Appendix

A More related works

Besides proportionality, in another parallel line of research, envy-freeness and its relaxations, namely envy-free up to one item (EF1) and envy-free up to any item (EFX), are also widely studied. It was shown in [35] and [11] for goods and chores, respectively, that an EF1 allocation exists for the monotone combinatorial functions. However, the existence of EFX allocations is still unknown even with additive functions. Therefore, approximation algorithms were proposed in [2, 42] for additive functions and in [39, 16] for subadditive functions. We refer the readers to [3] for a detailed survey on fair allocation of indivisible items.

B Missing materials in preliminaries

B.1 Impossibility result for general cost functions

We provide an example to show that no bounded approximation ratio can be achieved for general cost functions. Note that there exist simpler examples, but we choose the following one because it represents a particular combinatorial structure – minimum spanning tree. Let $G = (V, E)$ be a graph shown in the left sub-figure of Figure 2, where the vertices V are the items that are to be allocated, i.e., $M = V$. There are two agents $N = \{1, 2\}$ who have different weights on the edges as shown in the middle and right sub-figures of Figure 2. The cost functions are measured by the minimum spanning tree in their received subgraphs. Particularly, for any $S \subseteq V$, $v_i(S)$ equals the weight of the minimum spanning tree on $G[S]$ – the induced subgraph of S in G – under agent i's weights. Thus, $\mathsf{MMS}_i = 0$, for both $i = 1, 2$, where an MMS defining partition for agent 1 is $\{v_1, v_2\}$ and $\{v_3, v_4\}$ and that for agent 2 is $\{v_1, v_4\}$ and $\{v_2, v_3\}$. However, it can be verified that no matter how the vertices are allocated to the agents, there is one agent whose cost is at least 1, which implies that no bounded approximation is possible for general costs.

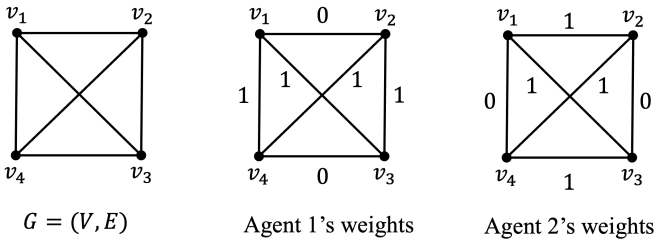

Figure 2: An instance with unbounded approximation ratio

B.2 Proof of Observation 1

To prove the observation, it suffices to show $\mathsf{MMS}_i^d \leq \lceil \frac{n}{d} \rceil \cdot \mathsf{MMS}_i^n$ for any agent $i \in N$. Let $\mathbf{X} = (X_1, \ldots, X_n)$ be an MMS defining partition for agent i, which satisfies $v_i(X_j) \leq \mathsf{MMS}_i^n$ for every $j \in [n]$. Consider a d-partition $\mathbf{X}' = (X'_1, \ldots, X'_d)$ built by evenly distributing the n bundles in $\mathbf{X}$ to the d bundles in $\mathbf{X}'$; that is, the number of bundles distributed to the bundles in $\mathbf{X}'$ differs by at most one. Clearly, $\mathbf{X}'$ satisfies $v_i(X'_j) \leq \lceil \frac{n}{d} \rceil \cdot \mathsf{MMS}_i^n$ for every $j \in [d]$. By the definition of 1-out-of-d MMS, it follows that

$$\mathsf{MMS}_i^d \leq \max_{j \in [d]} v_i(X'_j) \leq \lceil \frac{n}{d} \rceil \cdot \mathsf{MMS}_i^n,$$

thus completing the proof.

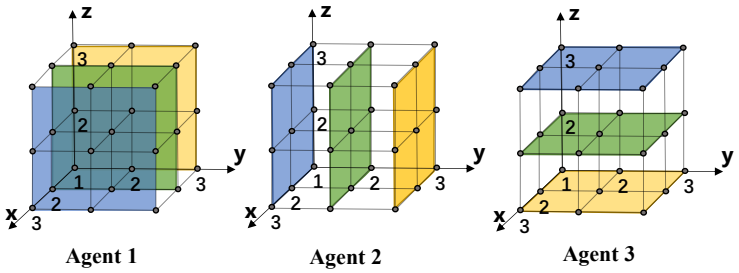

Figure 3: An instance with 3 agents and 27 items

C Missing materials in general subadditive cost setting

C.1 An example that helps understand Theorem 1

The example is illustrated in Figure 3 where each agent has three covering planes. Take agent 1 for example, her three covering planes contain the items whose x coordinates are 1, 2, 3, respectively. If there exists an allocation that is better than 3-MMS, then each agent is allocated items from at most 2 of her covering planes. Without loss of generality, we assume that agent 1 (or agents 2 and 3 respectively) is not allocated any item whose x (or y and z respectively) coordinate is 1. Then, the item $(1, 1, 1)$ is not allocated to any agent, a contradiction.

C.2 Proof of Corollary 1

We consider the same instance that is designed in Theorem 1. In this instance, we have proved that no matter how the items are allocated among the agents, there is at least one agent, say i, whose cost is n. Moreover, by the design of the cost functions, for any integer d, it can be observed that $\mathsf{MMS}_i^d = \lceil \frac{n}{d} \rceil$. Note that $\lceil \frac{n}{d} \rceil$ is always smaller than n for all $d \geq 2$, thus the allocation is not 1-out-of-d MMS to i.

D Missing materials in bin packing setting

D.1 The IDO reduction

For a bin packing or job scheduling instance I, the IDO instance I' is constructed by setting the size of each item $e_j \in M$ to each agent $i \in N$ in I' to the j-th largest size of the items to i in I. Then the IDO reduction is formally presented in the following lemma.

Lemma 1 *For the bin packing or job scheduling setting, if there exists an allocation $\mathbf{A}' = (A'_1, \ldots, A'_n)$ in the IDO instance I' such that $v'_i(A'_i) \leq \alpha \cdot \mathsf{MMS}_i^d(I')$ for all $i \in N$, then there exists an allocation $\mathbf{A} = (A_1, \ldots, A_n)$ in the original instance I such that $v_i(A_i) \leq \alpha \cdot \mathsf{MMS}_i^d(I)$ for all $i \in N$.*

Proof. We design Algorithm 4 that given I, I' and $\mathbf{A}'$, computes the desired allocation $\mathbf{A}$. In the algorithm, we look at the items from e_m to e_1. For each item, we let the agent who receives it in I' pick her smallest unallocated item in I.

To prove the lemma, we first show that $v_i(A_i) \leq v'_i(A'_i)$ for all $i \in N$. Consider the iteration where we look at the item e_g. We suppose that in this iteration agent i picks item $e_{g'}$; that is, $e_g \in A'_i$, $e_{g'} \in A_i$ and $e_{g'}$ is the smallest unallocated item for i. Since an item is removed from the set R after it is allocated, exactly $m - g$ items have been allocated before $e_{g'}$ is allocated. Therefore, $e_{g'}$ is among the top $m - g + 1$ smallest items for agent i. Recall that e_g is the item with the exactly $(m-g+1)$-th smallest size to i, hence $s_{i,g'} \leq s'_{i,g}$. The same reasoning can be applied to other items in A'_i and A_i, and to other agents. It follows that for any $i \in N$, any $e_g \in A'_i$ and the corresponding $e_{g'} \in A_i$, $s_{i,g'} \leq s'_{i,g}$. For the bin packing or job scheduling setting, this implies $v_i(A_i) \leq v'_i(A'_i)$. Since the maximin share depends on the sizes of the items but not on the order, the maximin share

of agent i in I' is the same as that in I, i.e., $\mathsf{MMS}_i^d(I') = \mathsf{MMS}_i^d(I)$. Hence, the condition that $v_i'(A_i') \leq \alpha \cdot \mathsf{MMS}_i^d(I')$ gives $v_i(A_i) \leq \alpha \cdot \mathsf{MMS}_i^d(I)$, which completes the proof. ■

Algorithm 4 IDO reduction for the bin packing and job scheduling settings

Input: A general instance I, the IDO instance I' and an allocation $\mathbf{A}' = (A_1', ..., A_n')$ for the IDO instance such that $v_i'(A_i') \leq \alpha \cdot \mathsf{MMS}_i^d(I')$ for all $i \in N$.

Output: An allocation $\mathbf{A} = (A_1, ..., A_n)$ such that $v_i(A_i) \leq \alpha \cdot \mathsf{MMS}_i^d(I)$ for all $i \in N$.

1: For all $i \in N$ and $e_g \in A_i'$, set $p_g \leftarrow i$.
2: Initialize $A_i \leftarrow \emptyset$ for all $i \in N$, and $R \leftarrow M$.
3: **for** $g = m$ to 1 **do**
4: Pick $e_{g'} \in \arg\min_{e_k \in R}\{s_{p_g,k}\}$.
5: $A_{p_g} \leftarrow A_{p_g} \cup \{e_{g'}\}$, $R \leftarrow R \setminus \{e_{g'}\}$.
6: **end for**

D.2 Lower bound instance

We present an instance for the bin packing setting where no allocation can be better than 2-MMS. We first recall the impossibility instance given by Feige et al. [20]. In this instance there are three agents and nine items as arranged in a three by three matrix. The three agents' costs are shown in the matrices V_1, V_2 and V_3.

$$V_1 = \begin{pmatrix} 6 & 15 & 22 \\ 26 & 10 & 7 \\ 12 & 19 & 12 \end{pmatrix} \qquad V_2 = \begin{pmatrix} 6 & 15 & 23 \\ 26 & 10 & 8 \\ 11 & 18 & 12 \end{pmatrix}$$

$$V_3 = \begin{pmatrix} 6 & 16 & 22 \\ 27 & 10 & 7 \\ 11 & 18 & 12 \end{pmatrix}$$

Feige et al. [20] proved that for this instance the MMS value of every agent is 43, however, in any allocation, at least one of the three agents gets cost no smaller than 44.

We can adapt this instance to the bin packing setting and obtain a lower bound of 2. In particular, we also have three agents and nine items. The numbers in matrices V_1, V_2 and V_3 are the sizes of the items to agents $1, 2$ and 3, respectively. Let the capacities of the bins be $c_i = 43$ for all $i \in \{1, 2, 3\}$. Accordingly, we have $\mathsf{MMS}_i = 1$ for all $i \in \{1, 2, 3\}$. Since in any allocation, there is at least one agent who gets items with total size no smaller than 44, for this agent, she has to use two bins to pack the assigned items, which means that no allocation can be better than 2-MMS.

D.3 Computing $\frac{3}{2}\mathsf{MMS} + 1$ allocations

Recall that in the proof of Corollary 2, it has been shown that each agent $i \in N$ can use MMS_i bins to pack all items in $W_i(A_i \setminus \{e_i^*\})$ and another MMS_i bins to pack all items in $J_i(A_i \setminus \{e_i^*\}) \cup \{e_i^*\}$. Actually, since all items in $J_i(A_i \setminus \{e_i^*\}) \cup \{e_i^*\}$ are small for i and at least two small items can be put into one bin, i only needs $\lceil \frac{\mathsf{MMS}_i}{2} \rceil$ bins to pack all items in $J_i(A_i \setminus \{e_i^*\}) \cup \{e_i^*\}$. Therefore, each agent i can use no more than $\frac{3}{2}\mathsf{MMS}_i + 1$ bins to pack all the items allocated to her.

E Missing materials in job scheduling setting

E.1 Another interpretation to the job scheduling setting

An alternative way to explain the job scheduling setting is to view each agent i as a group of k_i small agents and MMS_i^d as the *collective maximin share* for these k_i small agents. We believe this notion of collective maximin share is of independent interest as a group-wise fairness notion. We remark that this notion is different from the group-wise (and pair-wise) maximin share defined in [7] and [15], where the max-min value is defined for each single agent. In our definition, however, a set of agents share the same value for the items allocated to them.

E.2 Algorithm

E.2.1 Part 1: partitioning the items into d bundles

We first partition the items into d bundles $\mathbf{B} = (B_1, \ldots, B_d)$ in a round-robin fashion. Specifically, we allocate the items in descending order of their sizes to the bundles by turns, from the first bundle to the last one. Each time, we allocate one item to one bundle, and when every bundle receives an item, we start over from the first bundle and so on. For any set of items S, let $S[l]$ be the l-th largest item in S, then the algorithm is formally presented in Algorithm 5.

Algorithm 5 Partitioning the items into d bundles

Input: An IDO job scheduling instance $(N, M, \{v_i\}_{i\in N}, \{s_i\}_{i\in N})$.
Output: A d-partition of M: $\mathbf{B} = (B_1, \ldots, B_d)$.

```
Initialize B_j ← ∅ for every j ∈ [d], and r ← 1.
while r ≤ m do
    for j = 1 to d do
        if r ≤ m then
            B_j ← B_j ∪ {M[r]}.
            r ← r + 1.
        end if
    end for
end while
```

By the characteristic of the round-robin fashion, we have the following important observation.

Observation 2 *For each bundle $B_j \in \mathbf{B}$ and each item $e_k \in B_j \setminus \{B_j[1]\}$ (if exists), the $d-1$ items before e_k (i.e., items $e_{k-1}, e_{k-2}, \ldots, e_{k-d+1}$) have at least the same sizes as e_k.*

E.2.2 Part 2: imaginary assignment

Next, for each bundle $B_j \in \mathbf{B}$ computed in the first part and each agent $i \in N$, we imaginatively assign the items in $B_j \setminus B_j[1]$ to i's machines as follows. We greedily assign the items with larger sizes to i's machines with faster speeds (in other words, with larger capacities), as long as the total workload on one machine does not exceed the its capacity. The first time when the workload exceeds the capacity, we move to the next machine and so on. The algorithm is formally presented in Algorithm 6 and illustrated in Figure 4. For each $l \in P_i$, $C^I_{i,l}$ contains the items imaginatively assigned to machine l that do not make the total workload exceed l's capacity, and $t_{i,l}$ is the last item assigned to l that makes the total workload exceed the capacity. Note that $C^I_{i,l}$ may be empty and $t_{i,l}$ may be null. For simplicity, let $t_{i,0} = B_j[1]$; that is, $B_j[1]$ is assigned to an imaginary machine 0. The items in $\bigcup_{l\in[k_i]} C^I_{i,l}$ are called *internal items* (as shown by the *dark* boxes in Figure 4), and $\{t_{i,0}, \ldots, t_{i,k_i}\}$ are called *external items* (as shown by the *light* boxes).

Algorithm 6 Imaginary assignment

Input: A bundle $B_j \in \mathbf{B}$ computed in the first part and an agent $i \in N$.
Output: Sets of internal items $\{C^I_{i,1}, \ldots, C^I_{i,k_i}\}$ and external items $\{t_{i,0}, \ldots, t_{i,k_i}\}$.

```
Initialize C^I_{i,l} ← ∅, t_{i,l} ← null for every l ∈ [k_i], and r ← 1.
while r ≤ |B_j| do
    for l = 1 to k_i do
        t_{i,l-1} ← B_j[r], r ← r + 1.
        while r ≤ |B_j| and s_i(C^I_{i,l} ∪ {B_j[r]}) ≤ c_{i,l} do
            C^I_{i,l} ← C^I_{i,l} ∪ {B_j[r]}, r ← r + 1.
        end while
    end for
end while
```

For each bundle $B_j \in \mathbf{B}$ and each agent $i \in N$, the imaginary assignment has the following important properties.

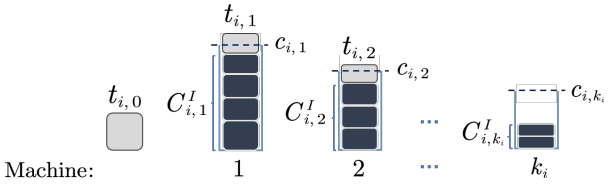

Figure 4: The imaginary assignment of B_j to agent i

- **Property 1**: all items in $B_j \setminus \{B_j[1]\}$ can be assigned to agent i's machines. Besides, the last machine k_i does not have an external item; that is, t_{i,k_i} is null.
- **Property 2**: for any $1 \le l \le k_i$, the total size of the internal items $C^I_{i,l}$ does not exceed the capacity of machine l, i.e., $s_i(C^I_{i,l}) \le c_{i,l}$
- **Property 3**: for any $1 \le l \le k_i$, the external item $t_{i,l-1}$ (if not null) has size no larger than the capacity of machine l, i.e., $s_i(t_{i,l-1}) \le c_{i,l}$.

Proof. The first property holds since otherwise, $s_i(B_j \setminus \{B_j[1]\}) > \sum_{l\in[k_i]} c_{i,l}$. By Observation 2, it follows that

$$s_i(M) > d \cdot s_i(B_j \setminus \{B_j[1]\}) > d \cdot \sum_{l\in[k_i]} c_{i,l}.$$

However, since all items can be assigned to i's machines in i's 1-out-of-d MMS defining partition, we have $s_i(M) \le d \cdot \sum_{l\in[k_i]} c_{i,l}$, a contradiction.

The second property directly follows the algorithm. For the third property, $s_i(t_{i,0}) \le c_{i,1}$ follows two facts that $t_{i,0}$ is assigned to some machine in i's 1-out-of-d MMS defining partition and $c_{i,1}$ is the largest capacity of the machines. We then consider $l \in [k_i - 1]$ and show $s_i(t_{i,l}) \le c_{i,l+1}$ (if $t_{i,l}$ is not null). The same reasoning can be applied to any other $l' \in [k_i - 1]$. Let $S_1 = \bigcup_{p\in[l]}(C^I_{i,p} \cup \{t_{i,p}\})$. From the algorithm, we know that $s_i(S_1) > \sum_{p\in[l]} c_{i,p}$ and $t_{i,l}$ is the smallest item in S_1. By Observation 2, there exist another $d-1$ disjoint sets of items $\{S_2, \ldots, S_d\}$ such that $s_i(S_k) \ge s_i(S_1)$ for every $k \in [2, d]$ and $t_{i,l}$ is also the smallest item in $\bigcup_{k\in[d]} S_k$. Hence, $\sum_{k\in[d]} s_i(S_k) > d \cdot \sum_{p\in[l]} c_{i,p}$. This implies that in i's 1-out-of-d MMS defining partition, at least one item in $\bigcup_{k\in[d]} S_k$ is assigned to machine $p \ge l+1$. Combining with the fact that $t_{i,l}$ is the smallest item in $\bigcup_{k\in[d]} S_k$, we have $s_i(t_{i,l}) \le c_{i,l+1}$. ■

By these properties, for each machine $l \in P_i$, we can assign either the internal items $C^I_{i,l}$ or the external item $t_{i,l-1}$ to l, such that its completion time does not exceed MMS^d_i. This intuition guides the allocation of the items to the agents in the following part.

E.2.3 Part 3: allocating the items to the agents

Lastly, for any bundle $B_j \in B$, we arbitrarily choose two agents $i_1, i_2 \in N$ and allocate them the items in B_j as formally described in Algorithm 7. Recall that in the imaginary assignment of B_j to each agent $i \in \{i_1, i_2\}$, the items in B_j are divided into internal items $\bigcup_{l\in[k_i]} C^I_{i,l}$ and external items $\{t_{i,0}, \ldots, t_{i,k_i}\}$. Let $E = \{e^*_1, \ldots, e^*_{|E|}\}$ contain all external items shared by i_1 and i_2. Note that $e^*_1 = t_{i_1,0} = t_{i_2,0}$. We allocate the items in B_j to agents i_1 and i_2 in $|E|$ rounds. In each round $q \in [|E|]$, we first find the machines of i_1 and i_2 to which the shared external items e^*_q and e^*_{q+1} are assigned (denoted by l_1, l_2, l'_1 and l'_2, respectively. If $q = |E|$, simply let $l'_1 = k_{i_1}$ and $l'_2 = k_{i_2}$). We then find the agent $i_k \in \{i_1, i_2\}$ whose machine $l_k + 1$ has more internal items. We allocate i_k her internal items from machine $l_k + 1$ to machine l'_k, and allocate the other agent i_k's external items from machine l_k to machine $l'_k - 1$.

Since $2 \cdot d = 2 \cdot \lfloor \frac{n}{2} \rfloor \le n$, no more than n agents are needed to allocate all items. Thus to prove Theorem 4, it remains to show that each agent can assign her allocated items to her machines such that the total workload on each of the machines does not exceed its capacity.

Algorithm 7 Allocating the items to the agents

Input: A d-partition of the items $\mathbf{B} = (B_1, \ldots, B_d)$ returned by Algorithm 5.

Output: An allocation $\mathbf{A} = (A_1, \ldots, A_n)$ such that $v_i(A_i) \leq \mathsf{MMS}_i^d$ for all $i \in N$.

Initialize $A_i \leftarrow \emptyset$ for every $i \in N$.
for $j = 1$ to d **do**
  Arbitrarily choose 2 agents $i_1, i_2 \in N$, $N \leftarrow N \setminus \{i_1, i_2\}$.
  $\{C^I_{i_1,1}, \ldots, C^I_{i_1,k_{i_1}}\}, \{t_{i_1,0}, \ldots, t_{i_1,k_{i_1}}\} \leftarrow$ *Algorithm* 6(B_j, i_1).
  $\{C^I_{i_2,1}, \ldots, C^I_{i_2,k_{i_2}}\}, \{t_{i_2,0}, \ldots, t_{i_2,k_{i_2}}\} \leftarrow$ *Algorithm* 6(B_j, i_2).
  $E \leftarrow \{t_{i_1,0}, \ldots, t_{i_1,k_{i_1}}\} \cap \{t_{i_2,0}, \ldots, t_{i_2,k_{i_2}}\}$. Re-label $E \leftarrow \{e^*_1, \ldots, e^*_{|E|}\}$. // Shared external items by i_1 and i_2
  for $q = 1$ to $|E|$ **do**
    Find $l_1 \in [0, k_{i_1}]$ and $l_2 \in [0, k_{i_2}]$ such that $e^*_q = t_{i_1,l_1} = t_{i_2,l_2}$.
    if $q < |E|$ **then**
      Find $l'_1 \in [0, k_{i_1}]$ and $l'_2 \in [0, k_{i_2}]$ such that $e^*_{q+1} = t_{i_1,l'_1} = t_{i_2,l'_2}$.
    else
      $l'_1 = k_{i_1}$ and $l'_2 = k_{i_2}$.
    end if
    if $|C^I_{i_1,l_1+1}| \geq |C^I_{i_2,l_2+1}|$ **then**
      $A_{i_1} \leftarrow \bigcup_{l=l_1+1}^{l'_1} C^I_{i_1,l}$, $A_{i_2} \leftarrow \bigcup_{l=l_1}^{l'_1-1} t_{i_1,l}$.
    else
      $A_{i_2} \leftarrow \bigcup_{l=l_2+1}^{l'_2} C^I_{i_2,l}$, $A_{i_1} \leftarrow \bigcup_{l=l_2}^{l'_2-1} t_{i_2,l}$.
    end if
  end for
end for

Proof of Theorem 4. Consider any bundle $B_j \in \mathbf{B}$ and assume the two chosen agents are $i_1, i_2 \in N$. We first look at the first round of the process of allocating the items in B_j to i_1 and i_2. Without loss of generality, assume that the first machine of i_1 contains more internal items than that of i_2, i.e., $C^I_{i_1,1} \geq C^I_{i_2,1}$. From the algorithm, the items i_1 takes are $\bigcup_{l=1}^{l'_1} C^I_{i_1,l}$. By the second property of the imaginary assignment, these items can be assigned to the first l'_1 machines of i_1 such that the total workload on each machine does not exceed its capacity. Besides, the items i_2 takes are $\bigcup_{l=0}^{l'_1-1} t_{i_1,l}$, which are e^*_1 and a subset of $\bigcup_{l=2}^{l'_2} C^I_{i_2,l}$. By the second and third properties of the imaginary assignment, these items can be assigned to the first l'_2 machines of i_2 such that the total workload on each machine does not exceed its capacity. The same reasoning can be applied to all following rounds. By induction, it follows that both i_1 and i_2 can assign their allocated items to their machines such that the total workload on each machine does not exceed its capacity. This means that both i_1 and i_2 receive costs no more than their 1-out-of-d MMS, which completes the proof. ■

For the multiplicative relaxation of MMS, by Theorem 4 and Observation 1, a $\lceil \frac{n}{\lfloor \frac{n}{2} \rfloor} \rceil$-MMS allocation is guaranteed. As the bin packing setting, after a slight modification, Algorithm 5 computes a 2-MMS allocation, which is better than $\lceil \frac{n}{\lfloor \frac{n}{2} \rfloor} \rceil$-MMS.

Proof of Corollary 3. We show that by replacing the value of d with n, Algorithm 5 computes a 2-MMS allocation. Particularly, in the new version of Algorithm 5, we partition the items in M into n bundles in a round-robin fashion and allocate each of the n bundles to one agent in N. By the properties of the imaginary assignment, for each agent, the makespan of processing either the internal items or the external items in her bundle using her machines does not exceed MMS_i^n. This implies that for each agent, the cost of her bundle does not exceed $2 \cdot \mathsf{MMS}_i^n$, which completes the proof. ■

F Proportionality up to one or any item

We now discuss two other relaxations for proportionality, i.e., proportional up to one item (PROP1) and proportional up to any item (PROPX), which are also widely studied for additive costs.

Definition 2 (α-PROP1 and α-PROPX) *An allocation $\mathbf{A} = (A_1, \ldots, A_n)$ is α-approximate proportional up to one item (α-PROP1) if $v_i(A_i \backslash \{e\}) \leq \alpha \cdot \frac{v_i(M)}{n}$ for all agents $i \in N$ and some item $e \in A_i$. It is α-approximate proportional up to any item (α-PROPX) if $v_i(A_i \backslash \{e\}) \leq \alpha \cdot \frac{v_i(M)}{n}$ for all agents $i \in N$ and any item $e \in A_i$. The allocation is PROP1 or PROPX if $\alpha = 1$.*

It is easy to see that a PROPX allocation is also PROP1. Although exact PROPX or PROP1 allocations are guaranteed to exist for additive costs, when the costs are subadditive, no algorithm can be better than n-PROP1 or n-PROPX. Consider an instance with n agents and $n+1$ items. The cost function is $v_i(S) = 1$ for all agents $i \in N$ and any non-empty subset $S \subseteq M$. Clearly, the cost function is subadditive since $v_i(S) + v_i(T) \geq v_i(S \cup T)$ for any $S, T \subseteq M$. By the pigeonhole principle, at least one agent i receives two or more items in any allocation of M. After removing any item $e \in A_i$, A_i is still not empty. That is, $v_i(A_i \backslash \{e\}) = 1 = n \cdot \frac{v_i(M)}{n}$ for any $e \in A_i$. This example can be easily extended to the bin packing and job scheduling settings, and thus we have the following theorem.

Theorem 5 *For the bin packing and job scheduling settings, no algorithm performs better than n-PROP1 or n-PROPX.*

Proof. For the bin packing setting, consider an instance with n agents and $n+1$ items. The capacity of each agent's bins is 1, i.e, $c_i = 1$ for all $i \in N$. Each item is very tiny so that every agent can pack all items in just one bin, e.g., $s_{i,j} = \frac{1}{n+1}$ for any $i \in N$ and $e_j \in M$. Therefore, we have $v_i(M) = 1$ and $\mathsf{PROP}_i = \frac{1}{n}$ for each agent $i \in N$. By the pigeonhole principle, at least one agent i receives two or more items in any allocation of M. After removing any item $e \in A_i$, agent i still needs one bin to pack the remaining items. Hence, we have $v_i(A_i \backslash \{e\}) = 1 = n \cdot \mathsf{PROP}_i$ for any $e \in A_i$.

For the job scheduling setting, consider an instance with $2n$ agents and $2n+1$ items where each agent possesses $2n$ machines with the same speed of 1, and the size of each item is 1 for every agent. It can be easily seen that for every agent $i \in N$, the maximum completion time of her machines is minimized when assigning two items to one machine and one item to each of the remaining $2n-1$ machines. Therefore, we have $v_i(M) = 2$ and $\mathsf{PROP}_i = \frac{2}{2n} = \frac{1}{n}$ for any $i \in N$. Similarly, by the pigeonhole principle, at least one agent i receives two or more items in any allocation of M. This implies that $v_i(A_i \backslash \{e\}) = 1 = n \cdot \mathsf{PROP}_i$ for any $e \in A_i$, thus completing the proof. ■