View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Dagstuhl Research Online Publication Server

Restricted Max-Min Allocation: Approximation and Integrality Gap

Siu-Wing Cheng
Department of Computer Science and Engineering, HKUST, Hong Kong
scheng@cse.ust.hk

Yuchen Mao
Department of Computer Science and Engineering, HKUST, Hong Kong
ymaoad@cse.ust.hk

Abstract

Asadpour, Feige, and Saberi proved that the integrality gap of the configuration LP for the restricted max-min allocation problem is at most 4. However, their proof does not give a polynomial-time approximation algorithm. A lot of efforts have been devoted to designing an efficient algorithm whose approximation ratio can match this upper bound for the integrality gap. In ICALP 2018, we present a $(6+\delta)$-approximation algorithm where δ can be any positive constant, and there is still a gap of roughly 2. In this paper, we narrow the gap significantly by proposing a $(4+\delta)$-approximation algorithm where δ can be any positive constant. The approximation ratio is with respect to the optimal value of the configuration LP, and the running time is $poly(m,n) \cdot n^{poly(\frac{1}{\delta})}$ where n is the number of players and m is the number of resources. We also improve the upper bound for the integrality gap of the configuration LP to $3+\frac{21}{26} \approx 3.808$.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases fair allocation, configuration LP, approximation, integrality gap

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.38

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1905.06084`.

1 Introduction

Background

In the max-min fair allocation problem, we are given a set P of n players, a set R of m indivisible resources, and a set of non-negative values $\{v_{pr}\}_{p\in P, r\in R}$. For each $r \in R$ and each $p \in P$, resource r is worth a value of v_{pr} to player p. An allocation is a partition of R into disjoint subsets $\{D_p\}_{p\in P}$ so that each player p is assigned the resources in D_p. The goal is to find an allocation that maximizes the welfare of the least lucky player, that is, we want to maximize $\min_{p\in P} \sum_{r\in D_p} v_{pr}$. Unfortunately, unless P = NP, no polynomial-time algorithm can achieve an approximation ratio smaller than 2 [6].

Bezáková and Dani [6] tried to solve the problem using the assignment LP – a technique for the classic scheduling problem of makespan minimization [16]. However, they showed that the integrality gap of the assignment LP is unbounded, so rounding the assignment LP gives no guarantee on the approximation ratio. Later, Bansal and Sviridenko [4] proposed a stronger LP relaxation, the configuration LP, for the max-min allocation problem. Asadpour and Saberi [3] developed a polynomial-time rounding scheme for the configuration LP that gives an approximation ratio of $O(\sqrt{n}\log^3 n)$. Saha and Srinivasan [18] improved it to $O(\sqrt{n\log n})$. These approximation ratios almost match the lower bound of $\Omega(\sqrt{n})$ for the integrality gap of the configuration LP proved by Bansal and Svirodenko [4]. Bateni et al. [5]

© Siu-Wing Cheng and Yuchen Mao;
licensed under Creative Commons License CC-BY
46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 38; pp. 38:1–38:13
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

and Chakrabarty et al. [7] established a trade-off between the approximation ratio and the running time. For any $\delta > 0$, they can achieve an approximation ratio of $O(n^{\delta})$ with $O(n^{1/\delta})$ running time.

In this paper, we study the restricted max-min allocation problem. In the restricted case, we have $v_{pr} \in \{v_r, 0\}$. That is, each resource r has an intrinsic value v_r, and it is worth value v_r to those players who desire it and value 0 to those who do not. Assuming P $\neq$ NP, the restricted case has a lower bound of 2 for the approximation ratio. The integrality gap of configuration LP for the restricted case also has a lower bound of 2. 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 [11] proved that the integrality gap of the configuration LP is bounded by a constant, albeit large and unspecified. His proof was later made constructive by Haeupler et al. [12], and hence a constant approximation can be found in polynomial time. Asadpour et al. [2] viewed the restricted max-min allocation problem as a bipartite hyper-graph matching problem. Let T^* be the optimal value of the configuration LP. By adapting Haxell's [13] alternating tree technique for bipartite hyper-graph matchings, they proposed a local search algorithm that returns an allocation where every player receives at least $T^*/4$ worth of resources, and hence proved that the integrality gap of the configuration LP is at most 4. However, their algorithm is not known to run in polynomial time. A lot of efforts have been devoted to making their algorithm run in polynomial time. Polacek and Svensson [17] showed that the local search can be done in quasi-polynomial time by building the alternating tree in a more careful way. Annamalai, Kalaitzis and Svensson [1] carried out the local search in a more structured way. Together with two new *greedy* and *lazy update* strategies, they can find in polynomial time an allocation in which every player receives a value of at least $T^*/(6 + 2\sqrt{10} + \delta)$. Recently, we proposed a more flexible, aggressive greedy strategy that improves the approximation ratio to $6 + \delta$ [9]. Davies et al. [10] claimed a $(6 + \delta)$-approximation algorithm for the restricted max-min allocation problem by reducing it to the fractional matroid max-min allocation problem.

Our Contribution

We adapt the framework in [1] by introducing two new strategies: *layer-level node-disjoint paths* and *limited blocking*. The performance of our framework is determined by three parameters, and a trade-off between the running time and the quality of solution can be achieved by tuning these parameters. On one extreme, our framework acts exactly the same as the original local search in [2], which achieves a ratio of 4 but not necessarily run in polynomial time. On the other extreme, it becomes something like the algorithm in [1], which achieves a polynomial running time but a much worse ratio. We show that, in order to achieve a polynomial running time, one doesn't have to go from one extreme to the other – a marginal movement is sufficient. As a result, a ratio slightly worse than 4 can be achieved in polynomial time.

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

Although the algorithm we present takes the optimal value of the configuration LP as its input, one can avoid solving the configuration LP by combining our algorithm with binary search to zoom into the optimal value of configuration LP. The binary search technique is similar to that in [1, 9].

We also show that the integrality gap of the configuration LP is at most $3 + \frac{21}{26} \approx 3.808$ by giving a better analysis of the AFS algorithm. This improves the bound of $3 + \frac{5}{6} \approx 3.833$ recently obtained in [8, 15].

Primal

$$\sum_{C \in \mathcal{C}_p(T)} x_{p,C} \geqslant 1 \forall p \in P$$

$$\sum_{p \in P} \sum_{C \in \mathcal{C}_p(T): r \in C} x_{p,C} \leqslant 1 \forall r \in R$$

$$x_{p,C} \geqslant 0$$

Dual

$$\max \quad \sum_{p \in P} y_p - \sum_{r \in R} z_r$$

$$s.t. \quad y_p \leqslant \sum_{r \in C} z_r \qquad \forall p \in P, \forall C \in \mathcal{C}_p(T)$$

$$y_p \geqslant 0 \qquad \forall p \in P$$

$$z_r \geqslant 0 \qquad \forall r \in R$$

Figure 1 The configuration LP and its dual.

▶ **Theorem 2.** *The integrality gap of the configuration LP for the restricted max-min allocation problem is at most* $3 + \frac{21}{26} \approx 3.808$.

We focus on only the proof of Theorem 1 in the main text. The proof of Theorem 2 can be found in the full version of the paper. Other omitted proofs can also be found in the full version of the paper.

2 Preliminaries

2.1 The Configuration LP

Suppose that we hope to find an allocation where every player receives at least T worth of resources. A *configuration* for a player p is a subset D of the resources desired by p such that $\sum_{r \in D} v_r \geqslant T$. Let $\mathcal{C}_p(T)$ denote the set of all configurations for p.

The configuration LP is given on the left of Figure 1. Given a target T, the configuration LP, denoted as $CLP(T)$, associates a variable $x_{p,C}$ with each player p and each configuration C in $\mathcal{C}_p(T)$. Its first constraint ensures that each player receives at least 1 unit of configurations, and the second constraint guarantees that every resource r is used in at most 1 unit of configurations. The optimal value of the configuration LP is the largest T for which $CLP(T)$ is feasible. We denote this optimal value by T^*. Without loss of generality, we assume that $T^* = 1$ for the rest of the paper. Although the configuration LP may have an exponential number of variables, it can be solved within any constant relative error in polynomial time [4]. Viewing the objective function of the configuration LP as a minimization of a constant, one can get the dual LP on the right of the Figure 1.

2.2 Fat and thin edges

Our goal is to find an allocation in which every player receives at least λ worth of resources for some $\lambda \in (0, 1)$. In particular, our approximation algorithm sets $\lambda = \frac{1}{4+\delta}$ where δ is a positive constant. For each resource $r \in R$, we call r *fat* if $v_r \geqslant \lambda$, and *thin* otherwise. To find the target allocation, it suffices to assign each player p either a fat resource desired by p or a subset D of the thin resources desired by p with $\sum_{r \in D} v_r \geqslant \lambda$.

For every $p \in P$ and every fat resource r desired by p, we call $\{p, r\}$ a *fat edge*. For every $p \in P$ and every subset D of the thin resources desired by p, we call (p, D) a *thin edge* if $\sum_{r \in D} v_r \geqslant \lambda$. Two edges are *compatible* if they share no common resource. We say that a fat edge $\{p, r\}$ *covers* p and r. Similarly, a thin edge (p, D) *covers* p and the resources in D. A player or a resource is covered by a set of edges if it is covered by some edge in the set. For any $w \geqslant 0$, a thin edge (p, D) is a *w-minimal* if $\sum_{r \in D} v_r \geqslant w$ and $\sum_{r \in D'} v_r < w$ for any $D' \subsetneq D$. For a w-minimal thin edge (p, D), it is not hard to see that $w \leqslant \sum_{r \in D} v_r < w + \lambda$.

Given the above definitions of fat and thin edges, finding the target allocation is equivalent to finding a set of mutually compatible edges that covers all the players.

2.3 A local search idea

The following local search idea is initially proposed by Asadpour et al. [2], and is also used in [1, 9].

Let G be the bipartite graph formed by the players, the fat resources, and the fat edges. We maintain a set M of fat edges and a set $\mathcal{E}$ of thin edges such that: (i) M is a maximum matching of G, (ii) edges in $\mathcal{E}$ are λ-*minimal* and are mutually compatible, and (iii) each player is covered by at most one edge in $M \cup \mathcal{E}$. We call such M and $\mathcal{E}$ a partial allocation. Initially, M is an arbitrary maximum matching of G, and $\mathcal{E}$ is empty. The set $M \cup \mathcal{E}$ is updated and grown iteratively so that one more player is covered in each iteration. The final set $M \cup \mathcal{E}$ covers all the players and induces our target allocation.

Let p_0 be a player not yet covered by $M \cup \mathcal{E}$. We need to update $M \cup \mathcal{E}$ to cover p_0 without losing any player that are already covered. The simplest case is that we can find a player q_0 such that q_0 is covered by a thin edge a compatible with $\mathcal{E}$ and there is an alternating path [14] with respect to M from p_0 to q_0. Let π be this alternating path. We first update M by taking the symmetric difference $M \oplus \pi$, i.e., remove the edges in $\pi \cap M$ from the matching and add the edges in $\pi \setminus M$ to the matching. $M \oplus \pi$ is also a maximum matching of G. After the update, p_0 becomes matched while q_0 becomes unmatched. Then we add a to $\mathcal{E}$ to cover q_0 again. Here we slight abuse the notion of alternating paths in the sense that wen allow an alternating path with no edge. The $\oplus$ can easily extend to alternating paths with no edge.

It is possible that no edge covering q_0 is compatible with $\mathcal{E}$. Let a be an edge covering q_0. Suppose that b is an edge in $\mathcal{E}$ that is not compatible with a. We say b *blocks* a. Let p_1 be the player covered by b. In order to add a to $\mathcal{E}$, we have to release b from $\mathcal{E}$. But we cannot lose p_1, so before we release b, we need to find another edge to cover p_1. Now p_1 has a similar role as p_0.

2.4 Node-disjoint alternating paths

In order to achieve a polynomial running time, our algorithm updates M using multiple node-disjoint alternating paths from unmatched players to players . In this section, we define a problem of finding a largest set of node-disjoint paths. We also extend the $\oplus$ operation to a set of node-disjoint paths.

For any maximum matching M of G, we define G_M to be the directed graph obtained from G by orienting edges of G from r to p if $\{p, r\} \in M$, and from p to r if $\{p, r\} \notin M$. Let S be a subset of the players not matched by M. Let T be a subset of the players. Finding the largest set of node-disjoint alternating paths from S to T is equivalent to finding the largest set of node-disjoint paths in G_M from S to T. Let $G_M(S, T)$ denote the problem of finding the largest set of node-disjoint paths from S to T in G_M. Let $f_M(S, T)$ denotes the maximum number of such paths. Note that when $S \cap T \neq \emptyset$, a path consisting of a single node is allowed. Such path is called a trivial path. Paths with at least one edge is non-trivial. Let Π be a feasible solution for $G_M(S, T)$. The paths in Π originate from a subset of S, which we call the *sources* and denote as src_Π, and terminate in a subset of T, which we call the *sinks* and denote as $sink_\Pi$. We extend the $\oplus$ operation to Π. Viewing Π as a set of edges, $M \oplus \Pi$ stands for removing the edges in $\Pi \cap M$ from the matching and adding the edges in $\Pi \setminus M$ to the matching. One can see that $M \oplus \Pi$ is a maximum matching of G.

The problem $G_M(S, T)$ can be solved in polynomial time. Please refer to the full version of the paper for more details.

3 An Approximation Algorithm

We discuss below a few techniques used by our algorithm. Some of them are used in [1, 9, 10]. The *limited blocking* strategy is brand new, and is crucial to achieving an approximation ratio of $4 + \delta$. In the following discussion, one can interpret addable edges as thin edges that we hope to add to $\mathcal{E}$, and blocking edges as edges in $\mathcal{E}$ that are not compatible with addable edges. The precise definition will be given later.

Layers. As in [1, 9], we maintain a stack of layers, where each layer consists of addable edges and their blocking edges. The key to achieving a polynomial running time is to guarantee a geometric growth in the number of blocking edges from the bottom to the top of the stack.

Layer-level node-disjoint paths. We require that the players covered by the addable edges in a layer can be simultaneously reached via node-disjoint paths in G_M from the players covered by the blocking edges in the previous layers [10]. It has the same effect as the globally node-disjoint path used in [1]: if lots of addable edges in a layer become unblocked, then a significant update can be made. The advantage of our strategy is that it offers more flexibility when building a new layer.

Lazy update. When having an unblocked addable edge, one may be tempted to update M and $\mathcal{E}$ immediately. However, as in [1], in order to achieve a polynomial running time, we should wait until there are lots of unblocked addable edges, and then a significant update can be made in one step. The laziness is controlled by a small constant μ that will be defined later.

Greedy and Limited Blocking. Recall that the key to achieving a polynomial running time is to guarantee a geometric growth in the number of blocking edges from the bottom to the top of the stack. In [2], every addable edge is λ-minimal, and each blocking edge blocks exactly one addable edge. If using this strategy, in worst case, one may get a layer consisting of one addable edge that is blocked by many blocking edges. After some of these blocking edges are released from $\mathcal{E}$, we may be left with a layer of a single addable edge that is blocked by a single blocking edge, which breaks the geometric growth in the number of blocking edges. To resolve this issue, Annamalai et al. [1] allow a blocking edge to block as many addable edges as possible. However, it brings a new trouble: one may get a layer consisting of many addable edges that are blocked by one blocking edge. As a consequence, they have to introduce another strategy *Greedy*. They require every addable edge to be $\frac{1}{2}$-minimal. If such an addable edge is blocked, at least $\frac{1}{2} - \lambda$ worth of its resources must be occupied by blocking edges. Provided that a blocking edge is λ-minimal and covers at most 2λ worth of resources, the greedy strategy ensures that, in a layer, the number of blocking edges cannot be too small comparing with the number of addable edges. Analysis shows that although the greedy strategy makes the algorithm faster, it deteriorates the approximation ratio. Our strategy is a generalization of those used in [2] and [1]. We allow a blocking edge to block more than one addable edge, but once it shares strictly more than $\beta\lambda$ worth of resources with the addable edges blocked by it, we stop it from blocking more edges. We use greedy too. In our algorithm, addable edges are $(1 + \gamma)\lambda$-minimal for some constant γ.

If we set β, γ, μ to be 0, then our algorithm acts exactly the same as the local search in [2], which achieves a ratio of 4 but may not run in polynomial time. If β, γ are set to be some large constant, then our algorithm acts like the algorithm in [1] which achieves a polynomial running time but a much worse ratio. We show that carefully selected tiny β and tiny γ guarantee a polynomial running time but barely hurt the approximation ratio.

3.1 The algorithm

Let $M \cup \mathcal{E}$ be the current partial allocation. Let p_0 be a player that is not yet covered by $M \cup \mathcal{E}$. The algorithm alternates between two phases to update and extend $M \cup \mathcal{E}$ so that the partial allocation covers p_0 eventually without losing any covered player. In the building phase, it pushes new layers onto a stack, where each layer stores some addable edges and their blocking edges. In the collapse phase, it uses unblocked addable edges to release some blocking edges in some layer from $\mathcal{E}$.

Since we frequently talk about resources covered by thin edges and take sum of values over a set of resources, we define the following notations. Given a thin edge e, R_e denotes the set of resources covered by e. Given a set $\mathcal{S}$ of thin edges, $R(\mathcal{S})$ denotes the set of thin resources covered by $\mathcal{S}$. Given a set D of resources, define $v[D] = \sum_{r \in D} v_r$.

3.1.1 Building phase

The algorithm maintains a stack of layers. The layer index starts with 1 from the bottommost layer in the stack. The i-th layer L_i is a tuple $(\mathcal{A}_i, \mathcal{B}_i, d_i, z_i)$, where $\mathcal{A}_i$ is a set of addable edges that we want to add to $\mathcal{E}$, $\mathcal{B}_i$ is a set of blocking edges that prevent us from doing so, and d_i and z_i are two values maintained for the sake of analysis. The algorithm also maintains a set $\mathcal{I}$ of addable edges that are compatible with $\mathcal{E}$. We will define addable edges and blocking edges later. We use ℓ to denote the number of layers in the current stack. The state of the algorithm is specified by $(M, \mathcal{E}, \mathcal{I}, (L_1, \ldots, L_\ell))$.

For each $\mathcal{A}_i$, we use A_i to denote the set of players covered by $\mathcal{A}_i$. Similarly, B_i and I denote the set of players covered by $\mathcal{B}_i$ and $\mathcal{I}$, respectively. For $i \in [1, \ell]$, define $\mathcal{B}_{\leqslant i} = \bigcup_{j=1}^{i} \mathcal{B}_j$, $B_{\leqslant i} = \bigcup_{j=1}^{i} B_j$, and $\mathcal{A}_{\leqslant i} = \bigcup_{j=1}^{i} \mathcal{A}_j$. We do not define $A_{\leqslant i}$ because although two addable edges in different layers do not share any resource, they may cover the same player. This is a consequence of the layer-level node-disjoint paths technique.

For simplicity, we define the first layer L_1 to be $(\emptyset, \{(p_0, \emptyset)\}, 0, 0)$. That is, $\mathcal{A}_1 = \emptyset$, $\mathcal{B}_1 = \{(p_0, \emptyset)\}$, and $d_1 = z_1 = 0$.

The layers are built inductively. Initially, there is only the layer L_1 and $\mathcal{I} = \emptyset$. Let ℓ be the number of layers in the current stack. Consider the construction of the $(\ell + 1)$-th layer.

▶ **Definition 3.** *Let $\beta \geqslant 0$ be a constant to be specified later. A thin resource r is **inactive** if* (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) *$r \in R_b$ for some $b \in \mathcal{B}_{\ell+1}$ and $v[R_b \cap R(\mathcal{A}_{\ell+1})] > \beta\lambda$. If a thin resource is not inactive, then it is **active**.*

We will define addable edges so that they use only active thin resources.

▶ **Definition 4.** *A player p is **addable** if $f_M(B_{\leqslant \ell}, A_{\ell+1} \cup I \cup \{p\}) = f_M(B_{\leqslant \ell}, A_{\ell+1} \cup I) + 1$.*

The activeness of thin resources and the addability of the players depend on $\mathcal{A}_{\ell+1}$ and $\mathcal{I}$ ($A_{\ell+1}$ and I), so they may be affected as we add edges to $\mathcal{A}_{\ell+1}$ and $\mathcal{I}$.

▶ **Definition 5.** *A thin edge (p, D) is **addable** if p is addable and D is a set of **active** thin resources desired by p with $v[D] \geqslant \lambda$. The **blocking edges** of an addable edge (p, D) are $\{e \in \mathcal{E} : R_e \cap D \neq \emptyset\}$. An addable edge (p, D) is **unblocked** if $v[D \setminus R(\mathcal{E})] \geqslant \lambda$.*

Recall that, for any $w > 0$, a thin edge (p, D) is a *w-minimal* if $v[D] \geqslant w$ and $v[D'] < w$ for any $D' \subsetneq D$. Our algorithm considers two kinds of addable edges. The first kind is unblocked addable edges that are λ-minimal. It is easy to see that if an unblocked addable edge is λ-minimal, then it must be compatible with $\mathcal{E}$. We use $\mathcal{I}$ to keep such addable edges. The second kind is blocked addable edges that are $(1 + \gamma)\lambda$-minimal, where γ is a constant

Table 1 Some facts about $\mathcal{I}$ and the layers in the stack.

Fact 1	Edges in $\mathcal{A}_{\leqslant \ell} \cup \mathcal{I}$ are mutually compatible.
Fact 2	Edges in $\mathcal{I}$ are compatible with edges in $\mathcal{E}$.
Fact 3	For any $i \in [1, \ell]$, no two edges in $\mathcal{A}_i \cup \mathcal{I}$ cover the same player.
Fact 4	$\{\mathcal{B}_2, \ldots, \mathcal{B}_\ell\}$ are disjoint subsets of $\mathcal{E}$. Note that $\mathcal{B}_1 = \{(p_0, \emptyset)\}$ does not share any resource with $\mathcal{B}_i$ for $i \in [2, \ell]$.

to be specified later. Such edges will be added to $\mathcal{A}_{\ell+1}$. Once a $(1+\gamma)\lambda$-minimal addable edge (p, D) becomes unblocked, we can easily extract a λ-minimal unblocked addable edge (p, D') with $D' \subseteq D$.

Consider condition (iii) in Definition 3. Let b be a blocking edge in $\mathcal{B}_{\ell+1}$. When R_b and $R(\mathcal{A}_{\ell+1})$ share strictly more than $\beta\lambda$ worth of resources, all resources in R_b become inactive. Any addable edge to be added to $\mathcal{A}_{\ell+1}$ in the future cannot use these inactive resources, and hence, will not be blocked by b. This is how we achieve "limited blocking" mentioned before.

We call BUILD below to construct the $(\ell+1)$-th layer. Note that after adding an addable edge to $\mathcal{A}_{\ell+1}$, we immediately add its blocking edges to $\mathcal{B}_{\ell+1}$ in order to keep the inactive/active status of thin resources up-to-date.

BUILD$(M, \mathcal{E}, \mathcal{I}, (L_1, \cdots, L_\ell))$

1. Initialize $\mathcal{A}_{\ell+1} = \emptyset$ and $\mathcal{B}_{\ell+1} = \emptyset$.
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, D) that is $(1+\gamma)\lambda$-minimal
   - 3.1 add (p, D) to $\mathcal{A}_{\ell+1}$. (Note that (p, D) must be blocked; otherwise, we could extract from it a λ-minimal unblocked addable edge, which should be added to $\mathcal{I}$ in step 2.)
   - 3.2 add to $\mathcal{B}_{\ell+1}$ the edges in $\mathcal{E}$ that block (p, D).
4. Set $d_{\ell+1} := f_M(B_{\leqslant \ell}, A_{\ell+1} \cup I)$, $z_{\ell+1} := |A_{\ell+1}|$, and $L_{\ell+1} := (\mathcal{A}_{\ell+1}, \mathcal{B}_{\ell+1}, d_{\ell+1}, z_{\ell+1})$.
5. Update $\ell := \ell + 1$

Table 1 lists a few facts about the layers.

3.1.2 Collapse phase

When some layer becomes collapsible, the algorithm enters the collapse phase. Let $(M, \mathcal{E}, \mathcal{I}, (L_1, \cdots, L_\ell))$ be the current state of the algorithm. In order to determine whether a layer is collapsible or not, we need to compute the following decomposition of $\mathcal{I}$. Let $(\mathcal{I}_1, \ldots, \mathcal{I}_{\ell-1})$ be some disjoint subsets of $\mathcal{I}$. Let I_i denote the set of players covered by $\mathcal{I}_i$. For $i \in [1, \ell-1]$, we use $\mathcal{I}_{\leqslant i}$ and $I_{\leqslant i}$ to denote $\bigcup_{j=1}^{i} \mathcal{I}_j$ and $\bigcup_{j=1}^{i} I_j$, respectively.

▶ **Definition 6.** *A collection of disjoint subsets $(\mathcal{I}_1, \ldots, \mathcal{I}_{\ell-1})$ of $\mathcal{I}$ is a **canonical decomposition** of $\mathcal{I}$ if for all $i \in [1, \ell-1]$, $f_M(B_{\leqslant i}, I_{\leqslant i}) = f_M(B_{\leqslant i}, I) = |I_{\leqslant i}|$. A solution Γ for $G_M(B_{\leqslant \ell-1}, I)$ is a **canonical solution** with respect to the canonical decomposition $(\mathcal{I}_1, \ldots, \mathcal{I}_{\ell-1})$ if Γ can be partitioned into disjoint subsets $(\Gamma_1, \ldots, \Gamma_{\ell-1})$ such that for every $i \in [1, \ell-1]$, Γ_i is a set of $|I_i|$ paths from B_i to I_i in G_M.*

Although it is not clear from the definition, invariant 1 in Table 2 implies that $(\mathcal{I}_1, \ldots, \mathcal{I}_{\ell-1})$ is indeed a partition of $\mathcal{I}$. The following lemma is analogous to its counterpart in [1, 9]. Its proof is omitted.

▶ **Lemma 7.** *Let ℓ be the number of layers in the stack. A canonical decomposition of $\mathcal{I}$ and a corresponding canonical solution for $G_M(B_{\leqslant \ell-1}, I)$ can be computed in poly(ℓ, m, n) time.*

All the edges in $\mathcal{I}_i$ are compatible with $\mathcal{E}$, and all the players in I_i can be reached from B_i by node-disjoint paths Γ_i in G_M, so every edge in $\mathcal{I}_i$ can be used to release one blocking edge in $\mathcal{B}_i$ from $\mathcal{E}$. A layer is collapsible if a certain fraction of its blocking edges can be released.

▶ **Definition 8.** *Let μ be a constant to be specified later. A layer L_i is **collapsible** if there is a canonical decomposition $(\mathcal{I}_1, \ldots, \mathcal{I}_{\ell-1})$ of $\mathcal{I}$ such that $|I_i| > \mu |B_i|$.*

Note that although there can be more than one canonical decomposition, the collapsibility of a layer is independent of the choice of canonical decompositions, because by definition, we always have $|I_i| = f_M(B_{\leqslant i}, I) - f_M(B_{\leqslant i-1}, I)$.

When some layer is collapsible, we enter the collapse phase, call Collapse to shrink collapsible layers until no layer is collapsible, and then return to the build phase.

Collapse$(M, \mathcal{E}, \mathcal{I}, (L_1, \cdots, L_\ell))$

1. Compute a canonical decomposition $(\mathcal{I}_1, \ldots, \mathcal{I}_{\ell-1})$ and a corresponding canonical solution $\Gamma_1 \cup \cdots \cup \Gamma_{\ell-1}$ for $G_M(B_{\leqslant \ell-1}, I)$. If no layer is collapsible, go to the build phase; otherwise, let L_t be the collapsible layer with the smallest index.
2. Remove all the layers above L_t from the stack. Set $\mathcal{I} := \mathcal{I}_{\leqslant t-1}$.
3. Recall that $src_{\Gamma_t} \subseteq B_t$ by Definition 6. Let $\mathcal{B}^\Gamma$ denote the set of edges in $\mathcal{B}_t$ that are incident to players in src_{Γ_t}. We use $\mathcal{I}_t$ and Γ_t to release the edges in $\mathcal{B}^\Gamma$.
   - 3.1 Update M by flipping the paths in Γ_t, i.e., set $M := M \oplus \Gamma_t$.
   - 3.2 Add to $\mathcal{E}$ the edges in $\mathcal{I}_t$, i.e., set $\mathcal{E} := \mathcal{E} \cup \mathcal{I}_t$.
   - 3.3 Each player in src_{Γ_t} is now covered by either a fat resource or a thin edge from $\mathcal{I}_t$. If $t = 1$, then p_0 is already covered, and the algorithm terminates. Assume that $t \geqslant 2$. Edges in $\mathcal{B}^\Gamma$ can be safely released from $\mathcal{E}$. Set $\mathcal{E} := \mathcal{E} \setminus \mathcal{B}^\Gamma$ and $\mathcal{B}_t := \mathcal{B}_t \setminus \mathcal{B}^\Gamma$.
4. If $t \geqslant 2$, we need to update $\mathcal{A}_t$ because some edges in $\mathcal{A}_t$ may become unblocked due to the release of blocking edges. For every edge (p, D) in $\mathcal{A}_t$ that becomes unblocked,
   - 4.1 Remove (p, D) from $\mathcal{A}_t$,
   - 4.2 if $f_M(B_{\leqslant t-1}, I \cup \{p\}) = f_M(B_{\leqslant t-1}, I) + 1$, then extract a λ-minimal unblocked addable edge (p, D') from (p, D), and add (p, D') to $\mathcal{I}$.
5. Update $\ell := t$. Go to step 1.

4 Analysis of the approximation algorithm

4.1 Some invariants

Table 2 lists a few invariants, where ℓ is the number of layers in the stack. Lemmas 9 and 10 below have analogous versions in [1, 9] and can be proved similarly. We omit their proofs.

▶ **Lemma 9.** Build *and* Collapse *maintain the invariants in Table 2.*

▶ **Lemma 10.** *Let $(L_1, \ldots, L_\ell)$ be the stack of layers. If no layer is collapsible, then* (i) *$|I| \leqslant \mu |B_{\leqslant \ell-1}|$, and* (ii) *for all $i \in [1, \ell]$, $|A_i| \geqslant z_i - \mu |B_{\leqslant i-1}|$.*

Table 2 Invariants maintained by the algorithm.

Invariant 1	$f_M(B_{\leqslant \ell-1}, I) = \lvert I\rvert$.
Invariant 2	For all $i \in [1, \ell-1]$, $f_M(B_{\leqslant i}, A_{i+1} \cup I) \geqslant d_{i+1}$.
Invariant 3	For all $i \in [1, \ell]$, $\lvert A_i\rvert \leqslant z_i$.
Invariant 4	For all $i \in [1, \ell]$, $d_i \geqslant z_i$.

4.2 Bounding the number of blocking edges

Lemma 11 – 13 are consequences of our greedy and limited blocking strategies. Basically, they bound the number of blocking edges in a layer in terms of the number of addable edges.

▶ **Lemma 11.** *Let $L_i = (\mathcal{A}_i, \mathcal{B}_i, d_i, z_i)$ be an arbitrary layer in the stack. For each 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. Let b be an edge in $\mathcal{B}_i$. Sort the edges in $\mathcal{A}_i$ in chronological order of their additions into $\mathcal{A}_i$. Let a be the last edge in $\mathcal{A}_i$ that is blocked by b. By our choice of a, the edges in $\mathcal{A}_i$ that are added after a cannot be blocked by b, so they do not share any common resource with b. Among the edges added before a, let S be the subset of their resources that are also covered by b. We claim that $v[S] \leqslant \beta\lambda$. If not, all resources in R_b would be inactive before the addition of a by definition of inactive resources. So no resource in R_b could be included in a. But $R_a \cap R_b$ must be non-empty as b blocks a, a contradiction. ◀

▶ **Lemma 12.** *Let $L_i = (\mathcal{A}_i, \mathcal{B}_i, d_i, z_i)$ be an arbitrary layer in the stack. We have $\lvert A_i\rvert < (1 + \frac{\beta}{\gamma})\lvert B_i\rvert$.*

Proof. By Lemma 11, 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. $\lvert\mathcal{A}_i^0\rvert \leqslant \lvert B_i\rvert$.

Let $\mathcal{A}_i^1 = \mathcal{A}_i \setminus \mathcal{A}_i^0$. For every $b \in \mathcal{B}_i$, $v[R_b \cap R(\mathcal{A}_i^1)] \leqslant v[R_b \cap R(\mathcal{A}_i \setminus \{a_b\})] \leqslant \beta\lambda$. Taking sum over all edges in $\mathcal{B}_i$, we get $v[R(\mathcal{B}_i) \cap R(\mathcal{A}_i^1)] \leqslant \beta\lambda\lvert B_i\rvert$. On the other hand, each edge a in $\mathcal{A}_i^1$ is $(1+\gamma)\lambda$-minimal and is blocked, so it must have more than $\gamma\lambda$ worth of its resources occupied by edges in $\mathcal{B}_i$, i.e., $v[R(\mathcal{B}_i) \cap R_a] > \gamma\lambda$. Taking sum over all the edges in $\mathcal{A}_i^1$ gives $v[R(\mathcal{B}_i) \cap R(\mathcal{A}_i^1)] > \gamma\lambda\lvert\mathcal{A}_i^1\rvert$. Hence, $\beta\lambda\lvert B_i\rvert \geqslant v[R(\mathcal{B}_i) \cap R(\mathcal{A}_i^1)] > \gamma\lambda\lvert\mathcal{A}_i^1\rvert$.

Finally we get $\lvert B_i\rvert = \lvert\mathcal{A}_i^0\rvert + \lvert\mathcal{A}_i^1\rvert \leqslant \lvert B_i\rvert + \frac{\beta}{\gamma}\lvert B_i\rvert < (1 + \frac{\beta}{\gamma})\lvert B_i\rvert$. ◀

▶ **Lemma 13.** *Let $L_i = (\mathcal{A}_i, \mathcal{B}_i, d_i, z_i)$ be an arbitrary layer in the stack. Let $\mathcal{B}_i'$ be the set of edges in $\mathcal{B}_i$ that share strictly more than $\beta\lambda$ resources with edges in $\mathcal{A}_i$. More precisely, $\mathcal{B}_i' = \{e \in \mathcal{B}_i : v[R_e \cap R(\mathcal{A}_i)] > \beta\lambda\}$. We have $\lvert\mathcal{B}_i'\rvert < \frac{2+\gamma}{\beta}\lvert A_i\rvert$.*

Proof. Taking the sum of $v[R_e \cap R(\mathcal{A}_i)]$ over all edges e in $\mathcal{B}_i'$, we obtain $v[R(\mathcal{B}_i') \cap R(\mathcal{A}_i)] > \beta\lambda\lvert\mathcal{B}'_i\rvert$. On the other hand, $v[R(\mathcal{B}_i') \cap R(\mathcal{A}_i)] \leqslant v[R(\mathcal{A}_i)] < (2+\gamma)\lambda\lvert A_i\rvert$. The last inequality is because that edges in $\mathcal{A}_i$ are $(1+\gamma)\lambda$-minimal. Combining the above two inequality, we obtain $\beta\lambda\lvert\mathcal{B}'_i\rvert < (2+\gamma)\lambda\lvert A_i\rvert \Rightarrow \lvert\mathcal{B}_i'\rvert < \frac{2+\gamma}{\beta}\lvert A_i\rvert$. ◀

4.3 Geometric growth in the number of blocking edges

Now we are ready to prove that the number of blocking edges grow geometrically from bottom to top. Lemma 14 states that there are lots of addable edges in a layer *immediately after* its construction. Previous Lemma 10(ii) ensures that as long as there is no collapsible layer, every layer cannot lose too many addable edges. Therefore, Lemma 14 implies that

when no layer is collapsible, then every layer must have lots of addable edges, even if they are not newly constructed. Then since the number of blocking edges in a layer is lower bounded in terms of the number of addable edges by Lemma 12, we can conclude that there must be lots of blocking edges in a layer when no layer in the stack is collapsible (Lemma 17).

▶ **Lemma 14.** *Let $(M, \mathcal{E}, \mathcal{I}, (L_1, \ldots, L_{\ell+1}))$ be the state of the algorithm **immediately after** the construction of $L_{\ell+1}$. If no layer is collapsible, then $z_{\ell+1} = |\mathcal{A}_{\ell+1}| \geqslant 2\mu|\mathcal{B}_{\leqslant\ell}|$.*

Proof. We give a proof by contradiction. Suppose that $z_{\ell+1} = |\mathcal{A}_{\ell+1}| < 2\mu|\mathcal{B}_{\leqslant\ell}|$. We will show that the dual of $CLP(1)$ is unbounded, which implies that $CLP(1)$ is infeasible, contradicting the assumption that the configuration LP has optimal value $T^* = 1$.

Consider the moment immediately after we finish adding edges to $\mathcal{A}_{\ell+1}$ during the construction of $L_{\ell+1}$. At this moment, there is no $(1+\gamma)\lambda$-minimal addable edge left. The rest of the proof is with respect to this moment.

Let Π be an optimal solution for $G_M(B_{\leqslant\ell}, A_{\ell+1} \cup I)$. Note that $M \oplus \Pi$ is a maximum matching of G. Consider the directed graph $G_{M\oplus\Pi}$ obtained by orienting the edges of G according to whether they are in $M \oplus \Pi$ or not. Let P^+ be the set of players that can be reached in $G_{M\oplus\Pi}$ from $B_{\leqslant\ell} \setminus src_\Pi$. Let R_f^+ be the set of fat resources that can be reached in $G_{M\oplus\Pi}$ from $B_{\leqslant\ell} \setminus src_\Pi$. Let R_t^+ be the set of inactive thin resources.

▷ Claim 15. (i) Players in P^+ are still addable after we finish adding edges to $\mathcal{A}_{\ell+1}$ (ii) Players in P^+ have in-degree at most 1 in $G_{M\oplus\Pi}$. (iii) Resources in R_f^+ have out-degree exactly 1 in $G_{M\oplus\Pi}$.

We define a dual solution $(\{y_p^*\}_{p\in P}, \{z_r^*\}_{r\in R})$ as follows.

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

▷ Claim 16. $(\{y_p^*\}_{p\in P}, \{z_r^*\}_{r\in R})$ is a feasible solution, and it has a positive objective function value.

Suppose that Claim 16 holds. Then $(\{\alpha y_p^*\}_{p\in P}, \{\alpha z_r^*\}_{r\in R})$ is also a feasible solution for any $\alpha > 0$. As α goes to infinity, the objective function value goes to infinity, yielding the contradiction that we look for. ◀

We omit the proof of Claim 15. We give the proof of Claim 16 below.

Feasibility. We need to show that $\forall p \in P$, $\forall C \in \mathcal{C}_p(1)$, $y_p^* \leqslant \sum_{r\in C} z_r^*$. If $p \notin P^+$, then $y_p^* = 0$, and the inequality holds since z_r^* is non-negative. Assume that $p \in P^+$. So $y_p^* = 1-(1+\gamma)\lambda$. Let C be any configuration for p. We show that $\sum_{r\in C} z_r^* \geqslant 1-(1+\gamma)\lambda$.

Case 1: C contains a fat resource r_f. Since p desires r_f, $G_{M\oplus\Pi}$ has either an edge (p, r_f) or an edge (r_f, p). By the definition of P^+, there is a path π in $G_{M\oplus\Pi}$ from $B_{\leqslant\ell} \setminus src_\Pi$ to p. If $G_{M\oplus\Pi}$ has an edge (p, r_f), we can reach r_f from $B_{\leqslant\ell} \setminus src_\Pi$ by following π and then (p, r_f). So $r_f \in R_f^+$. If $G_{M\oplus\Pi}$ has an edge (r_f, p), then p is matched by $M \oplus \Pi$. Since players in $B_{\leqslant\ell} \setminus src_\Pi$ are not matched by $M \oplus \Pi$, $p \notin (B_{\leqslant\ell} \setminus src_\Pi)$. By Claim 15, in $G_{M\oplus\Pi}$, the in-degree of p is at most one, so (r_f, p) is the only edge entering p. To reach p, π must reach r_f first. Hence, we can follow π to reach r_f from $B_{\leqslant\ell} \setminus src_\Pi$, which implies $r_f \in R_f^+$. In both cases, we have $\sum_{r\in C} z_r^* \geqslant z_{r_f}^* = 1-(1+\gamma)\lambda$.

Case 2: C contains only thin resources. By Claim 15, p is still addable after we finish adding edges to $\mathcal{A}_{\ell+1}$. However, when we finish adding edges to $\mathcal{A}_{\ell+1}$, there is no $(1+\gamma)\lambda$-minimal addable edge left. It must be that p does not have enough active resources to form an addable edge. The active thin resources in C must have a total value less than $(1+\gamma)\lambda$. Recall that $v[C] \geqslant 1$. At least $1-(1+\gamma)\lambda$ worth of thin resources in C are inactive. Since $z^*_r = v_r$ for inactive thin resources, $\sum_{r\in C} z^*_r \geqslant 1-(1+\gamma)\lambda$. ◁

Positive Objective Function Value. We need to show that $\sum_{p\in P} y^*_p - \sum_{r\in R} z^*_r > 0$. By our setting of y^*_p and z^*_r, $\sum_{p\in P} y^*_p - \sum_{r\in R} z^*_r = \sum_{p\in P^+} y^*_p - \sum_{r\in R^+_f} z^*_r - \sum_{r\in R^+_t} z^*_r$.

First consider $\sum_{p\in P^+} y^*_p - \sum_{r\in R^+_f} z^*_r$. Since y^*_p and z^*_r have the same value $1-(1+\gamma)\lambda$ for $p\in P^+$ and $r\in R^+_f$, it suffices to bound $|P^+|-|R^+_f|$ from below. For each $r_f\in R^+_f$, by Claim 15, r_f has exactly one out-going edge to some player p in $G_{M\oplus\Pi}$. Since r_f is reachable from $B_{\leqslant\ell}\setminus src_\Pi$, so is p. That is, $p\in P^+$. We charge r_f to p. By Claim 15, each player in P^+ has in-degree at most 1 in $G_{M\oplus\Pi}$, so each of them is charged at most once. Note that players in $B_{\leqslant\ell}\setminus src_\Pi$ obviously belong to P^+ because they can be reached by themselves. Moreover, they have zero in-degree in $G_{M\oplus\Pi}$ as they are not matched by $M\oplus\Pi$, so they are not charged. Therefore, $|P^+|-|R^+_f| \geqslant |B_{\leqslant\ell}\setminus src_\Pi| \geqslant |B_{\leqslant\ell}|-|A_{\ell+1}|-|I|$. The last inequality is because Π is an optimal solution for $G_M(B_{\leqslant\ell}, A_{\ell+1}\cup I)$. In summary,

$$\sum_{p\in P^+} y^*_p - \sum_{r\in R^+_f} z^*_r \geqslant (1-(1+\gamma)\lambda)\left(|B_{\leqslant\ell}|-|A_{\ell+1}|-|I|\right). \tag{1}$$

Now consider $\sum_{r\in R^+_t} z^*_r$. By definition of inactive resources, R^+_t can be divided into three parts: those covered by $\mathcal{A}_{\leqslant\ell}\cup\mathcal{B}_{\leqslant\ell}$, those covered by $\mathcal{A}_{\ell+1}\cup\mathcal{I}$, and those covered by $\mathcal{B}'_{\ell+1} = \{e\in\mathcal{B}_{\ell+1} : v[R_e\cap R(\mathcal{A}_{\ell+1})] > b\lambda\}$. We handle these three parts separately.

Every edge in $\mathcal{A}_{\leqslant\ell}$ is blocked by some edges in $\mathcal{B}_{\leqslant\ell}$, so it has less than λ worth of thin resources not used by $\mathcal{B}_{\leqslant\ell}$. Every edge in $\mathcal{B}_{\leqslant\ell}$ is λ-minimal, so it covers less than 2λ worth of thin resources. Thus, $v[R(\mathcal{A}_{\leqslant\ell}\cup\mathcal{B}_{\leqslant\ell})] < \lambda|\mathcal{A}_{\leqslant\ell}| + 2\lambda|\mathcal{B}_{\leqslant\ell}| \leqslant \left(3+\frac{\beta}{\gamma}\right)\lambda|B_{\leqslant\ell}|$. The last inequality is by Lemma 12. Edges in $\mathcal{A}_{\ell+1}$ are $(1+\gamma)\lambda$-minimal, so each of them covers less than $(2+\gamma)\lambda$ worth of resources. Edges in $\mathcal{I}$ are λ-minimal, so each of them covers less than 2λ worth of thin resources. Therefore, $v[R(\mathcal{A}_{\ell+1}\cup\mathcal{I})] < (2+\gamma)\lambda|A_{\ell+1}| + 2\lambda|I|$. Edges in $\mathcal{B}'_{\ell+1}$ are λ-minimal, so $v[R(\mathcal{B}'_{\ell+1})] < 2\lambda|\mathcal{B}'_{\ell+1}| < \frac{4+2\gamma}{\beta}\lambda|A_{\ell+1}|$. The second inequality is by Lemma 13. Combining the above three parts gives

$$\sum_{r\in R^+_t} z^*_r = \sum_{r\in R^+_t} v_r < \left(3+\frac{\beta}{\gamma}\right)\lambda|B_{\leqslant\ell}| + 2\lambda|I| + \left(2+\gamma+\frac{4+2\gamma}{\beta}\right)\lambda|A_{\ell+1}|. \tag{2}$$

Combining (1) and (2) gives that

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

By the contrapositive assumption at the beginning of the proof of Lemma 14, $|A_{\ell+1}| < 2\mu|B_{\leqslant\ell}|$. Moreover, since no layer is collapsible, by Lemma 10(i), $|I| \leqslant \mu|B_{\leqslant\ell}|$. Substituting

these two inequalities into the above gives

$$\begin{aligned}&\sum_{p\in P^+} y_p^* - \sum_{r\in R_f^+} z_r^* - \sum_{r\in R_t^+} z_r^*\\ &> \left((1-3\mu) - \left(4+\gamma+\frac{\beta}{\gamma}+3\mu+\frac{(8+4\gamma)\mu}{\beta}-\gamma\mu\right)\lambda\right)|B_{\leqslant\ell}|.\end{aligned}$$

Let $\beta=\gamma^2$ and $\mu=\gamma^3$. We have

$$\sum_{p\in P^+} y_p^* - \sum_{r\in R_f^+} z_r^* - \sum_{r\in R_t^+} z_r^* > \left(\left(1-3\gamma^3\right) - \left(4+10\gamma+4\gamma^2+3\gamma^3-\gamma^4\right)\lambda\right)|B_{\leqslant\ell}|.$$

Recall that $\lambda=\frac{1}{4+\delta}$ for some $\delta>0$. As $\gamma\to 0$, $\frac{1-3\gamma^3}{4+10\gamma+4\gamma^2+3\gamma^3-\gamma^4}\to\frac{1}{4}$. Hence, there is a sufficiently small γ that makes $\frac{1-3\gamma^3}{4+10\gamma+4\gamma^2+3\gamma^3-\gamma^4}>\frac{1}{4+\delta}=\lambda$, thereby proving $\sum_{p\in P^+} y_p^* - \sum_{r\in R_f^+} z_r^* - \sum_{r\in R_t^+} z_r^* > 0$. Moreover, one can verify that $\frac{1}{\gamma}=O(\frac{1}{\delta})$. ◁

▶ **Lemma 17.** *Let $(M,\mathcal{E},\mathcal{I},(L_1,\ldots,L_\ell))$ be a state of the algorithm. If no layer is collapsible, then for $i\in[1,\ell-1]$, $|B_{i+1}|\geqslant\frac{\gamma^3}{1+\gamma}|B_{\leqslant i}|$.*

Proof. Fix an $i\in[1,\ell-1]$. Consider the period from the *most recent* construction of layer L_{i+1} until now. During this period, none of the layers below L_{i+1} has ever been collapsed; otherwise, L_{i+1} would be removed, contradiction. Hence, blocking edges in the layers below L_{i+1} have never been touched during this period. In other words, at the time L_{i+1} was constructed, the set of blocking edges in the layers below L_{i+1} was exactly $\mathcal{B}_{\leqslant i}$. Also the constant z_{i+1} is unchanged. By Lemma 14, $z_{i+1}\geqslant 2\mu|\mathcal{B}_{\leqslant i}|$. Although addable edges may be removed from the L_{i+1} during this period, there are still lots of addable edges left. By Lemma 10(ii), $|A_{i+1}|\geqslant z_{i+1}-\mu|B_{\leqslant i}|\geqslant\mu|B_{\leqslant i}|$. By Lemma 12, $|B_{i+1}|>\frac{1}{(1+\beta/\gamma)}|A_{i+1}|\geqslant\frac{\mu}{(1+\beta/\gamma)}|\mathcal{B}_{\leqslant i}|$. Recall that we set $\beta=\gamma^2$ and $\mu=\gamma^3$ in the proof of Claim 16. Replacing β by γ^2 and μ by γ^3 proves the lemma. ◀

▶ **Lemma 18.** *In $poly(m,n)\cdot n^{poly(\frac{1}{\delta})}$ time, the algorithm extends $M\cup\mathcal{E}$ to cover one more player.*

Given Lemma 17, Lemma 18 can be proved in a way similar to that of [1, 9]. We sketch the proof here. Consider all non-collapsible states ever reached by the algorithm. By non-collapsible, we mean that no layer is collapsible in this state. Let $h=\frac{\gamma^3}{1+\gamma}$. For each non-collapsible state $(M,\mathcal{E},\mathcal{I},(L_1,\ldots,L_\ell))$, we define its signature vector $(s_1,\ldots,s_\ell,\infty)$ 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 non-collapsible state to another, the signature vector decreases lexicographically. Moreover, the sum of the coordinates is bounded by U^2 where $U=\log n\cdot O(\frac{1}{\mu h}\log\frac{1}{h})$. Each signature can be regarded as a partition of an integer less than or equal to U^2. Summing up the number of partitions of an integer i over all $i\in[1,U^2]$, we get the upper bound of $n^{O(\frac{1}{uh}\log\frac{1}{h})}$ on the number of distinct signatures. Recall that $u=\gamma^3$, $h=\frac{\gamma^3}{1+\gamma}$, and $\frac{1}{\gamma}=O(\frac{1}{\delta})$. As a consequence, the number of non-collapsible states ever reached by the algorithm is bounded by $n^{poly(\frac{1}{\delta})}$. Between two consecutive non-collapsible states, there is one Build and at most $\log_{h+1} n$ Collapse, which take $poly(m,n)\cdot n^{poly(\frac{1}{\delta})}$ time in total. The total running time is thus $poly(m,n)\cdot n^{poly(\frac{1}{\delta})}$.

References

1. C. Annamalai, C. Kalaitzis, and O. Svensson. Combinatorial Algorithm for Restricted Max-Min Fair Allocation. *ACM Transactions on Algorithms*, 13(3):37:1–37:28, 2017.
2. A. Asadpour, U. Feige, and A. Saberi. Santa Claus meets hypergraph matchings. *ACM Transactions on Algorithms*, 8(3):24:1–24:9, 2012.
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*, pages 114–121, 2007.
4. N. Bansal and M. Sviridenko. The Santa Claus problem. In *Proceedings of the 38th ACM Symposium on Theory of Computing*, pages 31–40, 2006.
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*, pages 543–552, 2009.
6. I. Bezáková and Varsha Dani. Allocating indivisible goods. *SIGecom Exchanges*, 5(3):11–18, 2005.
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*, pages 107–116, 2009.
8. S.-W. Cheng and Y. Mao. Integrality Gap of the Configuration LP for the Restricted Max-Min Fair Allocation. *CoRR*, abs/1807.04152, 2018. `arXiv:1807.04152`.
9. S.-W. Cheng and Y. Mao. Restricted Max-Min Fair Allocation. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming*, pages 37:1–37:13, 2018.
10. S. Davies, T. Rothvoss, and Y. Zhang. A Tale of Santa Claus, Hypergraphs and Matroids. *CoRR*, abs/1807.07189, 2018. `arXiv:1807.07189`.
11. U. Feige. On allocations that maximize fairness. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms*, pages 287–293, 2008.
12. B. Haeupler, B. Saha, and A. Srinivasan. New Constructive Aspects of the Lovász Local Lemma. *Journal of the ACM*, 58(6):28:1–28:28, 2011.
13. P.E. Haxell. A condition for matchability in hypergraphs. *Graphs and Combinatorics*, 11(3):245–248, 1995.
14. J.E. Hopcroft and R.M. Karp. An $n^{5/2}$ algorithm for maximum matching in bipartitie graphs. *SIAM Journal on Computing*, 2:225–231, 1973.
15. K. Jansen and L. Rohwedder. A note on the integrality gap of the configuration LP for restricted Santa Claus. *CoRR*, abs/1807.03626, 2018. `arXiv:1807.03626`.
16. J.K. Lenstra, D.B. Shmoys, and É. Tardos. Approximation Algorithms for Scheduling Unrelated Parallel Machines. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pages 217–224, 1987.
17. L. Polacek and O. Svensson. Quasi-polynomial Local Search for Restricted Max-Min Fair Allocation. In *39th International Colloquium on Automata, Languages, and Programming*, pages 726–737, 2012.
18. B. Saha and A. Srinivasan. A New Approximation Technique for Resource-Allocation Problems. In *Proceedings of the 1st Symposium on Innovations in Computer Science*, pages 342–357, 2010.