say Π is a set of node-disjoint paths if no two paths in Π share a vertex.

Let Π be a set of node-disjoint paths such that $src(\Pi)$ and $sink(\Pi)$ consist of players only, and no player in $src(\Pi)$ is matched by M. In this case, for every path $\pi \in \Pi$, if π is not a singleton vertex, the edge in π incident to $sink(\pi)$ must be a matching edge in M. We extend the $\oplus$ operation to Π. Viewing Π as a set of edges, we have

$$M \oplus \Pi = (M \cup \Pi) \setminus (M \cap \Pi),$$

and $M \oplus \Pi$ is also a maximum matching of G. The ability to update M using Π motivates the following problem definition.

Definition 14. Let S be a set of players not matched by M. Let T be some set of players. Define $G_M[S, T]$ to be the problem of finding a maximum set of node-disjoint paths from S to T in G_M. Every set Π of node-disjoint paths from S to T is a feasible solution for $G_M[S, T]$, and Π is an optimal solution if it achieves the maximum cardinality. Define $f_M[S, T]$ to be the number of paths in an optimal solution for $G_M[S, T]$.

The problem $G_M[S, T]$ can be reduced to maximum flow and can thus be solved in polynomial time by the Ford-Fulkerson algorithm [18, Chapter 7]. Given a feasible solution Π of $G_M[S, T]$, $G_{M\oplus\Pi}$ is well-defined as $M \oplus \Pi$ is a maximum matching of G. In flow terminology, Π can be regarded as a flow in G_M, $G_{M\oplus\Pi}$ is the residual graph of G_M with respect to Π, and every path in $G_{M\oplus\Pi}$ from $S \setminus src(\Pi)$ to $T \setminus sink(\Pi)$ is an augmenting path for Π. The following observation is a restatement of a well-known result for max flow problem.

Observation 15. *Let S be a set of players that are not matched by M. Let T be some set of players. A feasible solution Π for $G_M[S, T]$ is optimal if and only if there is no path in $G_{M\oplus\Pi}$ from $S \setminus src(\Pi)$ to $T \setminus sink(\Pi)$. Moreover, if there is such a path π, then Π can be augmented in polynomial time to another feasible solution Π' such that*

- $|\Pi'| = |\Pi| + 1$,
- $src(\Pi') = src(\Pi) \cup \{src(\pi)\}$,
- $sink(\Pi') = sink(\Pi) \cup \{sink(\pi)\}$, *and*
- *the set of vertices in Π' is a subset of those in $\Pi \cup \{\pi\}$.*

4.3 Description of the Algorithm

The algorithm maintains a stack $\Sigma = [L_0, L_1, \ldots]$ of layers, where each layer L_i is a quadruple $(\mathcal{A}_i, \mathcal{B}_i, d_i, z_i)$. The set $\mathcal{A}_i$ consists of blocked addable edges, and $\mathcal{B}_i$ is the set of $\mathcal{A}_i$'s blocking edges. The definitions of addable and blocking edges will be given later. The values d_i and z_i are kept only for the sake of analysis. The algorithm also maintains a set $\mathcal{I}$ as a global variable to accumulate unblocked addable edges. We use ℓ to denote the index of the topmost layer in the stack Σ. The state of the algorithm is fully characterized by $(M, \mathcal{M}, \Sigma, \ell, \mathcal{I})$.

Initially, $\mathcal{I} = \emptyset$, $\ell = 0$, and $\Sigma = [(\mathcal{A}_0, \mathcal{B}_0, 0, 0)]$ where $\mathcal{A}_0 = \emptyset$, $\mathcal{B}_0 = \{(p_0, \emptyset)\}$, and p_0 is the unmatched player that we want to match next. The algorithm alternates between the build phase and the collapse phase until p_0 is matched. In the build phase, the algorithm grows Σ by adding a new layer. When some layer becomes *collapsible* (definition to be given later), it switches to the collapse phase. In the collapse phase, the algorithm removes collapsible layers, and updates M and $\mathcal{M}$. When no collapsible layer is left, it switches back to the build phase.

Define $A_i = \{p(e) : e \in \mathcal{A}_i\}$, $B_i = \{p(e) : e \in \mathcal{B}_i\}$, and $I = \{p(e) : e \in \mathcal{I}\}$. We use $\mathcal{A}_{\leqslant k}$ to denote $\bigcup_{i=1}^{k} \mathcal{A}_i$. $\mathcal{B}_{\leqslant k}$, and $B_{\leqslant k}$ are similarly defined.

4.3.1 Build Phase

Suppose that the current state of the algorithm is $(M, \mathcal{M}, \Sigma, \ell, \mathcal{I})$ and that the algorithm is about to construct a new layer $L_{\ell+1}$. The sets $\mathcal{A}_{\ell+1}$ and $\mathcal{B}_{\ell+1}$ are initialized to be empty, and they will grow as the build phase progresses. We first define *active* and *inactive* thin resources, addability of players and edges, and blocking edges.

Definition 16. Let $\beta \geqslant 0$ be a constant to be specified later. A thin resource r is either *active* or *inactive*, and r is inactive if and only if one of the following conditions is satisfied:

(i) $r \in R(\mathcal{A}_{\leqslant \ell} \cup \mathcal{B}_{\leqslant \ell})$, or

(ii) $r \in R(\mathcal{A}_{\ell+1} \cup \mathcal{I})$, or

(iii) $r \in R(e)$ for some $e \in \mathcal{B}_{\ell+1}$ such that $v[R(e) \cap R(\mathcal{A}_{\ell+1})] > \beta\lambda$.

Definition 17. A player p is *addable* if $f_M[B_{\leqslant \ell}, A_{\ell+1} \cup I \cup \{p\}] = f_M[B_{\leqslant \ell}, A_{\ell+1} \cup I] + 1$. A hyper-edge (p, S) is *addable* if p is addable, all the thin resources in S are active, and $v[S] \geqslant \lambda$.

Definition 18. Given an addable edge a, if an edge b in $\mathcal{M}$ shares some resource with a, i.e., $R(a) \cap R(b) \neq \emptyset$, then b is a *blocking* edge of a. We also say that a is *blocked by* b. An addable edge a is *unblocked* if $v[R(a) \setminus R(\mathcal{M})] \geqslant \lambda$, and *blocked* otherwise.

Condition (iii) in Definition 16 is the key to the limited blocking strategy. When a blocking edge in $\mathcal{B}_{\ell+1}$ shares more than $\beta\lambda$ worth of resources with addable edges in $\mathcal{A}_{\ell+1}$, all of its resources become inactive, and hence, cannot be included in future addable edges. As a result, no future addable edge can be blocked by this blocking edge.

Remark 19. Definitions 16 and 17 involve $\mathcal{A}_{\ell+1}$, $\mathcal{B}_{\ell+1}$, and $\mathcal{I}$. As we add edges to $\mathcal{A}_{\ell+1}$, $\mathcal{B}_{\ell+1}$, and $\mathcal{I}$, the activeness and the addability of resources and edges may be altered.

Our algorithm considers unblocked addable edges that are λ-minimal and blocked addable edges that are $(1+\gamma)\lambda$-minimal, where γ is a value to be specified later. Edges of the former kind are stored in $\mathcal{I}$. Edges of the latter kind are added to $\mathcal{A}_{\ell+1}$.

The build phase constructs a new layer $L_{\ell+1}$ in Σ by calling the routine BUILD in Figure 6.

BUILD$(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell)$

1. Initialize $\mathcal{A}_{\ell+1}$ and $\mathcal{B}_{\ell+1}$ to be empty sets.
2. While there is an unblocked addable edge that is λ-minimal, add it to $\mathcal{I}$.
3. While there is an addable edge (p, S) that is $(1+\gamma)\lambda$-minimal:
   3.1 add (p, S) to $\mathcal{A}_{\ell+1}$;
   3.2 add the blocking edges of (p, S) to $\mathcal{B}_{\ell+1}$.
4. Set $d_{\ell+1} := f_M[B_{\leqslant \ell}, A_{\ell+1} \cup I]$, and $z_{\ell+1} := |A_{\ell+1}|$.
5. Push $L_{\ell+1} = (\mathcal{A}_{\ell+1}, \mathcal{B}_{\ell+1}, d_{\ell+1}, z_{\ell+1})$ onto Σ. Update $\ell := \ell + 1$.

Figure 6: Routine BUILD.

4.3.2 Collapse Phase

A layer is collapsible if more than a fraction μ of its blocking edges can be released. The formal definition is given below.

Definition 20. Let μ be a constant to be specified later. Given a state $(M, \mathcal{M}, \Sigma, \ell, \mathcal{I})$ of the algorithm, the layer L_0 is *collapsible* if $f_M[B_0, I] > \mu|B_0| = \mu$, and for $i \in [1, \ell]$, the layer L_i is *collapsible* if $f_M[B_{\leqslant i}, I] - f_M[B_{\leqslant i-1}, I] > \mu|B_i|$.

In order to release the blocking edges using the $\oplus$ operation, we need to identify $f_M[B_{\leqslant i}, I] - f_M[B_{\leqslant i-1}, I]$ node-disjoint paths in G_M. This brings about the concept of a canonical solution.

Definition 21. Given a state $(M, \mathcal{M}, \Sigma, \ell, \mathcal{I})$ of the algorithm, an optimal solution Π^c for $G_M[B_{\leqslant \ell}, I]$ is a *canonical solution* if, for every $i \in [0, \ell]$, Π^c contains $f_M[B_{\leqslant i}, I]$ node-disjoint paths from $B_{\leqslant i}$ to I.

We will show later in Lemma 23 that a canonical solution always exists and it can be computed in polynomial time. Suppose a layer L_i is collapsible. By Definitions 20 and 21, a canonical solution Π^c of $G_M[B_{\leqslant \ell}, I]$ contains $f_M[B_{\leqslant i}, I] - f_M[B_{\leqslant i-1}, I] > \mu|B_i|$ node-disjoint paths from B_i to I. Each of these paths can be used to release a blocking edge in $\mathcal{B}_i$ via an $\oplus$ operation with the current maximum matching M. In total, more than $\mu|B_i|$ blocking edges in $\mathcal{B}_i$ can be released. There are $\ell + 1$ groups of paths in Π^c from B_i to I for $i \in [0, \ell]$. Let I_i be the set of the sinks of the group from B_i to I. Let $I_{\leqslant j} = \bigcup_{i=0}^{j} I_i$. Let $\mathcal{I}_i$ and $\mathcal{I}_{\leqslant j}$ denote the subsets of edges in $\mathcal{I}$ that are incident to players in I_i and $I_{\leqslant j}$, respectively.

Our algorithm enters the collapse phase whenever a layer becomes collapsible. Then, the routine COLLAPSE in Figure 7 is invoked repeatedly until no collapsible layer is left.

The following observation is clear from the description of BUILD and COLLAPSE.

Observation 22. *Let $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell)$ be a state of the algorithm before the invocation of* BUILD.

- *No two edges in $\mathcal{A}_{\leqslant \ell} \cup \mathcal{I}$ share a resource.*
- *For every $i \in [1, \ell]$, no two edges in $\mathcal{A}_i \cup \mathcal{I}$ are incident to the same player.*
- *No edge in $\mathcal{I}$ shares a resource with any edge in $\mathcal{M}$.*
- *$\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_\ell$ are disjoint subsets of $\mathcal{M}$. Recall that $\mathcal{B}_0$ is defined to be $\{(p_0, \emptyset)\}$.*

COLLAPSE$(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell)$

1. Let L_k be the collapsible layer with the smallest index.
2. Compute a canonical solution Π^c for $G_M[B_{\leqslant \ell}, I]$. For $i \in [0, \ell]$, let $\Pi_i^c = \{\pi \in \Pi^c : src(\pi) \in B_i\}$, let $I_i = sink(\Pi_i^c)$, and let $\mathcal{I}_i$ be the set of edges in $\mathcal{I}$ that are incident to players in I_i.
3. Remove all layers above L_k from the stack. Set $\mathcal{I} := \mathcal{I}_{\leqslant k-1}$.
4. Let $\mathcal{B}^* = \{e \in \mathcal{B}_k : p(e) \in src(\Pi_k^c)\}$. We use $\mathcal{I}_k$ and Π_k^c to release the blocking edges in $\mathcal{B}^*$ as follows.
   - 4.1 $M := M \oplus \Pi_k^c$ and $\mathcal{M} := \mathcal{M} \cup \mathcal{I}_k$.
   - 4.2 If $k = 0$, then p_0 is already matched and the algorithm terminates.
   - 4.3 If $k \geqslant 1$, each player that is incident to a hyper-edge in $\mathcal{B}^*$ has been matched by a new edge, so we release $\mathcal{B}^*$ from $\mathcal{M}$ by setting $\mathcal{M} := \mathcal{M} \setminus \mathcal{B}^*$ and $\mathcal{B}_k := \mathcal{B}_k \setminus \mathcal{B}^*$.
5. If $k \geqslant 1$, we need to update $\mathcal{A}_k$ because the removal of the blocking edges in $\mathcal{B}^*$ from $\mathcal{B}_k$ may make some addable edges in $\mathcal{A}_k$ unblocked. For each addable edge $(p, S) \in \mathcal{A}_k$ that becomes unblocked, perform the following operations:
   - 5.1 Remove (p, S) from $\mathcal{A}_k$.
   - 5.2 If $f_M[B_{\leqslant k-1}, I \cup \{p\}] = f_M[B_{\leqslant k-1}, I] + 1$, then extract a λ-minimal unblocked addable edges (p, S') from (p, S) and add (p, S') to $\mathcal{I}$.
6. Update $\ell := k$.

Figure 7: Routine COLLAPSE.

Table 1: Invariants with respect to the state $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell)$.

Invariant 1	$f_M[B_{\leqslant \ell-1}, I] = \lvert I \rvert$.
Invariant 2	For every $i \in [0, \ell-1]$, $f_M[B_{\leqslant i}, A_{i+1} \cup I] \geqslant d_{i+1}$.
Invariant 3	For every $i \in [0, \ell]$, $\lvert A_i \rvert \leqslant z_i$ and $d_i \geqslant z_i$.

Lemma 23 below shows that a canonical solution for $G_M[B_{\leqslant \ell}, I]$ can be computed in polynomial time.

Lemma 23. *Given a state $(M, \mathcal{M}, \Sigma, \ell, \mathcal{I})$ of the algorithm, a canonical solution Π^c for $G_M[B_{\leqslant \ell}, I]$ always exists and can be computed in poly(m, n, ℓ) time.*

Proof. We start with an optimal solution Π_0 for $G_M[B_0, I]$ which can be computed in $poly(m, n)$ time by the Ford-Fulkerson algorithm. For $j = 1, 2, \ldots, \ell$, we obtain an optimal solution Π_j for $G_M[B_{\leqslant j}, I]$ from Π_{j-1} by repeated augmentation. By Observation 15, for any $j \in [0, \ell]$, $src(\Pi_j) \subseteq src(\Pi_\ell)$. Note that $\lvert src(\Pi_j) \rvert = f_M[B_{\leqslant j}, I]$ as Π_j is an optimal solution for $G_M[B_{\leqslant j}, I]$. Therefore, for $j \in [0, \ell]$, Π_ℓ contains exactly $f_M[B_{\leqslant j}, I]$ node-disjoint paths from $B_{\leqslant j}$ to I, and hence is a canonical solution. □

4.4 Invariants

We prove that our algorithm maintains some invariants as listed in Table 1. At the beginning, our algorithm initializes $\ell = 0$, $\mathcal{I} = \mathcal{A}_0 = \emptyset$, $\mathcal{B}_0 = \{(p_0, \emptyset)\}$, $d_0 = 0$, and $z_0 = 0$. So invariants 1 to 3 are satisfied trivially. Before analyzing the effects of Build and Collapse on the invariants, we start with a property of node-disjoint paths.

Lemma 24. *Let M be a maximum matching of G. Let S be a set of players not matched by M. Let T be some set of players. For every player p, if $f_M[S, T \cup \{p\}] = f_M[S, T] + 1$, then for every subset $T' \subseteq T$, $f_M[S, T' \cup \{p\}] = f_M[S, T'] + 1$.*

Proof. Suppose that $f_M[S, T \cup \{p\}] = f_M[S, T] + 1$. Clearly, $p \notin T$. Let T' be an arbitrary subset of T. Then, $f_M[S, T'] + 1$ is a trivial upper bound on $f_M[S, T' \cup \{p\}]$. We show that it is also a lower bound.

Let Π_1 be an optimal solution for $G_M[S, T']$. Since $p \notin T \supseteq T'$, we have $p \notin sink(\Pi_1)$. Note that Π_1 is a feasible solution for $G_M[S, T \cup \{p\}]$. Let Π_2 be an optimal solution for $G_M[S, T \cup \{p\}]$ obtained from Π_1 by repeated augmentation. By Observation 15, $sink(\Pi_1) \subseteq sink(\Pi_2)$.

If $p \in sink(\Pi_2)$, then Π_2 contains $\lvert src(\Pi_1) \cup \{p\} \rvert = f_M[S, T'] + 1$ node-disjoint paths from S to $T' \cup \{p\}$. Therefore, $f_M[S, T' \cup \{p\}] \geqslant f_M[S, T'] + 1$.

If $p \notin sink(\Pi_2)$, then Π_2 is a feasible solution for $G_M[S, T]$. But then $f_M[S, T \cup \{p\}] = \lvert \Pi_2 \rvert \leqslant f_M[S, T]$, contradicting the assumption that $f_M[S, T \cup \{p\}] = f_M[S, T] + 1$. □

We prove that Build preserves the invariants in Table 1.

Lemma 25. Build *maintains the invariants in Table 1.*

Proof. Let $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell)$ be the state of the algorithm before Build. Suppose that Build is about to construct a new layer $L_{\ell+1}$ and that all the invariants hold. We show that these invariants hold after the construction.

Consider Invariant 1. It holds before the construction of layer $L_{\ell+1}$, so $f_M[B_{\leqslant \ell-1}, I] = \lvert I \rvert$. Clearly, $\lvert I \rvert = f_M[B_{\leqslant \ell-1}, I] \leqslant f_M[B_{\leqslant \ell}, I] \leqslant \lvert I \rvert$. It follows that $f_M[B_{\leqslant \ell}, I] = \lvert I \rvert$ until step 2

of BUILD changes $\mathcal{I}$ by adding unblocked addable edges to it. Consider the moment that an unblocked addable edge (p, S) is about to be inserted to $\mathcal{I}$. By Definition 17, p must be an addable player, that is,

$$f_M[B_{\leqslant \ell}, A_{\ell+1} \cup I \cup \{p\}] = f_M[B_{\leqslant \ell}, A_{\ell+1} \cup I] + 1.$$

Then, Lemma 24 implies that

$$f_M[B_{\leqslant \ell}, I \cup \{p\}] = f_M[B_{\leqslant \ell}, I] + 1.$$

Therefore, when we add an unblocked addable edge to $\mathcal{I}$, both the left and right hand sides of invariant 1 increase by 1. Hence, invariant 1 is preserved.

Consider Invariant 2. Since old layers are not changed and $\mathcal{I}$ is only enlarged by BUILD, for $i \in [0, \ell-1]$, the value of $f_M[B_{\leqslant i}, A_{i+1} \cup I]$ does not decrease, which implies that the inequality $f_M[B_{\leqslant i}, A_{i+1} \cup I] \geqslant d_{i+1}$ continues to hold. For $i = \ell$, the inequality holds because $d_{\ell+1}$ is set to be $f_M[B_{\leqslant \ell}, A_{\ell+1} \cup I]$ for the new layer $L_{\ell+1}$.

Consider Invariant 3. Since BUILD does not change any old layer, the inequalities in Invariant 3 for $i \in [0, \ell]$ are preserved. For $i = \ell + 1$, the inequality $|A_{\ell+1}| \leqslant z_{\ell+1}$ holds because $z_{\ell+1}$ is set to be $|A_{\ell+1}|$ after the construction of $L_{\ell+1}$. The set $\mathcal{A}_{\ell+1}$ was initialized to be empty at the beginning of the construction of $L_{\ell+1}$. Whenever an addable edge (p, S) is added to $\mathcal{A}_{\ell+1}$, Definition 17 implies that the value of $f_M[B_{\leqslant \ell}, A_{\ell+1} \cup I]$ increases by 1. Therefore, after the construction of $L_{\ell+1}$, we have

$$d_{\ell+1} = f_M[B_{\leqslant \ell}, A_{\ell+1} \cup I] \geqslant |A_{\ell+1}| = z_{\ell+1}.$$

□

We move on to discuss how COLLAPSE maintains the invariants in Table 1. We need some notations to facilitate the discussion.

Definition 26. Let $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell)$ be a state of the algorithm. Given a canonical solution Π^c of $G_M[B_{\leqslant \ell}, I]$, let $\Pi^c_i = \{\pi \in \Pi^c : src(\pi) \in B_i\}$ and let $I_i = sink(\Pi^c_i)$ for $i \in [0, \ell]$. We say that $(\Pi^c_i, I_i)_{i \in [0,\ell]}$ is *the partition induced* by Π^c. Define $\Pi^c_{\leqslant i} = \bigcup_{j=0}^{i} \Pi^c_j$ and $\Pi^c_{\geqslant i+1} = \bigcup_{j=i+1}^{\ell} \Pi^c_j$ for $i \in [0, \ell]$. Also, let $I_{\geqslant i+1} = \bigcup_{j=i+1}^{\ell} I_j$ for $i \in [0, \ell-1]$; I_i and $I_{\leqslant i}$ were defined previously.

We first prove a technical lemma.

Lemma 27. *Let $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell)$ be a state of the algorithm. Let Π^c be a canonical solution for $G_M[B_{\leqslant \ell}, I]$. Let $(\Pi^c_i, I_i)_{i \in [0,\ell]}$ be the partition induced by Π^c. For every $i \in [0, \ell-1]$, $G_M[B_{\leqslant i}, A_{i+1} \cup I_{\leqslant i}]$ and $G_M[B_{\leqslant i}, A_{i+1} \cup I]$ share a common optimal solution that is node-disjoint from $\Pi^c_{\geqslant i+1}$.*

Proof. Fix some $i \in [0, \ell-1]$. By the definitions of a canonical solution and the partition induced by Π^c, $\Pi^c_{\leqslant i}$ is an optimal solution for $G_M[B_{\leqslant i}, I]$ and hence a feasible solution for $G_M[B_{\leqslant i}, A_{i+1} \cup I]$. Let Π^* be an optimal solution for $G_M[B_{\leqslant i}, A_{i+1} \cup I]$ obtained from $\Pi^c_{\leqslant i}$ by repeated augmentation. By Observation 15, $I_{\leqslant i} \subseteq sink(\Pi^*)$.

We claim that for any player $p \in I \setminus I_{\leqslant i}$, $p \notin sink(\Pi^*)$. Otherwise, Π^* would contain $|I_{\leqslant i} \cup \{p\}| = f_M[B_{\leqslant i}, I] + 1$ node-disjoint paths from $B_{\leqslant i}$ to I, contradicting the optimality of $f_M[B_{\leqslant i}, I]$.

By our claim, Π^* is also an optimal solution for $G_M[B_{\leqslant i}, A_{i+1} \cup I_{\leqslant i}]$.

Next we show that Π^* is node-disjoint from $\Pi^c_{\geqslant i+1}$. If $\Pi^c_{\geqslant i+1} = \emptyset$, the node-disjointness is trivial. Suppose that $\Pi^c_{\geqslant i+1} \neq \emptyset$. Assume to the contrary that Π^* is not node-disjoint from

$\Pi^c_{\geqslant i+1}$. Recall that Π^* is obtained from $\Pi^c_{\leqslant i}$ by repeated augmentation, and that $\Pi^c_{\leqslant i}$ is node-disjoint from Π^c_{i+1}. It must be that during the repeated augmentation, some augmenting path used intersects with some path in $\Pi^c_{\geqslant i+1}$. Let π be the first such augmenting path. Let Π be the feasible solution for $G_M[B_{\leqslant i}, A_{i+1} \cup I]$ immediately before the augmentation using π. Since Π is node-disjoint from $\Pi^c_{\geqslant i+1}$, the paths in $\Pi^c_{\geqslant i+1}$ remain to be paths in $G_{M\oplus\Pi}$. Since π intersects some path in $\Pi^c_{\geqslant i+1}$, we can get an augmenting path π' from $src(\pi)$ to a player in $sink(\Pi^c_{\geqslant i+1}) = I_{\geqslant i+1}$by following π and switching at the intersecting vertex to a path in $\Pi^c_{\geqslant i+1}$. Using π', we can augment Π to another feasible solution Π' for $G_M[B_{\leqslant i}, A_{i+1} \cup I]$. By Observation 15, $I_{\leqslant i} \subseteq sink(\Pi')$ and there is a player $p = sink(\pi') \in I_{\geqslant i+1}$ that is also in $sink(\Pi')$. Then Π' contains $|I_{\leqslant i} \cup \{p\}| = f_M[B_{\leqslant i}, I] + 1$ node-disjoint paths from $B_{\leqslant i}$ to I, contradicting the optimality of $f_M[B_{\leqslant i}, I]$. This completes the proof. ◻

We show that Collapse preserves the invariants.

Lemma 28. Collapse *maintains the invariants in Table 1.*

Proof. Collapse removes all the layers above L_k, so it suffices to prove the invariants with ℓ replaced by k. Steps 1, 2, and 6 of Collapse clearly have no effect on these invariants. We show that the invariants are preserved by steps 3, 4, and 5 of Collapse.

Consider Invariant 1. Step 3 removes all the layers above L_k and sets $\mathcal{I} := \mathcal{I}_{k-1}$, so we should show that $f_M[B_{\leqslant k-1}, I_{\leqslant k-1}] = |I_{\leqslant k-1}|$. Recall that $\Pi^c_{\leqslant k-1}$ is a set of node-disjoint paths in G_M originating from $B_{\leqslant k-1}$ and that $I_{\leqslant k-1} = sink(\Pi^c_{\leqslant k-1})$. Hence, $\Pi^c_{\leqslant k-1}$ certifies that $f_M[B_{\leqslant k-1}, I_{\leqslant k-1}] = |I_{\leqslant k-1}|$ immediately after step 3. In step 4, M is updated by taking the symmetric difference with Π^c_k. As $\Pi^c_{\leqslant k-1}$ is node-disjoint from Π^c_k, after executing $M := M \oplus \Pi^c_k$, $\Pi^c_{\leqslant k-1}$ remains as a set of node-disjoint paths in G_M and certifies that $f_M[B_{\leqslant k-1}, I_{\leqslant k-1}] = |I_{\leqslant k-1}|$ after step 4. Step 5 does not break the equation either because step 5 inserts a new addable edge into $\mathcal{I}$ only if $f_M[B_{\leqslant k-1}, I]$ will increase by 1. Therefore, invariant 1 is preserved.

Consider Invariant 2. Fix an $i \in [0, k-1]$. Before Collapse starts, by Lemma 27, $G_M[B_{\leqslant i}, A_{i+1} \cup I_{\leqslant i}]$ and $G_M[B_{\leqslant i}, A_{i+1} \cup I]$ share a common optimal solution, say Π^*, that is node-disjoint from $\Pi^c_{\geqslant i+1}$. As the invariants hold before Collapse starts,

$$|\Pi^*| = f_M[B_{\leqslant i}, A_{i+1} \cup I_{\leqslant i}] = f_M[B_{\leqslant i}, A_{i+1} \cup I] \geqslant d_{i+1}.$$

Obviously Π^* is not affected by step 3. Step 4 has no effect on Π^* because Π^* is node-disjoint from $\Pi^c_{\geqslant i+1}$ which includes Π^c_k. Hence, after step 4, Π^* certifies that

$$f_M[B_{\leqslant i}, A_{i+1} \cup I_{\leqslant k-1}] \geqslant |\Pi^*| \geqslant d_{i+1}.$$

In step 5.1, addable edges may be removed from $\mathcal{A}_k$. If $i < k-1$, this does not change $f_M[B_{\leqslant i}, A_{i+1} \cup I_{\leqslant k-1}]$. Suppose that $i = k-1$ and that $f_M[B_{\leqslant k-1}, A_k \cup I_{\leqslant k-1}]$ decreases after an addable edge (p, S) is removed from $\mathcal{A}_k$. That is,

$$f_M[B_{\leqslant k-1}, A_k \cup I] = f_M[B_{\leqslant k-1}, (A_k \setminus \{p\}) \cup I] + 1.$$

Then Lemma 24 implies that

$$f_M[B_{\leqslant k-1}, I \cup \{p\}] = f_M[B_{\leqslant k-1}, I] + 1.$$

As a consequence, step 5.2 will extract an unblocked addable edge (p, S') from (p, S) and add (p, S') to $\mathcal{I}$. After step 5.2, $f_M[B_{\leqslant k-1}, A_k \cup I]$ returns to its value prior to the removal of (p, S) from $\mathcal{A}_k$. Therefore, invariant 2 is preserved.

Invariant 3 is preserved because, for all $i \in [1, k]$, d_i and z_i do not change once they are set in Build, and $\mathcal{A}_i$ does not grow in Collapse. ◻

Remark 29. By Invariant 1 in Table 1, whenever $\ell \geqslant 1$, the top layer in Σ is not collapsible because $f_M[B_{\leqslant \ell}, I] - f_M[B_{\leqslant \ell-1}, I] \leqslant |I| - |I| = 0$. It also follows that $I_\ell = \emptyset$ in the partition induced by any canonical solution of $G_M[B_{\leqslant \ell}, I]$.

4.5 Relations among $|\mathcal{A}_i|$, $|\mathcal{B}_i|$, and $|\mathcal{I}|$

We first show that when the size of $\mathcal{I}$ is large enough or a layer loses a lot of its blocked addable edges, then some layer must be collapsible.

Lemma 30. *Let $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell)$ be a state of the algorithm. If no layer in Σ is collapsible, the following properties are satisfied.*

(i) $|I| \leqslant \mu |B_{\leqslant \ell-1}|$.

(ii) *For every $i \in [0, \ell-1]$, $|A_{i+1}| \geqslant z_{i+1} - \mu |B_{\leqslant i}|$.*

Proof. We first prove property (i). Since no layer is collapsible, Definition 20 implies that $f_M[B_0, I] \leqslant \mu |B_0|$ and that for $i \in [1, \ell-1]$, $f_M[B_{\leqslant i}, I] - f_M[B_{\leqslant i-1}, I] \leqslant \mu |B_i|$. Summing up these inequalities gives $f_M[B_{\leqslant \ell-1}, I] \leqslant \mu |B_0| + \cdots + \mu |B_{\ell-1}| = \mu |B_{\leqslant \ell-1}|$. Then by Invariant 1 in Table 1, we have $|I| = f_M[B_{\leqslant \ell-1}, I] \leqslant \mu |B_{\ell-1}|$.

Consider property (ii). Fix an $i \in [0, \ell-1]$. From the analysis of property (i), we obtain $f_M[B_{\leqslant i}, I] = f_M[B_0, I] + \cdots + f_M[B_i, I] \leqslant \mu |B_0| + \cdots + \mu |B_i| \leqslant \mu |B_{\leqslant i}|$. By Invariant 2 in Table 1, we have $f_M[B_{\leqslant i}, A_{i+1} \cup I] \geqslant d_{i+1}$. The first inequality implies that there are at most $\mu |B_{\leqslant i}|$ node-disjoint paths in G_M from $B_{\leqslant i}$ to I. The second inequality implies that there are at least d_{i+1} node-disjoint paths in G_M from $B_{\leqslant i}$ to $A_{i+1} \cup I$. Therefore, there are at least $d_{i+1} - \mu |B_{\leqslant i}|$ node-disjoint paths in G_M from $B_{\leqslant i}$ to A_{i+1}, which implies that $|A_{i+1}| \geqslant d_{i+1} - \mu |B_{\leqslant i}|$. Since $d_{i+1} \geqslant z_{i+1}$ by Invariant 3 in Table 1, we have $|A_{i+1}| \geqslant z_{i+1} - \mu |B_{\leqslant i}|$. ◻

The next two results show that the number of blocking edges in a layer is asymptotically bounded from below by the number of addable edges in the same layer.

Lemma 31. *Let $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell)$ be a state of the algorithm. For every $i \in [0, \ell]$ and every blocking edge $b \in \mathcal{B}_i$, there is an edge $a \in \mathcal{A}_i$ such that $v[R(b) \cap R(\mathcal{A}_i \setminus \{a\})] \leqslant \beta\lambda$.*

Proof. Sort the edges in $\mathcal{A}_i$ in chronological order of their additions to $\mathcal{A}_i$. Take any blocking edge $b \in \mathcal{B}_i$. Let a be the last edge in $\mathcal{A}_i$ that is blocked by b. By our choice of a, hyper-edges in $\mathcal{A}_i$ that were added after a cannot be blocked by b, so they do not share any resource with b. Since a is blocked by b, at the time when a was added to $\mathcal{A}_i$, some resource in b must be active. By Definition 16, in order that some resource in b is active, the edges in $\mathcal{A}_i$ that were added before a can share at most $\beta\lambda$ worth of resources with b. This completes the proof. ◻

Lemma 32. *Let $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell)$ be a state of the algorithm. For every $i \in [0, \ell]$, $|A_i| < \left(1 + \frac{\beta}{\gamma}\right) |B_i|$.*

Proof. By Lemma 31, for each $b \in \mathcal{B}_i$, we can identify an edge $a_b \in \mathcal{A}_i$ so that

$$v[R(b) \cap R(\mathcal{A}_i \setminus \{a_b\})] \leqslant \beta\lambda.$$

Let $\mathcal{A}_i^0 = \{a_b : b \in \mathcal{B}_i\}$ be the set of edges identified. Clearly $|\mathcal{A}_i^0| \leqslant |B_i|$. Let $\mathcal{A}_i^1 = \mathcal{A}_i \setminus \mathcal{A}_i^0$. For all $b \in \mathcal{B}_i$,

$$v[R(b) \cap R(\mathcal{A}_i^1)] \leqslant v[R(b) \cap R(\mathcal{A}_i \setminus \{a_b\})] \leqslant \beta\lambda.$$

Summing the above inequality over all edges b in $\mathcal{B}_i$, we get

$$v[R(\mathcal{B}_i) \cap R(\mathcal{A}_i^1)] \leqslant \beta\lambda|B_i|. \tag{6}$$

On the other hand, each edge a in $\mathcal{A}_i^1$ is blocked, so it must have more than $\gamma\lambda$ worth of its resources occupied by edges in $\mathcal{B}_i$. So $v[R(\mathcal{B}_i) \cap R(a)] > \gamma\lambda$. No two edges in $\mathcal{B}_i$ share a resource as they are matching edges in $\mathcal{M}$. Therefore, summing the inequality over all edges in $\mathcal{A}_i^1$ gives

$$v[R(\mathcal{B}_i) \cap R(\mathcal{A}_i^1)] > \gamma\lambda|\mathcal{A}_i^1|. \tag{7}$$

Combining (6) and (7) gives

$$\beta\lambda|B_i| \geqslant v[R(\mathcal{B}_i) \cap R(\mathcal{A}_i^1)] > \gamma\lambda|\mathcal{A}_i^1| \Rightarrow |\mathcal{A}_i^1| < \frac{\beta}{\gamma}|B_i|.$$

Finally, we have $|A_i| = |\mathcal{A}_i^0| + |\mathcal{A}_i^1| < |B_i| + \frac{\beta}{\gamma}|B_i| = (1 + \frac{\beta}{\gamma})|B_i|$. ◻

The last result in this section shows that not many blocking edges in $\mathcal{B}_i$ share more than $\beta\lambda$ worth of resources with addable edges in $\mathcal{A}_i$.

Lemma 33. *Let $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell)$ be a state of the algorithm. For every $i \in [0, \ell]$, the cardinality of $\mathcal{B}_i' = \{b \in \mathcal{B}_i : v[R(b) \cap R(\mathcal{A}_i)] > \beta\lambda\}$ is less than $\frac{2+\gamma}{\beta}|A_i|$.*

Proof. No two edges in $\mathcal{B}_i$ share a resource as they are matching edges in $\mathcal{M}$. Then, since $v[R(b) \cap R(\mathcal{A}_i)] > \beta\lambda$ for every $b \in \mathcal{B}_i'$, summing over all edges b in $\mathcal{B}_i'$ gives

$$v[R(\mathcal{B}_i') \cap R(\mathcal{A}_i)] > \beta\lambda|\mathcal{B}'_i|.$$

Every edge in $\mathcal{A}_i$ is $(1+\gamma)\lambda$-minimal, so $v[R(\mathcal{A}_i)] < (2+\gamma)\lambda|A_i|$. Therefore,

$$v[R(\mathcal{B}_i') \cap R(\mathcal{A}_i)] \leqslant v[R(\mathcal{A}_i)] < (2+\gamma)\lambda|A_i|.$$

Combining the above two inequalities, we obtain

$$\beta\lambda|\mathcal{B}'_i| < (2+\gamma)\lambda|A_i| \Rightarrow |\mathcal{B}_i'| < \frac{2+\gamma}{\beta}|A_i|.$$

◻

4.6 Bounding the initial size of $|\mathcal{A}_i|$

The key result in this section is Lemma 38: immediately after a layer $L_{\ell+1}$ is constructed, $L_{\ell+1}$ has a lot of addable edges, or some layer is collapsible. We first introduce some notations.

Definition 34. Let $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell+1)$ be the state of the algorithm **immediately after** the construction of a new layer $L_{\ell+1}$. Let Π be an arbitrary optimal solution for $G_M[B_{\leqslant\ell}, A_{\ell+1} \cup I]$. The *associated players and resources* (P^+, R_f^+, R_t^+) with respect to Π consist of:

- P^+ is the set of players that are reachable from $B_{\leqslant\ell} \setminus src(\Pi)$ in $G_{M\oplus\Pi}$,
- R_f^+ is the set of fat resources that are reachable from $B_{\leqslant\ell} \setminus src(\Pi)$ in $G_{M\oplus\Pi}$, and
- $R_t^+ = R(\mathcal{A}_{\leqslant\ell+1} \cup \mathcal{B}_{\leqslant\ell} \cup I) \cup R(\mathcal{B}'_{\ell+1})$, where $\mathcal{B}'_{\ell+1} = \{e \in \mathcal{B}_{\ell+1} : v[R(e) \cap R(\mathcal{A}_{\ell+1})] > \beta\lambda\}$.

The *associated dual solution* with respect to Π for the dual of $CLP(1)$ is defined as:

$$y_p^* = \begin{cases} 1-(1+\gamma)\lambda, & \text{if } p \in P^+, \\ 0, & \text{otherwise.} \end{cases} \qquad z_r^* = \begin{cases} 1-(1+\gamma)\lambda, & \text{if } r \in R_f^+, \\ v_r, & \text{if } r \in R_t^+, \\ 0, & \text{otherwise.} \end{cases}$$

Let's first draw some conclusions about $G_{M\oplus\Pi}$.

Lemma 35. *Let $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell+1)$ be the state of the algorithm immediately after the construction of the new layer $L_{\ell+1}$. Let (P^+, R_f^+, R_t^+) be the associated players and resources with respect to an optimal solution Π of $G_M[B_{\leqslant\ell}, A_{\ell+1}\cup I]$. In $G_{M\oplus\Pi}$, every player has in-degree at most 1, every player in $B_{\leqslant\ell}\setminus src(\Pi)$ has in-degree 0, and every resource in R_f^+ has in-degree 1.*

Proof. In $G_{M\oplus\Pi}$, a player has an incoming edge if it is matched by $M\oplus\Pi$ and no incoming edge otherwise. Hence, in-degree of a player is at most 1.

Players in $B_{\leqslant\ell}$ are not matched by M. If we update M using Π, only players in $src(\Pi)$ may become matched. Hence, players in $B_{\leqslant\ell}\setminus src(\Pi)$ are not matched by $M\oplus\Pi$; their in-degrees in $G_{M\oplus\Pi}$ are 0.

By definition, for every $r \in R_f^+$, there is a path in $G_{M\oplus\Pi}$ from a player not matched by $M\oplus\Pi$ to r. If r is not matched by $M\oplus\Pi$, we can augment with this path to increase the size of $M\oplus\Pi$, contradicting the fact that $M\oplus\Pi$ is a maximum matching of G. Hence, every resource in R_f^+ is matched by $M\oplus\Pi$, and therefore, its out-degree in $G_{M\oplus\Pi}$ is equal to 1. ◻

Now we are ready to analyze the associated dual solution. First, we show that the associated dual solution is feasible.

Lemma 36. *Let $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell+1)$ be the state of the algorithm immediately after the construction of the new layer $L_{\ell+1}$. The associated dual solution $(\{y_p^*\}_{p\in P}, \{z_r^*\}_{r\in R})$ with respect to any optimal solution of $G_M[B_{\leqslant\ell}, A_{\ell+1}\cup I]$ is feasible. That is, $y_p^* \leqslant \sum_{r\in C} z_r^*$ for every $p \in P$ and every $C \in \mathcal{C}_p(1)$.*

Proof. Let Π be an optimal solution of $G_M[B_{\leqslant\ell}, A_{\ell+1}\cup I]$. Let (P^+, R_f^+, R_t^+) be the associated players and resources with respect to Π. Let $(\{y_p^*\}_{p\in P}, \{z_r^*\}_{r\in R})$ be the associated dual solution with respect to Π.

Fix some player $p \in P$. If $p \notin P^+$, then $y_p^* = 0$ and the inequality $y_p^* \leqslant \sum_{r\in C} z_r^*$ holds because z_r^* is non-negative. Suppose that $p \in P^+$. We have $y_p^* = 1-(1+\gamma)\lambda$. Take any configuration $C \in \mathcal{C}_p(1)$. We show that $\sum_{r\in C} z_r^* \geqslant 1-(1+\gamma)\lambda$.

Case 1: C contains a fat resource r_f. By the definition of P^+, there is a path π in $G_{M\oplus\Pi}$ from $B_{\leqslant\ell}\setminus src(\Pi)$ to p. Since p desires r_f, the graph $G_{M\oplus\Pi}$ contains either the edge (p, r_f) or the edge (r_f, p). We show below that $r_f \in R_f^+$. Then, $z_{r_f}^* = 1-(1+\gamma)\lambda$ by definition, which implies that $\sum_{r\in C} z_r^* \geqslant z_{r_f}^* = 1-(1+\gamma)\lambda$.

- If $G_{M\oplus\Pi}$ contains the edge (p, r_f), we can reach r_f from $B_{\leqslant\ell}\setminus src(\Pi)$ by following π and then the edge (p, r_f). So $r_f \in R_f^+$.
- If $G_{M\oplus\Pi}$ contains the edge (r_f, p), then p must be matched by $M\oplus\Pi$. We have $p \notin B_{\leqslant\ell}\setminus src(\Pi)$ because players in $B_{\leqslant\ell}\setminus src(\Pi)$ are not matched by $M\oplus\Pi$. By Lemma 35, the in-degree of p in $G_{M\oplus\Pi}$ is at most one, so (r_f, p) is the only edge entering p. In order to reach p from $B_{\leqslant\ell}\setminus src(\Pi)$, we must go through r_f. Hence, $r_f \in R_f^+$.

Case 2: C contains thin resources only. Consider the completion of the construction of $L_{\ell+1}$. At that moment, we added the last addable edge to $\mathcal{A}_{\ell+1}$ (and its blocking edges to $\mathcal{B}_{\ell+1}$) and

found that no addable edge is left. By Definition 17 and Observation 15, every player $p \in P^+$ was addable players at that moment. However, there was no addable edge for p. The reason must be that, among the thin resources desired by p, the total value of active ones was less than $(1+\gamma)\lambda$. Note that $v[C] \geqslant 1$. It follows that at least $1-(1+\gamma)\lambda$ worth of thin resources in C were inactive. By Definition 16, R_t^+ were exactly the set of inactive thin resources at that moment. Therefore, we have

$$\sum_{r \in C} z_r^* \geqslant \sum_{r \in C \cap R_t^+} z_r^* = \sum_{r \in C \cap R_t^+} v_r^* \geqslant 1-(1+\gamma)\lambda.$$

This completes the proof. ◻

Second, we establish a lower bound on the objective function value of the associated dual solution.

Lemma 37. *Let $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell+1)$ be the state of the algorithm immediately after the construction of the new layer $L_{\ell+1}$. Let $(\{y_p^*\}_{p \in P}, \{z_r^*\}_{r \in R})$ be the associated dual solution with respect to any optimal solution of $G_M[B_{\leqslant \ell}, A_{\ell+1} \cup I]$. Then,*

$$\begin{aligned}\sum_{p \in P} y_p^* - \sum_{r \in R} z_r^* \;>\; & \left(1 - \left(4 + \gamma + \frac{\beta}{\gamma}\right)\lambda\right)|B_{\leqslant \ell}| \\ & - \left(1 + \left(1 + \frac{4+2\gamma}{\beta}\right)\lambda\right)|A_{\ell+1}| \\ & - (1 + (1-\gamma)\lambda)\,|I|.\end{aligned}$$

Proof. Let Π be an optimal solution of $G_M[B_{\leqslant \ell}, A_{\ell+1} \cup I]$. Let $(\{y_p^*\}_{p \in P}, \{z_r^*\}_{r \in R})$ be the associated dual solution with respect to Π. By our setting of y_p^* and z_r^*,

$$\sum_{p \in P} y_p^* - \sum_{r \in R} z_r^* = \sum_{p \in P^+} y_p^* - \sum_{r \in R_f^+} z_r^* - \sum_{r \in R_t^+} z_r^*.$$

We bound $\sum_{p \in P^+} y_p^* - \sum_{r \in R_f^+} z_r^*$ and $\sum_{r \in R_t^+} z_r^*$ separately.

Consider $\sum_{p \in P^+} y_p^* - \sum_{r \in R_f^+} z_r^*$. Since $y_p^* = z_r^* = 1-(1+\gamma)\lambda$ for $p \in P^+$ and $r \in R_f^+$,

$$\sum_{p \in P^+} y_p^* - \sum_{r \in R_f^+} z_r^* \geqslant (1-(1+\gamma)\lambda)\left(|P^+| - |R_f^+|\right).$$

We derive a lower bound for $|P^+| - |R_f^+|$. By the definition of R_f^+, r_f is reachable from $B_{\leqslant \ell} \setminus src(\Pi)$ in $G_{M \oplus \Pi}$. For each $r_f \in R_f^+$, by Lemma 35, r_f has exactly one out-going edge to some player p in $G_{M \oplus \Pi}$. Thus, p is also reachable from $B_{\leqslant \ell} \setminus src(\Pi)$; $p \in P^+$. We charge r_f to p. By Lemma 35, each player in P^+ has in-degree at most one in $G_{M \oplus \Pi}$, so each player in P^+ is charged at most once. Players in $B_{\leqslant \ell} \setminus src(\Pi)$ obviously belong to P^+ because they are reachable from themselves. By Lemma 35, players in $B_{\leqslant \ell} \setminus src(\Pi)$ have in-degrees of 0 in $G_{M \oplus \Pi}$ and hence are not charged. Consequently,

$$|P^+| - |R_f^+| \;\geqslant\; |B_{\leqslant \ell} \setminus src(\Pi)| \;\geqslant\; |B_{\leqslant \ell}| - |\Pi|.$$

Since Π is an optimal solution for $G_M[B_{\leqslant \ell}, A_{\ell+1} \cup I]$, we have $|\Pi| \leqslant |A_{\ell+1}| + |I|$. It follows that

$$|P^+| - |R_f^+| \;\geqslant\; |B_{\leqslant \ell}| - |A_{\ell+1}| - |I|.$$

In summary,

$$\sum_{p\in P^+} y_p^* - \sum_{r\in R_f^+} z_r^* \geqslant (1-(1+\gamma)\lambda)\left(|B_{\leqslant\ell}| - |A_{\ell+1}| - |I|\right). \tag{8}$$

Consider $\sum_{r\in R_t^+} z_r^*$. Recall that $R_t^+ = R(\mathcal{A}_{\leqslant\ell+1}\cup\mathcal{B}_{\leqslant\ell}\cup I)\cup R(\mathcal{B}'_{\ell+1})$ where $\mathcal{B}'_{\ell+1} = \{e\in\mathcal{B}_{\ell+1} : v[R(e)\cap R(\mathcal{A}_{\ell+1})] > \beta\lambda\}$. We handle $R(\mathcal{A}_{\leqslant\ell}\cup\mathcal{B}_{\leqslant\ell})$, $R(\mathcal{A}_{\ell+1}\cup I)$, and $R(\mathcal{B}'_{\ell+1})$ separately.

Every edge in $\mathcal{A}_{\leqslant\ell}$ is blocked, so by Definition 18, less than λ worth of its thin resources are not included in $\mathcal{B}_{\leqslant\ell}$. Every edge in $\mathcal{B}_{\leqslant\ell}$ is λ-minimal, so it includes less than 2λ worth of thin resources. Thus,

$$\begin{aligned}\sum_{r\in R(\mathcal{A}_{\leqslant\ell}\cup\mathcal{B}_{\leqslant\ell})} v_r &< \lambda|\mathcal{A}_{\leqslant\ell}| + 2\lambda|\mathcal{B}_{\leqslant\ell}| \\ &\leqslant \left(3+\frac{\beta}{\gamma}\right)\lambda|B_{\leqslant\ell}| \qquad (\because \text{Lemma 32}).\end{aligned} \tag{9}$$

Edges in $\mathcal{A}_{\ell+1}$ are $(1+\gamma)\lambda$-minimal, so each of them includes less than $(2+\gamma)\lambda$ worth of resources. Edges in $\mathcal{I}$ are λ-minimal, so each of them include less than 2λ worth of thin resources. Therefore,

$$\sum_{r\in R(\mathcal{A}_{\ell+1}\cup\mathcal{I})} v_r < (2+\gamma)\lambda|A_{\ell+1}| + 2\lambda|I|. \tag{10}$$

Edges in $\mathcal{B}'_{\ell+1}$ are λ-minimal, so

$$\begin{aligned}\sum_{r\in R(\mathcal{B}'_{\ell+1})} v_r &< 2\lambda|\mathcal{B}'_{\ell+1}| \\ &< \frac{4+2\gamma}{\beta}\lambda|A_{\ell+1}| \qquad (\because \text{Lemma 33}).\end{aligned} \tag{11}$$

Combining (9), (10), and (11) gives

$$\begin{aligned}\sum_{r\in R_t^+} z_r^* &= \sum_{r\in R_t^+} v_r \\ &< \left(3+\frac{\beta}{\gamma}\right)\lambda|B_{\leqslant\ell}| + 2\lambda|I| + \left(2+\gamma+\frac{4+2\gamma}{\beta}\right)\lambda|A_{\ell+1}|.\end{aligned} \tag{12}$$

Combining (8) and (12) shows that

$$\begin{aligned}\sum_{p\in P^+} y_p^* - \sum_{r\in R_f^+} z_r^* - \sum_{r\in R_t^+} z_r^* > &\left(1-\left(4+\gamma+\frac{\beta}{\gamma}\right)\lambda\right)|B_{\leqslant\ell}| \\ &-\left(1+\left(1+\frac{4+2\gamma}{\beta}\right)\lambda\right)|A_{\ell+1}| \\ &-(1+(1-\gamma)\lambda)\,|I|.\end{aligned}$$

□

We are ready to prove the key result in this section, Lemma 38, that a newly constructed layer has a lot of addable edges, or some layer is collapsible. We prove it by contradiction. If the lemma is not true, we use the analysis of the associated dual solution in this section to show that the dual of $CLP(1)$ is unbounded, which implies the contradiction that the value 1 is not feasible for configuration LP.

Lemma 38. *Let $\lambda = \frac{1}{4+\delta}$ for any $\delta \in (0,1)$. There exists a value $c_0 = \Theta(\delta)$ such that if $\gamma \leqslant c_0$, $\beta = \gamma^2$, and $\mu = \gamma^3$, then immediately after the construction of a new layer $L_{\ell+1}$, some layer in the stack is collapsible, or $z_{\ell+1} = |A_{\ell+1}| > 2\mu|B_{\leqslant l}|$.*

Proof. Suppose, for the sake of contradiction, that $|A_{\ell+1}| \leqslant 2\mu|B_{\leqslant\ell}|$ and that no layer is collapsible. We show that the dual of $CLP(1)$ is unbounded. Let $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell+1)$ be the state of the algorithm immediately after the construction of layer $L_{\ell+1}$. Let $(\{y_p^*\}_{p\in P}, \{z_r^*\}_{r\in R})$ be the associated dual solution with respect to some optimal solution of $G_M[B_{\leqslant\ell}, A_{\ell+1} \cup I]$.

By Lemma 36, $(\{y_p^*\}_{p\in P}, \{z_r^*\}_{r\in R})$ is feasible for the dual of $CLP(1)$. We claim that the objective function value of the associated dual solution is positive given the setting of γ, β, and μ in the lemma. Assuming that the claim is true, we can multiply the associated dual solution with an arbitrarily large constant to obtain an arbitrarily large objective function value while preserving feasibility. But then the dual of $CLP(1)$ is unbounded and hence $CLP(1)$ is infeasible, a contradiction.

Now all we need to do is to show that there exists a value $c_0 = \Theta(\delta)$ such that we can make $\sum_{p\in P} y_p^* - \sum_{r\in R} z_r^*$ positive by setting $\gamma = c$ for any $c \leqslant c_0$, $\beta = c^2$, and $\mu = c^3$. By Lemma 37,

$$\begin{aligned}\sum_{p\in P} y_p^* - \sum_{r\in R} z_r^* \;>\; & \left(1 - \left(4 + \gamma + \frac{\beta}{\gamma}\right)\lambda\right)|B_{\leqslant\ell}| \\ & - \left(1 + \left(1 + \frac{4+2\gamma}{\beta}\right)\lambda\right)|A_{\ell+1}| \\ & - (1 + (1-\gamma)\lambda)\,|I|.\end{aligned}$$

Since no layer is collapsible, by Lemma 30, $|I| \leqslant \mu|B_{\leqslant\ell}|$. Also, by our assumption, $|A_{\ell+1}| \leqslant 2\mu|B_{\leqslant\ell}|$. Therefore,

$$\begin{aligned}\sum_{p\in P} y_p^* - \sum_{r\in R} z_r^* \;>\; & \left(1 - \left(4 + \gamma + \frac{\beta}{\gamma}\right)\lambda\right)|B_{\leqslant\ell}| \\ & - \left(1 + \left(1 + \frac{4+2\gamma}{\beta}\right)\lambda\right)\cdot 2\mu|B_{\leqslant\ell}| \\ & - (1 + (1-\gamma)\lambda)\,\mu|B_{\leqslant\ell}| \\ = \;& \left((1-3\mu) - \left(4 + \gamma + \frac{\beta}{\gamma} + 3\mu + \frac{(8+4\gamma)\mu}{\beta} - \gamma\mu\right)\lambda\right)|B_{\leqslant\ell}| \\ = \;& \left(\left(1 - 3c^3\right) - \left(4 + 10c + 4c^2 + 3c^3 - c^4\right)\lambda\right)|B_{\leqslant\ell}|.\end{aligned}$$

As $c \to 0$,

$$\frac{1-3c^3}{4+10c+4c^2+3c^3-c^4} \to \frac{1}{4}.$$

There is a sufficiently small c that makes $\frac{1-3c^3}{4+10c+4c^2+3c^3-c^4} > \frac{1}{4+\delta} = \lambda$, and makes

$$\sum_{p\in P} y_p^* - \sum_{r\in R} z_r^* > 0.$$

One can verify that the largest value for c is $\Theta(\delta)$ for the above conclusion to hold. ◻

When a new layer $L_{\ell+1}$ is constructed, if $|z_{\ell+1}| \leqslant 2\mu|B_{\leqslant\ell}|$, then by Lemma 38, some layer in the stack must be collapsible. By Remark 29, the top layer $L_{\ell+1}$ cannot be collapsible, so the collapsible layer must be below $L_{\ell+1}$. As a consequence, $L_{\ell+1}$ will be removed by Collapse immediately after its construction.

Corollary 39. *Let λ, γ, β, μ be set as in Lemma 38. At the completion of the construction of a new layer $L_{\ell+1}$, if $|z_{\ell+1}| \leqslant 2\mu|B_{\leqslant\ell}|$, then $L_{\ell+1}$ will be removed by* Collapse *immediately.*

4.7 Exponential growth of layer size and polynomial running time

In this section, we show that our algorithm terminates in polynomial time. The completion of our algorithm means that it returns an allocation with a max-min value at least λ, thereby proving an approximation ratio of $1/\lambda = 4 + \delta$. The next result shows that the size of $\mathcal{B}_i$ grows exponentially with respect to i, which is essential to establishing a logarithmic bound on the depth of the stack Σ.

Lemma 40. *Let $\lambda = \frac{1}{4+\delta}$ for any $\delta \in (0, 1)$. There exists a value $c_0 = \Theta(\delta)$ such that if $\gamma \leqslant c_0$, $\beta = \gamma^2$, and $\mu = \gamma^3$, then for every state $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell)$ of the algorithm, some layer in Σ is collapsible or $|B_{i+1}| > \frac{\gamma^3}{1+\gamma}|B_{\leqslant i}|$ for every $i \in [0, \ell - 1]$.*

Proof. Take a layer L_{i+1} in Σ. Since the construction of L_{i+1}, no layer below L_{i+1} has ever been collapsed; otherwise, L_{i+1} would have been removed by Collapse. As a result, the set $\mathcal{B}_{\leqslant i}$ has not changed since the construction of L_{i+1}. Then, $z_{i+1} > 2\mu|B_{\leqslant i}|$ by Corollary 39.

Suppose that no layer is collapsible. We show that $|B_{i+1}| > \frac{\gamma^3}{1+\gamma}|B_{\leqslant i}|$. By Lemma 30, we have $|A_{i+1}| \geqslant z_{i+1} - \mu|B_{\leqslant i}|$. Therefore,

$$|A_{i+1}| \geqslant z_{i+1} - \mu|B_{\leqslant i}| > 2\mu|B_{\leqslant i}| - \mu|B_{\leqslant i}| = \mu|B_{\leqslant i}|.$$

By Lemma 32,

$$|B_{i+1}| > \frac{\gamma}{\gamma + \beta}|A_{i+1}| > \frac{\gamma\mu}{\gamma + \beta}|B_{\leqslant i}|.$$

Substituting $\beta = \gamma^2$ and $\mu = \gamma^3$ into the above inequality, we obtain

$$|B_{i+1}| > \frac{\gamma^3}{1 + \gamma}|B_{\leqslant i}|.$$

□

A direct consequence of Lemma 40 is that there are $O(\log n)$ layers in the stack.

Corollary 41. *Suppose that λ, γ, β, and μ are set as in Lemma 40. For every state $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell)$ of the algorithm, $\ell \leqslant \lceil \log_{1+h} n \rceil$, where $h = \frac{\gamma^3}{1+\gamma}$.*

Proof. Recall that n is the number of players. We show that whenever $\ell = \lceil \log_{1+h} n \rceil$, some layer in the stack must be collapsible. Then Collapse will be invoked, so the depth of the stack cannot go above $\lceil \log_{1+h} n \rceil$. Suppose for the sake of contradiction that $\ell = \lceil \log_{1+h} n \rceil$ but no layer is collapsible. By Lemma 40, $B_{i+1} > h|B_{\leqslant i}|$ for every $i \in [0, \ell - 1]$. Equivalently, $B_{\leqslant i+1} > (1 + h)|B_{\leqslant i}|$ for every $i \in [0, \ell - 1]$. As $|B_0| = 1$, we have $|B_{\leqslant \ell}| > (1 + h)^\ell \geqslant n$. But this is impossible because $\mathcal{B}_1, \ldots, \mathcal{B}_\ell$ are disjoint subsets of $\mathcal{M}$ and $\mathcal{B}_0 = \{(p_0, \emptyset)\}$, where p_0 is not matched by $\mathcal{M}$. □

We make use of Corollary 41 to show that our algorithm calls Build and Collapse a polynomial number of times.

Lemma 42. *There exists a value $c_0 = \Theta(\delta)$ such that if $\gamma = c_0$, $\beta = \gamma^2$, and $\mu = \gamma^3$, then the unmatched player p_0 will be matched after at most $n^{poly(1/\delta)}$ calls of* Build *and* Collapse.

Proof. The proof is essentially same as the one used in [1, 9]. We sketch it below.

Let $h = \frac{\gamma^3}{1+\gamma}$. Let $\mathcal{S}$ be the set of all possible states of the algorithm in which no layer is collapsible. For each state $(M, \mathcal{M}, \mathcal{I}, \Sigma, \ell)$ in $\mathcal{S}$, we define its signature to be $(s_0, \ldots, s_\ell, \infty)$, where $s_i = \log_{1/(1-\mu)} \frac{|B_i|}{h^{i+1}}$. One can verify that the coordinates of the signature vector are

non-decreasing, and that as the algorithm goes from one state in $\mathcal{S}$ to another state in $\mathcal{S}$, the signature vector decreases lexicographically. It follows that the same state in $\mathcal{S}$ cannot be reached twice by the algorithm. To prove the lemma, it suffices to bound the number of distinct states in $\mathcal{S}$ because each call of BUILD or COLLAPSE moves the algorithm from one state to another.

By Corollary 41, $\ell \leqslant \lceil \log_{1+h} n \rceil$. By some calculations one can veirfy that the sum of the coordinates in any signature vector is bounded by U^2 where $U = \log n \cdot O(\frac{1}{\mu h} \log \frac{1}{h})$. Each signature vector can be regarded as a partition of an integer less than or equal to U^2. Summing up the number of distinct partitions of an integer i over all $i \in [1, U^2]$, we get the upper bound of $n^{O(\frac{1}{\mu h} \log \frac{1}{h})}$ on the number of distinct signature vectors. Recall that $h = \frac{\gamma^3}{1+\gamma}$, $\mu = \gamma^3$, and $\gamma = \Theta(\delta)$. As a consequence, $|\mathcal{S}| \leqslant n^{poly(1/\delta)}$. ◻

Next, we argue that both BUILD and COLLAPSE run in polynomial time.

Lemma 43. BUILD *runs in* $poly(m, n)$ *time.*

Proof. It suffices to show that steps 2 and 3 of BUILD run in polynomial time. By Observation 15, we can tell whether a player is addable in polynomial time. Consider an addable player p. Let R_p is the subset of thin resources that are desired by p.

Step 2 determines whether there is a λ-minimal unblocked addable edge incident to p. Let R'_p be the subset of resources in R_p that are active and not used by $\mathcal{M}$. R'_p can be identified in polynomial time. If $v[R'_p] \geqslant \lambda$, we can form a λ-minimal unblocked addable edge (p, S) by including thin resources in R'_p in decreasing order of values. If $v[R'_p] < \lambda$, then p is not incident to any λ-minimal unblocked addable edge.

Step 3 determines whether there is a $(1+\gamma)\lambda$-minimal blocked addable edge incident to p. Let R''_p be the subset of resources in R_p that are active. If $v[R''_p] \geqslant (1+\gamma)\lambda$, we can form a $(1+\gamma)\lambda$-minimal blocked addable edge (p, S) by including thin resources in R''_p in decreasing order of values. If $v[R''_p] < (1+\gamma)\lambda$, then p is not incident to any $(1+\gamma)\lambda$-minimal blocked addable edge. ◻

Lemma 44. COLLAPSE *runs in* $poly(m, n)$ *time.*

Proof. By Lemma 23, step 2 of COLLAPSE takes $poly(m, n, \ell)$ time. By Observation 15, step 5.2 takes $poly(m, n)$ time. The rest of steps takes $poly(m, n, \ell)$. By Corollary 41, $\ell = O(\log n)$. Therefore, COLLAPSE runs in $poly(m, n)$ time. ◻

We now have all the pieces in place. One can solve the configuration LP in polynomial time as described in [4] to obtain its optimal value (with arbitrarily small additive error). Using this optimal value, we can run our approximation algorithm.

Theorem 2. *There is an algorithm for the restricted max-min allocation problem for n players and m indivisible resources that guarantees an approximation ratio of $4+\delta$ for any $\delta \in (0, 1)$. Its running time is $poly(m, n) \cdot n^{poly(1/\delta)}$.*

Proof. By Lemma 42, 43, and 44, for $\lambda = \frac{1}{4+\delta}$ with any $\delta \in (0, 1)$, the algorithm can match an unmatched player by modifying M and $\mathcal{M}$ in $poly(m, n) \cdot n^{poly(1/\delta)}$ time. Repeating it for at most n time gives an allocation with max-min value $\lambda = \frac{1}{4+\delta}$. Since the optimal value of the configuration LP is 1, the approximation ratio of the algorithm is $4 + \delta$. ◻

If we set $\delta = \gamma = \beta = \mu = 0$, Lemma 40 still holds, but it no longer guarantees a geometric growth in the number of blocking edges. All we have is $|B_{i+1}| > 0$. As a consequence, we cannot guarantee a polynomial running time for the algorithm. However, the algorithm still works. The guarantee of $|B_{i+1}| > 0$ means that the algorithm can always build a non-empty layer, and therefore, in finite number of steps, it can match the unmatched player p_0 (for $\lambda = \frac{1}{4}$). In fact, it is the same as the local search in Section 3.2 except that the addable edges and blocking edges are now organized in layers.

If we set $\beta = 2$, then it is like the algorithm in [1]: a blocking edge can block as many addable edges as possible. Consider the term $1 - (4 + \gamma + \beta/\gamma)\lambda$ in the proof of Lemma 38. In order to make Lemma 38 valid, the best λ is roughly $\frac{1}{4+2\sqrt{2}}$ and is achieved when $\gamma = \sqrt{2}$. The approximation ratio is bounded away from 4.

As in [1], we can avoid solving the configuration LP by using binary search, and hence make the algorithm fully combinatorial. Let $\widehat{T}$ be a guess of the optimal value T^* of the configuration LP. If $\widehat{T} \leqslant T^*$, our algorithm is guaranteed to return an allocation with max-min value at least $\widehat{T}/(4+\delta)$, and we increase our guess $\widehat{T}$. When $\widehat{T} > T^*$, there are two cases. We may be lucky enough that Lemma 38 still holds, so the algorithm returns in polynomial time an allocation with objective value $\widehat{T}/(4+\delta)$. We increase our guess $\widehat{T}$ in this case. If we are not that lucky, after some call of BUILD, we will encounter a violation of Lemma 38. That is, we detect that $z_{\ell+1} = |A_{\ell+1}| \leqslant 2\mu|B_{\leqslant\ell}|$ but no layer is collapsible. We abort the algorithm and decrease our guess $\widehat{T}$. Since Lemma 38 holds before the first violation, the algorithm runs for at most $poly(m,n) \cdot n^{poly(1/\delta)}$ time before it is aborted. One can see that $\frac{\sum_{r\in R} v_r}{n}$ is a trivial upper bound on T^*. Therefore, we can get an allocation with max-min value at least $T^*/(4+\delta)$ after $\log(\frac{\sum_{r\in R} v_r}{n})$ round of binary search, and each round takes at most $poly(m,n) \cdot n^{poly(1/\delta)}$ time.

5 Conclusion and Discussion

We show that the integrality gap of the configuration LP for the restricted max-min fair allocation problem is at most $\frac{99}{26} \approx 3.808$. This is not likely to be the tight bound for the integrality gap, not even for the local search of Asadpour et al. [2]. Given the current lower bound of 2, a natural question is whether one can narrow the gap between the upper and lower bound further. Another interesting challenge, which is proposed by the authors of [16], is to prove or disprove that the running time of the local search of Asadpour et al. is polynomial.

We propose a polynomial time algorithm that guarantees an approximation ratio of $4 + \delta$ for any $\delta \in (0, 1)$. Another polynomial-time approximation method used in the literature is based on rounding the optimal solution of the configuration LP, which gives an approximation ratio of $O\big(\frac{\log\log n}{\log\log\log n}\big)$ [4]. Is there a rounding scheme that gives an $O(1)$ approximation ratio?

References

[1] C. ANNAMALAI, C. KALAITZIS, AND O. SVENSSON, *Combinatorial algorithm for restricted max-min fair allocation*, ACM Transactions on Algorithms, 13 (2017), pp. 37:1–37:28.

[2] A. ASADPOUR, U. FEIGE, AND A. SABERI, *Santa Claus meets hypergraph matchings*, ACM Transactions on Algorithms, 8 (2012), pp. 24:1–24:9.

[3] A. ASADPOUR AND A. SABERI, *An approximation algorithm for max-min fair allocation of indivisible goods*, in Proceedings of the 39th ACM Symposium on Theory of Computing, 2007, pp. 114–121.

[4] N. Bansal and M. Sviridenko, *The Santa Claus problem*, in Proceedings of the 38th ACM Symposium on Theory of Computing, 2006, pp. 31–40.

[5] M. Bateni, M. Charikar, and V. Guruswami, *Max-min allocation via degree lower-bounded arborescences*, in Proceedings of the 41st Annual ACM Symposium on Theory of Computing, 2009, pp. 543–552.

[6] I. Bezáková and V. Dani, *Allocating indivisible goods*, SIGecom Exchanges, 5 (2005), pp. 11–18.

[7] D. Chakrabarty, J. Chuzhoy, and S. Khanna, *On allocating goods to maximize fairness*, in Proceedings of the 50th IEEE Symposium on Foundations of Computer Science, 2009, pp. 107–116.

[8] S. Cheng and Y. Mao, *Integrality gap of the configuration LP for the restricted max-min fair allocation*, CoRR, abs/1807.04152 (2018).

[9] ——, *Restricted max-min fair allocation*, in Proceedings of the 45th International Colloquium on Automata, Languages, and Programming, 2018, pp. 37:1–37:13.

[10] S. Cheng and Y. Mao, *Restricted max-min allocation: Approximation and integrality gap*, in Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, 2019, pp. 38:1–38:13.

[11] S. Davies, T. Rothvoss, and Y. Zhang, *A tale of santa claus, hypergraphs and matroids*, CoRR, abs/1807.07189v1 (2018).

[12] S. Davies, T. Tothvoss, and Y. Zhang, *A tale of Santa Claus, hypergraphs and matroids*, in Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms, 2020, pp. 2748–2757.

[13] U. Feige, *On allocations that maximize fairness*, in Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms, 2008, pp. 287–293.

[14] B. Haeupler, B. Saha, and A. Srinivasan, *New constructive aspects of the Lovász local lemma*, Journal of the ACM, 58 (2011), pp. 28:1–28:28.

[15] P. Haxell, *A condition for matchability in hypergraphs*, Graphs and Combinatorics, 11 (1995), pp. 245–248.

[16] K. Jansen and L. Rohwedder, *On the configuration-LP of the restricted assignment problem*, in Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms, 2017, pp. 2670–2678.

[17] ——, *A note on the integrality gap of the configuration LP for restricted santa claus*, CoRR, abs/1807.03626 (2018).

[18] J. Kleinberg and E. Tardos, *Algorithms Design*, Pearson/Addison-Wesley, 2006.

[19] J. Lenstra, D. Shmoys, and É. Tardos, *Approximation algorithms for scheduling unrelated parallel machines*, in Proceedings of the 28th IEEE Symposium on Foundations of Computer Science, 1987, pp. 217–224.

[20] L. Polacek and O. Svensson, *Quasi-polynomial local search for restricted max-min fair allocation*, in 39th International Colloquium on Automata, Languages, and Programming, 2012, pp. 726–737.

[21] B. Saha and A. Srinivasan, *A new approximation technique for resource-allocation problems*, in Proceedings of the 1st Symposium on Innovations in Computer Science, 2010, pp. 342–357.