Restricted Max-Min Allocation: Approximation and Integrality Gap

[An extended abstract of this full version is to appear in ICALP2019]

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 $\delta$ 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 $\delta$ can be any positive constant. The approximation ratio is with respect to the optimal value of the configuration LP, and the running time is $\text{poly}(m,n) \cdot n^{\text{poly}(1/\delta)}$ 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

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 [1]. 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 Sviridenko [4]. Bateni et al. [5] and Chakrabarty et al. [7] established a trade-off between the approximation ratio and the running time. For any $\delta > 0$, they can achieve an approximation ratio of $O(n^\delta)$ with $O(n^{1/\delta})$ running time.

© Siu-Wing Cheng and Yuchen Mao; licensed under Creative Commons License CC-BY

Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
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 \( P \neq \text{NP} \), 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 \( O\left(\frac{\log \log n}{\log \log \log n}\right) \)-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 \( [11] \), 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 \( \text{poly}(m, n) \cdot n^{\log(\frac{1}{\delta})} \) 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 \( [11, 9] \).

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 \( [8, 15] \).

**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 and defer the proof of Theorem 2 to Appendix C.

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 $\sum_{r \in D} v_r \geq T$. Let $C_p(T)$ 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 $\text{CLP}(T)$, associates a variable $x_{p,C}$ with each player $p$ and each configuration $C$ in $C_p(T)$. 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 $\text{CLP}(T)$ 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 $\lambda$ worth of resources for some $\lambda \in (0, 1)$. In particular, our approximation algorithm sets $\lambda = \frac{1}{4+\delta}$ where $\delta$ is a positive constant. For each resource $r \in R$, we call $r$ fat if $v_r \geq \lambda$, 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 $\sum_{r \in D} v_r \geq \lambda$.

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 $\sum_{r \in D} v_r \geq \lambda$. 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 $w \geq 0$, a thin edge $(p, D)$ is $w$-minimal if $\sum_{r \in D} v_r \geq w$ and $\sum_{r \in D'} v_r < w$ for any $D' \subset D$. For a $w$-minimal thin edge $(p, D)$, it is not hard to see that $w \leq \sum_{r \in D} v_r < w + \lambda$.

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 $\lambda$-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 in $\mathcal{E}$ and there is an alternating path \[14\] with respect to $M$ from $p_0$ to $q_0$. Let $\pi$ 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 slightly abuse the notion of alternating paths in the sense that when we 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 $b$ black $a$. 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 $\Pi$ be a feasible solution for $G_M(S, T)$. The paths in $\Pi$ originate from a subset of $S$, which we call the sources and denote as $src_\Pi$, and terminate in a subset of $T$, which we call the sinks and denote as $sink_\Pi$. We extend the $\oplus$ operation to $\Pi$. Viewing $\Pi$ 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 see the appendix \[A\] for more about this problem.
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 \(E\), and blocking edges as edges in \(E\) 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 lower 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 \(E\) 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. We will define a constant \(\mu\) to control the laziness.

**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 \(\lambda\)-minimal, and each blocking edge blocks exactly one addable edge. Using this strategy, in worst case, one may get a layer that has one addable edge being blocked by many blocking edges. After some of these blocking edges are released from \(E\), we may be left with a layer that has a single addable edge being 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 causes a new trouble: one may get a layer that has many addable edges being blocked by one blocking edge. Again, this breaks the geometric growth in the number of blocking edges. As a consequence, they have to introduce another strategy Greedy. They require every addable edge to be \(\frac{1}{2}\)-minimal. Such an addable edge contains much more resources than necessary. 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 \(\lambda\)-minimal and covers at most \(2\lambda\) 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 \(b\) to block more than one addable edge, but once \(b\) shares strictly more than \(\beta\lambda\) worth of resources with the addable edges blocked by it, we stop \(b\) from blocking more edges. We use greedy too. In our algorithm, addable edges in layers are \((1 + \gamma)\lambda\)-minimal for some constant \(\gamma\).

If we set \(\beta, \gamma, \mu\) 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 \(\beta, \gamma\) 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 \(\beta\) and tiny \(\gamma\) 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 $(A_i, B_i, d_i, z_i)$, where $A_i$ is a set of addable edges that we want to add to $\mathcal{E}$, and $B_i$ is a set of blocking edges that prevent us from doing so. The two numeric values $d_i$ and $z_i$ are 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 $\ell$ 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 $A_i$, we use $A_i$ to denote the set of players covered by $A_i$. Similarly, $B_i$ and $I$ denote the set of players covered by $B_i$ and $\mathcal{I}$, respectively. For $i \in [1, \ell]$, define $B_{\leq i} = \bigcup_{j=1}^{i} B_j$, $B_{< i} = \bigcup_{j=1}^{i-1} B_j$, and $A_{\leq i} = \bigcup_{j=1}^{i} A_j$.

For simplicity, we define the first layer $L_1$ to be $((\emptyset, \{(p_0, \emptyset)\}), 0, 0)$. That is, $A_1 = \emptyset$, $B_1 = \{(p_0, \emptyset)\}$, and $d_1 = z_1 = 0$.

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

- **Definition 3.** Let $\beta \geq 0$ be a constant to be specified later. A thin resource $r$ is **inactive** if (i) $r \in R(A_{\leq \ell} \cup B_{\ell})$, or (ii) $r \in R(A_{\ell+1} \cup \mathcal{I})$, or (iii) $r \in R_b$ for some $b \in B_{\ell+1}$ and $v[R_b \cap R(A_{\ell+1})] > \beta \lambda$. 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 $f_M(B_{\leq \ell}, A_{\ell+1} \cup I \cup \{p\}) = f_M(B_{\leq \ell}, A_{\ell+1} \cup I)$ + 1.

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

- **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 $v[D] \geq \lambda$. 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 $v[D \setminus R(\mathcal{E})] \geq \lambda$.

Recall that, for any $w > 0$, a thin edge $(p, D)$ is $w$-minimal if $v[D] \geq w$ and $v[D'] < w$ for any $D' \subseteq D$. Our algorithm considers two kinds of addable edges. The first kind is unblocked addable edges that are $\lambda$-minimal. It is easy to see that if an unblocked addable edge is $\lambda$-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 $\gamma$ is a constant to be specified later. Such edges will be added to $A_{\ell+1}$. Once a $(1 + \gamma)\lambda$-minimal addable
When some layer becomes collapsible, the algorithm enters the collapse phase. Let we use Definition 6. Put its proof to Appendix B.

Let $G$ be some disjoint subsets of $\mathcal{I}$. For every $i \in [1, \ell]$, $R(A_i) \cap R(A_j) = \emptyset$.

**Fact 2** Edges in $\mathcal{I}$ are mutually compatible, and they are also compatible with edges in $E$. For every $i \in [1, \ell]$, $R(A_i) \cap R(\mathcal{I}) = \emptyset$.

**Fact 3** $\{B_1, \ldots, B_\ell\}$ are disjoint subsets of $E$. Note that $B_1 = \{(p_1, \emptyset)\}$ does not share any resource with $B_i$ for $i \in [2, \ell]$.

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

edge $(p,D)$ becomes unblocked, we can easily extract a $\lambda$-minimal unblocked addable edge $(p,D')$ with $D' \subseteq D$.

Consider condition (iii) in Definition 3. Let $b$ be a blocking edge in $B_{\ell+1}$. When $R_b$ and $R(A_{\ell+1})$ share strictly more than $\beta \lambda$ worth of resources, all resources in $R_b$ become inactive. Any addable edge to be added to $A_{\ell+1}$ 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 $A_{\ell+1}$, we immediately add its blocking edges to $B_{\ell+1}$ in order to keep the inactive/active status of thin resources up-to-date.

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

1. Initialize $A_{\ell+1} = \emptyset$ and $B_{\ell+1} = \emptyset$.
2. While there is an unblocked addable edge that is $\lambda$-minimal, add it to $\mathcal{I}$.
3. While there is no addable edge $(p,D)$ that is $(1+\gamma)\lambda$-minimal
   a. 1. add $(p,D)$ to $A_{\ell+1}$. (Note that $(p,D)$ must be blocked; otherwise, we could extract from it a $\lambda$-minimal unblocked addable edge, which should be added to $\mathcal{I}$ in step 2)
   b. 2. add to $B_{\ell+1}$ the edges in $\mathcal{E}$ that block $(p,D)$.
4. Set $d_{\ell+1} := f_M(B_{\ell+1}, A_{\ell+1}, I_{\ell+1})$, $z_{\ell+1} := |A_{\ell+1}|$, and $L_{\ell+1} := (A_{\ell+1}, B_{\ell+1}, d_{\ell+1}, z_{\ell+1})$.
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 $(M, \mathcal{E}, \mathcal{I}, (L_1, \ldots, L_\ell))$ 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 $\mathcal{I}_{\leq i}$ and $I_{\leq i}$ 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 $i \in [1, \ell - 1]$, $f_M(B_{\leq i}, I_{\leq i}) = f_M(B_{\leq i}, I_i) = |I_{\leq i}|$. A solution $\Gamma$ for $G_M(B_{\leq \ell-1}, I)$ is a canonical solution with respect to the canonical decomposition $(\mathcal{I}_1, \ldots, \mathcal{I}_{\ell-1})$ if $\Gamma$ can be partitioned into disjoint subsets $(\Gamma_1, \ldots, \Gamma_{\ell-1})$ such that for every $i \in [1, \ell - 1]$, $\Gamma_i$ 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 [10]. We put its proof to Appendix B.
Theorem 4. Let $\ell$ be the number of layers in the stack. A canonical decomposition of $\mathcal{I}$ and a corresponding canonical solution for $G_{\ell}(B_{\leq \ell-1}, I)$ can be computed in $\text{poly}(\ell, m, n)$ 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 $\Gamma_i$ in $G_M$, so every edge in $\mathcal{I}_i$ can be used to release one blocking edge in $B_i$ from $\mathcal{E}$. A layer is collapsible if a certain fraction of its blocking edges can be released.

Definition 8. Let $\mu$ 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 $|I_i| = f_M(B_{\leq i}, I) - f_M(B_{\leq i-1}, I)$.

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.

\[ \text{Collapse}(\mathcal{M}, \mathcal{E}, \mathcal{I}, (L_1, \ldots, L_\ell)) \]

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 $G_M(B_{\leq \ell-1}, I)$. 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 $\mathcal{I} := \mathcal{I}_{\leq t-1}$.

3. Recall that $\text{src}_{\mathcal{I}_t} \subseteq B_t$ by Definition 6. Let $\mathcal{B}^T$ denote the set of edges in $B_t$ that are incident to players in $\text{src}_{\mathcal{I}_t}$. We use $\mathcal{I}_t$ and $\Gamma_t$ to release the edges in $\mathcal{B}^T$.

4. If $t \geq 2$, 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 $f_M(B_{\leq \ell-1}, I \cup \{p\}) = f_M(B_{\leq \ell-1}, I) + 1$, then extract a $\lambda$-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 $\ell$ is the number of layers in the stack. Lemma 9 gives a few invariants maintained by the algorithm. Lemma 10 states that when no layer is collapsible, $\mathcal{I}$ cannot have too many unblocked addable edges, nor can a layer lose too many addable edges. Lemmas 8 and 10 below have analogous versions in [1, 9] and can be proved similarly. We put their proofs to Appendix B.

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

Lemma 10. Let $(L_1, \ldots, L_\ell)$ be a stack of layers. If no layer is collapsible, then
Invariants maintained by the algorithm.

(ii) \(|I| \leq \mu|B_{\ell-1}|\), and

(iii) for all \(i \in [1, \ell]\), \(|A_i| \geq z_i - \mu|B_{i-1}|\).

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 \(L_i = (A_i, B_i, d_i, z_i)\) be an arbitrary layer in the stack. For each edge \(b \in B_i\), there is an edge \(a \in A_i\) such that \(v[R_b \cap R(A_i \setminus \{a\})] \leq \beta\).

Proof. Let \(b\) be an edge in \(B_i\). Sort the edges in \(A_i\) in chronological order of their additions into \(A_i\). Let \(a\) be the last edge in \(A_i\) that is blocked by \(b\). By our choice of \(a\), the edges in \(A_i\) 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 \(v[S] \leq \beta\). 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 \(L_i = (A_i, B_i, d_i, z_i)\) be an arbitrary layer in the stack. We have \(|A_i| < (1 + \frac{\gamma}{\beta})|B_i|\).

Proof. By Lemma 11 for each \(b \in B_i\), we can identify an edge \(a_b \in A_i\) so that \(v[R_b \cap R(A_i \setminus \{a_b\})] \leq \beta\). Let \(A_i^0 = \{a_b: b \in B_i\}\) be the set of edges identified. \(|A_i^0| \leq |B_i|\).

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

Finally we get \(|A_i| = |A_i^0| + |A_i^1| < |B_i| + \frac{2}{\gamma}|B_i| = (1 + \frac{2}{\gamma})|B_i|\).

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

Proof. Taking the sum of \(v[R_e \cap R(A_i)]\) over all edges \(e\) in \(B'_i\), we obtain \(v[R(B'_i) \cap R(A_i)] > \beta\lambda|B'_i|\). On the other hand, \(v[R(B'_i) \cap R(A_i)] \leq v[R(A_i)] < (2 + \gamma)\lambda|A_i|\). The last inequality is because that edges in \(A_i\) are \((1 + \gamma)\lambda\)-minimal. Combining the above two inequalities, we obtain \(\beta\lambda|B'_i| < (2 + \gamma)\lambda|A_i| \Rightarrow |B'_i| < \frac{2 + \gamma}{\beta\lambda}|A_i|\).
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 and Lemma 10(ii) together imply that as long as no layer is collapsible, every layer has lots of addable edges. 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 \((M,E,I,(L_1,\ldots,L_{\ell+1}))\) be the state of the algorithm immediately after the construction of \(L_{\ell+1}\). If no layer is collapsible, then \(z_{\ell+1} = |A_{\ell+1}| \geq 2\mu|B_{\leq \ell}|\).

Proof. We give a proof by contradiction. Suppose that \(z_{\ell+1} = |A_{\ell+1}| < 2\mu|B_{\leq \ell}|\). 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 \(A_{\ell+1}\) 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 \(\Pi\) be an optimal solution for \(G_M(B_{\leq \ell}, A_{\ell+1} \cup I)\). Note that \(M \oplus \Pi\) is a maximum matching of \(G\). Recall that \(G_{M\oplus\Pi}\) is a directed graph obtained from \(G\) 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 \(B_{\leq \ell} \setminus \text{src} \Pi\). Let \(R_f^+\) be the set of fat resources that can be reached in \(G_{M\oplus\Pi}\) from \(B_{\leq \ell} \setminus \text{src} \Pi\). 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 \(A_{\ell+1}\) (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 \(\{y_p^*, z_r^*\} \in P, \{z_r^*\} \in R\) as follows.

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

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

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

Suppose that Claim 15 holds. Then \(\{\alpha y_p^*, \alpha z_r^*\} \in P, \{\alpha z_r^*\} \in R\) is also a feasible solution for any \(\alpha > 0\). As \(\alpha\) goes to infinity, the objective function value goes to infinity, yielding the contradiction that we look for.

We defer the proof of Claim 15 to Appendix B and we give the proof of Claim 16 below.

Feasibility. We need to show that \(\forall p \in P, \forall C \in C_p(1), y_p^* \leq \sum_{r \in C} z_r^*\). 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 \(\sum_{r \in C} z_r^* \geq 1 - (1 + \gamma)\lambda\).

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 \(\pi\) in \(G_{M\oplus\Pi}\) from \(B_{\leq \ell} \setminus \text{src} \Pi\) to \(p\). If \(G_{M\oplus\Pi}\) has an edge \((p,r_f)\), we can reach \(r_f\) from \(B_{\leq \ell} \setminus \text{src} \Pi\) by following \(\pi\) and then
Lemma 13. Combining the above three parts gives
\( B \) than resources not used by \( B \) three parts: those covered by inequality is because are not charged. Therefore, Claim 15, for \( p \) must reach \( r_f \) first. Hence, we can follow \( \pi \) to reach \( r_f \) from \( B_{\leq t} \setminus \sigma_{\Pi} \), which implies \( r_f \in R^+_t \). In both cases, we have \( \sum_{r \in C} z^*_r \geq z^*_r \geq 1 - (1 + \gamma)\lambda \).

Case 2: \( C \) contains only thin resources. By Claim 15 \( p \) is still addable after we finish adding edges to \( A_{t+1} \). However, when we finish adding edges to \( A_{t+1} \), 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 \( v[C] \geq 1 \). At least \( 1 - (1 + \gamma)\lambda \) worth of thin resources in \( C \) are inactive. Since \( z^*_r = v_r \) for inactive thin resources, \( \sum_{r \in C} z^*_r \geq 1 - (1 + \gamma)\lambda \).

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 \( z^*_r \), \( \sum_{p \in P} y^*_p - \sum_{r \in R} z^*_r = \sum_{p \in P^+} y^*_p - \sum_{r \in R^+_t} z^*_r - \sum_{r \in R^+_f} z^*_r \).

First consider \( \sum_{p \in P^+} y^*_p - \sum_{r \in R^+_t} z^*_r \). Since \( y^*_p \) and \( z^*_r \) have the same value \( 1 - (1 + \gamma)\lambda \) for \( p \in P^+ \) and \( r \in R^+_t \), 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 \( G_{\leq \ell} \). Since \( r_f \) is reachable from \( B_{\leq t} \setminus \sigma_{\Pi} \), 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 \( G_{\leq \ell} \), so each of them is charged at most once. Note that players in \( B_{\leq t} \setminus \sigma_{\Pi} \) obviously belong to \( P^+ \) because they can be reached by themselves. Moreover, they have zero in-degree in \( G_{\leq \ell} \) as they are not matched by \( M \) \& \( \Pi \), so they are not charged. Therefore, \( |P^+| - |R^+_f| \geq |B_{\leq t} \setminus \sigma_{\Pi}| \geq |B_{\leq t}| - |A_{t+1}| - |I| \). The last inequality is because \( I \) is an optimal solution for \( G_M(B_{\leq t}, A_{t+1} \cup I) \).

In summary,
\[
\sum_{p \in P^+} y^*_p - \sum_{r \in R^+_t} z^*_r \geq (1 - (1 + \gamma)\lambda)(|B_{\leq t}| - |A_{t+1}| - |I|). \tag{1}
\]

Now consider \( \sum_{r \in R^+_f} z^*_r \). By definition of inactive resources, \( R^+_f \) can be divided into three parts: those covered by \( A_{\leq t} \cup B_{\leq t} \), those covered by \( A_{t+1} \cup I \), and those covered by \( B'_{t+1} = \{ e \in B_{t+1} : v[R_e \cap R(A_{t+1})] > \beta \lambda \} \). We handle these three parts separately.

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

Combining (1) and (2) gives that
\[
\sum_{p \in P^+} y^*_p - \sum_{r \in R^+_t} z^*_r - \sum_{r \in R^+_f} z^*_r > \left( 1 - \left( 4 + \gamma + \frac{\beta}{\gamma} \right) \lambda \right) |B_{\leq t}| - \left( 1 + \left( 1 + \frac{4 + 2\gamma}{\beta} \right) \lambda \right) |A_{t+1}| - (1 - (1 + \gamma)\lambda) |I|.
\]
By the contrapositive assumption at the beginning of the proof of Lemma 14, \(|A_{i+1}| < 2\mu|B_{\leq i}|\). Moreover, since no layer is collapsible, by Lemma 10(ii), \(|I| \leq \mu|B_{\leq i}|\). Substituting these two inequalities into the above gives
\[
\sum_{p \in P^+} y_p^* - \sum_{r \in R_i^+} z_r^* - \sum_{r \in R_i^+} z_r^* > \left((1 - 3\mu) - \left(4 + \gamma + \frac{\beta}{\gamma} + 3\mu + \frac{(8 + 4\gamma)\mu}{\beta} - \gamma\mu\right)\lambda\right)|B_{\leq i}|.
\]
Let \(\beta = \gamma^2\) and \(\mu = \gamma^3\). We have
\[
\sum_{p \in P^+} y_p^* - \sum_{r \in R_i^+} z_r^* - \sum_{r \in R_i^+} z_r^* > \left((1 - 3\gamma^3) - (4 + 10\gamma + 4\gamma^2 + 3\gamma^3 - \gamma^4)\lambda\right)|B_{\leq i}|.
\]
Recall that \(\lambda = \frac{1}{1+\delta}\) 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 \(\gamma\) that makes \(\frac{1-3\gamma^3}{4+10\gamma+4\gamma^2+3\gamma^3-\gamma^4} > \frac{1}{4}\), thereby proving
\[
\sum_{p \in P} y_p^* - \sum_{r \in R} z_r^* = \sum_{p \in P^+} y_p^* - \sum_{r \in R_i^+} z_r^* - \sum_{r \in R_i^+} z_r^* > 0.
\]
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 \(i \in [1, \ell-1]\), \(|B_{i+1}| \geq \frac{\gamma^3}{\ell+1}|B_{\leq i}|\).

**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 \(B_{\leq i}\). Also the constant \(z_{i+1}\) is unchanged. By Lemma 14, \(z_{i+1} \geq 2\mu|B_{\leq i}|\). Although addable edges may be removed from \(L_{i+1}\) during this period, there are still lots of addable edges left. By Lemma 10(ii), \(|A_{i+1}| \geq z_{i+1} - \mu|B_{\leq i}| \geq \mu|B_{\leq i}|\). By Lemma 12, \(|B_{i+1}| \geq \frac{1}{(\ell+1)\gamma}|A_{i+1}| \geq \frac{\mu}{(\ell+1)\gamma}|B_{\leq i}|\). Recall that we set \(\beta = \gamma^2\) and \(\mu = \gamma^3\) in the proof of Claim 16. Replacing \(\beta\) by \(\gamma^2\) and \(\mu\) by \(\gamma^3\) proves the lemma.

**Lemma 18.** In \(\text{poly}(m, n) \cdot n^{\text{poly}(\frac{1}{4})}\) 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 [11, 12]. 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 \(s_i = \log_{1/(1-\mu)} \frac{|B_i|}{\mu} \). 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 \(U = \log n \cdot O\left(\frac{1}{\mu} \log \frac{1}{\gamma}\right)\). 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 \(n^{O\left(\frac{1}{\mu} \log \frac{1}{\gamma}\right)}\) on the number of distinct signatures. Recall that \(u = \gamma^3\), \(h = \frac{\gamma^3}{1+\gamma}\), and \(\frac{1}{\gamma} = O\left(\frac{1}{\delta}\right)\). As a consequence, the number of non-collapsible states ever reached by the algorithm is bounded by \(n^{\text{poly}(\frac{1}{4})}\). Between two consecutive non-collapsible states, there is one \textsc{build} and at most \(\log_{1+\mu} n\) \textsc{collapse}, which take \(\text{poly}(m, n) \cdot n^{\text{poly}(\frac{1}{4})}\) time in total. The total running time is thus \(\text{poly}(m, n) \cdot n^{\text{poly}(\frac{1}{4})}\).
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.
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. Rothvoess, and Y. Zhang. A tale of santa claus, hypergraphs and matroids. *CoRR*, abs/1807.07189, 2018.
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 bipartite 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.
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. Polack 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.
Restricted Max-Min Allocation

A Node-disjoint Alternating Paths

Let $M$ be a maximum matching of $G$. Let $S$ be a subset of the players not matched by $M$. Let $T$ be a subset of the players. Recall that the problem $G_M(S,T)$ seeks the largest set of node-disjoint paths in $G_M$ from $S$ to $T$, and $f_M(S,T)$ denotes the maximum number of such paths. One may already observe that this problem can be easily solved after being reduced to a maximum flow problem. However, for the sake of future analysis, we understand it from the perspective of matchings. The following lemma is analogous to the well-known sufficient and necessary condition for maximum flow.

Lemma 19. Let $M$ be a maximum matching of $G$. Let $S$ be a subset of the players not matched by $M$. Let $T$ be a subset of $P$. Let $\Pi$ be a feasible solution for $G_M(S,T)$. $\Pi$ is an optimal solution for $G_M(S,T)$ if and only if there is no path in $G_{M\oplus \Pi}$ from $S \setminus src_\Pi$ to $T \setminus sink_\Pi$. (Recall that $M \oplus \Pi$ is a maximum matching, so $G_{M\oplus \Pi}$ is well-defined.)

Proof of if part. Suppose that $\Pi$ is not optimal. Recall that every path in $\Pi$ is an alternating path with respect to $M$. We show that, with respect to the maximum matching $M \oplus \Pi$, there is an alternating path from $S \setminus src_\Pi$ to $T \setminus sink_\Pi$. This alternating path is a path in $G_{M\oplus \Pi}$ from $S \setminus src_\Pi$ to $T \setminus sink_\Pi$. First we observe that $\Pi$ cannot use any player in $S \setminus src_\Pi$ because every of these players is not matched by $M$ and has in-degree 0 in $G_M$.

If there is a player $p \in (S \setminus src_\Pi)$ that belongs to $T$, we claim that $p$ itself forms our target alternating path. $p$ is not matched by $M$ and is not used by alternating paths in $\Pi$, so $p$ remains unmatched in $M \oplus \Pi$ and $p \notin sink_\Pi$. The player $p$ itself forms a trivial alternating path with respect to $M \oplus \Pi$, and it is from $S \setminus src_\Pi$ to $T \setminus sink_\Pi$.

Assume that no player in $S \setminus src_\Pi$ belongs to $T$. Let $\Pi^*$ be an optimal solution for $G_M(S,T)$. Clearly, $|\Pi^*| > |\Pi|$. Consider $src_\Pi \setminus src_\Pi$. It is non-empty as $|\Pi^*| > |\Pi|$. No node in $src_\Pi \setminus src_\Pi$ belongs to $T$ as $src_\Pi \setminus src_\Pi$ is a subset of $S \setminus src_\Pi$. The alternating paths in $\Pi^*$ that start with nodes in $src_\Pi \setminus src_\Pi$ must be non-trivial as these nodes are not in $T$. Now consider the edge set

$$E_{\oplus} = (M \oplus \Pi) \oplus (M \oplus \Pi^*) = \Pi \oplus \Pi^*.$$ 

$E_{\oplus}$ is the symmetric difference of two maximum matchings, so it consists of some non-trivial even-length paths and possibly some cycles [13]. All the paths in $E_{\oplus}$ are alternating paths with respect to $M \oplus \Pi$, and they are node-disjoint. We claim that

(i) every player in $src_\Pi \setminus src_\Pi$ is an endpoint of some non-trivial path in $E_{\oplus}$, and

(ii) if a non-trivial path has an endpoint in $src^*_\Pi \setminus src_\Pi$, then the other endpoint must be in either $src_\Pi \setminus src^*_\Pi$ or $sink^*_\Pi \setminus sink_\Pi$.

Suppose that our claim holds. Because $|\Pi^*| > |\Pi|$, we have $|src_\Pi \setminus src_\Pi| > |src_\Pi \setminus src^*_\Pi|$. By pigeonhole principle, there must be at least one path in $E_{\oplus}$ that goes from $src_\Pi \setminus src_\Pi$ to $sink^*_\Pi \setminus sink_\Pi$, and this is our target alternating path with respect to $M \oplus \Pi$.

To see why our claim holds, first consider the degrees of nodes in $E_{\oplus}$. Viewing $E_{\oplus}$ as $\Pi \oplus \Pi^*$, we observe that

- every player in $src_\Pi \setminus src_\Pi$ has degree 1 in $E_{\oplus}$, because they have degree 1 in $\Pi^*$ and degree 0 in $\Pi$;

- every player in $src_\Pi \setminus src^*_\Pi$, $sink^*_\Pi \setminus sink_\Pi$, and $sink_\Pi \setminus sink^*_\Pi$ may have odd degree in $E_{\oplus}$;

- any other players or fat resources must have even degree in $E_{\oplus}$, and hence they cannot be endpoints of paths in $E_{\oplus}$. 

Our claim (i) simply follows by observation (i). To prove claim (ii), it suffices to show that if a path has an endpoint in srcΠ \ srcΠ, then its other endpoint cannot be in srcΠ \ srcΠ or sinkΠ \ sinkΠ. Recall that all the paths in $E_\oplus$ are alternating paths with respect to $M \oplus \Pi$. Players in $(src_{\Pi^+} \setminus src_{\Pi}) \subseteq (S \setminus src_{\Pi})$ are not matched by $M \oplus \Pi$, so they cannot be reached by a non-trivial alternating path with respect to $M \oplus \Pi$ from another unmatched node in srcΠ \ srcΠ. Similarly, players in sinkΠ \ sinkΠ are not matched by $M \oplus \Pi$, so cannot be reached by a non-trivial alternating path from another unmatched node in srcΠ \ srcΠ. This completes the proof.

Proof of only-if part. Let $\pi$ be a path in $G_{M \oplus \Pi}$ from $S \setminus src_{\Pi}$ to $T \setminus sink_{\Pi}$. In other words, $\pi$ is an alternating path with respect to $M \oplus \Pi$ from $S \setminus src_{\Pi}$ to $T \setminus sink_{\Pi}$. We show how to update $\Pi$ to a large feasible solution for $G_M(S,T)$. If $\pi$ is a trivial alternating path consisting of a player $p$ alone, then $p$ must be in $S \cap T$, and hence is not matched by $M$. Also $p$ must be node-disjoint from $\Pi$. Therefore, $p$ itself forms a (trivial) alternating path with respect to $M$, and adding this path to $\Pi$ yields a larger feasible solution for $G_M(S,T)$.

Suppose that $\pi$ is non-trivial. Let src$\pi$ be the starting point of $\pi$, and let sink$\pi$ be the ending point of $\pi$. Let $\Pi^0$ be the set of trivial alternating paths in $\Pi$, and let $\Pi^+$ be the set of non-trivial alternating paths in $\Pi$. Note that players involved in $\Pi^0$ remain unmatched in $M \oplus \Pi$ and have in-degree 0 in $G_{M \oplus \Pi}$, so they can not be touched by the non-trivial path $\pi$ which starts at another player. $\Pi^0$ and $\pi$ are node-disjoint. Consider the edge set

$$E_\oplus = M \oplus (M \oplus \Pi \oplus \pi) = \Pi \oplus \pi = \Pi^+ \oplus \pi.$$ 

$E_\oplus$ is the symmetric difference of two maximum matchings, so it consists of some non-trivial even-length paths and possibly some cycles. All the paths in $E_\oplus$ are alternating paths with respect to $M$, and they are node-disjoint. There are two cases: (i) $sink_{\pi} \notin src_{\Pi^+}$, and (ii) $sink_{\pi} \in src_{\Pi^+}$. In case (i), similar to that in the proof of if part, if viewing $E_\oplus$ as the symmetric difference of two sets of paths, players in $src_{\Pi^+} \cup \{src_{\pi}\}$ and $sink_{\Pi} \cup \{sink_{\pi}\}$ have degree 1 in $E_\oplus$, so they are endpoints of paths in $E_\oplus$. All the other players and fat resources have even degree. As a result, there are exactly $|src_{\Pi^+}| + 1 = |\Pi^+| + 1$ non-trivial paths in $E_\oplus$. These paths, together with trivial paths in $\Pi^0$, form a larger feasible solution for $G_M(S,T)$. In case (ii), one player $p^* = sink_{\pi} \in src_{\Pi^+}$ has degree 0 or 2 in $E_\oplus$, so $p^*$ cannot be an endpoint of any path in $E_\oplus$. As a consequence, $E_\oplus$ contains only $|\Pi^+|$ non-trivial paths. We claim that the degree of $p^*$ in $E_\oplus$ must be 0. Then $p^*$ can be regarded as an additional trivial alternating path. This trivial alternating path, together with the $|\Pi^+|$ non-trivial paths in $E_\oplus$ and the trivial paths in $\Pi^0$, form a larger feasible solution for $G_M(S,T)$.

To see why $p^*$ cannot have degree 2 in $E_\oplus$, we should interpret $E_\oplus$ as the symmetric difference of two maximum matchings. A node with degree 2 in $E_\oplus$ must be matched in both matchings. However, since $p^* \in src_{\Pi^+} \subseteq S$, $p^*$ is not matched by $M$. Therefore, $p^*$ cannot have degree 2 in $E_\oplus$. This completes the proof.

**Corollary 20.** Let $M$ be a maximum matching of $G$. Let $S$ be a subset of the players not matched by $M$. Let $T$ be a subset of $P$. Let $\Pi$ be a feasible solution for $G_M(S,T)$. If $\Pi$ is not optimal, there exists a path $\pi$ in $G_{M \oplus \Pi}$ from $S \setminus src_{\Pi}$ to $T \setminus sink_{\Pi}$. In polynomial time, we can augment $\Pi$ to a larger solution $\Pi'$ for $G_M(S,T)$ such that $|\Pi'| = |\Pi| + 1$, $src_{\Pi'} = src_{\Pi} \cup \{src_{\pi}\}$, and $sink_{\Pi'} = sink_{\Pi} \cup \{sink_{\pi}\}$. Moreover, the collection of nodes used by $\Pi'$ is a subset of those used by $\Pi$ and $\pi$. 

The above proofs immediately imply the following.

$\blacktriangleright$
Corollary 21. Let $M$ be a maximum matching of $G$. Let $S$ be a subset of the players not matched by $M$. Let $T$ be a subset of $P$. An optimal solution for $G_M(S,T)$ can be computed in polynomial time.

The following lemma states that if we get one more node-disjoint path by including $p$ in $T$, so can we by including $p$ to any subset of $T$.

Lemma 22. Let $M$ be a maximum matching of $G$. Let $S$ be any subset of the unmatched players. Let $T$ be any subset of $P$. Let $p$ be an arbitrary player in $P$. If $f_M(S,T \cup \{p\}) = f_M(S,T) + 1$, then for every $T' \subseteq T$, $f_M(S,T' \cup \{p\}) = f_M(S,T') + 1$.

Proof. Suppose that $f_M(S,T \cup \{p\}) = f_M(S,T) + 1$. Obviously, $p \notin T$. Let $T'$ be an arbitrary subset of $T$. Let $\Pi_1$ be an optimal solution for $G_M(S,T')$. $p \notin \text{sink}_{\Pi_1}$. Note that $\Pi_1$ is also a feasible solution for $G_M(S,T \cup \{p\})$. Let $\Pi_2$ be an optimal solution for $G_M(S,T \cup \{p\})$ obtained by augmenting $\Pi_1$ (using Corollary 20). Then, $\text{sink}_{\Pi_1} \subseteq \text{sink}_{\Pi_2}$. If $p \in \text{sink}_{\Pi_2}$, then $(\text{sink}_{\Pi_1} \cup \{p\}) \subseteq \text{sink}_{\Pi_2}$, implying that there are $|\Pi_1| + 1 = f_M(S,T') + 1$ node-disjoint paths from $S$ to $T' \cup \{p\}$, and thus establishing the lemma. If $p \notin \text{sink}_{\Pi_2}$, then $\Pi_2$ is a feasible solution for $G_M(S,T)$. But then $f_M(S,T \cup \{p\}) = |\Pi_2| \leq f_M(S,T)$, a contradiction to the assumption.

B Omitted Proofs in the Analysis of the Approximation Algorithm

B.1 Proof of Lemma 7

We first compute an optimal solution $\Pi_i$ for $G_M(B_1,I)$. For $j = 2, \ldots, \ell - 1$, we compute an optimal solution $\Pi_j$ for $G_M(B_{\leq j}, I)$ by successively augmenting $\Pi_{j-1}$ using Corollary 20. Augmentation ensures that $\text{src}_{\Pi_{j-1}} \subseteq \text{src}_{\Pi_j}$. Therefore, we inductively maintain the property that for all $i \in [1, j]$, $\Pi_j$ contains $|\Pi_i| = f_M(B_{\leq i}, I)$ node-disjoint paths from $B_{\leq i}$ to $I$. In the end, we obtain $\Pi_{\ell-1}$. By invariant 1 in Table 2 $\text{sink}_{\Pi_{\ell-1}} = I$. We obtain the canonical decomposition and the canonical solution as follows: for every $i \in [1, \ell - 1]$, let $\Gamma_i$ be the subset of paths in $\Pi_{\ell-1}$ starting at $B_i$, and let $I_i = \text{sink}_{\Gamma_i}$, let $I_i$ be the subset of edges in $I$ that cover the players in $I_i$.

B.2 Proof of Lemma 9

We break the proof into two parts, handling BUILD and COLLAPSE separately.

Lemma 23. BUILD maintains the invariants in Table 2.

Proof. Suppose that the invariants hold before BUILD. Let $L_{\ell+1}$ be the layer newly constructed by BUILD. We show that the invariants hold after the construction of $L_{\ell+1}$.

Consider invariant 1 and the moment when BUILD is about to add an edge $(p, D)$ to $I$. By definition of addable players, we have

$$f_M(B_{\leq \ell}, A_{\ell+1} \cup I \cup \{p\}) = f_M(B_{\leq \ell}, A_{\ell+1} \cup I) + 1.$$  

Lemma 22 implies that

$$f_M(B_{\leq \ell}, I \cup \{p\}) = f_M(B_{\leq \ell}, I) + 1.$$  

Hence, whenever we add an edge to $I$, both of the left hand side and right hand side of invariant 1 increase by 1. Other steps obviously have no effect on this invariant.
Consider invariant 2. BUILD does not change any old layer. Also, BUILD does not delete any edge from \(\mathcal{I}\). Therefore, for all \(i \in [1, \ell - 1]\), \(f_M(B_{\leq i}, A_{t+1} \cup I)\) cannot decrease and the inequality still holds. By construction, BUILD sets \(d_{t+1} := f_M(B_{\leq t}, A_{t+1} \cup I)\). So invariant 2 is preserved.

Consider invariant 3 and 4. For \(i \in [1, \ell]\), the inequalities continue to hold because BUILD does not change any old layer. It remains to show that the inequalities hold for \(i = \ell + 1\). By definition, \(z_{\ell+1} = |A_{\ell+1}|\), so invariant 3 holds. At the beginning of BUILD, \(A_{\ell+1} = \emptyset\). Whenever we add an edge \((p, D)\) to \(A_{t+1}\), the value of \(f_M(B_{\leq t}, A_{t+1} \cup I)\) increases by 1 because \(p\) is an addable player. Therefore, at the end, the value of \(f_M(B_{\leq t}, A_{t+1} \cup I)\) is at least \(|A_{t+1}|\).

\[d_{t+1} = f_M(B_{\leq t}, A_{t+1} \cup I) \geq |A_{t+1}| = z_{t+1}.
\]

So Invariant 4 is preserved.

To prove that COLLAPSE preserves the invariants in Table[2] we need the following result.

\textbf{Lemma 24.} Let \((\mathcal{I}_1, \ldots, \mathcal{I}_{\ell-1})\) be a canonical decomposition of \(\mathcal{I}\). Let \(\mathcal{I}_1 \cup \cdots \cup \mathcal{I}_{\ell-1}\) be a corresponding canonical solution for \(G_M(B_{\leq \ell-1}, I)\). For every \(i \in [1, \ell - 2]\), \(G_M(B_{\leq i}, A_{t+1} \cup I_{\leq i})\) and \(G_M(B_{\leq i}, A_{t+1} \cup I)\) share a common optimal solution that is node-disjoint from \(\mathcal{I}_i \cup \cdots \cup \mathcal{I}_{\ell-1}\).

\textbf{Proof.} Fix some \(i \in [1, \ell - 2]\). We compute an optimal solution for \(G_M(B_{\leq i}, A_{t+1} \cup I)\) as follows. Let \(\Pi_0 = \mathcal{I}_1 \cup \cdots \cup \mathcal{I}_i\). \(\Pi_0\) is a feasible solution for \(G_M(B_{\leq i}, A_{t+1} \cup I)\) as paths in it go from \(B_{\leq i}\) to \(I_{\leq i}\). We iteratively augment \(\Pi_0\) using Corollary[20] \(\Pi_0, \Pi_1, \ldots, \Pi_k\) be the intermediate solutions obtained during the repeated augmentations, where \(\Pi_k\) is the optimal solution for \(G_M(B_{\leq i}, A_{t+1} \cup I)\) we obtain at the end.

We show that \(\Pi_k\) is node-disjoint from \(\mathcal{I}_{>i} = \mathcal{I}_{i+1} \cup \cdots \cup \mathcal{I}_{\ell-1}\). The set of sinks of \(\mathcal{I}_{>i}\) is \(I_{i+1} \cup \cdots \cup I_{\ell-1}\), which is denote as \(I_{>i}\).

For \(j \in [0, k - 1]\), let \(\pi_j\) be the path in \(G_{M*\Pi_j}\) that is used to augment \(\Pi_j\) to \(\Pi_{j+1}\). We claim that every \(\pi_j\) is node-disjoint from \(\mathcal{I}_{>i}\). Suppose not. Let \(\pi_{j^*}\) be the first such path that shares some node with \(\mathcal{I}_{>i}\). Since \(\Pi_0\) and every \(\pi_j\) with \(j < j^*\) are node-disjoint from \(\mathcal{I}_{>i}\), \(\Pi_{j^*}\) must be node-disjoint from \(\mathcal{I}_{>i}\). Hence, \(\mathcal{I}_{>i}\) remains to be paths in \(G_{M*\Pi_{j^*}}\). Since \(\pi_{j^*}\) share some node with \(\mathcal{I}_{>i}\), we can construct a path \(\pi^*\) in \(G_{M*\Pi_{j^*}}\), as follows: start with \(s_{\pi_{j^*}}\), follow \(\pi_{j^*}\), switch at the first common node of \(\pi_{j^*}\) and \(\mathcal{I}_{>i}\), and follow a path in \(\mathcal{I}_{>i}\) to some player in \(\mathcal{I}_{>i}\). If we augment \(\Pi_{j^*}\) using \(\pi^*\), we will get a feasible solution \(\Pi^*\) for \(G_M(B_{\leq i}, A_{t+1} \cup I)\). All the players in \(I_{\leq i} = \text{sink}_{\Pi_{j^*}}\) are sinks of \(\Pi^*\). The player \(p^* \in I_{>i}\) is also a sink of \(\Pi^*\). Consequently, there are \(|I_{\leq i}| + 1\) node-disjoint paths in \(\Pi^*\) from \(B_{\leq i}\) to \(I\), which implies that \(f_M(B_{\leq i}, I) \geq |I_{\leq i}| + 1\). This contradicts the definition of canonical decomposition, thereby establishing our claim.

By our claim, all \(\pi_j\)'s are nodes-disjoint from \(\mathcal{I}_{>i}\). So the repeated augmentations to produce \(\Pi_k\) do not generate any node-sharing with \(\mathcal{I}_{>i}\). That is, \(\Pi_k\) is node-disjoint from \(\mathcal{I}_{>i}\).

Since \(\Pi_k\) is node-disjoint from \(\mathcal{I}_{>i}\), no player from \(\mathcal{I}_{>i}\) can be a sink of \(\Pi_k\). Therefore, \(\Pi_k\) is also an optimal solution for \(G_M(B_{\leq i}, A_{t+1} \cup I_{\leq i})\).
Restricted Max-Min Allocation

\( \Gamma_{\leq t-1} \) are node-disjoint paths in \( G_M \) from \( B_{\leq t-1} \) to every player in \( I_{\leq t-1} \). Obviously steps 2 and 3 do not affect \( \Gamma_{\leq t-1} \). Step 4 updates \( M \), and may affect the paths in \( G_M \). The paths in \( \Gamma_{\leq t-1} \) are node-disjoint from those in \( \Gamma_t \), so after updating \( M \) using \( \Gamma_t \), the paths in \( \Gamma_{\leq t-1} \) remain to be alternating paths with respect to the updated \( M \). Hence, after step 3, \( \Gamma_{\leq t-1} \) is still a set of paths in \( G_M \) from \( B_{\leq t-1} \) to every player in \( I_{\leq t-1} \). It certifies that

\[
f_M(B_{\leq t-1}, I_{\leq t-1}) = |I_{\leq t-1}|.
\]

Consider invariant 2. Since \( L_t \) is going to be the topmost layer, we only need to show that these inequalities hold for \( i \in [1, t-1] \) after steps 2, 4. Fix some \( i \in [1, t-1] \). By Lemma 24, \( G_M(B_{\leq i-1}, A_{i+1} \cup I_{\leq i}) \) and \( G_M(B_{\leq i}, A_i \cup I) \) share a common optimal solution that is node-disjoint from \( \Gamma_t \). Let \( \Pi^* \) be that optimal solution. By inductive hypothesis, we have \( |\Pi^*| \geq d_{i+1} \). \( \Pi^* \) is not affected by step 2 because all its sinks belong to \( A_{\leq i+1} \cup I_{\leq i} \). Similar to the proof of invariant 2, since \( \Pi^* \) is node-disjoint from \( \Gamma_t \), it is a set of paths in \( G_M \) after step 3 updates \( M \) using \( \Gamma_t \). Thus, \( \Pi^* \) certifies that

\[
f_M(B_{\leq i}, A_{i+1} \cup I) \geq d_{i+1}
\]

at the end of step 3. In step 4, the removal of edges from \( A_i \) may decrease the value of \( f_M(B_{\leq t-1}, A_i \cup I) \), but it does not change \( f_M(B_{\leq i}, A_{i+1} \cup I) \) for \( i \in [1, t-2] \). Suppose that \( f_M(B_{\leq t-1}, A_i \cup I) \) decreases after removing an edge \((p, D)\) from \( A_i \), that is,

\[
f_M(B_{\leq t-1}, A_i \cup I) = f_M(B_{\leq t-1}, (A_i \setminus \{p\}) \cup I) + 1.
\]

Then when we reach step 4.2, Lemma 22 implies that

\[
f_M(B_{\leq t-1}, I \cup \{p\}) = f_M(B_{\leq t-1}, I) + 1,
\]

so step 4.2 will add \( p \) to \( I \). Afterwards, \( f_M(B_{\leq t-1}, A_i \cup I) \) returns to its value prior to the removal of \((p, D)\) from \( A_i \). As a result, invariant 2 holds after step 4.

Invariant 3 and 4 hold because \textsc{collapse} neither grows any \( A_i \) nor changes any \( d_i \) and \( z_i \) for \( i \in [1, t] \). \hfill \( \blacksquare \)

### B.3 Proof of Lemma 10

Consider (i). By invariant 1 in Table 2, \( f_M(B_{\leq t-1}, I) = |I| \). Hence, in the canonical decomposition of \( I \), we have \( I_{\leq t-1} = I \). If \( |I_{\leq t-1}| = |I| > \mu|B_{\leq t-1}| \), by the pigeonhole principle, there exists an index \( i \in [1, t-1] \) such that \( |I_i| > \mu|B_{\leq i-1}| \). But then layer \( L_i \) is collapsible, a contradiction.

Consider (ii). Assume to the contrary that there exists \( i \in [1, t] \) such that \( |A_i| < z_i - \mu|B_{\leq i-1}| \). Equivalently, \( \frac{z_i}{|A_i| + \mu|B_{\leq i-1}|} \geq \frac{z_{i+1}}{d_{i+1}} \). By invariants 2 and 4 in Table 2, \( f_M(B_{\leq i-1}, A_i \cup I) \geq d_i \geq z_i > |A_i| + \mu|B_{\leq i-1}| \). Therefore, any optimal solution for \( G_M(B_{\leq i-1}, A_i \cup I) \) contains at least \( \mu|B_{\leq i-1}| + 1 \) node-disjoint paths from \( B_{\leq i-1} \) to \( I \). It follows that \( f_M(B_{\leq i-1}, I) \geq \mu|B_{\leq i-1}| + 1 \). By the definition of canonical decomposition, \( |I_{\leq i-1}| = f_M(B_{\leq i-1}, I) \geq \mu|B_{\leq i-1}| + 1 \). By the pigeonhole principle, there exists some \( j \in [1, i-1] \) such that \( |I_j| > \mu|B_j| \). But then layer \( L_j \) is collapsible, a contradiction.

### B.4 Proof of Claim 15

Consider (i). Recall that \( \Pi \) is an optimal solution for \( G_M(B_{\leq t}, A_{t+1} \cup I) \). The players in \( B_{\leq t} \setminus \text{src}_{\Pi} \) cannot be sinks of \( \Pi \) because they have in-degree 0 in \( G_M \) and cannot be reached.
by paths not starting with them. In \( G_{M \oplus \Pi} \), the players in \( \text{sink}_I \) have in-degree 0, and hence cannot be reached from the other players. In particular, they cannot be reached from the players in \( B_{\leq \ell} \setminus \text{src}_I \). Let \( p \) be an arbitrary player in \( P \). By definition, \( p \) is reachable in \( G_{M \oplus \Pi} \) from \( B_{\leq \ell} \setminus \text{src}_I \), so \( p \notin \text{sink}_I \). Note that \( \Pi \) is a feasible solution for \( G_M(B_{\leq \ell}, A_{\ell+1} \cup I \cup \{p\}) \), and \( \Pi \) cannot be optimal by Lemma 19 because there is a path in \( G_{M \oplus \Pi} \) from \( B_{\leq \ell} \setminus \text{src}_I \) to \( p \notin \text{sink}_I \). In other words,

\[
f_M(B_{\leq \ell}, A_{\ell+1} \cup I \cup \{p\}) > f_M(B_{\leq \ell}, A_{\ell+1} \cup I).
\]

So \( p \) is addable.

(ii) is easy to prove because in \( G_{M \oplus \Pi} \), a player has degree 0 if it is not matched by \( M \oplus \Pi \), and degree 1 otherwise.

To prove (iii), it suffices to show that every fat resource in \( R^f_j \) must be matched by \( M \oplus \Pi \). Let \( r_f \) be an arbitrary fat resource in \( R^f_j \). By definition, \( r_f \) is reachable in \( G_{M \oplus \Pi} \) from some player \( p^* \in B_{\leq \ell} \setminus \text{src}_I \). Note that \( p^* \) is not matched by \( M \oplus \Pi \). If \( r_f \) is not matched neither, then the path between \( p^* \) and \( r_f \) would be an alternating path from an unmatched player to an unmatched fat resource, which can be used to increase the size of \( M \oplus \Pi \) [14]. But \( M \oplus \Pi \) is already a maximum matching of \( G \), contradiction. Hence, \( r_f \) must be matched. This completes the proof.

\section{Integrality Gap: Proof of Theorem 2}

We present a tighter analysis for the local search algorithm in [2]. We shows that an allocation can be computed such that every player receives at least \( \lambda = \frac{22}{9} \) worth of resources. Recall that the optimal value \( T^* \) of the configuration LP is assumed to be 1. This proves that the integrality gap is at most \( \frac{22}{9} \approx 3.808 \). The computation time is not known to be polynomial though.

\subsection{The local search algorithm}

We present the algorithm in a way slightly different from that in [2], in order to show the similarity between this algorithm and the approximation algorithm in Section 3. Let \( M \) and \( \mathcal{E} \) be the current maximum matching of \( G \) and the current set of thin edges maintained by the algorithm, respectively. Let \( p_0 \) be a player not yet covered by \( M \cup \mathcal{E} \).

The algorithm maintains a stack of tuples \( \Sigma = [(a_1, B_1), (a_2, B_2), \ldots] \), where \( a_i \) is an addable edge and \( B_i \) is the set of blocking edges of \( a_i \). We will give the definitions of addable edges and blocking edges soon. For \( i < j \), \( (a_i, B_i) \) is pushed into \( \Sigma \) before \( (a_j, B_j) \). For simplicity, we define \( a_1 = \text{null} \) and \( B_1 = \{(p_0, \emptyset)\} \). We use \( \ell \) to denote the length of \( \Sigma \). We use \( R(\Sigma) \) to denote the set of thin resources covered by \( \{a_1, \ldots, a_\ell\} \cup B_1 \cdots \cup B_\ell \). For \( i \in [1, \ell] \), \( B_i \) denotes the set of players covered by \( B_i \), \( B_{\leq i} \) denotes \( \bigcup_{j=1}^i B_j \), and \( B_{\leq i} \) denotes \( \bigcup_{j=1}^i B_j \).

The stack \( \Sigma \) is built inductively. Initially, \( \Sigma = [(a_1, B_1)] \). Consider the construction of \( (a_{\ell+1}, B_{\ell+1}) \). Recall that \( G_M \) is the directed graph obtained from \( G \) by orienting edges in \( G \) from \( r_f \) to \( p \) if \( \{p, r_f\} \in M \) and \( p \) to \( r_f \) otherwise. Also recall that, for any \( w > 0 \), a thin edge \( (p, D) \) is \( w \)-minimal if \( v[D] \geq w \) and \( v[D'] < w \) for any \( D' \subset D \).

\begin{definition}
\textbf{Definition 26.} Given the current partial allocation \( M \cup \mathcal{E} \) and the current stack \( \Sigma = [(a_1, B_1), \ldots, (a_\ell, B_\ell)] \), a player \( p \) is \textbf{addable} if, in \( G_M \), there is a path to \( p \) from some player in \( B_{\leq \ell} \).
\end{definition}
Definition 27. Given the current partial allocation \( M \cup \mathcal{E} \) and the current stack \( \Sigma = [(a_1, B_1), \ldots, (a_{\ell}, B_{\ell})] \), a thin edge \((p, D)\) is \textit{addable} if (i) \( p \) is addable, (ii) \( D \cap R(\Sigma) = \emptyset \), and (iii) \((p, D)\) is \( \lambda \)-minimal. For an addable edge \( a \), an edge \( b \) in \( \mathcal{E} \) is a \textit{blocking} edge of \( a \) if \( b \) shares some common resource with \( a \). If an addable edge has no blocking edge, it is unblocked; otherwise, it is blocked.

The construction of \((a_{\ell+1}, B_{\ell+1})\) is specified in the following routine \( \text{Build} \).

\[
\begin{aligned}
\text{Build}(M, \mathcal{E}, \Sigma, \ell) \\
1. & \quad \text{Arbitrarily pick an addable edge } a_{\ell+1}. \\
2. & \quad B_{\ell+1} := \{ e \in \mathcal{E} : e \text{ is a blocking edge of } a_{\ell+1} \}. \\
3. & \quad \text{Append } (a_{\ell+1}, B_{\ell+1}) \text{ to } \Sigma. \text{ Set } \ell := \ell + 1.
\end{aligned}
\]

Once some addable edge \( a_i \) in \( \Sigma \) is unblocked, i.e., \( B_i = \emptyset \), the following routine is invoked to update \( \Sigma, M, \) and \( \mathcal{E} \).

\[
\begin{aligned}
\text{Contract}(M, \mathcal{E}, \Sigma, \ell) \\
1. & \quad \text{Let } a^* \text{ be the unblocked addable edge with the smallest index. Let } p_{a^*} \text{ be the player covered by } a^*. \\
2. & \quad \text{Let } t \text{ be the smallest index such that, in } G_M, \text{ there is a path } \pi \text{ to } p_{a^*} \text{ from a player } p_t \text{ covered by some blocking edge } b_t \text{ in } B_t. \text{ Note that } \pi \text{ is an alternating path with respect to } M. \\
3. & \quad \text{Delete all the tuples } (a_i, B_i)'s \text{ with } i > t. \text{ Set } \ell := t. \\
4. & \quad \text{Update } M \text{ and } \mathcal{E} \text{ as the following.} \\
4.1 & \quad \text{Update } M \text{ using } \pi, \text{ i.e., } M := M \oplus \pi, \\
4.2 & \quad \text{add } a^* \text{ to } \mathcal{E} \text{ and release } b_t, \text{ i.e., } \mathcal{E} := (\mathcal{E} \setminus \{b_t\}) \cup \{a^*\}. \\
5. & \quad \text{If } t = 1, \text{ step } 3 \text{ already matches } p_0, \text{ so the algorithm terminates. If } t \geq 2, \text{ as } b_t \text{ is released from } \mathcal{E}, \text{ it is no longer a blocking edge. Set } B_t := B_t \setminus \{b_t\}.
\end{aligned}
\]

The algorithm keeps calling \( \text{Contract} \) until no unblocked addable edge remains in \( \Sigma \). Then it calls \( \text{Build} \) to grow the stack \( \Sigma \) again. It alternates between calling \( \text{Build} \) and \( \text{Contract} \) until \( p_0 \) is covered.

C.2 Analysis

We prove an invariant maintained by the algorithm. It is implicitly used in step 2 of \( \text{Contract} \) to guarantee that \( t \) always exists.

Lemma 28. Let \( M \cup \mathcal{E} \) be the current partial allocation. Let \( \Sigma = [(a_1, B_1), \ldots, (a_{\ell}, B_{\ell})] \) be the current stack. For all \( i \in [2, \ell] \), there is always a path in \( G_M \) that goes from some player in \( B_{\leq i-1} \) to the player covered by \( a_i \).

Proof. We prove the lemma by induction. Initially, \( \Sigma = \{(a_1, B_1)\} \) and \( \ell = 1 \). So the invariant trivially holds.

We show that \( \text{Build} \) preserves the invariant. Let \((a_{\ell+1}, B_{\ell+1})\) be the tuple newly constructed by \( \text{Build} \). For \( i \in [1, \ell] \), since \( \text{Build} \) does not change the first \( \ell \) tuples in \( \Sigma \) nor \( M \cup \mathcal{E} \), by the inductive hypothesis, the player covered by \( a_i \) is always reachable from some player in \( B_{\leq i-1} \). The player covered by \( a_{\ell+1} \) must be addable. By definition, there is a path in \( G_M \) that goes from some player in \( B_{\leq \ell} \) to the player covered by \( a_i \).

We show that \( \text{Contract} \) also preserves the lemma. Since all the tuples with index greater than \( t \) are deleted in step 3, we only need to verify the invariant for the remaining \( t \) tuples. Let \( p_{a^*}, t, p_t, \pi \) be defined as in the description of \( \text{Contract} \). By our choice of \( t \),
players in \( B_{\leq t-1} \) cannot reach \( p_\ell \) by any path in \( G_M \). Hence, any path \( \pi' \) starting with some player in \( B_{\leq t-1} \) must be node-disjoint from \( \pi \), since otherwise, we can find a path from \( B_{\leq t-1} \) to \( p_\ell \) by first following \( \pi' \), switching at the common node of \( \pi' \) and \( \pi \), and then following \( \pi \). Therefore, paths originating from \( B_{\leq t-1} \) are not affected by the operation \( M \oplus \pi \).

Lemma 29 below shows that the algorithm never gets stuck. Recall that \( R(\Sigma) \) is the set of thin resources covered by the thin edges in \( \Sigma \). Also recall that the restricted max-min allocation problem can be modeled as a configuration LP. Consider the dual of the configuration LP. Intuitively, we show that if the lemma does not hold, then some lower bound for the total dual value of the thin resources in \( R(\Sigma) \) would exceed its upper bound, which is a contradiction. Asadpour et al. \[2\] set the dual value of a thin resource \( r \) to be \( v_r \), and then used a worst-case upper bound and a worst-case lower bound for the total value of the resources in \( R(\Sigma) \). Their proof works only for \( \lambda \leq \frac{1}{2} \). We observe that the worst-case upper bound and the worst-case lower bound used in \[2\] cannot occur simultaneously. More specifically, the worst-case upper bound occurs only when all the thin resources have values nearly \( \lambda \), while the worst-case lower bound occurs only when all the thin resources have values nearly 0. Our approach is to magnifying the dual value of the thin resources with small \( v_r \). It helps us to derive a better lower bound without deteriorating the upper bound. Hence, we can obtain a proof working for a larger \( \lambda \).

▶ Lemma 29. Let \( M \cup \mathcal{E} \) be the current partial allocation. Let \( \Sigma = [(a_1, B_1), \ldots, (a_\ell, B_\ell)] \) be the current stack. If \( \Sigma \) is non-empty, either some addable edge in \( \Sigma \) is unblocked or there is an addable edge to be added to \( \Sigma \).

**Proof.** We prove by contradiction. Assume that all addable edges in \( \Sigma \) are blocked, i.e., \(|B_i| \geq 1\) for all \( i \in [1, \ell] \), and that there is no addable edge to be added to \( \Sigma \). Recall that the optimal value of the configuration LP is assumed to be 1. We show that the dual of the configuration LP \( \text{CLP}(1) \) is unbounded, which implies the contradiction that \( \text{CLP}(1) \) is infeasible.

Consider the following solution for the dual of \( \text{CLP}(1) \). Let \( P^+ \) be the set of players that are reachable in \( G_M \) from some player in \( B_{\leq t} \). Let \( R_f^+ \) be the set of fat resources that are reachable in \( G_M \) from some player in \( B_{\leq t} \). Recall that \( R(\Sigma) \) is the set of thin resources covered by thin edges in \( \Sigma \). We set \( \lambda = \frac{26}{59} \). For every \( p \in P \) and every \( r \in R \), set the dual variable \( y_p^* \) and \( z_r^* \) as follows.

\[
y_p^* = \begin{cases} 
1 - \frac{21}{26} \lambda & \text{if } p \in P^+, \\
0 & \text{otherwise}.
\end{cases}
\]

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

Figure 2 plots \( z_r^* \) and \( z_r^*/v_r \) versus \( v_r \) for a thin resource \( r \) in \( R(\Sigma) \). In future analysis, we will draw some conclusions directly from Figure 2 without giving a formal proof. One can verify these conclusions by easy numeric calculation.

▷ Claim 30 (Feasibility). \( y_p^* \leq \sum_{r \in C} z_r^* \) for any \( p \in P \) and any \( C \in C_P(1) \).
We prove that \( v \) goes to infinity. Therefore, the dual is unbounded, a contradiction. By examining subcases 2.1 – 2.4 below.

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

**Proof of Claim 30.** We need to prove that \( y_p^* \leq \sum_{r \in C} z_r^* \) for any \( p \in P \) and any \( C \in C_p(1) \). Consider any player \( p \in P \) and any configuration \( C \in C_p(1) \). If \( p \notin P^+ \), then \( y_p^* = 0 \), and the inequality holds because \( z_r^* \) is non-negative. Assume that \( p \in P^+ \). So \( y_p^* = 1 - \frac{21}{26} \lambda \). We prove that \( \sum_{r \in C} z_r^* \geq 1 - \frac{21}{26} \lambda \) by a case analysis.

**Case 1.** \( C \) contains a fat resource \( r_f \). Since \( p \in P^+ \), \( p \) is reachable in \( G_M \) from some player in \( B_{\leq \ell} \). Since player \( p \) desires \( r_f \), \( G_M \) contains either an edge from \( p \) to \( r_f \) or an edge from \( r_f \) to \( p \). In the former case, obviously \( r_f \) is reachable in \( G_M \) from some player in \( B_{\leq \ell} \).

In the latter case, \( p \) must be a player matched to \( r_f \) by \( M \), so \( p \notin B_{\leq \ell} \) as players in \( B_{\leq \ell} \) are not matched by \( M \). Moreover, the edge from \( r_f \) to \( p \) is the only edge entering \( p \) in \( G_M \). Any path entering \( p \) must go through \( r \). Therefore, \( r_f \) must be reachable from some player in \( B_{\leq \ell} \). In either case, \( r_f \in R_f^L \). Therefore, \( z_{r_f} = 1 - \frac{21}{26} \lambda \). We have

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

**Case 2.** \( C \) contains only thin resources. Since \( p \in P^+ \), \( p \) is reachable from some player in \( B_{\leq \ell} \). So \( p \) is an addable player. However, by our assumption, no more addable edge can be added to \( \Sigma \). Thus, the total value of thin resources in \( C \setminus R(\Sigma) \) must be less than \( \lambda \), since otherwise there would be an addable edge formed by \( p \) and the thin resources in \( C \setminus R(\Sigma) \). Given \( \lambda = \frac{26}{99} \),

\[
\sum_{r \in C \setminus R(\Sigma)} v_r \geq \sum_{r \in C} v_r - \sum_{r \in C \setminus R(\Sigma)} v_r > 1 - \lambda = \frac{73}{26} \lambda.
\]

We prove that

\[
\sum_{r \in C \setminus R(\Sigma)} z_r^* \geq 3\lambda \cdot \frac{26}{99} \cdot 1 - \frac{21}{26} \lambda.
\]

by examining subcases 2.1 – 2.4 below.
Case 2.1. \( C \cap R(\Sigma) \) contains at least three resources with their values in \([\frac{3\lambda}{4}, \lambda)\). Denote these three resources as \( r_1, r_2, \) and \( r_3 \). We have that
\[
\sum_{r \in C \cap R(\Sigma)} z_r^* \geq z_{r_1}^* + z_{r_2}^* + z_{r_3}^* = 3\lambda.
\]

Case 2.2. \( C \cap R(\Sigma) \) contains exactly one resource with value in \([\frac{3\lambda}{4}, \lambda)\). Denote this resource as \( r_1 \). Let \( R' = (C \cap R(\Sigma)) \setminus \{r_1\} \).
\[
\sum_{r \in C \cap R(\Sigma)} v_r = \sum_{r \in C \cap R(\Sigma)} v_r - v_{r_1} \geq 73 \cdot 26 \lambda - \lambda = 47 \cdot 26 \lambda. \tag{4}
\]
Every resource in \( R' \) has value in the range \((0, \frac{3\lambda}{4})\). As illustrated in Figure 2(b), \( \frac{r'}{r} \geq \frac{6}{5} \) for every \( r \in R' \). Therefore,
\[
\sum_{r \in R'} z_r^* \geq \frac{6}{5} \sum_{r \in R'} v_r \geq \frac{141}{65} \lambda.
\]
Then,
\[
\sum_{r \in C \cap R(\Sigma)} z_r^* = \sum_{r \in R'} z_r^* = \sum_{r \in R'} z_{r_1}^* > \frac{151}{65} \lambda + \lambda > 3\lambda
\]

Case 2.3. \( C \cap R(\Sigma) \) contains no resource with value in \([\frac{3\lambda}{4}, \lambda)\). As illustrated in Figure 2(b), \( \frac{r'}{r} \geq \frac{6}{5} \) for every \( r \in (0, \frac{3\lambda}{4}) \). Then,
\[
\sum_{r \in C \cap R(\Sigma)} z_r^* \geq \frac{6}{5} \sum_{r \in C \cap R(\Sigma)} v_r \geq \frac{6}{5} \cdot 73 \cdot 26 \lambda > 3\lambda.
\]

Case 2.4. The only remaining case is that \( C \cap R(\Sigma) \) contains exactly two resources with values in \([\frac{3\lambda}{4}, \lambda)\). Denote these two resources as \( r_1 \) and \( r_2 \). \( r_1 \) and \( r_2 \) together contribute \( 2\lambda \) to the sum \( \sum_{r \in C \cap R(\Sigma)} z_r^* \). Let \( R' = (C \cap R(\Sigma)) \setminus \{r_1, r_2\} \). To prove that \( \sum_{r \in C \cap R(\Sigma)} z_r^* \geq 3\lambda \), it suffices to show that
\[
\sum_{r \in R'} z_r^* \geq \lambda.
\]

Let \( V_0 \) denote the multi-set of values of thin resources in \( R' \). Note that \( v \in (0, \frac{3\lambda}{4}) \) for any \( v \in V_0 \), and that
\[
\sum_{v \in V_0} v = \sum_{r \in R'} v_r = \left( \sum_{r \in C \cap R(\Sigma)} v_r \right) - v_{r_1} - v_{r_2} \geq 73 \cdot 26 \lambda - 2\lambda = 21 \cdot 26 \lambda.
\]

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

To derive a lower bound, we will transform \( V_0 \) step by step to another multiset \( V_2 \) of values such that
1. \( \sum_{v \in V_2 \cap (0, \frac{3\lambda}{4})} g(v) + \sum_{v \in V_2 \cap (\frac{3\lambda}{4}, \frac{3\lambda}{2})} h(v) \leq \sum_{v \in V_0 \cap (0, \frac{3\lambda}{4})} g(v) + \sum_{v \in V_0 \cap (\frac{3\lambda}{4}, \frac{3\lambda}{2})} h(v) \),
Restricted Max-Min Allocation

(2) \( \sum_{v \in V_3} v = \frac{2\lambda}{3} \), and

(3) \( V_2 \) contains exactly two values, one belonging to \((0, \frac{a}{2})\) and the other belonging to \([\frac{a}{2}, \frac{3\lambda}{4}]\). Then we give a lower bound for \( \sum_{v \in V_3(0, \frac{a}{2})} g(v) + \sum_{v \in V_3(\frac{a}{2}, \frac{3\lambda}{4})} h(v) \).

As illustrated in Figure 2(a), both \( g(v) \) and \( h(v) \) are increasing functions of \( v \). Hence, if we decrease the values in \( V_3 \) to some smaller positive values, \( \sum_{v \in V_3(0, \frac{a}{2})} g(v) + \sum_{v \in V_3(\frac{a}{2}, \frac{3\lambda}{4})} h(v) \) does not increase. We keep decreasing the values in \( V_3 \) in an arbitrary fashion until \( \sum_{v \in V_3} v = \frac{2\lambda}{3} \). Let \( V_1 \) be the resulting multi-set. All values in \( V_1 \) are in the range \((0, \frac{a}{2})\).

If \( V_1 \) contains exactly one value in \((0, \frac{a}{2})\), then \( V_1 \) must contain exactly one value in \([\frac{a}{2}, \frac{3\lambda}{4}]\) because \( \sum_{v \in V_1} v = \frac{2\lambda}{3} \). Hence, \( V_1 \) meets condition (3), and it is the multi-set we desire. We set \( V_2 = V_1 \). Suppose that \( V_1 \) contains at least two values \( a \) and \( b \) in \((0, \frac{a}{2})\). If \( a + b \leq \frac{a}{2} \), then we merge \( a \) and \( b \) into \( c = a + b \). If \( c < \frac{a}{2} \),

\[ g(c) = \frac{3\lambda}{2c} = \frac{3\lambda}{2a + b} (a + b) = \frac{3\lambda}{2a + b} b = g(a) + g(b). \]

If \( c = \frac{a}{2} \),

\[ h(c) = \frac{3\lambda}{3\lambda - c} c = \frac{3\lambda}{2a + c} (a + b) = \frac{3\lambda}{2a + b} b = g(a) + g(b). \]

Hence, \( \sum_{v \in V_1(0, \frac{a}{2})} g(v) + \sum_{v \in V_1(\frac{a}{2}, \frac{3\lambda}{4})} h(v) \) does not increase.

If \( a + b > \frac{a}{2} \), we replace them with \( c = a + b - \frac{a}{2} \) and \( d = \frac{a}{2} \). Note that \( c < \frac{a}{2} \) because \( a + b < \lambda \). Also, \( h(d) = \frac{3\lambda}{3\lambda - d} d = \frac{3\lambda}{3\lambda - \frac{a}{2}} \). One can verify that

\[ g(c) + h(d) = \frac{3\lambda}{2c} + \frac{3\lambda}{2d} = 6\lambda \left( 1 - \frac{4\lambda^2 + (c + d)\lambda}{4\lambda^2 + 2(c + d)\lambda + cd} \right), \]

and that

\[ g(a) + g(b) = \frac{3\lambda}{2a + c} a + \frac{3\lambda}{2b + d} b = 6\lambda \left( 1 - \frac{4\lambda^2 + (a + b)\lambda}{4\lambda^2 + 2(a + b)\lambda + ab} \right). \]

Since \( a + b = c + d \) and \( ab - cd = ab - (a + b)(\frac{a}{2} + \frac{\lambda^2}{4}) = (\frac{a}{2} - a)(\frac{a}{2} - b) > 0, \)
\[ g(c) + h(d) < g(a) + g(b). \]

We conclude that \( \sum_{v \in V_1(0, \frac{a}{2})} g(v) + \sum_{v \in V_1(\frac{a}{2}, \frac{3\lambda}{4})} h(v) \) does not increase.

The above operation reduces the number of values in \((0, \frac{a}{2})\) by exactly 1 while preserving the sum of values. Repeating the above gives a multi-set \( V_2 \) that meets condition (3).

Now we are ready to derive a lower bound for \( \sum_{v \in V_3(0, \frac{a}{2})} g(v) + \sum_{v \in V_3(\frac{a}{2}, \frac{3\lambda}{4})} h(v) \). Let \( a \in [\frac{a}{2}, \frac{3\lambda}{4}] \) be the larger value in \( V_2 \). Then the smaller value is \( (c - a) \) where \( c = \frac{2\lambda}{3} \).

\[ \sum_{v \in V_2(0, \frac{a}{2})} g(v) + \sum_{v \in V_2(\frac{a}{2}, \frac{3\lambda}{4})} h(v) \]

\[ = g(c - a) + h(a) \]
\[ = \frac{3\lambda(c - a)}{2a + c - a} + \frac{3\lambda a}{3\lambda - a} \]
\[ = 3\lambda \left( 1 - \frac{2\lambda}{2a + c - a} + \frac{3\lambda}{3\lambda - a} - 1 \right) \]
\[ = 3\lambda \left( \frac{2\lambda}{2a + c - a} + \frac{3\lambda}{3\lambda - a} \right). \]
One can verify that when \( a \in (\frac{\lambda}{2}, \frac{3\lambda}{4}) \)
\[
\frac{d}{da} (g(c - a) + h(a)) = 3\lambda \left( \frac{-2\lambda}{(2\lambda + c - a)^2} + \frac{3\lambda}{(3\lambda - a)^2} \right) > 0
\]
So we get the minimum when \( a = \frac{\lambda}{2} \). Therefore, we have
\[
\sum_{r \in R} z_r^* \geq g(c - a) + h(a) \geq g\left(\frac{21}{26}\lambda - \frac{1}{2}\lambda\right) + h\left(\frac{1}{2}\lambda\right) = \lambda.
\]
\(\triangleright\)

Before proving Claim 31, we first establish the following result.

\(\triangleright\) Claim 32. Let \( e \) be a thin edge that appears in \( \Sigma \). That is, \( e \) is either an addable edge \( a_i \) or a blocking edge in \( B_i \) for some \( i \). Let \( R_e \) be the set of thin resources covered by \( e \). Let \( r_0 \) be the resource with the least value in \( R_e \). Then,
\[
\sum_{r \in R_e} z_r^* \leq \frac{3\lambda}{2} + \frac{z_{r_0}^*}{2}.
\]

Proof. If \( v_{r_0} \geq \frac{\lambda}{2} \), all resources in \( R_e \) have values at least \( \frac{\lambda}{2} \). Since \( e \) is \( \lambda \)-minimal, \( R_e \) contains exactly two thin resources, including \( r_0 \). Let \( r_1 \) denote the other resource in \( R_e \).
\[
\sum_{r \in R_e} z_r^* = z_{r_0}^* + z_{r_1}^* \leq \frac{z_{r_0}^*}{2} + \frac{\lambda}{2} + \frac{z_{r_1}^*}{2} + \frac{3\lambda}{2}.
\]

Suppose that \( v_{r_0} < \frac{\lambda}{2} \). Let \( r_1 \) be the resources with the largest value in \( R_e \). If \( v_{r_0} \geq \lambda - v_{r_1} \), then \( r_0 \) and \( r_1 \) have a total value of at least \( \lambda \). Thus \( R_e \) does not contain any other resource because \( e \) is \( \lambda \)-minimal. We get \( \sum_{r \in R_e} z_r^* = z_{r_0}^* + z_{r_1}^* \leq \frac{z_{r_0}^*}{2} + \frac{3\lambda}{2} \) as before. Suppose that \( v_{r_0} < \lambda - v_{r_1} \). Consider an arbitrary resource \( r \in R_e \). If \( v_r \in (0, \frac{\lambda}{2}) \),
\[
\frac{z_r^*}{v_r} \leq 3\lambda \leq \frac{3\lambda}{2\lambda + v_{r_0}}.
\]
If \( v_r \in (\frac{\lambda}{2}, \frac{3\lambda}{4}) \),
\[
\frac{z_r^*}{v_r} = \frac{3\lambda}{3\lambda - v_r} = \frac{3\lambda}{2\lambda + (\lambda - v_r)} < \frac{3\lambda}{2\lambda + v_{r_0}}.
\]
If \( v_r \in (\frac{3\lambda}{4}, \lambda) \),
\[
\frac{z_r^*}{v_r} = \frac{\lambda}{3\lambda - v_r} = \frac{3\lambda}{2\lambda + (\lambda - v_r)} \leq \frac{3\lambda}{2\lambda + v_{r_0}}.
\]

In summary, for every \( r \in R_e \),
\[
\frac{z_r^*}{v_r} \leq \frac{3\lambda}{2\lambda + v_{r_0}}.
\]
Since \( e \) is \( \lambda \)-minimal,
\[
\sum_{r \in R_e} v_r^* < \lambda + v_{r_0}.
\]
Combining these two facts, we obtain
\[ \sum_{r \in R_e} z_r^+ \leq \frac{3\lambda}{2\lambda + v_{ro}} \sum_{r \in R_e} u_r^+ \]
\[ \leq \frac{3\lambda}{2\lambda + v_{ro}} (\lambda + v_{ro}) \]
\[ = \frac{3\lambda}{2\lambda + v_{ro}} \left( \lambda + \frac{1}{2} v_{ro} \right) + \frac{3\lambda}{2\lambda + v_{ro}} \frac{1}{2} v_{ro} \]
\[ \leq \frac{3\lambda}{2} + \frac{z_{ro}^+}{2}. \]

Now we are ready to prove Claim 31.

Proof of Claim 31 We shall prove that \( \sum_{p \in P} y_p^+ - \sum_{r \in R} z_r^+ > 0 \). By our construction of the dual solution, it is easy to see that
\[ \sum_{p \in P} y_p^+ - \sum_{r \in R} z_r^+ = \left( \sum_{p \in P^+} y_p^+ - \sum_{r \in R_f} z_r^+ \right) - \sum_{r \in R(\Sigma)} z_r^+ . \]

Consider \( \sum_{p \in P^+, y_p^+ - \sum_{r \in R_f^+} z_r^+ \right), \) for every player \( p \in P^+ \) and every fat resource \( r_f \in R_f^+ \), \( y_p^+ \) and \( z_r^+ \) have the same value \( 1 - \frac{21}{2\lambda} \). Therefore,
\[ \sum_{p \in P^+} y_p^+ - \sum_{r \in R_f^+} z_r^+ = \left( 1 - \frac{21}{2\lambda} \right) \left( |P^+| - |R_f^+| \right). \]

We will derive a lower bound for \(|P^+| - |R_f^+|\). The player in \( B_1 \) is \( p_0 \), and \( p_0 \) is not matched by \( M \). Players in other \( B_i \)'s are covered by \( E \), so they are not matched by \( M \). Therefore, no player in \( B_{\leq \ell} \) is matched by \( M \). Every fat resource that is reachable in \( G_M \) from some player in \( B_{\leq \ell} \) must be matched by \( M \); otherwise, such the path between the fat resource and the player would be an augmenting path with respect to \( M \), contradicting the fact that \( M \) is a maximum matching of \( G \). Hence, for every fat resource \( r_f \in R_f^+ \), \( r_f \) must have an out-going edge to some player \( p \) in \( G_M \). Since \( r_f \) is reachable from \( B_{\leq \ell} \) in \( G_M \), \( p \) is also reachable from \( B_{\leq \ell} \) in \( G_M \). So \( p \in P^+ \). We charge \( r_f \) to \( p \). Every player in \( P \) has in-degree at most 1 in \( G_M \), so it is charged at most once. Recall that \( B_{\leq \ell} \subseteq P^+ \). Players in \( B_{\leq \ell} \) do not have incoming edges in \( G_M \) because they are not matched by \( M \). So players in \( B_{\leq \ell} \) do not get charged. In conclusion,
\[ |P^+| - |R_f^+| \geq |B_{\leq \ell}|. \]

Putting things together, we get
\[ \sum_{p \in P^+} y_p^+ - \sum_{r \in R_f^+} z_r^+ \geq \left( 1 - \frac{21}{2\lambda} \right) |B_{\leq \ell}|. \] (5)

Next we bound \( \sum_{r \in R(\Sigma)} z_r^+ \). As in Figure 2(a), \( z_r^+ \) does not decrease as \( v_r \) increases. Hence, Claim 32 implies that for any \( e \in R(\Sigma) \) and for any resource \( r' \in R_e \),
\[ \sum_{r \in R_e} z_r^+ \leq \frac{3\lambda}{2} + \frac{z_{ro}^+}{2}. \] (6)
Now consider a tuple \((a_i, \mathcal{B}_i)\). By our assumption that every addable edge is blocked, \(|\mathcal{B}_i| \geq 1\). Let \(R_t\) be the set of thin resources covered by \(\{a_i\} \cup \mathcal{B}_i\). Consider a resource \(r \in R_t\). If \(r\) is covered by only one edge in \(\{a_i\} \cup \mathcal{B}_i\), then charge \(\tilde{z}_r^*\) to that edge. If \(r\) is covered by two edges in \(\{a_i\} \cup \mathcal{B}_i\), then charge half of \(\tilde{z}_r^*\) to one edge and the other half to the other edge. Take any edge \(e \in \{a_i\} \cup \mathcal{B}_i\). Because edges in \(\mathcal{B}_i\) are blocking edges of \(a_i\), \(e\) must share some resource, say \(r'\), with another edge in \(\{a_i\} \cup \mathcal{B}_i\). We conclude from (6) that the total charge on \(e\) is at most
\[
\sum_{r \in R_t} z_r^* - \frac{z_r^*}{2} \leq \frac{3\lambda}{2}.
\]

Taking sum over all edges in \(\{a_i\} \cup \mathcal{B}_i\) gives
\[
\sum_{r \in R_t} z_r^* \leq \left|\{a_i\} \cup \mathcal{B}_i\right| \cdot \frac{3\lambda}{2} = \frac{3\lambda}{2} (|\mathcal{B}_i| + 1) \leq 3\lambda|\mathcal{B}_i|.
\]

The last inequality follows from the fact that \(|\mathcal{B}_i| \geq 1\). Recall that \((a_1, \mathcal{B}_1) = (\text{null}, (p_0, \emptyset))\) covers no resource. Then, taking sum over \(i \in [2, \ell]\) gives
\[
\sum_{r \in R(\Sigma)} z_r^* = \sum_{i=2}^\ell \sum_{r \in R_i} z_r^* \leq 3\lambda (|\mathcal{B}_{\leq \ell}| - 1).
\]

Combining (5) and (7), we obtain
\[
\sum_{p \in P} y_p^* - \sum_{r \in R} z_r^* = \sum_{p \in P^+} y_p^* - \sum_{r \in R_1} z_r^* = \sum_{r \in R(\Sigma)} z_r^* \\
\quad \geq \left(1 - \frac{21}{26}\right) |\mathcal{B}_{\leq \ell}| - 3\lambda (|\mathcal{B}_{\leq \ell}| - 1) \\
= 3\lambda + \left(1 - \frac{99}{26}\right) |\mathcal{B}_{\leq \ell}|.
\]

Given \(\lambda = \frac{26}{99}\),
\[
\sum_{p \in P} y_p^* - \sum_{r \in R} z_r^* \geq 3\lambda > 0.
\]

We have completed the proof of Lemma 29, which says that the algorithm is always able to either find an unblocked edge and call Contract to shrink \(\Sigma\) or call Build to append an addable edge and its blocking edges to \(\Sigma\). It remains to show that the local search algorithm will make \(\Sigma\) empty in finite time (i.e., satisfies \(p_0\) eventually). This analysis has been given before in [2]. We repeat it here for completeness.

\begin{lemma}
The algorithm terminates after a finite number of calls of Build and Contract.
\end{lemma}

\begin{proof}
Define a signature vector \((|\mathcal{B}_1|, |\mathcal{B}_2|, \ldots, |\mathcal{B}_\ell|, \infty)\) with respect to the sequence of tuples. All \(\mathcal{B}_1, \ldots, \mathcal{B}_\ell\) are mutually disjoint subsets of \(\mathcal{E}\), so \(|\mathcal{B}_1| + \cdots + |\mathcal{B}_\ell| \leq n\). Therefore, the number of distinct signature vectors is at most \(n^n\). The signature vector evolves as the sequence is updated by the algorithm. After each invocation of Build, the signature vector decreases lexicographically because it gains a new second to last entry. After each invocation of Contract, the signature decreases lexicographically because it becomes shorter, and the second to last entry decreases by at least 1. Therefore, no signature vector is repeated. As a result, the algorithm terminates after at most \(n^n\) invocations of Build and Contract.
\end{proof}
By Lemma 33, we can obtain an allocation in which every player receives at least $\lambda = \frac{26}{29}$ worth of resources. This completes the proof of Theorem 2.