Title	General Max-Min Fair Allocation
Authors	Ko, Sheng-Yen; Chen, Ho-Lin; Cheng, Siu Wing; Hon, Wing-Kai; Liao, Chung-Shou
Source	Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v. 13025 LNCS, October 2021, p. 63-75
Version	Accepted Version
DOI	10.1007/978-3-030-89543-3_6
Publisher	Springer
Copyright	©Springer Nature Switzerland AG 2021. The final publication is available at Springer via https://doi.org/10.1007/978-3-030-89543-3_6

General Max-Min Fair Allocation*

Sheng-Yen Ko[1], Ho-Lin Chen[2], Siu-Wing Cheng[3], Wing-Kai Hong[4], and Chung-Shou Liao[1]

[1] Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu 30013, Taiwan,
s107034524@m107.nthu.edu.tw,csliao@ie.nthu.edu.tw

[2] Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, holinchen@ntu.edu.tw

[3] Department of Computer Science and Engineering, HKUST, Hong Kong, scheng@cse.ust.hk

[4] Department of Computer Science, National Tsing Hua University, Hsinchu 30013, Taiwan, wkhon@cs.nthu.edu.tw

Abstract. In the general max-min fair allocation, also known as the Santa Claus problem, there are m players and n indivisible resources, each player has his/her own utilities for the resources, and the goal is to find an assignment that maximizes the minimum total utility of resources assigned to a player. We introduce an over-estimation strategy to help overcome the challenges of each resource having different utilities for different players. When all resource utilities are positive, we transform it to the machine covering problem and find a $\left(\frac{c}{1-\epsilon}\right)$-approximate allocation in polynomial running time for any fixed $\epsilon \in (0,1)$, where c is the maximum ratio of the largest utility to the smallest utility of any resource. When some resource utilities are zero, we apply the approximation algorithm of Cheng and Mao [9] for the restricted max-min fair allocation problem. It gives a $\left(1+3\hat{c}+O(\delta\hat{c}^2)\right)$-approximate allocation in polynomial time for any fixed $\delta \in (0,1)$, where $\hat{c}$ is the maximum ratio of the largest utility to the smallest positive utility of any resource. The approximation ratios are reasonable if c and $\hat{c}$ are small constants; for example, when the players rate the resources on a 5-point scale.

Keywords: Max-min allocation, hypergraph matching, approximation algorithms

1 Introduction

We consider the general max-min fair allocation problem. The input consists of a set P of m players and a set R of n indivisible resources. Each player $p \in P$ has his/her own non-negative utilities for the n resources, and we denote the utility of the resource r for p by $v_{p,r}$. In other words, each resource r has a set

* Cheng is supported by Research Grants Council, Hong Kong. China (project no. 16207419). Liao is supported by MOST Taiwan under Grants MOST108-2634-F-007-010 and MOST109-2634-F-007-018.

of non-negative utilities $\{v_{p,r} : p \in P\}$, one for each player. For every subset S of resources, define the total utility of S for p to be $v_p(S) = \sum_{r \in S} v_{p,r}$. An allocation is a disjoint partition of R into $\{S_p : p \in P\}$ such that $S_p \subseteq R$ for all $p \in P$, and $S_p \cap S_q = \emptyset$ for any $p \neq q$. That is, p is assigned the resources in S_p. The *max-min fair allocation problem* is to find an allocation that maximizes $\min\{v_p(S_p) : p \in P\}$.

The problem has received considerable attention in recent decades. A related problem is a classic scheduling problem that minimizes the maximum makespan of scheduling on unrelated parallel machines. The problem has the same input as the max-min fair allocation problem. The only difference between them is that the goal of the scheduling problem is to minimize the maximum load over all machines. Lenstra et al. [16] proposed a 2-approximation algorithm by rounding the relaxation of the assignment linear programming model (LP). However, Bezáková and Dani [6] proved that the assignment LP cannot guarantee the same performance on the general max-min allocation problem.

The machine covering problem is a special case of the general max-min fair allocation problem where the objective is to assign n jobs to m parallel identical machines so that the minimum machine load is maximized. Every job (resource) has the same positive utility for every machine (player), i.e., every $r \in R$ has a positive value v_r such that $v_{p,r} = v_r > 0$ for all $p \in P$. Deuermeyer et al. [13] proved that the heuristic LPT algorithm returns a $\frac{4}{3}$-approximation allocation. Csirik et al. [11] improved the approximation ratio to $\frac{4m-2}{3m-1}$. Later, Woeginger [18] presented a polynomial time approximation scheme to develop a $\frac{1}{1-\epsilon}$-approximation algorithm, where $\epsilon > 0$. For another machine covering problem that considers the machine speed s_p and the processing time $v_{p,r} = v_r/s_p$, Azar et al. [4] also proposed a polynomial time approximation scheme. Furthermore, the online machine covering problem was studied in [14, 15] for identical machines.

Bansal and Sviridenko [5] proposed a stronger LP relaxation, the configuration LP, for the general max-min fair allocation problem. They showed that the integrality gap of the configuration LP is $\Omega(\sqrt{m})$, where m is the number of players. Based on the configuration LP, Asadpour and Saberi [3] developed an approximation algorithm that achieves an approximation ratio of $O(\sqrt{m}\log^3 m)$ by rounding. Later, Saha and Srinivasan [17] reduced the approximation ratio to $O(\sqrt{m \log m})$. Chakrabarty et al. [7] developed a method to provide a trade-off between the approximation ratio and the running time: for all $\delta \in (0, 1)$, an approximation ratio of $O(m^\delta)$ can be obtained in $O(m^{1/\delta})$ time.

Bansal and Sviridenko [5] also introduced an interesting restricted max-min fair allocation. In the restricted case, each resource has the same utility v_r for all players who are interested in it, that is, $v_{p,r} \in \{0, v_r\}$ for all $p \in P$. They proposed an $O\left(\frac{\log\log m}{\log\log\log m}\right)$-approximation algorithm by rounding the configuration LP. Later, Asadpour et al. [2] used the bipartite hypergraph matching technique to attack the restricted max-min fair allocation problem. They used local search to show that the integrality gap of the configuration LP is at most 4. However, it is not known whether the local search in [2] runs in polynomial time. Inspired by [2], Annamalai et al. [1] designed an approximation algorithm that enhances the local search with a *greedy player* strategy and a *lazy update* strategy. Their algorithm runs in polynomial time and achieves an approximation ratio of $12.325+\delta$. Cheng and Mao [8] adjusted the greedy strategy in a more flexible and aggressive way, and they successfully lowered the approximation ratio to $6+\delta$. Very recently, they introduced the *limited blocking* idea and improved the ratio to $4+\delta$ [9,10]. This ratio was also obtained by Davies et al. [12]. Table 1 lists the related results.

Table 1. Results on the max-min allocation problem

Problem	Approximation ratio	Running time	Ref.
Restricted	$O\left(\frac{\log\log m}{\log\log\log m}\right)$	$poly(m,n)$	[5]
General	$O\left(\sqrt{m}\log^3 m\right)$	$poly(m,n)$	[3]
General	$O\left(\sqrt{m}\log m\right)$	$poly(m,n)$	[17]
General	$O\left(m^{\delta}\right)$	$n^{O(1/\delta)}$	[7]
Restricted	$12.325+\delta$	$poly(m,n)\cdot m^{poly(1/\delta)}$	[1]
Restricted	$6+\delta$	$poly(m,n)\cdot m^{poly(1/\delta)}$	[8]
Restricted	$4+\delta$	$poly(m,n)\cdot m^{poly(1/\delta)}$	[9, 12]
General (positive utilities)	$c/(1-\epsilon)$	$O(n\log m)$	This paper
General	$1+3\hat{c}+O(\delta\hat{c}^2)$	$poly(m,n)\cdot m^{poly(1/\delta)}$	This paper

The major challenge for the general max-min fair allocation problem is that a resource may have different utilities for different players. Our key idea is to use a *player-independent* value to estimate the value of a particular set: for every resource $r\in R$, its over-estimated utility is $v_r^{max}=\max\{v_{p,r}:p\in P\}$. Consider an instance where every utility in the set $\{v_{p,r}:p\in P,\, r\in R\}$ is positive. Such a problem setting is closely related to the machine covering problem. The main difference is that players are not identical and players have their own preferences for the resources. Using the player-independent over-estimation, we can transform this case to the machine covering problem. Then, we can apply the currently best algorithm proposed by [18] to obtain an allocation in which every player gets at least $(1-\epsilon)T^*_{oe}$ worth of resources, where T^*_{oe} denotes the optimal solution for the transformed machine covering problem. Due to the over-estimation before the transformation, the allocation may be off by an additional factor of $c=\max\left\{v_{p,r}/v_{q,r}:r\in R\wedge p,q\in P\right\}$. This gives our first result which is summarized in Theorem 1 below. Further discussions are provided in Section 2.

Theorem 1. *For all $\epsilon\in(0,1)$, there is a polynomial-time $\left(\frac{c}{1-\epsilon}\right)$-approximation algorithm for the case that every resource has a positive utility for every player.*

The problem becomes much more complicated when some resource has zero utility for some players. Our strategy is to apply the algorithm of Cheng and

Mao [9, 10] for the restricted max-min fair allocation. To this end, we use the over-estimation strategy so that we can discuss active and inactive resources using maximum resource utilities when applying the technique of limited blocking as in [9, 10]. Interestingly, we use each player's own utilities for the resources when running the algorithm although players may have different utilities for the same resource. Then, in the analysis, we use the over-estimation strategy again to reconcile the analysis in [9, 10] for the restricted case with the general case that we consider. Since the approximation ratio given in [9] is $4+\delta$, one may think that the approximation ratio is 4 times the over-estimation factor plus some low-order terms. We adapt the analysis in [9, 10] to show a better approximation ratio of $1+3\hat{c}+O(\delta\hat{c}^2)$, where $\hat{c} = \max\bigl\{v_{p,r}/v_{q,r} : r \in R \wedge p,q \in P \wedge v_{q,r} > 0\bigr\}$.

Theorem 2. *For all $\delta \in (0,1)$, there is a $(1+3\hat{c}+O(\delta\hat{c}^2))$-approximation algorithm that runs in polynomial time for the case that some resources have zero utility for some players.*

The approximation ratios in Theorems 1 and 2 are reasonable if c and $\hat{c}$ are small; for example, when the players rate the resources on a 5-point scale.

2 Every resource has a positive utility for every player

In this section, we consider the general max-min fair allocation problem in which all utilities are positive. This model is similar to the machine covering problem, but the major difference is that every player has his/her own preferences for the resources. There is a polynomial time approximation scheme for the machine covering problem, proposed by Woeginger [18], that achieves an approximation ratio of $1/(1-\epsilon)$ in polynomial time.

Fr each resource $r \in R$, we use $v_r^{\max} = \max\{v_{p,r} : p \in P\}$ as the player-independent utility for r. Hence all players become identical. This transformed problem is exactly the machine covering problem which we denote by H'. Let H denote the original problem before the transformation. Let T^* be the optimal target value for H, and let T^*_{oe} be the optimal target value for H'. So $T^* \leq T^*_{oe}$. Using the PTAS algorithm [18], we can find an approximation allocation $\{S_p : p \in P\}$, where $\sum_{r\in S_p} v_r^{\max} \geq (1-\epsilon)T^*_{oe}$ for every $p \in P$. When we consider the actual value $v_p(S_p)$, we have to allow for the over-estimation factor $c = \max\{v_{p,r}/v_{q,r} : r \in R \wedge p,q \in P\}$.

The definition of c implies that $\sum_{r\in S_p} v_r^{\max} \leq c \cdot v_p(S_p)$, which further implies that $(1-\epsilon)T^*_{oe} \leq \sum_{r\in S_p} v_r^{\max} \leq c \cdot v_p(S_p)$. That is, we guarantee that every player receives at least $(1-\epsilon)T^*_{oe}/c$ worth of resources. Since $T^*_{oe} \geq T^*$, the allocation is a $\left(\frac{c}{1-\epsilon}\right)$-approximation for the original problem H. This completes the discussion of Theorem 1.

3 Some resources have zero utility for some players

As mentioned previously, we will combine the over-estimation and the algorithm in [9, 10] for the restricted max-min allocation problem. We will guess a target value T of the general max-min allocation problem and then try to find an allocation in which the resources assigned to every player have a total utility of at least λT. Depending on whether we succeed or not, we increase or decrease T correspondingly in order to zoom into the value T^* of the optimal max-min allocation. The initial range for T for binary search is $(0, \frac{1}{m}\sum_{r \in R} v_r^{max})$.

In the rest of this section, we assume that $T = 1$, which can be enforced by scaling all resource utilities, and we describe how to find an allocation such that every player obtains resources with a total utility of at least λ.

3.1 Resources and over-estimation

We call a resource r *fat* if $v_{p,r} \geq \lambda$ for all $p \in P$. Otherwise, there exists a player p such that $v_{p,r} < \lambda$, and we call r *thin* in this case. The input resources are thus divided into fat and thin resources.

Furthermore, we modify the resource utilities as follows: for every $r \in R$ and every $p \in P$, if $v_{p,r} > \lambda$, we reset $v_{p,r} := \lambda$. This modification does not affect our goal of finding an allocation in which the resources assigned to every player have a total utility of at least λ. Note that $v_{p,r}$ is left unchanged if it is at most λ. Therefore, fat resources remain fat, and thin resources remain thin. Note that r still has zero utility for those players who are not interested in r, and different players may have different utilities for the same resource.

Since we have reset each $v_{p,r}$ so that it is at most λ, we have $v_r^{\max} \leq \lambda$. For any subset D of thin resources, let $v_{\max}(D) = \sum_{r \in D} v_r^{\max}$. For every player p, $v_p(D)$ still denotes $\sum_{r \in D} v_{p,r}$.

3.2 Fat edges and thin edges

For better resource utilization, it suffices to assign a player p either a single fat resource (whose utilities are all equal to λ after the above modification), or a subset D of thin resources such that $v_p(D) \geq \lambda$. We model the above possible assignment of resources to players using a bipartite graph G and a bipartite hypergraph H. The vertices of G are the players and fat resources. For every player p and every fat resource r_f, G includes the edge (p, r_f) which we call a *fat edge*. The vertices of H are the players and thin resources. For every subset D of thin resources and every player p, the hypergraph H includes the edge (p, D) if $v_p(D) \geq \lambda$.

3.3 Overview of the algorithm

We focus on finding an allocation that corresponds to a maximum matching M in G and a subset $\mathcal{E}$ of hyperedges H such that every player is incident to an edge in M or $\mathcal{E}$, and no two edges in $M \cup \mathcal{E}$ share any resource.

To construct such an allocation, we start with an arbitrary maximum matching M of G and an empty $\mathcal{E}$, process unmatched players one by one in an arbitrary order, and update M and $\mathcal{E}$ in order to match the next unmatched player. Once the algorithm matches a player, that player remains matched until the end of the algorithm. Also, although M may be updated, it is always some maximum matching of G. We call any intermediate $M \cup \mathcal{E}$ a *partial allocation*.

Let G_M be a directed graph obtained by orienting the edges of G with respect to M of G as follows. If a fat edge $\{p, r_f\}$ belongs to the matching M, we orient $\{p, r_f\}$ from r_f to p in G_M. Conversely, if $\{p, r_f\}$ does not belong to the matching M, we orient $\{p, r_f\}$ from p to r_f in G_M.

Let p_0 be an arbitrary unmatched player with respect to the current partial allocation $M \cup \mathcal{E}$. We find a directed path π from p_0 to a player q_0 in G_M. If $p_0 = q_0$, then π a *trivial path*. In this case, if q_0 is covered by a thin edge a that does not share any resource with the edges in $M \cup \mathcal{E}$, then we can update the partial allocation to be $M \cup (\mathcal{E} \cup \{a\})$ to match p_0. If $p_0 \neq q_0$, then π a *non-trivial path*. Note that π has an even number of edges because both p_0 and q_0 are players. For $i \geq 0$, every $(2i+1)$-th edge in π does not belong to M, but every $(2i+2)$-th edge in π does. It is an *alternating path* in the matching terminology. The last edge in π is a matching edge (r_f, q_0) in M for some fat resource r_f. Suppose that q_0 is incident to a thin edge a that does not share any resource with any edge in $M \cup \mathcal{E}$. Then, we can update M to another maximum matching of G by flipping the edge in π. That is, delete every $(2i+2)$-th edge in π from M and add every $(2i+1)$-th edge in π to M. Denote this update of M by flipping π as $M \oplus \pi$. Consequently, p_0 is now matched by M. Although q_0 is no longer matched by M, we can regain q_0 by including the thin edge a. In all, the updated partial allocation is $(M \oplus \pi) \cup (\mathcal{E} \cup \{a\})$.

However, sometimes we cannot find a thin edge a that is incident to q_0 and shares no resource with the edges in $M \cup \mathcal{E}$. Let b be an edge in $M \cup \mathcal{E}$. If a and b share some resource, then a is blocked by b. That is, if we want to add a into $\mathcal{E}$, we must release the resources in b first. Thus, a is an *addable edge* and b is a *blocking edge* that forbids the addition of a. We will provide the formal definitions of addable and blocking edges shortly. To release the resources covered by b, we need to reconsider how to match the player covered by b. This defines a similar intermediate subproblem that needs to be solved first, namely, finding a thin edge that is incident to the player covered by b and shares no resource with the edges in $M \cup \mathcal{E}$. In general, the algorithm maintains a stack that consists of layers of addable and blocking edges; each layer correspond to some intermediate problems that need to be solved. Eventually, every blocking edge needs to be released in order that we can match p_0 in the end.

Annamalai et al. [1] introduced two ideas to enhance the above local search for the restricted max-min allocation problem. They are instrumental in obtaining a polynomial running time. First, when an *unblocked* addable edge is found, it is not used immediately to update the partial allocation. Instead, the algorithm waits until there are enough unblocked addable edges to reduce the number of blocking edges significantly. This ensures that the algorithm makes a substantial progress with each update of the partial allocation. This is called the *lazy update strategy*. Second, when the algorithm considers an addable edge (p, D), it requires $v_p(D)$ to be a constant factor larger than λ. As a result, (p, D) will induce more blocking edges, which will result in a *geometric growth* of the blocking edges in the layers from the bottom of the stack towards the top of the stack. This is called the *greedy player strategy*.

The greedy player strategy causes trouble sometimes, and a blocking edge may block too many addable edges. To this end, Cheng and Mao [9, 10] introduced *limited blocking* which stops the resources in a blocking edge b from being picked in an addable edge if b shares too many resources with addable edges.

We provide more details of the algorithm in the remaining subsections.

3.4 Layers of addable and blocking edges

For every thin edge e, we use R_e to denote the resources covered by e. Given a set $\mathcal{X}$ of thin edges, we use $R(\mathcal{X})$ to denote the set of resources covered by the edges in $\mathcal{X}$.

Let $\Sigma = (L_0, L_1, \dots, L_\ell)$ denote the current stack maintained by the algorithms, where each $L_i = (\mathcal{A}_i, \mathcal{B}_i)$ is a layer that consists of a set $\mathcal{A}_i$ of addable edges and a set $\mathcal{B}_i$ of blocking edges. That is, $\mathcal{B}_i = \{e \in \mathcal{E} : R_e \cap R(\mathcal{A}_i) \neq \emptyset\}$. The layer L_{i+1} is on top of the layer L_i. The layer $L_0 = (\mathcal{A}_0, \mathcal{B}_0)$ at the stack bottom is initialized to be $(\emptyset, \{(p_0, \emptyset)\})$. It signifies that there is no addable edge initially, and replacing $(p_0, \emptyset)$ by some edge is equivalent to finding an edge that covers p_0 without causing any blocking. In general, when building a new layer $L_{\ell+1}$ in Σ, the algorithm starts with $\mathcal{A}_{\ell+1} = \emptyset$, $\mathcal{B}_{\ell+1} = \emptyset$, and addable and blocking edges will be added to $\mathcal{A}_{\ell+1}$ and $\mathcal{B}_{\ell+1}$.

We use $\mathcal{A}_{\leq i}$ to denote $\mathcal{A}_0 \cup \ldots \cup \mathcal{A}_i$. Similarly, $\mathcal{B}_{\leq i} = \mathcal{B}_0 \cup \ldots \cup \mathcal{B}_i$.

The current configuration of the algorithm can be specified by a tuple $(M, \mathcal{E}, \Sigma, \ell, \mathcal{I})$, where $M \cup \mathcal{E}$ is the current partial allocation, Σ is the current stack of layers, ℓ is the index of the highest layer, and $\mathcal{I}$ is a set of thin edges in H such that they cover the players of some edges in $\mathcal{B}_{\leq \ell}$ and each edge in $\mathcal{I}$ does not share any resource with any edge in $\mathcal{E}$. Although the edges in $\mathcal{I}$ can be added to $\mathcal{E}$ immediately to release some blocking edges in $\mathcal{B}_{\leq \ell}$, we do not do so right away in order to accumulate a larger $\mathcal{I}$ which will release more blocking edges in the future.

Definition 1. *Let $(M, \mathcal{E}, \Sigma, \ell, \mathcal{I})$ be the current configuration. A thin resource r can be* **active** *or* **inactive**. *It is inactive if at least one of the following three conditions is satisfied: (a) $r \in R(\mathcal{A}_{\leq \ell} \cup \mathcal{B}_{\leq \ell})$, (b) $r \in R(\mathcal{A}_{\ell+1} \cup \mathcal{I})$, and (c) $r \in R_b$ for some $b \in \mathcal{B}_{\ell+1}$ and $v_{\max}\big(R_b \cap R(\mathcal{A}_{\ell+1})\big) > \beta\lambda$, where β is a positive parameter to be specified later. If none is satisfied, then r is active.*

Condition (c) is a modification of the *limited blocking strategy* introduced in [9, 10] that fits with our over-estimation strategy. The utilities of inactive resources are disregarded in judging whether a thin edge contributes enough total utility to be considered an addable edge. Avoiding inactive resources, especially those in condition (c), helps to improve the approximation ratio.

Let A_i, B_i, and I denote the sets of players covered by the edges in $\mathcal{A}_i$, $\mathcal{B}_i$, and $\mathcal{I}$, respectively. Let $A_{\leq i} = A_0 \cup \ldots A_i$, and let $B_{\leq i} = B_0 \cup \ldots \cup B_i$. Given two subsets of players S and T, we use $f_M[S,T]$ to denote the maximum number of node-disjoint paths from S to T in G_M. The alternating paths in G_M from $B_{\leq \ell}$ to I and other players are relevant. If we flip the alternating paths to I, we can release some blocking edges in $\mathcal{B}_{\leq \ell}$ because they will be matched to fat resources instead. Also, if there is an alternating path from $B_{\leq \ell}$ to a player p, then we can look for a thin edge that covers p to release a blocking edge.

Definition 2. *Let $(M,\mathcal{E},\Sigma,\ell,\mathcal{I})$ be the current configuration. A player p is* **addable** *if $f_M[B_{\leq \ell}, A_\ell \cup I \cup \{p\}] = f_M[B_{\leq \ell}, A_\ell \cup I] + 1$.*

Definition 3. *Let $(M,\mathcal{E},\Sigma,\ell,\mathcal{I})$ be the current configuration. Given an addable player p, a thin edge (p,D) in H is* **addable** *if D is a set of active thin resources and $v_p(D) \geq \lambda$.*

As mentioned before, the edges in $\mathcal{I}$ can be deployed any time to replace some blocking edges, but we only do so when we can release a significant number of blocking edges. When this is possible for a layer in Σ, we call that layer collapsible as defined below.

Definition 4. *Let $(M,\mathcal{E},\Sigma,\ell,\mathcal{I})$ be the current configuration. Let $\mu \in (0,1)$be a parameter to be specified later. The layer L_0 in Σ is collapsible if $f_M[B_0, I] = 1$ (note that $|B_0| = 1$), and for $i \in [1,\ell]$, the layer L_i is* **collapsible** *if $f_M[B_{\leq i}, I] - f_M[B_{\leq i-1}, I] > \mu|B_i|$.*

3.5 The local search step

We discuss how to match the next unmatched player p_0. Let $M \cup \mathcal{E}$ be the current partial allocation. Let $\Sigma = \big(L_0\big)$ be the initial stack. Let $\mathcal{I} = \emptyset$. We go into the Build phase to add a new layer to Σ. Afterwards, if some layer becomes collapsible, we go into the Collapse phase to prune Σ and update the current partial allocation. Afterwards, we go back into the Build phase to add new layers to Σ again. The above is repeated until Σ becomes empty, which means that p_0 is matched eventually. We describe the Build and Collapse phases below.

Build phase. We start with $\mathcal{A}_{\ell+1} = \mathcal{B}_{\ell+1} = \emptyset$. We grow $\mathcal{A}_{\ell+1}$ and $\mathcal{B}_{\ell+1}$ as long as we can find some appropriate thin edge (p,D) in H:

- Suppose that there is an unblocked addable thin edge (p,D). That is, $R(D) \cap R(\mathcal{E}) = \emptyset$. It is natural to add such an edge to $\mathcal{I}$, but for better resource utilization, there is no need to use the whole D if $v_p(D)$ is way larger than λ. We greedily extract a λ-*minimal thin edge* (p,D') from (p,D): (i) D' is a subset of D such that $v_p(D') \geq \lambda$, and (ii) $v_p(D'') < \lambda$ for all subset $D'' \subset D'$. We add (p,D') to $\mathcal{I}$.
- Assume that all addable thin edges are blocked. Suppose that there is blocked addable thin edge (p,D) that is $(1+\gamma)\lambda$-minimal for an appropriate $\gamma \in (0,1)$ that will be specified later. That is, $v_p(D) \geq (1+\gamma)\lambda$ and for all $D' \subset D$, $v_p(D') < (1+\gamma)\lambda$. This is in accordance with the greedy player strategy. Let E be the subset of thin edges in $\mathcal{E}$ that block (p,D), i.e., $E = \{e \in \mathcal{E} : R_e \cap R(D) \neq \emptyset\}$. Add (p,D) to $\mathcal{A}_{\ell+1}$ and update $\mathcal{B}_{\ell+1} := \mathcal{B}_{\ell+1} \cup E$.

Our definitions of λ-minimal and $(1+\gamma)\lambda$-minimal thin edges are player-dependent, in contrast to their player-independent counterparts in the restricted max-min case [9, 10].

If no more edge can be added to $\mathcal{I}$, or $\mathcal{A}_{\ell+1}$ and $\mathcal{B}_{\ell+1}$, then we push $(\mathcal{A}_{\ell+1}, \mathcal{B}_{\ell+1})$ onto Σ and increment ℓ. If some layer becomes collapsible, we go into the Collapse phase; otherwise, we repeat the Build phase to construct another new layer.

Collapse phase. Let L_k be the lowest collapsible layer in Σ. We are going to prune $\mathcal{B}_k$ which will make all layers above L_k invalid. Correspondingly, some of the unblocked addable edges in $\mathcal{I}$ also become invalid because they are generated using blocking edges in $\mathcal{B}_i$ for $i \in [k+1, \ell]$. So a key step is to decompose $\mathcal{I}$ into a disjoint partition $\bigcup_{i=0}^{\ell} \mathcal{I}_i$ such that, among the $f_M[B_{\leq \ell}, I]$ paths in G_M from $B_{\leq \ell}$ to I, there are exactly $|I_i|$ paths from B_i to I_i for $i \in [0, \ell]$, where I_i denotes the set of players covered by $\mathcal{I}_i$.

We remove L_i for $i \geq k+1$ from Σ, and we also forget about $\mathcal{I}_i$ for $i \geq k+1$. We change M by flipping the alternating paths from B_k to I_k. The sources of these paths form a subset of B_k, which are covered by a subset $\mathcal{B}_k^* \subseteq \mathcal{B}_k$. The flipping of the alternating paths from B_k to I_k has the effect of replacing $\mathcal{B}_k^*$ by $\mathcal{I}_k$ in $\mathcal{E}$.

If $k = 0$, it means that the next unmatched player p_0 is now matched and the local search has succeeded. Otherwise, some of the addable edges in $\mathcal{A}_k$ may no longer be blocked due to the removal of $\mathcal{B}_k^*$ from $\mathcal{E}$. We reset $\mathcal{I} := \mathcal{I}_{\leq k-1}$. For each edge $(p, D) \in \mathcal{A}_k$ that becomes unblocked, we delete (p, D) from $\mathcal{A}_k$, and if $f_M[B_{\leq k-1}, I \cup \{p\}] = f_M[B_{\leq k-1}, I] + 1$,[5] then we extract a λ-minimal thin edge (p, D') from (p, D) and add (p, D') to $\mathcal{I}$. After pruning $\mathcal{A}_k$, we reset $\ell := k$.

We repeat the above as long as some layer in Σ is collapsible. When this no longer the case and p_0 is not matched yet, we go back to the Build phase.

3.6 Analysis

For any $\delta \in (0, 1)$, we show that we can set $\gamma = \Theta(\delta)$, $\beta = \gamma^2$, and $\mu = \gamma^3$ so that the local search runs in polynomial time, and if the target value 1 is feasible, the local search returns an allocation that achieves a value of $\lambda = 1/(1 + 3\hat{c} + O(\delta\hat{c}^2))$. Recall that $\hat{c} = \max\{v_{p,r}/v_{q,r} : r \in R \wedge p, q \in P \wedge v_{q,r} > 0\}$.

The key is to show that the stack Σ has logarithmic depth for these choices of γ, β and μ. The numbers of blocking edges $|\mathcal{B}_i|$ for $i \in [0, \ell]$ induce a signature vector that increases lexicographically as the local search proceeds. Therefore, if Σ has logarithmic depth, the local search must terminate in polynomial time before we run out of all possible signature vectors. To show that Σ has logarithmic depth, we are to prove that the number of blocking edges increases geometrically from one layer to the next as we go up Σ.

First, we extract Lemmas 1–3 from [9, 10]; either the original proofs still hold or minor adaptations work for our setting.

[5] As proved in [9, 10], this is equivalent to checking the addability of player p.

Lemma 1. *For $i \in [0, \ell]$, let $z_i = |\mathcal{A}_i|$ right after the creation of the layer L_i. Whenever no layer in collapsible, $|A_{i+1}| \geq z_{i+1} - \mu|B_{\leq i}|$ for $i \in [0, \ell - 1]$.*

Lemma 2. *For each blocking edge $b \in \mathcal{B}_i$, there exists an edge $a \in \mathcal{A}_i$ such that $v_{max}\big(R_b \cap R(\mathcal{A}_i \setminus \{a\})\big) \leq \beta\lambda$.*

Lemma 3. *For $i \in [0, \ell]$, $|\mathcal{A}_i| < \big(1 + \frac{\beta}{\gamma}\big)|\mathcal{B}_i|$.*

The analogous version of Lemma 4 below in [9, 10] gives the inequality $|\mathcal{B}'_i| < \frac{(2+\gamma)}{\beta}|\mathcal{A}_i|$ in the restricted max-min case. We prove that a similar bound with the extra over-estimation factor $\hat{c}$ holds for the general max-min case.

Lemma 4. *Let $\mathcal{B}'_i$ be the subset of $\mathcal{B}_i$ such that all resources in $\mathcal{B}'_i$ are inactive, i.e., $\mathcal{B}'_i = \big\{b \in \mathcal{B}_i \mid v_{max}\big(R_b \cap R(\mathcal{A}_i)\big) > \beta\lambda\big\}$. Then, $|\mathcal{B}'_i| < \frac{(2+\gamma)\hat{c}}{\beta}|\mathcal{A}_i|$.*

Proof. Summing over the edges in $\mathcal{B}'_i$ gives $v_{max}\big(R(\mathcal{B}'_i) \cap R(\mathcal{A}_i)\big) > \beta\lambda|\mathcal{B}'_i|$. Every edge $(p, D) \in \mathcal{A}_i$ is $(1 + \gamma)\lambda$-minimal by definition, so $v_p(D) \leq (2 + \gamma)\lambda$. Summing over the edges in $\mathcal{A}_i$ gives $\sum_{(p,D)\in\mathcal{A}_i} v_p(D) \leq (2 + \gamma)\lambda|\mathcal{A}_i|$. The definition of $\hat{c}$ implies that $v_r^{\max} \leq \hat{c}\, v_{p,r}$, which implies that $v_{max}(R(\mathcal{A}_i)) \leq \sum_{(p,D)\in\mathcal{A}_i} \hat{c}\, v_p(D) \leq (2 + \gamma)\lambda\hat{c}\,|\mathcal{A}_i|$. Combining the inequalities above gives $\beta\lambda|\mathcal{B}'_i| < v_{max}(R(\mathcal{B}'_i) \cap R(\mathcal{A}_i)) \leq v_{max}(R(\mathcal{A}_i)) \leq (2 + \gamma)\lambda\hat{c}\,|\mathcal{A}_i|$, which implies that $|\mathcal{B}'_i| < \frac{(2+\gamma)\hat{c}}{\beta}|\mathcal{A}_i|$. □

Lemma 4 is instrumental to proving Lemma 5 which is the key to showing a geometric growth in the numbers of blocking edges. The proof of Lemma 5 is given in the appendix.

Lemma 5. *Assume that the target value 1 is feasible for the general max-min allocation problem. Then, immediately after the construction of a new layer $L_{\ell+1}$, if no layer is collapsible, then $|\mathcal{A}_{\ell+1}| > 2\mu|\mathcal{B}_{\leq \ell}|$.*

We show how to use Lemma 5 to obtain the geometric growth.

Lemma 6. *If no layer is collapsible, then $\big|\mathcal{B}_{i+1}\big| > \frac{\gamma^3}{1+\gamma}\big|\mathcal{B}_{\leq i}\big|$. Hence, $\big|\mathcal{B}_{\leq i+1}\big| > \big(1 + \frac{\gamma^3}{1+\gamma}\big|\mathcal{B}_{\leq i}\big|\big)$.*

Proof. Let $(L_0, L_1, \ldots, L_\ell)$ be the current stack Σ. Take any $i \in [0, \ell{-}1]$. Since the most recent construction of L_{i+1}, L_{i+1} and any layer below it is not collapsible. If not, L_{i+1} would be deleted, which means that there would be another construction of it after the most recent construction, a contradiction. Therefore, Lemma 5 implies that $z_{i+1} > 2\mu|B_{\leq i}|$, where z_{i+1} is the value of $|A_{i+1}|$ right after the construction of L_{i+1}. By Lemma 1, $|A_{i+1}| \geq z_{i+1} - \mu|B_{\leq i}|$. Substituting $z_{i+1} > 2\mu|B_{\leq i}|$ into this inequality gives $|A_{i+1}| > 2\mu|B_{\leq i}| - \mu|B_{\leq i}| = \mu|B_{\leq i}|$. Lemma 3 implies that $|B_{i+1}| > \frac{\gamma}{\gamma+\beta}|A_{i+1}| > \frac{\gamma\mu}{\gamma+\beta}|B_{\leq i}|$. Plugging in $\beta = \gamma^2$ and $\mu = \gamma^3$ gives $|B_{i+1}| > \gamma^3|B_{\leq i}|/(1 + \gamma)$. □

The next result shows that a polynomial running time follows from Lemma 6.

Lemma 7. *Suppose that the target value 1 is feasible for the general max-min allocation problem. Then, the local search matches a player in* $poly(m,n) \cdot m^{poly(1/\delta)}$.

Proof. The proof follows the argument in [1]. Let $h = \gamma^3/(1+\gamma)$. Define the signature vector $(s_1, s_2, ..., s_\ell, \infty)$, where $s_i = \left\lfloor \log_{1/(1-\mu)} \left(|\mathcal{B}_i| h^{-i-1} \right) \right\rfloor$. By Lemma 6, $|\mathcal{B}_\ell| \geq h|\mathcal{B}_{\leq \ell-1}| \geq h|\mathcal{B}_{\ell-1}|$. So $\infty > s_\ell \geq s_{\ell-1}$, which means that the coordinates of the signature vector are non-decreasing. When a (lowest) layer L_t is collapsed in the Collapse phase, we update $\mathcal{B}_t$ to $\mathcal{B}'_t$ where $(1-\mu)|\mathcal{B}_t| > |\mathcal{B}'_t|$. The signature vector is updated to $(s_1, s_2, ..., s'_t)$ where $s'_t \leq s_t - 1$. So the signature vector decreases lexicographically. By Lemma 6, the number of layers in Σ is at most $\log_{1+h} m$, where m is the number of players. One can verify that the sum of coordinates in every signature vector is at most U^2 where $U = \log m \cdot O(\frac{1}{\mu h} \log \frac{1}{h})$. Every signature vector corresponds to a distinct partition of an integer that is no more than U^2. By summing up the number of distinct partitions of integers that are no more than U^2, we get that the upper bound of $m^{O(\frac{1}{\mu h} \log \frac{1}{h})}$ on the number of signature vectors. Since $\gamma = \Theta(\delta)$, $\mu = \gamma^3$, and $h = \gamma^3/(1+\gamma)$, this upper bound is $m^{poly(1/\delta)}$. This also bounds the number of calls on Build and Collapse. It is not difficult to make the construction of a layer and the collapse of a layer run in polynomial time. □

We have not discussed how to handle the case that the target value 1 is infeasible for the general max-min allocation problem. In this case, the local search must fail at some point. From the previous proofs, we know that as long as the conclusion of Lemma 5 holds immediately after the construction of a new layer $L_{\ell+1}$, that is, if no layer is collapsible, then $|\mathcal{A}_{\ell+1}| > 2\mu|\mathcal{B}_{\leq \ell}|$, the local search must succeed and finish in polynomial time. As a result, we must encounter a situation that no layer is collapsible and yet $|\mathcal{A}_{\ell+1}| \leq 2\mu|\mathcal{B}_{\leq \ell}$ for the first time during the local search. This situation can be checked explicitly and we can abort and guess the next target value. Since the conclusion of Lemma 5 has held so far, the running time up to the point of abortion is polynomial.

4 Conclusion

We provide two solutions for the general max-min fair allocation problem. If every resource has a positive utility for every player, the problem can be transformed to the machine covering problem using our over-estimation strategy. By using an existing polynomial time approximation scheme for the machine covering problem, we obtain a $\left(\frac{c}{1-\epsilon}\right)$-approximation algorithm which runs in polynomial time, where ϵ is any constant in the range $(0,1)$, and $c = \max\{v_{p,r}/v_{q,r} : r \in R \wedge p, q \in P\}$. If some resource has zero utility for some players, we show how to combine the over-estimation strategy with the approximation algorithm in [9] for the restricted max-min allocation problem to obtain an approximation ratio of $1 + 3\hat{c} + O(\delta\hat{c}^2)$ for any $\delta \in (0,1)$ in polynomial time, where $\hat{c} = \max\{v_{p,r}/v_{q,r} : r \in R \wedge p, q \in P \wedge v_{q,r} > 0\}$. We conclude with two research questions. The first question is whether the approximation ratios presented here can be improved further. Despite its theoretical guarantee, the local search step is still quite challenging to implement. So the second question is whether there is a simpler algorithm that can also achieve a good approximation ratio in polynomial time.

References

1. Annamalai, C., Kalaitzis, C., Svensson, O.: Combinatorial algorithm for restricted max-min fair allocation. ACM Transactions on Algorithms **13**(3), 1–28 (2017)
2. Asadpour, A., Feige, U., Saberi, A.: Santa Claus meets hypergraph matchings. ACM Transactions on Algorithms **8**(3), 1–9 (2012)
3. Asadpour, A., Saberi, A.: An approximation algorithm for max-min fair allocation of indivisible goods. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing. pp. 114–121 (2007)
4. Azar, Y., Epstein, L.: Approximation schemes for covering and scheduling on related machines. In: Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization. pp. 39–47 (1998)
5. Bansal, N., Sviridenko, M.: The Santa Claus problem. In: Proceedings of the 38th Annual ACM Symposium on Theory of Computing. pp. 31–40 (2006)
6. Bezáková, I., Dani, V.: Allocating indivisible goods. ACM SIGecom Exchanges **5**(3), 11–18 (2005)
7. Chakrabarty, D., Chuzhoy, J., Khanna, S.: On allocating goods to maximize fairness. In: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science. pp. 107–116 (2009)
8. Cheng, S.W., Mao, Y.: Restricted max-min fair allocation. In: Proceedings of the 45th International Colloquium on Automata, Languages, and Programming. pp. 37:1–37:13 (2018)
9. Cheng, S.W., Mao, Y.: Restricted max-min allocation: Approximation and integrality gap. In: Proceedings of the 46th International Colloquium on Automata, Languages, and Programming. pp. 38:1–38:13 (2019)
10. Cheng, S.W., Mao, Y.: Restricted max-min allocation: Approximation and integrality gap. arXiv:1905.06084 (2019)
11. Csirik, J., Kellerer, H., Woeginger, G.: The exact LPT-bound for maximizing the minimum completion time. Operations Research Letters **11**(5), 281–287 (1992)
12. Davies, S., Tothvoss, T., Zhang, Y.: A tale of Santa Claus, hypergraphs and matroids. In: Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 2748–2757 (2020)
13. Deuermeyer, B., Friesen, D., Langston, M.: Scheduling to maximize the minimum processor finish time in a multiprocessor system. SIAM Journal on Algebraic and Discrete Methods **3**, 190–196 (Jun 1982)
14. He, Y., Jiang, Y.: Optimal semi-online preemptive algorithms for machine covering on two uniform machines. Theoretical Computer Science **339**(2), 293–314 (2005)
15. He, Y., Tan, Z.: Ordinal on-line scheduling for maximizing the minimum machine completion time. Journal of Combinatorial Optimization **6**, 199–206 (Feb 2002)
16. Lenstra, J.K., Shmoys, D.B., Tardos, E.: Approximation algorithms for scheduling unrelated parallel machines. In: Proceedings of the 28th Annual Symposium on Foundations of Computer Science. pp. 217–224 (1987)
17. Saha, B., Srinivasan, A.: A new approximation technique for resource-allocation problems. Random Structures and Algorithms **52**, 680–715 (2018)
18. Woeginger, G.J.: A polynomial-time approximation scheme for maximizing the minimum machine completion time. Operations Research Letters **20**(4), 149–154 (1997)

A Proof of Lemma 5

We prove Lemma 5 by contradiction. Suppose that $z_{\ell+1} = |\mathcal{A}_{\ell+1}| < 2\mu|\mathcal{B}_{\leq \ell}|$.

For every $p \in P$, a configuration C for p is a subset of resources (fat or thin) such that p is interested in the resources in C and $v_p(C) \geq 1$. Let $\mathcal{C}_p(1)$ denote the set of all configurations for p. The LP relaxation below is the configuration LP is for the general max-min allocation problem.

Primal

$$
\begin{aligned}
\textbf{Min} \quad & 0 \\
\textbf{s.t.} \quad & \sum_{C \in \mathcal{C}_p(1)} x_{p,C} \geq 1, && \forall p \in P \\
& \sum_{p \in P} \sum_{r \in C} x_{p,C} \leq 1, && \forall r \in R \\
& x_{p,C} \geq 0, && \forall p \in P,\ \forall C \in \mathcal{C}_p(1)
\end{aligned}
$$

Dual

$$
\begin{aligned}
\textbf{Max} \quad & \sum_{p \in P} y_p - \sum_{r \in R} z_r \\
\textbf{s.t.} \quad & y_p \leq \sum_{r \in C} z_r, && \forall p \in P,\ \forall C \in \mathcal{C}_p(1) \\
& y_p \geq 0, && \forall p \in P \\
& z_r \geq 0, && \forall r \in R
\end{aligned}
$$

We will define a solution for the dual of configuration LP and show that it is feasible and gives a positive objective function value. We can scale up this feasible dual solution arbitrarily, which gives an arbitrarily large objective function value. Therefore, the dual of the configuration LP is unbounded, which implies the contradiction that the configuration LP is infeasible. Our proof is largely the same as the counterpart of Lemma 5 in [9, 10], but adaptions to our setting is needed in defining the dual solution and in proving that the objective function value is positive.

To define the dual solution, we need classify certain subsets of players and resources. Consider the moment right after the completion of the construction of the layer $L_{\ell+1}$.

Let Π be the set of $f_M[B_{\leq \ell}, A_{\ell+1} \cup I]$ node-disjoint paths in G_M from $B_{\leq \ell}$ to $A_{\ell+1} \cup I$. Let $src(\Pi)$ denote the set of sources of the paths in Π. Let Π^+ be the set of non-trivial paths in Π, i.e., paths with at least one edge. Let $src(\Pi^+)$ denote the set of sources of the paths in Π^+. The non-trivial paths in Π are alternating, that is, for $i \geq 0$, every $(2i+2)$-th edge belongs to M and every $(2i+1)$-th edge does not.

If we flip the alternating paths in Π, M is updated to another maximum matching of G. Let $M \oplus \Pi$ denote the resulting maximum matching. Note that flipping a trivial path in Π does not change anything. The players in $src(\Pi^+)$ are not matched in M, but they are now matched in $M \oplus \Pi$, which means that there are directed edges in $G_{M\oplus\Pi}$ from fat resources to the players in $src(\Pi^+)$. So players in $src(\Pi^+)$ has in-degree exactly one in $G_{M\oplus\Pi}$.

Let P^+ be the set of players that can be reached in $G_{M\oplus\Pi}$ from $B_{\leq\ell} \setminus src(\Pi)$. Let R_f^+ be the set of fat resources that can be reached in $G_{M\oplus\Pi}$ from $B_{\leq\ell} \setminus src(\Pi)$. Let R_t^+ be the set of inactive thin resources. We can now define the desired dual solution $(\{y_p^*\}_{p\in P}, \{z_r^*\}_{r\in R})$ as follows:

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

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

We will need the following properties of $G_{M\oplus\Pi}$.

Proposition 1. *In $G_{M\oplus\Pi}$, the players in $B_{\leq\ell} \setminus src(\Pi)$ have zero in-degrees, and the fat resources in R_f^+ have out-degrees exactly one.*

Proof. The players in $B_{\leq\ell}$ are matched by thin edges in $\mathcal{E}$, so they are not matched by M to fat resources. Hence, players in $B_{\leq\ell}$ have in-degree zero in G_M. After flipping the paths in Π, only the player in $src(\Pi)$ may now be matched to fat resources in $M \oplus \Pi$, and therefore, players in $B_{\leq\ell} \setminus src(\Pi)$ are still unmatched in $M \oplus \Pi$. So players in $B_{\leq\ell} \setminus src(\Pi)$ have zero in-degrees.

By the definition of R_f^+, for every resource $r \in R_f^+$, there is a path to r from a player that is not matched in $M \oplus \Pi$. If r is not matched in $M \oplus \Pi$, we can flip this path to increase the matching size, contradicting the fact that $M \oplus \Pi$ is a maximum matching. So r is matched in $M \oplus \Pi$, implying that its out-degree in $G_{M\oplus\Pi}$ is exactly one. $\square$

Note that the rest of the proof considers the moment right after completing the construction of $L_{\ell+1}$, and there is no more $(1+\gamma)\lambda$-minimal addable edge that can be discovered.

A.1 Feasibility

We first prove that the dual solution is feasible, which requires us to show that $y_p^* \leq \sum_{r\in C} z_r^*$, $\forall p \in P$, $\forall C \in \mathcal{C}_i(1)$. The feasibility constraint is trivially satisfied if $p \notin P^+$ because $y_p^* = 0$ in this case and $z_r^* \geq 0$ for all $r \in R$. Take any $p \in P^+$. So $y_p^* = 1-(1+\gamma)\lambda$ by definition. We prove that $\sum_{r\in C} z_r^* \geq 1-(1+\gamma)\lambda$ for all $C \in \mathcal{C}_p(1)$.

Case 1: C contains a fat resource r_f. As $p \in P^+$, there is a path π from $B_{\leq \ell} \setminus src(\Pi)$ to p in $G_{M \oplus \Pi}$. Since r_f belongs to a configuration for p, p desires r, which means that $G_{M \oplus \Pi}$ contains either the directed edge (p, r_f) or the directed edge (r_f, p). We show that $r_f \in R_f^+$ below. It immediately follows that $z_{r_f}^* = 1 - (1+\gamma)\lambda$, so $\sum_{r \in C} z_r^* \geq z_{r_f}^* = 1 - (1+\gamma)\lambda$.

If $G_{M \oplus \Pi}$ contains the edge (p, r_f), then $r_f \in R_f^+$ because we can follow the path π to p and then the edge (p, r_f) to r_f. Suppose that $G_{M \oplus \Pi}$ contains the edge (r_f, p). By the property of matching, p has in-degree at most one in $G_{M \oplus \Pi}$, so (r_f, p) is the only edge entering p. It follows that the path π reaches r_f first before p, so the prefix of π up to r_f certifies that $r_f \in R_f^+$.

Case 2: C contains only thin resources. We consider the moment after completing the construction of $L_{\ell+1}$. The existence of the path π to p certifies that p is an addable player. However, we cannot find any $(1+\gamma)\lambda$-minimal addable edge for p. Since $C \in \mathcal{C}_p(1)$, we have $v_p(C) \geq 1$. Therefore, the total utility of active resources in C for p must be less than $(1+\gamma)\lambda$, which means that the total utility of inactive resources in C for p is greater than $1 - (1+\gamma)\lambda$. Since R_t^+ is the set of inactive thin resources, we have $\sum_{r \in C} z_r^* \geq \sum_{r \in C \cap R_t^+} z_r^* = \sum_{r \in C \cap R_t^+} v_r^{\max} \geq \sum_{r \cap C \cap R_t^+} v_{p,r} > 1 - (1+\gamma)\lambda$.

A.2 Positive objective function value

By definition, $y_p^* = 0$ if $p \notin P^+$, and $z_r^* = 0$ if $r \notin R_f^+ \cup R_t^+$. So we only need to prove that $\sum_{p \in P^+} y_p^* - \sum_{r \in R_f^+} z_r^* - \sum_{r \in R_t^+} z_r^* > 0$.

First we consider the value of $\sum_{p \in P^+} y_p^* - \sum_{r \in R_f^+} z_r^*$. By definition, $y_p^* = z_r^* = 1 - (1+\gamma)\lambda$ for all $p \in P^+$ and $r \in R_f^+$. Then

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

Take any $r \in R_f^+$. By Proposition 1, there is an edge (r_f, p) in $G_{M \oplus \Pi}$ for some player p, i..e, p is matched to r_f in $M \oplus \Pi$. There is a path from $B_{\leq \ell} \setminus src(\Pi)$ to r_f by the definition of R_f^+, which implies that there is path from $B_{\leq \ell} \setminus src(\Pi)$ to p. Hence, $p \in P^+$. We can thus charge every $r_f \in R_f^+$ to a player in P^+ such that, by the property of matching, no player in P^+ is charged more than once. As discussed previously, every player in $B_{\leq \ell} \setminus src(\Pi)$ is not matched in $M \oplus \Pi$, so players in $B_{\leq \ell} \setminus src(\Pi)$ are not charged. Moreover, every player in $B_{\leq \ell} \setminus src(\Pi)$ belongs to P^+ because there is a trivial path from every such player to himself/herself. As a result,

$$|P^+| - |R_f^+| \geq |B_{\leq \ell} \setminus src(\Pi)|.$$

Since Π is the maximum set of node-disjoint paths in G_M from $B_{\leq \ell}$ to $A_{\ell+1} \cup I$, we have $|src(\Pi)| = |\Pi| \leq |A_{\ell+1}| + |I|$. Therefore,

$$|P^+| - |R_f^+| \geq |B_{\leq \ell} \setminus src(\Pi)| \geq |\mathcal{B}_{\leq \ell}| - |\mathcal{A}_{\ell+1}| - |\mathcal{I}|.$$

We conclude that

$$\sum_{p_i \in P^+} y_i^* - \sum_{r_j \in R_f^+} z_j^* = \Big(1-(1+\gamma)\lambda\Big)\Big(|P^+| - |R_f^+|\Big)$$

$$\geq \Big(1-(1+\gamma)\lambda\Big)\Big(|\mathcal{B}_{\leq \ell}| - |\mathcal{A}_{\ell+1}| - |\mathcal{I}|\Big). \quad (1)$$

It remains to analyze $\sum_{r \in R_t^+} z_r^*$. Every resource $r \in R_t^*$ is inactive. By definition, the inactive resources appear in three different kinds of thin edges: (i) $\mathcal{A}_{\leq \ell} \cup \mathcal{B}_{\leq \ell}$, (ii) $\mathcal{A}_{\ell+1} \cup \mathcal{I}$, and (iii) $\mathcal{B}'_{\ell+1} = \big\{ b \in \mathcal{B}_{\ell+1} : v_{\max}\big(R_b \cap R(\mathcal{A}_{\ell+1})\big) > \beta\lambda \big\}$. It follows that

$$\sum_{r \in R_t^+} z_r^* = \sum_{r \in R_t^+} v_r^{\max}$$

$$= v_{\max}\big(R(\mathcal{A}_{\leq \ell} \cup \mathcal{B}_{\leq \ell})\big) + v_{\max}\big(R(\mathcal{A}_{\ell+1} \cup \mathcal{I})\big) + v_{\max}\big(R(\mathcal{B}'_{\ell+1})\big).$$

We first consider $v_{\max}\big(R(\mathcal{A}_{\leq \ell} \cup \mathcal{B}_{\leq \ell})\big)$. Every edge (p, D) in $\mathcal{A}_{\leq \ell}$ is a blocked addable edge. So $v_p\big(D \setminus R(\mathcal{B}_{\leq \ell})\big) < \lambda$. Every edge in $\mathcal{B}_{\leq \ell}$ is λ-minimal. Thus, we can use the over-estimate strategy to bound $v_{\max}\big(R(\mathcal{A}_{\leq \ell} \cup \mathcal{B}_{\leq \ell})\big)$:

$$\begin{aligned} v_{\max}(R(\mathcal{A}_{\leq \ell} \cup \mathcal{B}_{\leq \ell})) &\leq \sum_{(p,D) \in \mathcal{A}_{\leq \ell} \cup \mathcal{B}_{\leq \ell}} \hat{c} \cdot v_p(D) \\ &\leq \hat{c} \cdot \lambda |\mathcal{A}_{\leq \ell}| + \hat{c} \cdot 2\lambda |\mathcal{B}_{\leq \ell}| \\ &\overset{\text{Lemma 3}}{<} \left(3 + \frac{\beta}{\gamma}\right) \hat{c} \cdot \lambda |\mathcal{B}_{\leq \ell}|. \end{aligned} \quad (2)$$

We analyze $v_{\max}\big(R(\mathcal{A}_{\ell+1} \cup \mathcal{I})\big)$ next. Every edge in $\mathcal{A}_{\ell+1}$ is $(1+\gamma)\lambda$-minimal, and every edge in $\mathcal{I}$ is λ-minimal. Therefore,

$$\begin{aligned} v_{\max}\big(R(\mathcal{A}_{\ell+1} \cup \mathcal{I})\big) &\leq \sum_{(p,D) \in \mathcal{A}_{\ell+1} \cup \mathcal{I}} \hat{c} \cdot v_p(D) \\ &\leq (2+\gamma)\hat{c} \cdot \lambda |\mathcal{A}_{\ell+1}| + 2\hat{c} \cdot \lambda |\mathcal{I}|. \end{aligned} \quad (3)$$

Lastly, consider $v_{\max}\big(R(\mathcal{B}'_{\ell+1})\big)$. Edges in $\mathcal{B}'_{\ell+1}$ are λ-minimal. Therefore,

$$\begin{aligned} v_{\max}\big(R(\mathcal{B}'_{\ell+1})\big) &\leq \sum_{(p,D) \in \mathcal{B}'_{\ell+1}} \hat{c} \cdot v_p(D) \\ &\leq 2\hat{c} \cdot \lambda |\mathcal{B}'_{\ell+1}| \\ &\overset{\text{Lemma 4}}{<} \left(\frac{4+2\gamma}{\beta}\right) \hat{c}^2 \cdot \lambda |\mathcal{A}_{\ell+1}|. \end{aligned} \quad (4)$$

Summing (2), (3)) and (4) gives

$$\sum_{r \in R_t^+} z_r^* < \left(3 + \frac{\beta}{\gamma}\right) \hat{c}\lambda |\mathcal{B}_{\leq \ell}| + \left((2+\gamma)\hat{c} + \left(\frac{4+2\gamma}{\beta}\right)\hat{c}^2\right) \lambda |\mathcal{A}_{\ell+1}| + 2\hat{c}\lambda |\mathcal{I}|. \quad (5)$$

Combining (1) and (5) gives

$$\begin{aligned}
&\sum_{p\in P^+} y_p^* - \sum_{r\in R_f^+} z_r^* - \sum_{r\in R_t^+} z_r^* \\
&> \left(1-(1+\gamma)\lambda - \left(3+\frac{\beta}{\gamma}\right)\hat{c}\lambda\right)|B_{\le\ell}| \\
&\quad - \left(1-(1+\gamma)\lambda + (2+\gamma)\hat{c}\lambda + \left(\frac{4+2\gamma}{\beta}\right)\hat{c}^2\lambda\right)|A_{\ell+1}| \\
&\quad -\big(1-(1+\gamma)\lambda+2\hat{c}\lambda\big)|I|. \qquad (6)
\end{aligned}$$

We claim that $|I| \le \mu|B_{\le\ell}|$ because no layer is collapsible. If $|I| > \mu|B_{\le\ell}|$, there would be some $\mathcal{I}_i$ in the disjoint partition $\bigcup_{i=1}^{\ell}\mathcal{I}_i$ of $\mathcal{I}$ such that $|I_i| > \mu|B_i|$. But then by definition $f_M[B_{\le i}, I] - f_M[B_{\le i-1}, I] = |I_i| > \mu|B_i|$, which implies that the layer L_i is collapsible, a contradiction to the assumption that no layer is collapsible. This proves our claim.

We have assumed to the contrary of Lemma 5 that $|\mathcal{A}_{\ell+1}| \le 2\mu|\mathcal{B}_{\le\ell}|$.

Plugging $\beta=\gamma^2$, $\mu=\gamma^3$, and the two inequalities above into (6) gives

$$\begin{aligned}
&\sum_{p\in P^+} y_i^* - \sum_{r\in R_f^+} z_r^* - \sum_{r\in R_t^+} z_r^* \\
&> \big(1-3\gamma^3\big)|B_{\le\ell}| - \big(1+\gamma-3\gamma^3-3\gamma^4\big)\lambda|B_{\le\ell}| - \big(3+\gamma+6\gamma^3+2\gamma^4\big)\hat{c}\lambda|B_{\le\ell}| \\
&\quad -\big(8\gamma+4\gamma^2\big)\hat{c}^2\lambda|\mathcal{B}_{\le\ell}|.
\end{aligned}$$

One can set $\gamma = \Theta(\delta)$ so that

$$\begin{aligned}
&\frac{1-3\gamma^3}{(1+\gamma-3\gamma^3-3\gamma^4)+(3+\gamma+6\gamma^3+2\gamma^4)\hat{c}+(8\gamma+4\gamma^2)\hat{c}^2} \\
&> \frac{1}{1+3c+O(\delta\hat{c}^2)} \\
&= \lambda,
\end{aligned}$$

which implies that the objective function value is positive.