been studied under the multidimensional setting [18, 19]. The max-knapsack problem when the fairness constraint is an upper bound on weight is a special case of multidimensional knapsack problem. Another variant of the knapsack problem is the submodular knapsack problem, where the value function is submodular [29, 32, 42]. The knapsack problem has also been well-studied under an online setting [1]. For different variants of knapsack and related bin packing problems, we refer the readers to [14, 28].

Min-knapsack problem. The min-knapsack problem is a frequently encountered problem in optimization. The problem admits an FPTAS [15, 35]. As the problem appears as a key substructure of numerous other optimization problems, the polyhedral study of this problem has led to the development of important tools, such as the knapsack cover inequalities and there is a rich connection of the problem with extension complexity and sum-of-squares hierarchy [4, 16, 30].

Class-constrained knapsack and fair knapsack. To the best of our knowledge, the fairness notions for knapsack problems studied in this work have not been studied previously. The closest model to our model that has been studied before, is the class-constrained multiple knapsack problem [40]. This problem restricts the maximum number of classes from which items could be packed in a knapsack. The algorithm of the class constrained knapsack problem [40] uses two levels of dynamic programming, and is similar in spirit to our algorithm.

Fairness notions for knapsack under multi-agent valuations have also been studied [17]. They have several approaches to aggregate voters' preferences: individually best, diverse knapsack and fair knapsack. The main difference is that our model ensures fairness through constraints, while their model ensures fairness through objective function. The objective function used in the fair knapsack model in [17] is the Nash welfare function [34]. The diverse knapsack model in [17] only ensures one representative item from each category, whereas our model allows different lower and upper bounds for different groups. We focus on approximation schemes for our problems, whereas [17] focused on the parameterized complexity and the complexity of the problem under some restricted special domains.

Govind et al. [37] study some notion of group fairness for online/offline matching problem. Their notions of fairness are similar to ours in some respect.

3 ALGORITHM FOR BOUND ON VALUE IN MAX-KNAPSACK

Notations. Let the set $\{1, 2, \ldots, r\}$ be denoted by $[r]$. Let m denote the number of categories. The category number is denoted from the set $[m]$. Each input item belongs to some category. Let V_i denote the set of all items from category i, $\forall i \in [m]$. For $i \in [m]$, if V_i has k items, then $V_i := [k]$. Also, assume that $w_j^{(i)}$ and $v_j^{(i)}$ are the weight and the value of item $j \in V_i$, respectively, $\forall i \in [m]$. Also let $\max\{| V_i | \mid i \in [m]\} = n$. Let N be the total number of items. By our notations, $N \leq nm$.

optimal (or $1 - \varepsilon$ for maximization problems). A fully polynomial-time approximation scheme (FPTAS) is a PTAS with running time polynomial in both input size and $\frac{1}{\varepsilon}$.

Now, we formally define the fair max-knapsack problem with bound on total value of items from each category. We refer this problem by the acronym BV^{max}.

Problem 1 (BV^{max}). *Given a set of items, each belonging to one of m categories, a lower bound $l_i^v \geq 0$ and an upper bound u_i^v such that $u_i^v \geq l_i^v$ for each category i, the goal is to find a subset that maximizes the total value, such that the total value of items from the category i is in between l_i^v and u_i^v, $\forall i \in [m]$, and the total weight of the subset is at most the knapsack capacity B.*

We prove that it is even NP-hard to obtain a feasible solution of an instance of BV^{max} (Problem 1).

Theorem 3.1. *There is no polynomial time algorithm that outputs a feasible solution of BV^{max} (Problem 1), assuming $P \neq NP$.*

Proof. Subset sum is a classical NP-hard problem, where we are given a set I of non-negative rationals such that $\sum_{a \in I} a = 1$, and the goal is to find a subset $S \subseteq I$ such that $\sum_{a \in S} a = \frac{1}{2}$. Now given an instance of subset sum with the set I, we will construct an instance of BV^{max} (Problem 1). Let $m = 1$, $l_1^v = \frac{1}{2}$, $u_1^v = 1$, $B = \frac{1}{2}$. Let V_1 be the set of items in the instance of BV^{max}. The items in V_1 correspond to the numbers in I. An item corresponding to some $a \in I$ has a value and a weight equal to a. This proves that if we have an algorithm that doesn't violate the fairness constraint, then we can solve subset sum. But assuming P ≠ NP, this is not possible. □

We now give an algorithm for BV^{max} (Problem 1) which might violate fairness constraint for a category by small fraction. The algorithm uses dynamic programming tables. The entry in the dynamic programming tables represents the total weight of a subset of items. The respective subsets can be obtained by maintaining appropriate references in the tables. Details of it are trivial and hence they are not discussed in this paper.

Theorem 3.2. *For any $\varepsilon > 0$, there exists an algorithm for BV^{max} (Problem 1) that outputs a set S having total value at least $1 - \varepsilon$ times the optimal value of BV^{max}, such that $(1 - \varepsilon)l_k^v \leq \sum_{r \in S \cap V_k} v_r^{(k)} \leq (1+\varepsilon)u_k^v, \forall k \in [m]$, and the total weight of S is at most B. The running time of the algorithm is $O\left(\frac{n^2 m \log^3 m \log^3_{1+\varepsilon}\left(\frac{N v_{max}}{v_{min}}\right)}{\varepsilon}\right)$, where $v_{min} := \min\left\{v_j^{(i)} \mid i \in [m] \ \& \ j \in V_i \ \& \ v_j^{(i)} > 0\right\}$. and $v_{max} := \max\left\{v_j^{(i)} \mid i \in [m] \ \& \ j \in V_i\right\}$.*

Towards proving this theorem, we first study the following subproblem.

Problem 2. *Given $v > 0$, $\varepsilon > 0$ and a set $V' = [n]$ of items, with item $i \in V'$ having the value v_i' and the weight w_i', compute a subset $S \subseteq V'$ minimizing $\sum_{i \in S} w_i'$, such that $(1 - \varepsilon)v \leq \sum_{i \in S} v_i' \leq (1 + \varepsilon)v$.*

Problem 2 looks similar to the min-knapsack problem but it is different from it in the following way. The total value of output of min-knapsack problem is required to be more than the given lower bound, while the total value of the output of Problem 2 is required to be lying in a small range. We will use Algorithm 1 for Problem 2 to obtain bundles of items in V_i, $\forall i \in [m]$, such that the total value of different bundles are in different required ranges.

Algorithm 1: Algorithm for Problem 2

Input: $v > 0$, $\varepsilon > 0$, a set $V' = [n]$ of items, with item $i \in V'$ having value v_i' and weight w_i'.
Output: A subset $S \subseteq V'$ having total weight at most the optimal weight for Problem 2, such that $(1-3\varepsilon)v \le \sum_{i \in S} v_i' \le (1+3\varepsilon)v$.

(1) Remove all items from V' of value more than $(1+\varepsilon)v$.
(2) For each item $i \in V'$, let $v_i'' := \left\lfloor \frac{n v_i'}{\varepsilon v} \right\rfloor$.
(3) Let $\mathcal{H}(i, v'')$ for $v'' \in \left[\left\lceil \frac{(1+2\varepsilon)n}{\varepsilon} \right\rceil\right] \cup \{0\}$ and $i \in [n]$, be the DP table constructed in the following way.
  (a) $\mathcal{H}(1, v_1'') = w_1'$. $\mathcal{H}(1, 0) = 0$. $\mathcal{H}(1, v'') = \infty$, $\forall v'' \in \left[\left\lceil \frac{(1+2\varepsilon)n}{\varepsilon} \right\rceil\right] \setminus \{v_1''\}$.
  (b) $\forall i \in [n] \setminus \{1\}$, $\forall v'' \in \left[\left\lceil \frac{(1+2\varepsilon)n}{\varepsilon} \right\rceil\right]$,
  If $v'' < v_i''$, then $\mathcal{H}(i, v'') = \mathcal{H}(i-1, v'')$.
  Else,
$$\mathcal{H}(i, v'') := \min\{\mathcal{H}(i-1, v'' - v_i'') + w_i', \mathcal{H}(i-1, v'') \mid v_i'' \le v''\}.$$
(4) Output the subset S in the following way,
$$\min\left\{\mathcal{H}(n, v'') \mid \left\lfloor \frac{(1-2\varepsilon)n}{\varepsilon} \right\rfloor \le v'' \le \left\lfloor \frac{(1+2\varepsilon)n}{\varepsilon} \right\rfloor\right\}.$$

Theorem 3.3. *For any $\varepsilon > 0$ and $v > 0$, there exists an algorithm for Problem 2 that outputs a subset S having the total weight at most the optimal weight of Problem 2, and $(1-3\varepsilon)v \le \sum_{i \in S} v_i' \le (1+3\varepsilon)v$.*

Proof. The algorithm for the theorem is described in Algorithm 1. We give the proof of correctness of the algorithm below.

We claim that the entry $\mathcal{H}(i, v'')$,$\forall i \in [n]$, $\forall v'' \in \left[\left\lceil \frac{(1+2\varepsilon)n}{\varepsilon} \right\rceil\right] \cup \{0\}$, indicates the weight of the least weight subset from the first i items of V' having the total rounded value equal to v''. Let S' denote the subset satisfying above property for the entry $\mathcal{H}(i, v'')$. We can have two cases as below.

- If $i \in S'$, then $\mathcal{H}(i, v'')$ is the sum of the weight of i (w_i'), and the weight of minimum weight subset from first $i-1$ items having the total rounded value $v'' - v_i''$ ($\mathcal{H}(i-1, v'' - v_i'')$).
- If $i \notin S'$, then $\mathcal{H}(i, v'')$ is equal to the weight of minimum weight subset from first $i-1$ items having the total rounded value v'', i.e. $\mathcal{H}(i, v'') = \mathcal{H}(i-1, v'')$.

The recursion in Step 3b captures both of above possibilities. Step 3a does necessary initialization for the recursion in Step 3b. Let O denote the set of items in an optimal solution of Problem 2 and S be the set output by Step 4 of Algorithm 1. By the construction of the table $\mathcal{H}$, we have

$$\left\lfloor \frac{(1-2\varepsilon)n}{\varepsilon} \right\rfloor \le \sum_{i \in S} v_i'' \le \left\lfloor \frac{(1+2\varepsilon)n}{\varepsilon} \right\rfloor \tag{1a}$$

$$\Rightarrow \frac{(1-2\varepsilon)n}{\varepsilon} - 1 \le \sum_{i \in S} v_i'' \le \frac{(1+2\varepsilon)n}{\varepsilon}. \tag{1b}$$

Using the definition of v_i'' of Step 2, we obtain from the left inequality of (1b),

$$(1-2\varepsilon)v - \frac{\varepsilon v}{n} \le \sum_{i \in S} v_i'$$
$$\Rightarrow (1-3\varepsilon)v \le \sum_{i \in S} v_i'.$$

From right inequality of (1b),

$$\sum_{i \in S} (v_i'' + 1) \le \frac{(1+2\varepsilon)n}{\varepsilon} + n$$
$$\Rightarrow \sum_{i \in S} v_i' \le (1+3\varepsilon)v. \qquad \text{(Definition of } v_i''\text{)}$$

Now, we will prove that $\left\lfloor \frac{(1-2\varepsilon)n}{\varepsilon} \right\rfloor \le \sum_{i \in O} v_i'' \le \left\lfloor \frac{(1+2\varepsilon)n}{\varepsilon} \right\rfloor$. Since S is the least weight subset in the previous range, the above claim will imply that the total weight of S is at most the total weight of O. We know that $(1-\varepsilon)v \le \sum_{i \in O} v_i' \le (1+\varepsilon)v$. Using the definition of v_i'' in Step 2, we get

$$\frac{n(1-\varepsilon)}{\varepsilon} - n \le \sum_{i \in O} v_i'' \le \frac{(1+\varepsilon)n}{\varepsilon} + n$$
$$\Rightarrow \frac{n(1-2\varepsilon)}{\varepsilon} \le \sum_{i \in O} v_i'' \le \frac{(1+2\varepsilon)n}{\varepsilon}$$
$$\Rightarrow \left\lfloor \frac{n(1-2\varepsilon)}{\varepsilon} \right\rfloor \le \sum_{i \in O} v_i'' \le \left\lfloor \frac{(1+2\varepsilon)n}{\varepsilon} \right\rfloor.$$

The last inequality is true by the fact that v_i'' is an integer, $\forall i \in V'$. Note that, due to the initialization in Step 3a, the algorithm returns ∞, if it does not find a subset in Step 4.

Running time analysis. The size of the table $\mathcal{H}$ is $O\left(\frac{n^2}{\varepsilon}\right)$. We require $O(1)$ time to fill each entry. So, the total running time of the algorithm is $O\left(\frac{n^2}{\varepsilon}\right)$.

□

Proof of Theorem 3.2. The algorithm for theorem is described in Algorithm 2. The algorithm creates bundles of items from V_i, $\forall i \in [m]$, such that the total value of each bundle is in different ranges using Theorem 3.3 (Step 2). It stores these bundles in the table $\mathcal{W}$. After that the algorithm combines bundles from all categories in divide and conquer fashion to obtain the final solution using the dynamic programming table $\mathcal{X}$ (Step 3). The total value of each bundle is represented by some power of $(1+\varepsilon')$ in the tables $\mathcal{W}$ and $\mathcal{X}$. So, the algorithm might lose at most $(1+\varepsilon')$ fraction of total value in one iteration of Step 3. The total fraction of value lost in the calculation after combining the bundles from all the categories in Step 3b is at most $(1+\varepsilon')^{O(\log_2 m)}$. This is at most $(1+\varepsilon)$ because of the choice of ε' in Step 1. We describe the formal proof of the algorithm below.

Properties of $\mathcal{W}$. We claim that the entry $\mathcal{W}(i, j)$, $\forall i \in [m]$, $j \in W_{range}$, indicates the weight of the subset of V_i that satisfies the two properties listed below. The entry of $\mathcal{W}$ also corresponds to the respective subset. The entry of $\mathcal{W}$ could be ∞, which indicates no subset. We use the notation $\mathcal{W}(i, j)$ to indicate both the entry and the subset.

Algorithm 2: Algorithm for the BV^{max} (Problem 1)

Input: The sets of items: V_i for all $i \in [m]$, $0 \le l_i^v \le u_i^v$ for all $i \in [m]$, the knapsack capacity B, and $\varepsilon > 0$.

Output: S having the total value at least $1-\varepsilon$ times the optimal value of BV^{max}, such that $(1-\varepsilon)l_i^v \le \sum_{r\in S\cap V_i} v_r^{(i)} \le (1+\varepsilon)u_i^v, \forall i \in [m]$, and the total weight of S is at most B.

(1) Let $\varepsilon' = \left(1+\frac{3\varepsilon}{8}\right)^{\frac{1}{\log_2 m+1}} - 1$. Also let $W_{range} := \left[\left\lceil \log_{(1+\frac{\varepsilon}{8})}\left(\frac{nv_{max}}{v_{min}}\right)\right\rceil\right] \cup \{0\}$ and $X_{range} := \left[\left\lceil \log_{1+\varepsilon'}\left(\frac{Nv_{max}}{v_{min}}\right)\right\rceil\right] \cup \{-\log_2 m, \dots, -1, 0\}$.

(2) Let $\mathcal{W}(i,j), \forall i \in [m], \forall j \in W_{range}$, be the table, where the entry $\mathcal{W}(i,j)$ indicates the weight of a subset of V_i that is obtained by Theorem 3.3 by setting V_i as V', $v_{min}\left(1+\frac{\varepsilon}{8}\right)^j$ as v, and $\frac{\varepsilon}{6}$ as ε in Theorem 3.3.

(3) Let $\mathcal{X}(i,j), \forall i \in [2m-1], \forall j \in X_{range}$, be the DP table constructed as follows.

(a) $\forall i \in [m], \forall j \in X_{range}$,

$$\mathcal{X}(m-1+i,j) := \min\left\{\mathcal{W}(i,j'') \mid \left(1+\frac{\varepsilon}{8}\right)^{j''} \ge (1+\varepsilon')^j \right. \\ \& \ v_{min}\left(1+\frac{\varepsilon}{8}\right)^{j''+1} \ge l_i^v \ \& \\ v_{min}\left(1+\frac{\varepsilon}{8}\right)^{j''-1} \le u_i^v \ \& \\ \left. j'' \in W_{range}\right\}.$$

If the set satisfying above condition is empty, then set $\mathcal{X}(i,j)$ to ∞.

(b) $\forall i \in [m-1], \forall j', j'' \in X_{range}$, we have

$$\mathcal{X}(i,j) := \min\left\{\mathcal{X}(2i,j') + \mathcal{X}(2i+1,j'') \mid \right. \\ \left. (1+\varepsilon')^j \le (1+\varepsilon')^{j'} + (1+\varepsilon')^{j''}\right\}.$$

If the set satisfying above condition is empty, then set $\mathcal{X}(i,j)$ to ∞.

(4) Output the subset S in the following way,

$$\mathrm{argmax}_j\{\mathcal{X}(1,j) \mid \mathcal{X}(1,j) \le B\}.$$

(1) If the entry $\mathcal{W}(i,j)$ is finite, then the total value of the corresponding subset is in between $\left(1-\frac{\varepsilon}{2}\right)\left(1+\frac{\varepsilon}{8}\right)^j v_{min}$ and $\left(1+\frac{\varepsilon}{2}\right)\left(1+\frac{\varepsilon}{8}\right)^j v_{min}$.
(2) The total weight of the subset corresponding to $\mathcal{W}(i,j)$ is at most the total weight of any subset of V_i having the total value in between $v_{min}\left(1-\frac{\varepsilon}{6}\right)\left(1+\frac{\varepsilon}{8}\right)^j$ and $v_{min}\left(1+\frac{\varepsilon}{6}\right)\left(1+\frac{\varepsilon}{8}\right)^j$ (Since $\frac{\varepsilon}{8} < \frac{\varepsilon}{6}$, the total weight of $\mathcal{W}(i,j)$ will be less than or equal to the total weight of any subset of V_i having the total value in between $\left(1+\frac{\varepsilon}{8}\right)^j v_{min}$ and $\left(1+\frac{\varepsilon}{8}\right)^{j+1} v_{min}$).

The table $\mathcal{W}$ is created in Step 2 of the algorithm. This step uses Theorem 3.3. We get both above properties because of the guarantee of Theorem 3.3.

Let $\mathcal{T}$ be a perfect binary tree with m leaf nodes. For simplicity, assume that m is power of 2. Although, we can prove the same result by slight modification of the proof when m is not power of 2. The total number of nodes in $\mathcal{T}$ will be $2m-1$. Each node in $\mathcal{T}$ could be represented by an index number from 1 to $2m-1$ with root at index 1. The node at an index i has a left child at an index $2i$ and right child at an index $2i+1$, $\forall i \in [m-1]$. Let the leaf node at an index $(m-1)+i$ represent the category i, $\forall i \in [m]$. Let $\mathcal{T}(i)$ denote the set of categories represented by the leaves of sub tree rooted at i. Specifically, $\mathcal{T}(m-1+i) = \{i\}$, $\forall i \in [m]$. Also, $\mathcal{T}(i) = \mathcal{T}(2i) \cup \mathcal{T}(2i+1)$, $\forall i \in [m-1]$.

Properties of $\mathcal{X}$. We claim that the entry $\mathcal{X}(i,j)$, $\forall i \in [2m-1]$, $\forall j \in X_{range}$, indicates the weight of a subset of $\cup_{k\in\mathcal{T}(i)} V_k$ that satisfies three properties mentioned below. The entry of $\mathcal{X}$ also corresponds to the respective subset. The entry of $\mathcal{X}$ could be ∞, which indicates no subset. We use the notation $\mathcal{X}(i,j)$ to indicate both the entry and the subset.

(1) If the entry $\mathcal{X}(i,j)$ is finite, then

$$\sum_{k\in\mathcal{T}(i)} \sum_{r\in\mathcal{X}(i,j)\cap V_k} v_r^{(k)} \ge \left(1-\frac{\varepsilon}{2}\right)(1+\varepsilon')^j v_{min}$$

.

(2) If the entry $\mathcal{X}(i,j)$ is finite, then $\forall k \in \mathcal{T}(i)$,

$$\left(1-\frac{\varepsilon}{2}\right)\left(1+\frac{\varepsilon}{8}\right)^{-1} l_k^v \le \sum_{r\in\mathcal{X}(i,j)\cap V_k} v_r^{(k)} \le \left(1+\frac{\varepsilon}{2}\right)\left(1+\frac{\varepsilon}{8}\right) u_k^v.$$

(3) Let i be a node of $\mathcal{T}$, $\forall i \in [2m-1]$, having the distance t from leaves, $t \in \left[\log_2 m\right] \cup \{0\}$. For all $O' \subseteq \cup_{k\in\mathcal{T}(i)} V_k$ having the total value at least $(1+\varepsilon')^{j+t} v_{min}$, and $l_k^v \le \sum_{r\in O'\cap V_k} v_r^{(k)} \le u_k^v$, $\forall k \in \mathcal{T}(i)$, the total weight of the subset $\mathcal{X}(i,j)$ is at most the total weight of O'.

In Property 3, any considered set O' will have value at least v_{min}. Thus in Property 3, $t+j \ge 0$, i.e. $j \ge -\log_2 m$. Hence, the minimum value in X_{range} has been set to $-\log_2 m$ (Step 1).

Steps 3a of the algorithm chooses the subset $\mathcal{W}(i,j'')$ that satisfies the inequalities $\left(1+\frac{\varepsilon}{8}\right)^{j''+1} v_{min} \ge l_i^v$ and $\left(1+\frac{\varepsilon}{8}\right)^{j''-1} v_{min} \le u_i^v$. By Property 1 of $\mathcal{W}$, the total value of $\mathcal{W}(i,j'')$ is in between $\left(1-\frac{\varepsilon}{2}\right)\left(1+\frac{\varepsilon}{8}\right)^{-1} l_i^v$ and $\left(1+\frac{\varepsilon}{2}\right)\left(1+\frac{\varepsilon}{8}\right) u_i^v$. This proves that the subset corresponding to finite entry $\mathcal{X}(i,j)$, $\forall i \in [2m-1]$, $\forall j \in X_{range}$, satisfies Property 2.

The subset corresponding to finite entry $\mathcal{X}(m-1+i,j)$, $\forall i \in [m]$ $\forall j \in X_{range}$, will satisfy Property 1. This is true because of Property 1 of $\mathcal{W}$ and the condition $\left(1+\frac{\varepsilon}{8}\right)^{j''} \ge (1+\varepsilon')^j$ in Step 3a for selecting subset $\mathcal{W}(i,j'')$. Because of the condition $(1+\varepsilon')^j \le (1+\varepsilon')^{j'} + (1+\varepsilon')^{j''}$ in Step 3b of the algorithm, the subset corresponding to finite entry $\mathcal{X}(i,j)$, $\forall i \in [m-1]$, $\forall j \in X_{range}$, will satisfy Property 1.

We prove that the nodes in $\mathcal{T}$ will satisfy Property 3 by induction. In the base case, we prove that Property 3 is satisfied for all leaves. Let O'' be any subset of V_i that satisfies the fairness bounds such that the total value of O'' is at least $v_{min}(1+\varepsilon')^j$. Let $j'' \in W_{range}$ such that the total value of O'' is in between $v_{min}\left(1+\frac{\varepsilon}{8}\right)^{j''-1}$ and $v_{min}\left(1+\frac{\varepsilon}{8}\right)^{j''}$. So, the total value of O'' is in between $v_{min}\left(1-\frac{\varepsilon}{6}\right)\left(1+\frac{\varepsilon}{8}\right)^{j''}$ and $v_{min}\left(1+\frac{\varepsilon}{8}\right)^{j''}$. By Property 2 of $\mathcal{W}$, the total weight of $\mathcal{W}(i,j'')$ is at most O''. Since O'' satisfies the fairness bounds, the conditions $v_{min}\left(1+\frac{\varepsilon}{8}\right)^{j''+1} \ge l_i^v$ and

$v_{min}\left(1+\frac{\varepsilon}{8}\right)^{j''-1} \le u_i^v$ in Step 3a are also satisfied. So, the subset $\mathcal{W}(i, j'')$ is feasible for $\mathcal{X}(m-1+i, j)$ in Step 3a. So, the Property 3 is satisfied by $\mathcal{X}(m-1+i, j)$.

For any $t \in \lfloor \log_2 m - 1 \rfloor \cup \{0\}$, assume the hypothesis that all the nodes having distance t from leaves satisfy Property 3. We will prove by induction that for any node i with distance $t+1$ from leaves, the node i satisfy Property 3. Let $O' \subseteq \cup_{k=\mathcal{T}(i)} V_k$ that satisfies the fairness conditions for all categories in $\mathcal{T}(i)$. Let $j^* \in X_{range}$ that satisfies the following inequality

$$\left(1+\varepsilon'\right)^{j^*} v_{min} \le \sum_{k \in \mathcal{T}(2i)} \sum_{r \in O' \cap V_k} v_r^{(k)} \le \left(1+\varepsilon'\right)^{j^*+1} v_{min}. \quad (2)$$

If $j' = j^* - t$, the induction hypothesis implies the following :

$$\sum_{k \in \mathcal{T}(2i)} \sum_{r \in \mathcal{X}(2i, j') \cap V_k} w_r^{(k)} \le \sum_{k \in \mathcal{T}(2i)} \sum_{r \in O' \cap V_k} w_r^{(k)}. \quad (3)$$

Similarly, let $j^{**} \in X_{range}$ that satisfies the following inequality

$$\left(1+\varepsilon'\right)^{j^{**}} v_{min} \le \sum_{k \in \mathcal{T}(2i+1)} \sum_{r \in O' \cap V_k} v_r^{(k)} \le \left(1+\varepsilon'\right)^{j^{**}+1} v_{min}. \quad (4)$$

If $j'' = j^{**} - t$, the induction hypothesis implies the following

$$\sum_{k \in \mathcal{T}(2i+1)} \sum_{r \in \mathcal{X}(2i+1, j'') \cap V_k} w_r^{(k)} \le \sum_{k \in \mathcal{T}(2i+1)} \sum_{r \in O' \cap V_k} w_r^{(k)}. \quad (5)$$

We claim that the inequality $(1+\varepsilon')^j \le (1+\varepsilon')^{j'} + (1+\varepsilon')^{j''}$ is satisfied in Step 3b for any $j \in X_{range}$, such that the total value of O' is at least $v_{min}(1+\varepsilon')^{j+t+1}$. This is true because the maximum value of O' is at most $v_{min}\left((1+\varepsilon')^{j^*+1} + (1+\varepsilon')^{j^{**}+1}\right)$ (by Inequality 2 and Inequality 4), which is more than $v_{min}\left(1+\varepsilon'\right)^{j+t+1}$. So, the pair of subsets $\mathcal{X}(2i, j')$ and $\mathcal{X}(2i+1, j'')$ is feasible in Step 3b for all such j. By Equation 3 and Equation 5, Property 3 of $\mathcal{X}$ is satisfied for $\mathcal{X}(i, j)$ and O'.

Let OPT be the optimal value. Let j be the number in X_{range} such that

$$\left(1+\varepsilon'\right)^{j+\log_2 m} v_{min} \le \mathsf{OPT} \le \left(1+\varepsilon'\right)^{j+\log_2 m+1} v_{min}. \quad (6)$$

If we apply Property 3 of $\mathcal{X}$ to any optimal solution, we get that the total weight of the subset $\mathcal{X}(1, j)$ is less than or equal to the total weight of an optimal subset. So, the subset $\mathcal{X}(1, j)$ is feasible in Step 4. By Property 1, the total value of $\mathcal{X}(1, j)$ is at least $\left(1-\frac{\varepsilon}{2}\right)(1+\varepsilon')^j v_{min}$. By Inequality 6, this value is at least

$$\begin{aligned}\left(1-\frac{\varepsilon}{2}\right)\frac{\mathsf{OPT}}{(1+\varepsilon')^{\log_2 m+1}} &= \left(1-\frac{\varepsilon}{2}\right)\frac{\mathsf{OPT}}{\left(1+\frac{3\varepsilon}{8}\right)} \\ &> \left(1-\frac{\varepsilon}{2}\right)\left(1-\frac{3\varepsilon}{8}\right)\mathsf{OPT} \\ &> (1-\varepsilon)\mathsf{OPT}.\end{aligned}$$

If the entry $\mathcal{X}(1, j)$ is finite, Property 2 of $\mathcal{X}$ implies that the total value of items from $V_i \cap \mathcal{X}(1, j)$ is in between $\left(1-\frac{\varepsilon}{2}\right)\left(1+\frac{\varepsilon}{8}\right)^{-1} l_i^v$ and $\left(1+\frac{\varepsilon}{2}\right)\left(1+\frac{\varepsilon}{8}\right) u_i^v$ for all $i \in [m]$. The total value of items from $V_i \cap \mathcal{X}(1, j)$ is at least $\left(1-\frac{\varepsilon}{2}\right)\left(1+\frac{\varepsilon}{8}\right)^{-1} l_i^v > \left(1-\frac{\varepsilon}{2}\right)\left(1-\frac{\varepsilon}{8}\right) l_i^v > (1-\varepsilon) l_i^v$. The total value of items from $V_i \cap \mathcal{X}(1, j)$ is at most $\left(1+\frac{\varepsilon}{2}\right)\left(1+\frac{\varepsilon}{8}\right) u_i^v < \left(1+\frac{6}{8}\varepsilon\right) u_i^v < (1+\varepsilon)\, u_i^v$.

Running time analysis: The size of the $\mathcal{W}$ is $O\left(m \log_{1+\varepsilon}\left(\frac{N v_{max}}{v_{min}}\right)\right)$. We require $O\left(\frac{n^2}{\varepsilon}\right)$ time to fill each entry of the $\mathcal{W}$. So, the total time required to build the table $\mathcal{W}$ is $O\left(\frac{n^2 m \log_{1+\varepsilon}\left(\frac{N v_{max}}{v_{min}}\right)}{\varepsilon}\right)$. The total time required to build the table $\mathcal{X}$ is $O\left(m \log^3_{1+\varepsilon'}\left(\frac{N v_{max}}{v_{min}}\right)\right) = O\left(m \log^3 m \log^3_{1+\varepsilon}\left(\frac{N v_{max}}{v_{min}}\right)\right)$. So, the total running time of the algorithm is $O\left(\frac{n^2 m \log^3 m \log^3_{1+\varepsilon}\left(\frac{N v_{max}}{v_{min}}\right)}{\varepsilon}\right)$. □

4 ALGORITHM FOR BOUND ON NUMBER OF ITEMS IN MAX-KNAPSACK

The notations used in this section are same as in Section 3. We formally define the fair max-knapsack problem with bound on the number of items from each category below. We refer this problem by the acronym BN^{max}.

Problem 3 (BN^{max}). *Given a set of items, each belonging to one of m categories, and the numbers l_i^n and u_i^n for each category i, $\forall i \in [m]$, the problem is to find a subset that maximizes the total value, such that the number of items from category i is between l_i^n and u_i^n, $\forall i \in [m]$, and the total weight of the subset is at most the capacity of the knapsack B.*

Theorem 4.1. *For any $\varepsilon > 0$, there exists a $(1-\varepsilon)$-approximation algorithm for BN^{max} (Problem 3) with running time $O\left(\frac{nN^3}{\varepsilon^2}\right)$.*

Proof. The algorithm is described in Algorithm 3. The algorithm starts by rounding the values of all items so that the rounded values lie in a small range (Step 2). Then it creates bundles of different cardinality and total rounded value from items in $V_i, \forall i \in [m]$. It uses the dynamic programming table $\mathcal{A}$ for this (Step 3). After that the algorithm combines bundles from all categories to obtain the final solution using the dynamic programming table $\mathcal{B}$ (Step 4). The following describes formal proof of the algorithm.

Property of $\mathcal{A}$. We claim that $\mathcal{A}(i, j, v, t), \forall i \in [m], \forall t \in V_i \cup \{0\}$, $j \in V_i$, $\forall v \in \left[\left\lceil \frac{N^2}{\varepsilon} \right\rceil\right] \cup \{0\}$, denotes the weight of a minimum weight subset of cardinality t from first j items of V_i having the total rounded value v. Let S' be the subset satisfying above property for the entry $\mathcal{A}(i, j, v, t)$.

- If $j \in S'$, then $\mathcal{A}(i, j, v, t)$ is the sum of weight of j and the weight of minimum weight subset from first $j-1$ items of V_i having cardinality $t-1$ and the rounded value $v - v_j^{(i)}{}'$, which is equal to $\mathcal{A}(i, j-1, v - v_j^{(i)}{}', t-1) + w_j^{(i)}$.
- If $j \notin S'$, then $\mathcal{A}(i, j, v, t)$ is equal to the weight of minimum weight subset from first $j-1$ items of V_i having cardinality t and the rounded value v, which is equal to $\mathcal{A}(i, j-1, v, t)$.

The recursion in Step 3b captures both of above possibilities. The Step 3a initializes the base cases for the recursion in Step 3b. The entry of $\mathcal{A}$ also corresponds to the respective subset. The entry

Algorithm 3: Algorithm for BN^{max} (Problem 3)

Input: The sets V_i of items, $0 \le l_i^n \le u_i^n$ for $i \in [m]$, B the capacity of knapsack and $\varepsilon > 0$.

Output: S having the total value at least $1-\varepsilon$ times the optimal value of BN^{max} (Problem 3), such that $l_i^n \le |S \cap V_i| \le u_i^n, \forall i \in [m]$ and the total weight of S is at most the knapsack weight B.

(1) Remove all items $j \in V_i, \forall i \in [m]$ that do not come under any feasible solution. We can check whether an item comes under a feasible solution by checking the weight of least weight feasible subset containing the item is less than or equal to B. Let v_{max} be the maximum value of the remaining items.

(2) $\forall i \in [m], \forall j \in V_i$, let

$$v_j^{(i)\prime} := \left\lfloor \frac{v_j^{(i)}}{\varepsilon v_{max}} N \right\rfloor.$$

(3) Let $\mathcal{A}(i,j,v,t), \forall i \in [m], \forall j \in V_i, \forall t \in V_i \cup \{0\}$, $\forall v \in \left[\left\lceil \frac{N^2}{\varepsilon} \right\rceil\right] \cup \{0\}$, be the dynamic programming table constructed in the following way,

(a) $\forall i \in [m]$,
$\mathcal{A}(i,1,v_1^{(i)\prime},1) := w_1^{(i)}$. $\mathcal{A}(i,1,0,0) = 0$.
$\mathcal{A}(i,1,v',.) := \infty$ for $v' \in \left[\left\lceil \frac{N^2}{\varepsilon} \right\rceil\right] \setminus \{v_1^{(i)\prime}\}$.

(b) $\forall i \in [m], \forall j \in V_i \setminus \{1\}, \forall t \in V_i \cup \{0\}$,
$\forall v \in \left[\left\lceil \frac{N^2}{\varepsilon} \right\rceil\right] \cup \{0\}$,
If $v < v_j^{(i)}$ or $t = 0$, then $\mathcal{A}(i,j,v,t) = \mathcal{A}(i,j-1,v,t)$.
Else ,

$$\mathcal{A}(i,j,v,t) := \min \Big\{ \mathcal{A}(i,j-1,v-v_j^{(i)\prime},t-1) + w_j^{(i)}, \; \mathcal{A}(i,j-1,v,t) \mid 0 \le v_j^{(i)\prime} \le v \Big\}.$$

(4) Let $\mathcal{B}(i,v), \forall i \in [m], \forall v \in \left[\left\lceil \frac{N^2}{\varepsilon} \right\rceil\right] \cup \{0\}$ be another the dynamic programming table.

(a) $\forall v \in \left[\left\lceil \frac{N^2}{\varepsilon} \right\rceil\right] \cup \{0\}$,

$$\mathcal{B}(1,v) := \min\{\mathcal{A}(1,|V_1|,v,t) \mid l_1^n \le t \le u_1^n\}.$$

(b) For $i \in [m] \setminus \{1\}$ and $v \in \left[\left\lceil \frac{N^2}{\varepsilon} \right\rceil\right] \cup \{0\}$,

$$\mathcal{B}(i,v) := \min \Big\{ \mathcal{B}(i-1,v_r) + \mathcal{A}(i,|V_i|,v-v_r,t) \mid l_i^n \le t \le u_i^n \;\&\; v_r \le v \;\&\; v_r \in \left[\left\lceil \frac{N^2}{\varepsilon} \right\rceil\right] \cup \{0\} \Big\}.$$

(5) Output S in the following way,

$$\text{argmax}_v\{\mathcal{B}(m,v) \mid \mathcal{B}(m,v) \le B\}.$$

of $\mathcal{A}$ could be ∞, which indicates no subset. We use the notation $\mathcal{A}(i,j,v,t)$ to indicate both the entry and the subset.

Property of $\mathcal{B}$. We claim that $\mathcal{B}(i,v), \forall i \in [m], \forall v \in \left[\left\lceil \frac{N^2}{\varepsilon} \right\rceil\right] \cup \{0\}$, denotes the weight of a minimum weight subset of $\cup_{j=1}^{i} V_j$ having the total rounded value v, such that fairness constraints are satisfies for the subset for all the categories up to i. Let S' be the subset of $\cup_{j=1}^{i} V_j$ satisfying above property for $\mathcal{B}(i,v)$. Let $\sum_{j \in S' \cap V_i} v_j^{(i)\prime} = v_r$. $\mathcal{B}(i,v)$ is the sum of weights of minimum weight subset of $\cup_{j=1}^{i-1} V_j$ having the total rounded value $v - v_r$ that satisfies fairness condition for all categories up to $i-1$ ($\mathcal{B}(i-1, v - v_r)$), and the minimum weight subset of V_i having total rounded value v_r that satisfies fairness condition for category i ($\mathcal{A}(i,|V_i|,v_r,t)$ such that $l_i^n \le t \le u_i^n$). The recursion in Step 4b captures this for all possible v_r. The entry of $\mathcal{B}$ also corresponds to the respective subset. The entry of $\mathcal{B}$ could be ∞, which indicates no subset. We use the notation $\mathcal{B}(i,v)$ to indicate both the entry and the subset.

By the property of $\mathcal{B}$, the total rounded value of S (Step 5) is at least the total rounded value of any optimal solution. By Lemma 4.2, we can show that the total value of S is at least $(1-\varepsilon)$ times the optimal solution.

Running time analysis: The size of the table $\mathcal{A}$ is $O\left(\frac{nN^3}{\varepsilon}\right)$, and to fill each entry in the table $\mathcal{A}$, we require $O(1)$ time. The size of the table $\mathcal{B}$ is $O\left(\frac{N^2 m}{\varepsilon}\right)$, and to fill each entry in the table $\mathcal{B}$, we require $O\left(\frac{nN^2}{\varepsilon}\right)$ time. So, the total time required is $O\left(\frac{nN^3}{\varepsilon^2}\right)$. □

Lemma 4.2. *If* OPT *is the optimal objective value of* BN^{max} *(Problem 3), then the total value of a set returned by Algorithm 3 is at least* $(1-\varepsilon)$OPT.

Proof. Let $O \subseteq \cup_{i=1}^{m} V_i$ be the set of items in an optimal solution and $S \subseteq \cup_{i=1}^{m} V_i$ be the set of items selected by Algorithm 3. Since S is an optimal solution of rounded value in Algorithm 3, the total rounded value of S is more than the total rounded value of O.

$$\sum_{i=1}^{m} \sum_{j \in V_i \cap O} v_j^{(i)\prime} \le \sum_{i=1}^{m} \sum_{j \in V_i \cap S} v_j^{(i)\prime}. \tag{7}$$

Because of the rounding in Step 2, we have following inequalities $\forall i \in [m]$ and $\forall j \in V_i$,

$$\frac{\varepsilon v_{max} v_j^{(i)\prime}}{N} \le v_j^{(i)} \le \frac{\varepsilon v_{max}(v_j^{(i)\prime}+1)}{N} = \frac{\varepsilon v_{max} v_j^{(i)\prime}}{N} + \frac{\varepsilon v_{max}}{N}. \tag{8}$$

So we get,

$$\begin{aligned}
\sum_{i=1}^{m} \sum_{j \in V_i \cap O} v_j^{(i)} &\le \frac{\varepsilon v_{max}}{N} \left(\sum_{i=1}^{m} \sum_{j \in V_i \cap O} (v_j^{(i)\prime} + 1) \right) && \text{(From Ineq. 8)} \\
&\le \frac{\varepsilon v_{max}}{N} \left(\sum_{i=1}^{m} \sum_{j \in V_i \cap O} v_j^{(i)\prime} \right) + \varepsilon v_{max} \\
&\le \frac{\varepsilon v_{max}}{N} \left(\sum_{i=1}^{m} \sum_{j \in V_i \cap S} v_j^{(i)\prime} \right) + \varepsilon v_{max} && \text{(Ineq. 7)} \\
&\le \left(\sum_{i=1}^{m} \sum_{j \in V_i \cap S} v_j^{(i)} \right) + \varepsilon v_{max}. && \text{(Ineq. 8)}
\end{aligned}$$

Since Step 1 of the Algorithm 3 discards all the items which doesn't come in any feasible solution, we know that $v_{max} \le$ OPT.

Therefore, we get

$$\left(\sum_{i=1}^{m}\sum_{j\in V_i\cap S} v_j^{(i)}\right) \geq (1-\varepsilon)\mathrm{OPT}.$$

□

5 OTHER PROBLEMS

In this section, we formally define other fairness models that we consider. We give detailed algorithms and hardness results for these problems in the supplementary material.

- Bound on weight in max-knapsack: We refer this problem by the acronym BW^{max}.

Problem 4 (BW^{max}). *Given a set of items, each belonging to one of m categories, a lower bound l_i^w and an upper bound u_i^w for each category i, $\forall i \in [m]$, the goal is to find a subset that maximizes the total value, such that the total weight of items from category i is in between l_i^w and u_i^w, $\forall i \in [m]$ and the total weight of the subset is at most the capacity of the knapsack B.*

- Bound on number of items in min-knapsack: We refer this problem by the acronym BN^{min}.

Problem 5 (BN^{min}). *Given a set of items, each belonging to one of m categories, and numbers l_i^n and u_i^n for category i, $\forall i \in [m]$, the problem is to find a subset that minimizes the total weight, such that the number of items from category i is between l_i^n and u_i^n, $\forall i \in [m]$, and the total value of the subset is at least the given value lower bound L.*

- Bound on value in min-knapsack: We refer this problem by the acronym BV^{min}.

Problem 6 (BV^{min}). *Given a set of items, each belonging to one of m categories, and numbers l_i^v and u_i^v for category i, $\forall i \in [m]$, the problem is to find a subset that minimizes the total weight, such that the total value of items from category i is between l_i^v and u_i^v, $\forall i \in [m]$, and the total value of the subset is at least the given value lower bound L.*

- Bound on weight in min-knapsack: We refer this problem by the acronym BW^{min}.

Problem 7 (BW^{min}). *Given a set of items, each belonging to one of m categories, and a lower bound l_i^w and an upper bound u_i^w such that $0 \leq l_i^w \leq u_i^w$ for each category $i \in [m]$, the goal is to find a subset that minimizes the total weight, such that the total weight of items from each category i is in between the given bounds l_i^w and u_i^w, $\forall i \in [m]$, and the total value of the subset is at least the given bound L.*

6 EXPERIMENTAL RESULTS

Our theorems give theoretical guarantees for the quality of the solution output by our algorithms. In this section, we present an experimental evaluation of the running time of Algorithm 2. We experiment[2] with Algorithm 2 on randomly generated instances for Problem 1. We note that [26] also uses similar model of random instances for experimental results for the standard knapsack problems.

[2]The code is available on https://github.com/creatovolve/GroupFairness_Knapsack.

Table 2: Average running time in minutes

	epsilon		
	$\varepsilon = 0.1$	$\varepsilon = 0.2$	$\varepsilon = 0.5$
$m = 64$	14.5	3.2	0.6
$m = 32$	6.1	1.4	0.3
$m = 16$	2.4	0.7	0.2

We give experimental results for the implementation of Algorithm 2 in Table 2. The code was parallelized wherever possible to improve the running time. The model of GPU used by the code was GeForce GTX 1080 Ti. It has 3584 cores. The processor of host system was Intel(R) Xeon(R) Silver 4110 CPU, 2.10GHz.

The input instance is a randomly generated instance. The weight and value of each item is a randomly generated integer between 1 to 100. In all the instances, the number of items in each category is 157. So the total number of items N is 10048, 5024, and 2512, for $m = 64, 32, 16$, respectively. The knapsack weight is set to $\frac{1}{4}$ of expected total weight of items in each case. The lower bounds of categories are set to $\frac{1}{3}$ of expected total value of items in a category in each case. The upper bounds of categories are set to be the expected total value of items in a category in each case. We report our experimental results in Table 2. For each combination of m and epsilon, we report the average of running time (in minutes) over 10 different randomly generated input instances.

7 CONCLUSION

In this paper, we studied various fairness notions for knapsack problems. We studied six variants, three for each of max-knapsack and min-knapsack. These different variants encompass several interesting problems. Studying fairness notions in related problems such as multiple knapsack problem [24], multidimensional knapsack problem [18], submodular knapsack problem [32], is an interesting open problem. Study of different fairness notions for resource allocation and scheduling problems remains an interesting area.

ACKNOWLEDGMENTS

AL was supported in part by SERB Award ECR/2017/003296 and a Pratiksha Trust Young Investigator Award.

REFERENCES

[1] Susanne Albers, Arindam Khan, and Leon Ladewig. 2019. Improved Online Algorithms for Knapsack and GAP in the Random Order Model. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA (LIPIcs, Vol. 145)*, Dimitris Achlioptas and László A. Végh (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 22:1–22:23. https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2019.22

[2] Junaid Ali, Mahmoudreza Babaei, Abhijnan Chakraborty, Baharan Mirzasoleiman, Krishna P. Gummadi, and Adish Singla. 2019. On the Fairness of Time-Critical Influence Maximization in Social Networks. *ArXiv* abs/1905.06618 (2019).

[3] Haris Aziz, Barton E. Lee, and Nimrod Talmon. 2018. Proportionally Representative Participatory Budgeting: Axioms and Algorithms. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (Stockholm, Sweden) *(AAMAS '18)*. International Foundation for Autonomous

Agents and Multiagent Systems, Richland, SC, 23–31. http://dl.acm.org/citation.cfm?id=3237383.3237394

[4] Abbas Bazzi, Samuel Fiorini, Sangxia Huang, and Ola Svensson. 2017. Small Extended Formulation for Knapsack Cover Inequalities from Monotone Circuits. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*. SIAM, 2326–2341.

[5] Xiaohui Bei, Xinhang Lu, Pasin Manurangsi, and Warut Suksompong. 2019. The Price of Fairness for Indivisible Goods. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, (IJCAI) 2019, Macao, China, August 10-16, 2019*. ijcai.org, 81–87.

[6] Nawal Benabbou, Mithun Chakraborty, Edith Elkind, and Yair Zick. 2019. Fairness Towards Groups of Agents in the Allocation of Indivisible Items. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, (IJCAI) 2019, Macao, China, August 10-16, 2019*. IJCAI, 95–101.

[7] Nawal Benabbou, Mithun Chakraborty, Xuan-Vinh Ho, Jakub Sliwinski, and Yair Zick. 2018. Diversity Constraints in Public Housing Allocation. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (Stockholm, Sweden) *((AAMAS) '18)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 973–981.

[8] Suman Kalyan Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. 2019. Fair Algorithms for Clustering. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, (NIPS) 2019, 8-14 December 2019, Vancouver, BC, Canada*. NIPS, 4955–4966.

[9] Robert Bredereck, Piotr Faliszewski, Ayumi Igarashi, Martin Lackner, and Piotr Skowron. 2017. Multiwinner Elections with Diversity Constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI.

[10] Chandra Chekuri and Sanjeev Khanna. 2000. A PTAS for the Multiple Knapsack Problem. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms* (San Francisco, California, USA) *(SODA '00)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 213–222. http://dl.acm.org/citation.cfm?id=338219.338254

[11] Chandra Chekuri and Amit Kumar. 2004. Maximum Coverage Problem with Group Budget Constraints and Applications. In *Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques, APPROX/RANDOM 2004, Cambridge, MA, USA, August 22-24, 2004, Proceedings*. APPROX/RANDOM, 72–83.

[12] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. 2017. Fair Clustering Through Fairlets. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS) 2017, 4-9 December 2017, Long Beach, CA, USA*. NIPS, 5029–5037.

[13] Alexandra Chouldechova. 2017. Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments. *Big Data* 5, 2 (2017), 153–163.

[14] Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. 2017. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review* 24 (2017), 63–79.

[15] János Csirik. 1991. Heuristics for the 0-1 min-knapsack problem. *Acta Cybernetica* 10, 1-2 (1991), 15–20.

[16] Yuri Faenza, Igor Malinovic, Monaldo Mastrolilli, and Ola Svensson. 2018. On Bounded Pitch Inequalities for the Min-Knapsack Polytope. In *Combinatorial Optimization - 5th International Symposium, ISCO 2018, Marrakesh, Morocco, April 11-13, 2018, Revised Selected Papers*. ISCO, 170–182.

[17] Till Fluschnik, Piotr Skowron, Mervin Triphaus, and Kai Wilker. 2017. Fair Knapsack. *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (11 2017). https://doi.org/10.1609/aaai.v33i01.33011941

[18] A.M. Frieze and M.R.B. Clarke. 1984. Approximation algorithms for the m-dimensional 0−1 knapsack problem: Worst-case and probabilistic analyses. *European Journal of Operational Research* 15, 1 (1984), 100 – 109. https://doi.org/10.1016/0377-2217(84)90053-5

[19] Waldo Gálvez, Fabrizio Grandoni, Sandy Heydrich, Salvatore Ingala, Arindam Khan, and Andreas Wiese. 2017. Approximating Geometric Knapsack via L-Packings. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*. FOCS, 260–271.

[20] Ashish Goel, Anilesh K. Krishnaswamy, Sukolsak Sakshuwong, and Tanja Aitamurto. 2019. Knapsack Voting for Participatory Budgeting. *ACM Trans. Econ. Comput.* 7, 2, Article 8 (July 2019), 27 pages. https://doi.org/10.1145/3340230

[21] Sreenivas Gollapudi and Debmalya Panigrahi. 2014. Fair Allocation in Online Markets. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM)*. CIKM, 1179–1188.

[22] Oscar H. Ibarra and Chul E. Kim. 1975. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM (JACM)* 22, 4 (1975), 463–468.

[23] Yusuke Inoue, Takakazu Imada, Syunya Doi, Lei Chen, Takehito Utsuro, and Yasuhide Kawada. 2016. Selecting Web search results of diverse contents with search engine suggests and a topic model. In *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. WAINA, 455–460.

[24] Klaus Jansen. 2012. A Fast Approximation Scheme for the Multiple Knapsack Problem. In *SOFSEM 2012: Theory and Practice of Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 313–324.

[25] Matthew Joseph, Michael Kearns, Jamie H Morgenstern, and Aaron Roth. 2016. Fairness in Learning: Classic and Contextual Bandits. In *Advances in Neural Information Processing Systems 29 (NIPS)*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 325–333. http://papers.nips.cc/paper/6355-fairness-in-learning-classic-and-contextual-bandits.pdf

[26] Hans Kellerer, Ulrich Pferschy, and David Pisinger. 2004. *Knapsack problems*. Springer. https://doi.org/10.1007/978-3-540-24777-7

[27] Jalal Khamse-Ashari, Ioannis Lambadaris, George Kesidis, Bhuvan Urgaonkar, and Yiqiang Q. Zhao. 2018. An Efficient and Fair Multi-Resource Allocation Mechanism for Heterogeneous Servers. *(IEEE) Trans. Parallel Distrib. Syst.* 29, 12 (2018), 2686–2699.

[28] Arindam Khan. 2015. *Approximation algorithms for multidimensional bin packing*. Ph.D. Dissertation. Georgia Institute of Technology.

[29] Ariel Kulik, Hadas Shachnai, and Tami Tamir. 2013. Approximations for Monotone and Nonmonotone Submodular Maximization with Knapsack Constraints. *Math. Oper. Res.* 38, 4 (2013), 729–739. https://doi.org/10.1287/moor.2013.0592

[30] Adam Kurpisz, Samuli Leppänen, and Monaldo Mastrolilli. 2014. On the Lasserre/Sum-of-Squares Hierarchy with Knapsack Covering Inequalities. *CoRR* abs/1407.1746 (2014).

[31] Eugene L. Lawler. 1979. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research* 4, 4 (1979), 339–356.

[32] Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. 2009. Non-monotone submodular maximization under matroid and knapsack constraints. *CoRR* abs/0902.0353 (2009). arXiv:0902.0353 http://arxiv.org/abs/0902.0353

[33] Hanan Luss. 1999. On Equitable Resource Allocation Problems: A Lexicographic Minimax Approach. *Operations Research* 47, 3 (1999), 361–378.

[34] John Nash. 1950. The Bargaining Problem. *Econometrica* 18, 2 (1950), 155–162. https://EconPapers.repec.org/RePEc:ecm:emetrp:v:18:y:1950:i:2:p:155-162

[35] Manfred W Padberg, Tony J Van Roy, and Laurence A Wolsey. 1985. Valid linear inequalities for fixed charge problems. *Operations Research* 33, 4 (1985), 842–861.

[36] Deval Patel, Arindam Khan, and Anand Louis. 2021. Group Fairness for Knapsack Problems. arXiv:2006.07832 [cs.DS]

[37] Govind S. Sankar, Anand Louis, Meghana Nasre, and Prajakta Nimbhorkar. 2020. Matchings with Group Fairness Constraints: Online and Offline Algorithms. *Manuscript 2020* (2020).

[38] Erel Segal-Halevi and Warut Suksompong. 2018. Democratic Fair Allocation of Indivisible Goods. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, (IJCAI) 2018, July 13-19, 2018, Stockholm, Sweden*. IJCAI, 482–488.

[39] Meinolf Sellmann. 2004. The Practice of Approximated Consistency for Knapsack Constraints. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, Deborah L. McGuinness and George Ferguson (Eds.). AAAI Press / The MIT Press, 179–184. http://www.aaai.org/Library/AAAI/2004/aaai04-029.php

[40] Hadas Shachnai and Tami Tamir. 2000. Polynominal time approximation schemes for class-constrained packing problem. In *Approximation Algorithms for Combinatorial Optimization, Third International Workshop, APPROX 2000, Saarbrücken, Germany, September 5-8, 2000, Proceedings*. APPROX, 238–249.

[41] Ehud Shapiro and Nimrod Talmon. 2017. A Democratically-Optimal Budgeting Algorithm. *CoRR* abs/1709.05839 (2017). arXiv:1709.05839 http://arxiv.org/abs/1709.05839

[42] Maxim Sviridenko. 2004. A Note on Maximizing a Submodular Set Function Subject to a Knapsack Constraint. *Operations Research Letters* 32, 1 (2004), 41–43. https://doi.org/10.1016/S0167-6377(03)00062-2

[43] Alan Tsang, Bryan Wilder, Eric Rice, Milind Tambe, and Yair Zick. 2019. Group-Fairness in Influence Maximization. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, (IJCAI) 2019, Macao, China, August 10-16, 2019*. IJCAI, 5997–6005.

[44] François Vanderbeck. 1999. Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming* 86, 3 (01 Dec 1999), 565–594. https://doi.org/10.1007/s101070050105