A Tale of Santa Claus, Hypergraphs and Matroids

Sami Davies[*] Thomas Rothvoss[†] Yihao Zhang[‡]

Abstract

A well-known problem in scheduling and approximation algorithms is the Santa Claus problem. Suppose that Santa Claus has a set of gifts, and he wants to distribute them among a set of children so that the least happy child is made as happy as possible. Here, the value that a child i has for a present j is of the form $p_{ij} \in \{0, p_j\}$. A polynomial time algorithm by Annamalai et al. gives a 12.33-approximation and is based on a modification of Haxell's hypergraph matching argument.

In this paper, we introduce a *matroid* version of the Santa Claus problem. Our algorithm is also based on Haxell's augmenting tree, but with the introduction of the matroid structure we solve a more general problem with cleaner methods. Our result can then be used as a blackbox to obtain a $(4+\varepsilon)$-approximation for Santa Claus. This factor also compares against a natural, compact LP for Santa Claus.

1 Introduction

Formally, the *Santa Claus* problem takes as input a set M of children, a set J of gifts, and values $p_{ij} \in \{0, p_j\}$ for all $i \in M$ and $j \in J$. In other words, a child is only interested in a particular subset of the gifts, but then its value only depends on the gift itself. The goal is to find an assignment $\sigma : J \to M$ of gifts to children so that $\min_{i\in M} \sum_{j\in\sigma^{-1}(i)} p_{ij}$ is maximized.

The first major progress on this problem is due to Bansal and Sviridenko [BS06], who showed a $O(\log\log n/\log\log\log n)$-approximation based on rounding a *configuration LP*. The authors of [BS06] also realized that in order to obtain a $O(1)$-approximation, it suffices to answer a purely combinatorial problem: show that in a uniform bipartite hypergraph with equal degrees on all sides, there is a left-perfect matching that selects a constant fraction of nodes from original edges. This question was affirmatively answered by Feige [Fei08] who proved a large unspecified constant using the Lovász Local Lemma repeatedly. Then Asadpour, Feige and Saberi [AFS08] showed that one can answer the question of [BS06] by using a beautiful theorem on hypergraph matchings due to Haxell [Hax95]; their bound[1] of 4 has been slightly improved to 3.84 by Jansen and Rohwedder [JR18c] and Cheng and Mao [CM18a]. Recently, Jansen and Rohwedder [JR18a]

[*]University of Washington, Seattle. Email: daviess@uw.edu

[†]University of Washington, Seattle. Email: rothvoss@uw.edu. Supported by NSF CAREER grant 1651861 and a David & Lucile Packard Foundation Fellowship.

[‡]University of Washington, Seattle. Email: yihaoz93@uw.edu

[1]Note that the conference version of [AFS08] provides a factor of 5, which in the journal version [AFS12] has been improved to 4.

also showed (still non-constructively) that it suffices to compare to a linear program with as few as $O(n^3)$ many variables and constraints, in contrast to the exponential size configuration LP.

A *hypergraph* $\mathcal{H} = (X \dot\cup W, \mathcal{E})$ is called *bipartite* if $|e \cap X| = 1$ for all hyperedges $e \in \mathcal{E}$. A *(left-) perfect matching* is a set of hyperedges $F \subseteq \mathcal{E}$ that are disjoint but cover each node in X. In general, finding perfect matchings in even bipartite hypergraphs is **NP**-hard, but there is an intriguing sufficient condition:

Theorem 1 (Haxell [Hax95]). *Let $\mathcal{H} = (X \dot\cup W, \mathcal{E})$ be a bipartite hypergraph with $|e| \le r$ for all $e \in \mathcal{E}$. Then either $\mathcal{H}$ contains a left-perfect matching or there is a subset $C \subseteq X$ and a subset $U \subseteq W$ so that all hyperedges incident to C intersect U and $|U| \le (2r-3)\cdot(|C|-1)$.*

It is instructive to consider a "standard" bipartite graph with $r = 2$. In this case, if there is no perfect matching, then there is a set $C \subseteq X$ with at most $|C| - 1$ many neighbors — so Haxell's condition generalizes *Hall's Theorem*. Unlike Hall's Theorem, Haxell's proof is non-constructive and based on a possibly exponential time augmentation argument. Only very recently and with a lot of care, Annamalai [Ann16] managed to make the argument polynomial. This was accomplished by introducing some slack into the condition and assuming the parameter r is a constant. Preceding [Ann16], Annamalai, Kalaitzis and Svensson [AKS15] gave a non-trivially modified version of Haxell's argument for Santa Claus, which runs in polynomial time and gives a 12.33-approximation[2]. Recently, Cheng and Mao altered their algorithm to improve the approximation to $6 + \varepsilon$, for any constant $\varepsilon > 0$ [CM18b]. Our algorithm will also borrow a lot from [AKS15]. However, through a much cleaner argument we obtain a result that works in a more general matroid setting and implies a better approximation of $4 + \varepsilon$ for Santa Claus.

It should not go without mention that the version of the Santa Claus problem with arbitrary p_{ij} has also been studied before under the name *Max-Min Fair Allocation*. Interestingly, the integrality gap of the configuration LP is at least $\Omega(\sqrt{n})$ [BS06]. Still, Chakrabarty, Chuzhoy and Khanna [CCK09] found a (rather complicated) $O(\log^{10}(n))$-approximation algorithm in $n^{O(\log n)}$ time[3].

Santa Claus has a very well studied "dual" minmax problem. Usually it is phrased as *Makespan Scheduling* with *machines* $i \in M$ and *jobs* $j \in J$. Then we have a running time p_{ij} of job j on machine i, and the goal is to assign jobs to machines so that the maximum load of any machine is minimized. In this general setting, the seminal algorithm of Lenstra, Shmoys and Tardos [LST87] gives a 2-approximation — with no further improvement since then. In fact, a $(\frac{3}{2} - \varepsilon)$-approximation is **NP**-hard [LST87], and the configuration LP has an integrality gap of 2 [VW11]. In the restricted assignment setting with $p_{ij} \in \{p_j, \infty\}$, the breakthrough of Svensson [Sve11] provides a non-constructive 1.942-bound on the integrality gap of the configuration LP using a custom-tailored Haxell-type search method. Recently, this was improved by Jansen and Rohwedder [JR17] to 1.834. In an even more restricted variant called *Graph Balancing*, each job is admissable on exactly 2 machines. In this setting Ebenlendr, Krcál and Sgall [EKS08] gave a 1.75-approximation based on an LP-rounding approach, which has again been improved by Jansen and Rohwedder [JR18b] to 1.749 using a local search argument.

[2]To be precise they obtain a $(6 + 2\sqrt{10} + \varepsilon)$-approximation in time $n^{O(\frac{1}{\varepsilon^2}\log(\frac{1}{\varepsilon}))}$.

[3]The factor is n^{ε} if only polynomial time is allowed, where $\varepsilon > 0$ is arbitrary but fixed.

1.1 Our contributions

Let $\mathcal{M} = (X, \mathcal{I})$ be a *matroid* with *groundset* X and a family of *independent sets* $\mathcal{I} \subseteq 2^X$. Recall that a matroid is characterized by three properties:

(i) *Non-emptyness*: $\emptyset \in \mathcal{I}$;
(ii) *Monotonicity*: For $Y \in \mathcal{I}$ and $Z \subseteq Y$ one has $Z \in \mathcal{I}$;
(iii) *Exchange property*: For all $Y, Z \in \mathcal{I}$ with $|Y| < |Z|$ there is an element $z \in Z \setminus Y$ so that $Y \cup \{z\} \in \mathcal{I}$.

The *bases* $\mathcal{B}(\mathcal{M})$ of the matroid are all inclusion-wise maximal independent sets. The cardinalities of all bases are identical, with size denoted as $\text{rank}(\mathcal{M})$. The convex hull of all bases is called the *base polytope*, that is $P_{\mathcal{B}(\mathcal{M})} := \text{conv}\{\chi(S) \in \{0,1\}^X \mid S \text{ is basis}\}$, where $\chi(S)$ is the *characteristic vector* of S.

Now consider a bipartite graph $G = (X \dot\cup W, E)$ with the ground set X on one side and a set of *resources* W on the other side; each resource $w \in W$ has a *size* $p_w \geq 0$. In a problem that we call *Matroid Max-Min Allocation*, the goal is to find a basis $S \in \mathcal{B}(\mathcal{M})$ and an assignment $\sigma : W \to S$ with $(\sigma(w), w) \in E$ so that $\min_{i \in S} \sum_{w \in \sigma^{-1}(i)} p_w$ is maximized. To the best of our knowledge, this problem has not been studied before. In particular if $T \geq 0$ is the target objective function value, then we can define a linear programming relaxation $Q(T)$ as the set of vectors $(x, y) \in \mathbb{R}^X_{\geq 0} \times \mathbb{R}^E_{\geq 0}$ satisfying the constraints

$$x \in P_{\mathcal{B}(\mathcal{M})}; \quad \sum_{w \in N(i)} p_w y_{iw} \geq T \cdot x_i \ \forall i \in X; \quad y(\delta(w)) \leq 1 \ \forall w \in W; \quad y_{iw} \leq x_i \ \forall (i,w) \in E.$$

Here, the decision variable x_i expresses whether element i should be part of the basis, and y_{iw} expresses whether resource w should be assigned to element i. We abbreviate $N(i)$ as the neighborhood of i and $y(\delta(w))$ is shorthand for $\sum_{i:(i,w) \in E} y_{iw}$. Then our main technical result is:

Theorem 2. *Suppose $Q(T) \neq \emptyset$. Then for any $\varepsilon > 0$ one can find*

$$(x, y) \in Q\left(\left(\frac{1}{3} - \varepsilon\right) \cdot T - \frac{1}{3} \cdot \max_{w \in W} p_w\right)$$

with both x and y integral in time $n^{\Theta_\varepsilon(1)}$, where $n := |X| + |W|$. This assumes that membership in the matroid can be tested in time polynomial in n.

Previously this result was not even known with non-constructive methods. We see that Matroid Max-Min Allocation is a useful framework by applying it to the Santa Claus problem:

Theorem 3. *The Santa Claus problem admits a $(4 + \varepsilon)$-approximation algorithm in time $n^{\Theta_\varepsilon(1)}$.*

For a suitable threshold $0 < \delta < 1$, call a gift j *small* if $p_j \leq \delta \cdot OPT$ and *large* otherwise. Then the family of sets of children that can get assigned large gifts forms a *matchable set matroid*. We apply Theorem 2 to the *co-matroid* of the matchable set matroid. Then we obtain a basis $S := \{i \in M \mid x_i = 1\}$, which contains the children *not* receiving a large gift. These children can receive small gifts of total value $(\frac{1}{3} - \frac{\delta}{3} - \varepsilon) \cdot OPT$. The remaining children receive a large gift with value at least $\delta \cdot OPT$. Setting $\delta := \frac{1}{4}$ implies the claim. Note the approximation factor $4+\varepsilon$ will be with respect to a natural, compact linear program with $O(n^2)$ many variables and constraints. The smallest LP that was previously known to have a constant integrality gap was the $O(n^3)$-size LP of [JR18a].

2 An algorithm for Matroid Max-Min Allocation

In this section we provide an algorithm that proves Theorem 2.

2.1 Intuition for the algorithm

We provide some insight by starting with an informal overview of our algorithm. Let $G = (X \cup W, E)$ be the bipartite graph defined in Section 1.1. If $U \subset W$ and $i \in X$ with $(i, j) \in E$ for all $j \in U$, we can consider the pair (i, U) to be a hyperedge. Then for $0 < \nu < 1$ and $\mathrm{val}(\cdot)$ the function summing the value in a hyperedge's resources, we say that (i, U) is a *ν-edge* if it a hyperedge with minimal (inclusion wise) resources such that $\mathrm{val}(U) := \sum_{w \in U} p_w \geq \nu T$. By $\mathcal{E}_{\nu T}$ we denote the set of ν-edges.

Fix constants $0 < \beta < \alpha < 1$ and $0 < \delta < 1$, to be chosen later. The goal of the algorithm is to find a basis $S \in \mathcal{B}(\mathcal{M})$ and a hypergraph matching $M \subseteq \mathcal{E}_{\beta T}$ covering S. The algorithm is initialized with $S := \{i_0\}$, for any node $i_0 \in X$, and $M := \emptyset$. We perform $\mathrm{rank}(\mathcal{M})$ many phases, where in each phase we find a larger matching, and the set it covers in X is independent with respect to the matroid. In an intermediate phase, we begin with $S \in \mathcal{I}$ and $M \subseteq \mathcal{E}_{\beta T}$ a hypergraph matching covering $S \setminus \{i_0\}$ with one exposed node $i_0 \in X$. At the end of a phase, the algorithm produces an updated matching covering an independent set S', with $|S'| = |S|$. For $|S'| < \mathrm{rank}(\mathcal{M})$, there exists $i'_0 \in X \setminus S'$ such that $S' \cup \{i'_0\} \in \mathcal{I}$. Repeating this $\mathrm{rank}(\mathcal{M})$ times, we end with a basis which is well-covered by β-edges.

The algorithm generalizes the notion of an augmenting path used to find a maximum matchings in bipartite graphs to an *augmenting tree*. Though instead of swapping every other edge in an augmenting path, as is the case for a bipartite graph, the algorithm swaps sets of edges in the augmenting tree to find more space in the hypergraph. During a phase, the edges are swapped in such a way that the underlying set in X covered by the matching is always independent with respect to the matroid. The edges which are candidates for being swapped into the matching are called *adding edges* and denoted by A, while those which are candidates for being swapped out of the matching are called *blocking edges* and denoted by B. It is helpful to discuss the nodes covered by adding and blocking edges in each part, and so for hyperedges $H \subseteq \mathcal{E}_{\nu T}$ we define H_X and H_W as the nodes covered by H in X and W, respectively. The algorithm gives some slack by allowing the adding edges to be slightly larger than the blocking edges.

The parameters α and β determine the value of the adding and blocking edges, respectively, so the adding edges are a subset of $\mathcal{E}_{\alpha T}$ while the blocking edges are a subset of $\mathcal{E}_{\beta T}$. Set $\delta := \max_w p_w / T$, so that all elements in the basis receive resources with value at most δT. The following observations follow from minimality of the hyperedges:

1. A ν-edge has value less than $(\nu + \delta)T$. This implies that an add edge has value less than $(\alpha + \delta)T$ and a blocking edge has value less than $(\beta + \delta)T$.

2. Every blocking edge has value at most $\beta \cdot T$ not covered by an add edge.

To build the augmenting tree, the algorithm starts from the node in S uncovered by M, i_0, and chooses an edge $e \in \mathcal{E}_{\alpha T}$ covering i_0 which is added to A. If there is a large enough hyperedge $e' \in \mathcal{E}_{\beta T}$ such that $e' \subset e$ and e' is disjoint from M, then there is enough available resources that we simply update M by adding e' to it. Otherwise, e does not contain a set of resources with total value βT free from M. The edges of M intersecting e are added to the set of blocking edges, B. Nodes in $C = \{i_0\} \cup B_X$ are called *discovered* nodes, as they are the nodes covered by the hypermatching M which appear in the augmenting tree.

Continuing to build the augmenting tree in later iterations, the algorithm uses an *Expansion Lemma* to find a large set of disjoint hyperedges, $H \subset \mathcal{E}_{\alpha T}$, that cover a subset which can be swapped into S in place of some subset of C while maintaining independence in the matroid. The set of hyperedges H either (i) intersects many edges of M or (ii) has a constant fraction of edges which contain a hyperedge from $\mathcal{E}_{\beta T}$ that is disjoint from M.

In the first case, a subset of H which intersects M, denoted $A_{\ell+1}$, is added to A, and the edges of M intersecting $A_{\ell+1}$, denoted $B_{\ell+1}$, are added to B, for ℓ the index of the iteration. Note we naturally obtain *layers* which partition the adding and blocking edges in our augmenting tree. The layers for the adding and blocking edges respectively are denoted as A_ℓ and B_ℓ, with

$$A_{\le \ell} := \bigcup_{i=0}^{\ell} A_i \qquad \text{and} \qquad B_{\le \ell} := \bigcup_{i=0}^{\ell} B_i.$$

The layer indices are tracked because they are useful in proving the algorithm's runtime. In the second case, for the set of edges $H' \subset \mathcal{E}_{\alpha T}$ that have a hyperedge from $\mathcal{E}_{\beta T}$ disjoint from M, the algorithm finds a layer which has a large number of discovered nodes that can be swapped out for a subset of nodes which H' covers.

2.2 A detailed procedure

Recall, we fixed $\delta = \max_{w \in W} p_w / T$. Then, we set $\beta = \frac{1}{3} - \frac{\delta}{3} - \varepsilon$ and $\alpha = \frac{1}{3} - \frac{\delta}{3} - \frac{\varepsilon}{2}$, for $0 < \varepsilon < (1-\delta)/3$. Here lies the subtle but crucial difference to previous work. In [AKS15] the authors have to use adding edges that are a large constant factor bigger than blocking edges. In our setup we can allow adding edges that are only marginally larger than the blocking edges. This results in an improved approximation factor of $4+\varepsilon$ for Santa Claus compared to the 12.33 factor by [AKS15].

The algorithm is described in Figure 1. For later reference, the constant from Lemma 7 is $\frac{1-2\alpha-\beta-\delta}{1+\delta} = \frac{2\varepsilon}{1+\delta} \ge \varepsilon$, and the constant from Lemma 8 is $\frac{\alpha-\beta}{\delta+\alpha} = \frac{\varepsilon}{2(1+\delta)} \ge \varepsilon/4$. We use Lemma 9, with constant $c = \frac{1-2\alpha-\beta-\delta}{1+\delta} \cdot \frac{\alpha-\beta}{\delta+\alpha}$. Our bounds for constants do not use a specific choice of δ, and instead they only use the fact that $0 < \delta < 1$. Both cases in the algorithm are visualized in Figure 2 and Figure 3.

2.3 Correctness of the algorithm

Here, we prove several lemmas used in the algorithm which implies Theorem 2. We begin by building up to our *Expansion Lemma*, Lemma 7. Our algorithm takes a fixed independent set,

```
**Input:** Node $i_0$ and set $S \in \mathcal{I}$ with $i_0 \in S$. Matching $M \subseteq \mathcal{E}_{\beta T}$ with $M_X = S \setminus \{i_0\}$.
Initialize: $A = A_0 = \emptyset$, $B = B_0 = \emptyset$, $C = \{i_0\}$, $\ell = 0$.
while TRUE do
    Find disjoint $H \subseteq \mathcal{E}_{\alpha T}$ covering $D \subseteq (X \setminus S) \cup C$, s.t. $|D| \geq \varepsilon \cdot |C|$, $(S \setminus C) \cup D \in \mathcal{I}$, *and $H_W$ is disjoint from $A_W \cup B_W$.

// Build the next layer in the augmenting tree
    if $H$ intersects at least $\frac{\varepsilon}{4} \cdot |H| \geq \frac{\varepsilon^2}{4} \cdot |C|$ many edges $M$ on $W$-side then
        $B \leftarrow B \cup B_{\ell+1}$, $B_{\ell+1} = \{e \in M : e \cap H \neq \emptyset\}$
        // Find subset of $H$ to add to $A$
            for $b \in B_{\ell+1}$ do
                Choose one edge $h_b \in H$ such that $h_b \cap b \neq \emptyset$
                $A_{\ell+1} \leftarrow A_{\ell+1} \cup \{h_b\}$
            end for
        $C \leftarrow B_X \cup \{i_0\}$
        $\ell \leftarrow \ell + 1$

//  Swap sets and collapse layers
    else $H' = \{e \in H : \mathrm{val}(e_W \setminus M_W) \geq \beta T\}$ has size at least $\frac{\varepsilon}{4} \cdot |H|$
        For all $e \in H'$, choose one $e' \subset e$ with $e' \in \mathcal{E}_{\beta T}$ and $e'_W \cap M_W = \emptyset$. Replace $e$ for $e'$ in $H'$.
        // Find a set to swap in, $\tilde{D}$, and a set to swap out, $\tilde{C}$
            $D' \subseteq D$ are the nodes covered by $H'$
            $C' \subseteq C$ is such that $|C'| = |D'|$ and $S \setminus C' \cup D' \in \mathcal{I}$
            if $i_0 \in C'$ then
                Let $i_1 \in D'$ so that $S' := S \setminus \{i_0\} \cup \{i_1\} \in \mathcal{I}$ and let $e_1 \in H'$ be edge covering $i_1$.
                Return $M' := M \cup \{e_1\}$ covering all of $S'$ and terminate.
            end if
            Layer $\tilde{\ell} \leq \ell$ contains $\tilde{C} \subset C' \cap (B_{\tilde{\ell}})_X$, with $|\tilde{C}| \geq \gamma |C'|$. **
            Let $\tilde{D} \subset D'$ be such that $|\tilde{C}| = |\tilde{D}|$ and $S' := S \setminus \tilde{C} \cup \tilde{D} \in \mathcal{I}$.
            $\tilde{M} \subset M$ covers $\tilde{C}$ and $\tilde{H} \subset H'$ covers $\tilde{D}$.
        $M \leftarrow M \setminus \tilde{M} \cup \tilde{H}$, and $S \leftarrow S'$
        $A \leftarrow A_{\leq \tilde{\ell}}$, $B \leftarrow B_{\leq \tilde{\ell}} \setminus \tilde{M}$, $C \leftarrow B_X \cup \{i_0\}$
        $\ell \leftarrow \tilde{\ell}$
    end if***
end while
        * Possible by Lemma 7 with $W' := A_W \cup B_W$.
        ** By Lemma 9, such a $\tilde{C}$ exists.
        *** One of the conditionals occurs by Lemma 8.
```

Figure 1: Main algorithm

S, and swaps $C \subset S$ out of S for a set of nodes D in order to construct a new independent set of the same size. This is possible by Lemma 7.

Recall a variant of the so-called *Exchange Lemma*. For independent sets $Y, Z \in \mathcal{I}$, let $H_{\mathcal{M}}(Y,Z)$ denote the bipartite graph on parts Y and Z (if $Y \cap Z \neq \emptyset$, then have one copy of the intersection on the left and one on the right). For $i \in Y \setminus Z$ and $j \in Z \setminus Y$ we insert an edge (i, j) in $H_{\mathcal{M}}(Y,Z)$ if $Y \setminus \{i\} \cup \{j\} \in \mathcal{I}$. Otherwise, for $i \in Y \cap Z$, there is an edge between the left and right copies of i, and this is the only edge for both copies of i.

Lemma 4 (Exchange Lemma)**.** *For any matroid $\mathcal{M} = (X,\mathcal{I})$ and independent set $Y, Z \in \mathcal{I}$ with $|Y| \leq |Z|$, the exchange graph $H_{\mathcal{M}}(Y,Z)$ contains a left perfect matching.*

Next, we prove several lemmas about vectors in the base polytope with respect to sets containing swappable elements. Lemma 7 relies on a *Swapping Lemma*, Lemma 6, for which the next lemma serves as a helper function.

Lemma 5 (Weak Swapping Lemma)**.** *Let $\mathcal{M} = (X,\mathcal{I})$ be a matroid with an independent set $S \in \mathcal{I}$. For $C \subseteq S$, define*

$$U := \{i \in (X \setminus S) \cup C \mid (S \setminus C) \cup \{i\} \in \mathcal{I}\}.$$

Then for any vector $x \in P_{\mathcal{B}(\mathcal{M})}$ in the base polytope one has $\sum_{i \in U} x_i \geq |C|$.

Proof. Note that in particular $C \subseteq U$. Moreover, an equivalent definition of U is

$$U = \{i \in (X \setminus S) \cup C \mid \exists j \in C : (S \setminus \{j\}) \cup \{i\} \in \mathcal{I}\}.$$

Due to the integrality of the base polytope, there is a basis $B \in \mathcal{I}$ with $\sum_{i \in U} x_i \geq \sum_{i \in U}(\chi(B))_i = |U \cap B|$, where $\chi(B) \in \{0,1\}^X$ is the characteristic vector of B. As S and B are independent sets with $|S| \leq |B|$, from Lemma 4 there is a left-perfect matching in the exchange graph $H_{\mathcal{M}}(S,B)$. The neighborhood of C in $H_{\mathcal{M}}(S,B)$ is $U \cap B$. As there is a left-perfect matching, $|B \cap U|$ is least $|C|$ and hence $\sum_{i \in U} x_i \geq |U \cap B| \geq |C|$. □

Next, we derive a more general form of the Swapping Lemma (which coincides with the previous Lemma 5 if $D = \emptyset$):

Lemma 6 (Strong Swapping Lemma)**.** *Let $\mathcal{M} = (X,\mathcal{I})$ be a matroid with an independent set $S \in \mathcal{I}$. Let $C \subseteq S$ and $D \subseteq (X \setminus S) \cup C$ with $|D| \leq |C|$ and $S \setminus C \cup D \in \mathcal{I}$. Define*

$$U := \{i \in ((X \setminus S) \cup C) \setminus D \mid S \setminus C \cup D \cup \{i\} \in \mathcal{I}\}.$$

Then for any vector $x \in P_{\mathcal{B}(\mathcal{M})}$ in the base polytope one has $\sum_{i \in U} x_i \geq |C| - |D|$.

Proof. Partition $C = C_1 \dot\cup C_2$ so that $C \cap D \subseteq C_1$, $|C_1| = |D|$ and $S' := S \setminus C_1 \cup D \in \mathcal{I}$. Then note that

$$\begin{aligned} U &= \Big\{i \in X \setminus \underbrace{(S \setminus C \cup D)}_{=S' \setminus C_2} \mid \underbrace{S \setminus C \cup D}_{=S' \setminus C_2} \cup \{i\} \in \mathcal{I}\Big\} \\ &= \{i \in (X \setminus S') \cup C_2 \mid S' \setminus C_2 \cup \{i\} \in \mathcal{I}\}. \end{aligned}$$

Then applying Lemma 5 gives

$$\sum_{i \in U} x_i \geq |C_2| = |C| - |D|.$$

$\square$

Having proved our swapping lemma, we are equipped to prove the Expansion Lemma. Note that in our algorithm, layers are built to ensure that $|A_{\ell+1}| \le |B_{\ell+1}|$. Due to this and the minimality of the edges in $\mathcal{E}_{\alpha T}$ and $\mathcal{E}_{\beta T}$, $W' := A_W \cup B_W$ has $\text{val}(W') \le (\alpha+\beta+\delta)T \cdot |C|$.

Lemma 7 (Expansion Lemma)**.** *Let $C \subseteq S \in \mathcal{I}$, $W' \subseteq W$ with val$(W') \le (\alpha+\beta+\delta)T \cdot |C|$. Further, let $\mu := \frac{1-2\alpha-\beta-\delta}{1+\delta} > 0$ and assume that there exists $(x,y) \in Q(T)$. Then there is a set $D \subseteq (X \setminus S) \cup C$ of size $|D| \ge \lceil \mu \cdot |C| \rceil$ covered by a matching $H \subseteq \mathcal{E}_{\alpha T}$ so that $H_W \cap W' = \emptyset$ and $(S \setminus C) \cup D \in \mathcal{I}$.*

Proof. Note that D may contain elements from C. Greedily choose D and the matching H with $|D| = |H|$ one node/edge after the other. Suppose the greedy procedure gets stuck — no edge can be added without intersecting $W' \cup H_W$. For the sake of contradiction assume this happens when $|D| < \mu|C|$. First, let

$$U := \{i \in ((X \setminus S) \cup C) \setminus D \mid (S \setminus C) \cup D \cup \{i\} \in \mathcal{I}\}$$

be the nodes which could be added to D while preserving independence. Then for our fixed $x \in P_{\mathcal{B}(\mathcal{M})}$, by Lemma 6 one has

$$\sum_{i \in U} x_i \ge |C| - |D| > (1-\mu) \cdot |C|.$$

Let $W'' := W' \cup H_W$ be the right hand side resources that are being covered by the augmenting tree. Here, we let $W' = A_W \cup B_W$. Using the minimality of the adding and blocking edges,

$$\text{val}(W'') \le \mu|C|(\alpha+\delta)T + |C|(\beta+\alpha+\delta)T = |C|T(\mu(\alpha+\delta)+\beta+\alpha+\delta).$$

By the assumption that the greedy procedure is stuck, there is no edge $e \in \mathcal{E}_{\alpha T}$ with $e_X \in U$ and $e \cap W'' = \emptyset$. If $N(i)$ denotes the neighborhood of $i \in X$ in the bipartite graph G, then this means that $\text{val}(N(i) \setminus W'') < \alpha T$ for all $i \in U$. For every fixed $i \in U$ we can then lower bound the y-weight going into W'' as

$$\sum_{(i,w)\in E: w \in W''} p_w y_{i,w} = \underbrace{\sum_{w \in \delta(i)} p_w y_{i,w}}_{\ge T x_i} - \sum_{(i,w)\in E: w \notin W''} p_w \underbrace{y_{i,w}}_{\le x_i} \ge T x_i - x_i \underbrace{\Big(\sum_{(i,w)\in E: w\notin W''} p_w \Big)}_{<\alpha T} \ge T \cdot x_i \cdot (1-\alpha) \quad (*)$$

Then double counting the y-weight running between U and W'' with a lower and upper bound shows that

$$(1-\alpha)T \underbrace{\sum_{i \in U} x_i}_{\ge (1-\mu)|C|} \le \sum_{(i,w)\in E: i \in U, w \in W''} p_w y_{i,w} \le \sum_{w \in W''} p_w \underbrace{y(\delta(w))}_{\le 1} \le \text{val}(W'')$$

Simplifying the above,

$$(1-\alpha)\cdot(1-\mu)\cdot T|C| < (\mu(\alpha+\delta)+\beta+\alpha+\delta)\cdot T|C| \quad \Rightarrow \quad \frac{1-2\alpha-\beta-\delta}{1+\delta} < \mu.$$

Thus we reach a contradiction for our choice of μ. $\square$

The algorithm relies on the fact that from the set of hyperedges, H, guaranteed by the Expansion Lemma, there is either some constant fraction of H to swap into the matching, or a constant fraction of H is blocked by edges in the current matching. In the former, significant space is found in W for S. In the latter, enough edges of the matching are intersected to guarantee the next layer in the augmenting tree is large. The following lemma proves at least one of these conditions occurs.

Lemma 8. *Set $\mu := \frac{\alpha-\beta}{\delta+\alpha} > 0$. Let $M \subseteq \mathcal{E}_{\beta T}$ and $F \subseteq \mathcal{E}_{\alpha T}$ both be hypergraph matchings. Further, let*

$$H := \{e \in F \mid val(e_W \setminus M_W) \geq \beta T\}$$

be the edges in F that still have value βT after overlap with M is removed. Then either (i) $|H| \geq \mu|F|$ or (ii) F intersects at least $\mu|F|$ edges of M.

Proof. Let $W' := M_W \cap F_W$ be the right hand side nodes where the hypermatchings overlap and suppose for the sake of contradiction that neither of the two cases occur. Then double counting the value of W' gives

$$\mu \cdot (\beta+\delta) \cdot T \cdot |F| > (\beta+\delta)T \cdot \underbrace{(\#\text{edges in } M \text{ intersecting } W')}_{\mu|F|>} \geq \text{val}(W') \geq \underbrace{|F \setminus H|}_{\geq(1-\mu)\cdot|F|} \cdot (\alpha-\beta) \cdot T.$$

Rearranging and simplifying, the above implies $\mu > \frac{\alpha-\beta}{\delta+\alpha}$. Thus we contradict our choice of μ. ☐

Our last lemma will show that a constant fraction of the nodes which could be swapped out of the augmenting tree come from the same *layer* in the tree. This allows us to swap out enough nodes from the same layer to make substantial progress with each iteration. Here C' and $\tilde{C}$ are labelled the same as in the algorithm.

Lemma 9. *Let sets C' and $\{B_i\}_{i=0}^{\ell}$ be such that $C' \subset (B_{\leq \ell})_X$. Further, suppose there exists constant $c > 0$ such that $|C'| \geq c \cdot |B_{\leq \ell}|$ and $|B_{i+1}| \geq c \cdot |B_{\leq i}|$ for $i = 0, \ldots, \ell-1$. Then, there exists a layer $0 \leq \tilde{\ell} \leq \ell$ and constant $\gamma := \gamma(c) > 0$, such that $\tilde{C} := C' \cap (B_{\tilde{\ell}})_X$ has size $|\tilde{C}| \geq \gamma \cdot |C'|$.*

Proof. By induction, $|B_{\leq \ell}|$ can be written in terms of lower indexed sets as

$$|B_{\leq \ell}| \geq (1+c)^k \cdot |B_{\leq \ell - k}|,$$

for $k = 0, \ldots, \ell$. Therefore, the size of C' can be written as $|C'| \geq c(1+c)^k \cdot |B_{\leq \ell-k}|$. As c is a constant, take k large enough so $c(1+c)^k \geq 2$, namely $k \geq \frac{\log(\frac{2}{c})}{\log(1+c)}$. Then the collection of sets $(B_{\ell-i})_X$ for $i = 0, \ldots, k$ contain at least half of C', so one of them must contain at least $\gamma = \frac{1}{2(k+1)}$ of C'. ☐

2.4 Termination and runtime

As seen in Lemma 9,

$$|X| \geq |B_{\leq \ell}| \geq \left(1 + \frac{\varepsilon^2}{4}\right)^{\ell} |B_0|,$$

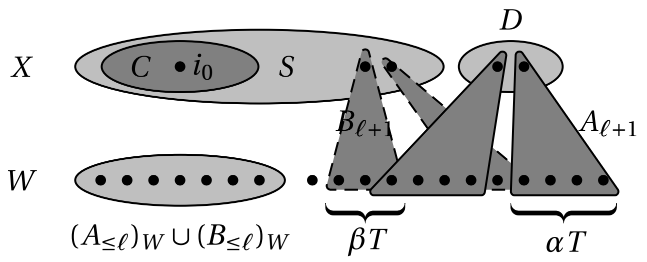

Figure 2: Case 1 of the algorithm, where a set $A_{\ell+1} \subseteq \mathcal{E}_{\alpha T}$ of hyperedges is found that intersects many new edges $B_{\ell+1} \subseteq (M \setminus B_{\leq\ell})$. In particular $|B_{\ell+1}| \geq \Omega_\varepsilon(|C|)$. Note that D might contain nodes from C.

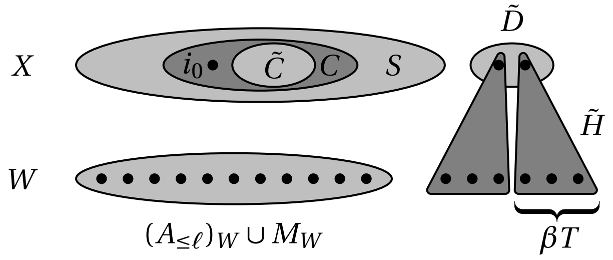

Figure 3: Case 2 of the algorithm, where $\tilde{H} \subseteq \mathcal{E}_{\beta T}$ of size $|\tilde{H}| \geq \Omega_\varepsilon(|C|)$ is found so that (i) $\tilde{H}$ is disjoint on the W-side to the matching M and the adding edges in the augmenting tree, (ii) $\tilde{H}$ covers a set $\tilde{D}$ with $S \setminus \tilde{C} \cup \tilde{D} \in \mathcal{I}$, and (iii) $\tilde{C}$ is from one layer of the augmenting tree. Here $\tilde{D}$ and $\tilde{C}$ do not have to be disjoint.

and solving for ℓ shows $\frac{\log(|X|)}{\log\left(1+\frac{\varepsilon^2}{4}\right)} \geq \ell$. Thus the total number of layers at any step in the algorithm is $O(\log|X|)$. Note after each collapse of the layers, the matching M and possibly the independent set S are updated. However, the fixed exposed node i_0 will remain in S until the very last iteration in which the algorithm finds an edge e_1 that augments the matching. Before we begin discussing the proof guaranteeing our algorithm terminates, we need a lemma to compare the number of blocking edges after a layer is collapsed to the number of blocking edges at the beginning of the iteration.

Lemma 10. *Let $\tilde{\ell}$ be the index of the collapsed layer and let B' be the updated blocking edges after a collapse step. Then, $|B'_{\leq\tilde{\ell}}| \leq |B_{\leq\tilde{\ell}}| \cdot (1 - \frac{\varepsilon^2}{4} \cdot \gamma)$.*

Proof. Recall $B'_{\tilde{\ell}} = B_{\tilde{\ell}} \setminus F$ for F the edges of M covering $\tilde{C}$. Further, the blocking edges in layers indexed less than $\tilde{\ell}$ are not effected in the iteration. Hence

$$|B'_{\leq\tilde{\ell}}| \;=\; |B'_{\leq\tilde{\ell}-1}| + |B'_{\tilde{\ell}}| = |B_{\leq\tilde{\ell}-1}| + |B'_{\tilde{\ell}}|$$

From Lemmas 7 and 8, $|B_{\ell+1}| \geq \frac{\varepsilon^2}{4}|B_{\leq\ell}|$. Then examining the collapsed layer by itself, we see

$$|B'_{\tilde{\ell}}| \;=\; |B_{\tilde{\ell}}| - |F| \leq |B_{\tilde{\ell}}| - \frac{\varepsilon^2}{4} \cdot \gamma |B_{\leq\tilde{\ell}}|.$$

Substituting back into $|B'_{\leq\tilde{\ell}}|$, we find that

$$\begin{aligned} |B'_{\leq\tilde{\ell}}| &\leq |B_{\leq\tilde{\ell}-1}| + |B_{\tilde{\ell}}| - \frac{\varepsilon^2}{4} \cdot \gamma |B_{\leq\tilde{\ell}}| \\ &= |B_{\leq\tilde{\ell}}| - \frac{\varepsilon^2}{4} \cdot \gamma |B_{\leq\tilde{\ell}}| = |B_{\leq\tilde{\ell}}| \cdot \left(1 - \frac{\varepsilon^2}{4} \cdot \gamma\right). \end{aligned}$$

□

To prove the algorithm terminates in polynomial time, we consider a signature vector $s = (s_0, s_1, \ldots, s_\ell, \infty)$, where $s_j = \lfloor \log_c |B_{\leq j}| \rfloor$ for $c = \frac{1}{1-\frac{\varepsilon^2}{4}\cdot\gamma}$. The signature vector and proof that the algorithm terminates is inspired by [AKS15], but it is subtly different.

Lemma 11. *The signature vector decreases lexicographically after each iterative loop in the algorithm.*

Proof. Let $s = (s_0, \ldots, s_\ell, \infty)$ be a signature vector at the beginning of a step in the algorithm, and let s' be the result of s through one iteration of the algorithm. For $\ell + 1$ denoting the newest built layer in the algorithm, if the newest set of hyperedges found intersects at least $\frac{\varepsilon^2}{4}|C|$ many edges of M, then another layer in the augmenting tree is built and no layer is collapsed. Then $s' = (s_0, \ldots, s_\ell, s_{\ell+1}, \infty)$ is lexicographically smaller than s.

Otherwise, layer $0 \leq \tilde{\ell} \leq \ell$ is collapsed. All finite coordinates above $s_{\tilde{\ell}}$ are deleted from the signature vector, and all coordinates before $s_{\tilde{\ell}}$ are unaffected. So it suffices to check that $s'_{\tilde{\ell}} < s_{\tilde{\ell}}$. Again, let B' be the updated blocking edges after a collapse step. As $B_{\tilde{\ell}}$ is the only set of blocking edges in $B_{\leq\tilde{\ell}}$ affected by the collapse, by Lemma 10 one has $|B'_{\leq\tilde{\ell}}| \leq |B_{\leq\tilde{\ell}}|(1-\frac{\varepsilon^2}{4}\cdot\gamma)$. Taking a log we compare the coordinates

$$s'_{\tilde{\ell}} = \left\lfloor \log_c\left(\left|B'_{\leq\tilde{\ell}}\right|\right)\right\rfloor \leq \left\lfloor \log_c\left(\left|B_{\leq\tilde{\ell}}\right|\left(1-\frac{\varepsilon^2}{4}\cdot\gamma\right)\right)\right\rfloor = \left\lfloor \log_c\left(\left|B_{\leq\tilde{\ell}}\right|\right)\right\rfloor - 1 = s_{\tilde{\ell}} - 1.$$

□

Choose the infinite coordinate to be some integer larger than $\log|X|$. Since for every layer ℓ, we have $|B_{\leq\ell}| \leq |X|$, then every coordinate of the signature vector is upper bounded by $U = O(\log|X|)$. Recall the number of layers, and thus the number of coordinates in the signature vector, is also upper bounded by U. Together, these imply that the sum of the coordinates of the signature vector is at most U^2.

As the signature vector has non-decreasing order, each signature vector corresponds to a partition of an integer $z \leq U^2$. On the other hand, every partition of some $z \leq U^2$ has a corresponding signature vector. Thus we apply a result of Hardy and Ramanujan to find the total number of signature vectors is $\sum_{k\leq U^2} e^{O(\sqrt{k})} = |X|^{O(1)}$. Since each iteration of the algorithm can be done in polynomial time and the signature vector decreases lexicographically after each iteration, the algorithm terminates after a total time of $n^{\Theta_\varepsilon(1)}$.

3 Application to Santa Claus

In this section, we show a polynomial time $(4+\varepsilon)$-approximation algorithm for the Santa Claus problem. Recall that for a given set of children M, and a set of presents J, the Santa Claus problem asks how Santa should distribute presents to children in order to maximize the minimum happiness of any child[4]. Here, present j is only wanted by some subset of children that we denote by $A_j \subseteq M$, and present j has value p_j to child $i \in A_j$. The happiness of child i is the sum of all p_j for presents j assigned to child i. We assume w.l.o.g. to know the integral objective function value T of the optimum solution, otherwise T can be found by binary search.

We partition gifts into two sets: *large* gifts $J_L := \{j \in J \mid p_j > \delta_2 T\}$ and *small* gifts $J_S := \{j \in J \mid p_j \leq \delta_1 T\}$, for parameters $0 < \delta_1 \leq \delta_2 < 1$ such that all gifts have values in $[0, \delta_1 T] \cup (\delta_2 T, T]$. Let $P(T, \delta_1, \delta_2)$ be the set of vectors $z \in \mathbb{R}_{\geq 0}^{J\times M}$ satisfying

$$\begin{array}{rcll} \sum_{j\in J_S : i\in A_j} p_j z_{ij} & \geq & T\cdot\Big(1 - \sum_{j\in J_L : i\in A_j} z_{ij}\Big) & \forall i \in M \\ \sum_{i\in A_j} z_{ij} & \leq & 1 & \forall j \in J \\ z_{ij} & \leq & 1 - \sum_{j'\in J_L : i\in A_{j'}} z_{ij'} & \forall j \in J_S\ \forall i \in A_j \end{array}$$

If $n = |J| + |M|$, then this LP has $O(n^2)$ many variables and $O(n^2)$ many constraints. To see that this is indeed a relaxation, take any feasible assignment $\sigma : J \to M$ with $\sum_{j\in\sigma^{-1}(i)} p_j \geq T$ for all $i \in M$. Now let $\sigma : J \to M \cup \{\emptyset\}$ be a modified assignment where we set $\sigma(j) = \emptyset$ for gifts that

[4]We assume Santa to be an equitable man– not one influenced by bribery, social status, etc.

we decide to drop. For each child $i \in M$ that receives at least one large gift we drop all small gifts and all but one large gift. Then a feasible solution $z \in P(T, \delta_1, \delta_2)$ is obtained by letting

$$z_{ij} := \begin{cases} 1 & \text{if } \sigma(j) = i \\ 0 & \text{otherwise.} \end{cases}$$

We will show that given a feasible solution $z \in P(T, \delta_1, \delta_2)$, there exists a feasible solution (x^*, y^*) to $Q(T)$. To do this, we will exploit two underlying matroids in the Santa Claus problem, allowing us to apply Theorem 2. Let

$$\mathcal{I} = \left\{ M_L \subseteq M \mid \exists \text{ left-perfect matching between } M_L \text{ and } J_L \text{ using edges } (i,j) : i \in A_j \right\},$$

be a family of independent sets. Then $\mathcal{M} = (M, \mathcal{I})$ constitutes a *matchable set matroid*. We denote the *co-matroid* of $\mathcal{M}$ by $\mathcal{M}^* = (M, \mathcal{I}^*)$. Recall that the independent sets of the co-matroid are given by

$$\mathcal{I}^* = \left\{ M_S \subseteq M \mid \exists M_L \in \mathcal{B}(\mathcal{M}) : M_S \cap M_L = \emptyset \right\}.$$

We can define a vector $x \in \mathbb{R}^M$ with $x_i = \sum_{j \in J_L : i \in A_j} z_{ij}$ that lies in the matroid polytope of $\mathcal{M}$. This fact follows easily from the integrality of the fractional matching polytope in bipartite graphs. It is instructive to think of x_i as the decision variable telling whether child $i \in M$ should receive a large present.

Unfortunately, x does not have to lie in the base polytope — in fact the sum $\sum_{i \in M} x_i$ might not even be integral. However, there always exists a vector x' in the base polytope that covers every child just as well with large presents as x does. This observation can be stated for general matroids:

Lemma 12. *Let $\mathcal{M} = (X, \mathcal{I})$ be any matroid and let x be a point in its matroid polytope. Then in polynomial time one can find a point x' in the base polytope so that $x' \geq x$ coordinate-wise.*

In fact the algorithm behind this claim is rather trivial: as long as $x \in P_{\mathcal{M}}$ is not in the base polytope, there is always a coordinate i and a $\mu > 0$ so that $x + \mu e_i \in P_{\mathcal{M}}$.

With the new vector $x' \in P_{\mathcal{B}(\mathcal{M})}$ at hand, we can redefine the z-assignments by letting

$$z'_{ij} = \begin{cases} z_{ij} & x_i = 1 \\ \frac{1 - x'_i}{1 - x_i} z_{ij} & x_i \neq 1. \end{cases}$$

for $j \in J_S$; the new values z'_{ij} for $j \in J_L$ can be obtained from the fractional matching that corresponds to x'_i. Note that $0 \leq z'_{ij} \leq z_{ij}$ for $j \in J_S$. The reader should be convinced that still $z' \in P(T, \delta_1, \delta_2)$, just that the corresponding vector x' now lies in $P_{\mathcal{B}(\mathcal{M})}$[5].

It is well known in matroid theory that the complementary vector $x^* := \mathbf{1} - x'$ lies in $P_{\mathcal{B}(\mathcal{M}^*)}$. Again, it is instructive to think of x^*_i as the decision variable whether child i has to be satisfied with small gifts. Finally, the assignments y^* are simply the restriction of z' on the coordinates

[5]There is an alternative proof without the need to replace x by x'. Add the constraint $\sum_{j \in J_L, i \in A_j} z_{ij} = \text{rank}(\mathcal{M})$ to $P(T, \delta_1, \delta_2)$. There is always a feasible integral solution satisfying this constraint. Then for any fractional solution $z \in P(T, \delta_1, \delta_2)$, the corresponding vector x will immediately lie in the base polytope.

$(i,j) \in M \times J_S$. The obtained pair (x^*, y^*) lies in $Q(T)$, where the matroid in the definition of $Q(T)$ is $\mathcal{M}^*$.

As $Q(T) \neq \emptyset$, we can apply Theorem 2 which results in a subset $M_S \in \mathcal{B}(\mathcal{M}^*)$ of the children and an assignment $\sigma : J_S \to M_S$, where each child in M_S receives happiness at least $\left(\frac{1}{3} - \frac{\delta_1}{3} - \varepsilon\right) \cdot T$ from the assignment of small gifts. Implicitly due to the choice of the matroid $\mathcal{M}^*$, we know that the remaining children $M \setminus M_S = M_L$ can all receive one large gift and this assignment can be computed in polynomial time using a matching algorithm. Overall, each child receives either one large present of value at least $\delta_2 \cdot T$ or small presents of total value at least $(\frac{1}{3} - \frac{\delta_1}{3} - \varepsilon) \cdot T$. Therefore each child receives value at least

$$\min\left\{\left(\frac{1}{3} - \frac{\delta_1}{3} - \varepsilon\right) \cdot T, \delta_2 \cdot T\right\} \geq \left(\frac{1}{4} - \varepsilon\right) \cdot T \tag{1}$$

for a choice of $\delta_2 = \delta_1 = \frac{1}{4}$. In some instances of Santa Claus, we can do better. Set δ_1 so that $\delta_1 \cdot T$ is the largest gift value that is at most $\frac{1}{4}T$, and set δ_2 so that $\delta_2 \cdot T$ is the smallest gift value that is at $\frac{1}{4}T$. Then the algorithm guarantees that each child receives value at least as in the left hand side of Equation 1. When δ_1 and δ_2 are bounded away from $1/4$, then the approximation improves. For example, when $\delta_2 \geq 1/3$ and $\delta_1 T$ is close to 0, such as in the case where all gifts have value either T or 1, we approach a $(3+\varepsilon)$-approximation.

References

[AFS08] Arash Asadpour, Uriel Feige, and Amin Saberi. Santa claus meets hypergraph matchings. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings*, pages 10–20, 2008.

[AFS12] Arash Asadpour, Uriel Feige, and Amin Saberi. Santa claus meets hypergraph matchings. *ACM Trans. Algorithms*, 8(3):24:1–24:9, July 2012.

[AKS15] Chidambaram Annamalai, Christos Kalaitzis, and Ola Svensson. Combinatorial algorithm for restricted max-min fair allocation. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1357–1372, 2015.

[Ann16] Chidambaram Annamalai. Finding perfect matchings in bipartite hypergraphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1814–1823, 2016.

[BS06] Nikhil Bansal and Maxim Sviridenko. The santa claus problem. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 31–40, New York, NY, USA, 2006. ACM.

[CCK09] Deeparnab Chakrabarty, Julia Chuzhoy, and Sanjeev Khanna. On allocating goods to maximize fairness. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 107–116, 2009.

[CM18a] Siu-Wing Cheng and Yuchen Mao. Integrality gap of the configuration lp for the restricted max-min fair allocation. *arXiv preprint arXiv:1807.04152*, 2018.

[CM18b] Siu-Wing Cheng and Yuchen Mao. Restricted max-min fair allocation. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 37:1–37:13, 2018.

[EKS08] Tomás Ebenlendr, Marek Krcál, and Jirí Sgall. Graph balancing: a special case of scheduling unrelated parallel machines. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 483–490, 2008.

[Fei08] Uriel Feige. On allocations that maximize fairness. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 287–293, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[Hax95] Penny E. Haxell. A condition for matchability in hypergraphs. *Graphs and Combinatorics*, 11(3):245–248, 1995.

[JR17] Klaus Jansen and Lars Rohwedder. On the configuration-lp of the restricted assignment problem. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2670–2678, 2017.

[JR18a] Klaus Jansen and Lars Rohwedder. Compact LP relaxations for allocation problems. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, pages 11:1–11:19, 2018.

[JR18b] Klaus Jansen and Lars Rohwedder. Local search breaks 1.75 for graph balancing. *CoRR*, abs/1811.00955, 2018.

[JR18c] Klaus Jansen and Lars Rohwedder. A note on the integrality gap of the configuration LP for restricted santa claus. *CoRR*, abs/1807.03626, 2018.

[LST87] Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, pages 217–224, 1987.

[Sve11] Ola Svensson. Santa claus schedules jobs on unrelated machines. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 617–626, 2011.

[VW11] José Verschae and Andreas Wiese. On the configuration-lp for scheduling on unrelated machines. In *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, pages 530–542, 2011.