



# Partitioning Problems with Splittings and Interval Targets

Samuel Bismuth  

Department of Computer Science, Ariel University, Israel

Vladislav Makarov 

Department of Mathematics and Computer Science, St. Petersburg State University, Russia

Erel Segal-Halevi  

Department of Computer Science, Ariel University, Israel

Dana Shapira  

Department of Computer Science, Ariel University, Israel

---

## Abstract

---

The  $n$ -way number partitioning problem is a classic problem in combinatorial optimization, with applications to diverse settings such as fair allocation and machine scheduling. All these problems are NP-hard, but various approximation algorithms are known. We consider three closely related kinds of approximations.

The first two variants optimize the partition such that: in the first variant some fixed number  $s$  of items can be *split* between two or more bins and in the second variant we allow at most a fixed number  $t$  of *splittings*. The third variant is a decision problem: the largest bin sum must be within a pre-specified interval, parameterized by a fixed rational number  $u$  times the largest item size.

When the number of bins  $n$  is unbounded, we show that every variant is strongly NP-complete. When the number of bins  $n$  is fixed, the running time depends on the fixed parameters  $s, t, u$ . For each variant, we give a complete picture of its running time.

For  $n = 2$ , the running time is easy to identify. Our main results consider any fixed integer  $n \geq 3$ . Using a two-way polynomial-time reduction between the first and the third variant, we show that  $n$ -way number-partitioning with  $s$  split items can be solved in polynomial time if  $s \geq n - 2$ , and it is NP-complete otherwise. Also,  $n$ -way number-partitioning with  $t$  splittings can be solved in polynomial time if  $t \geq n - 1$ , and it is NP-complete otherwise. Finally, we show that the third variant can be solved in polynomial time if  $u \geq (n - 2)/n$ , and it is NP-complete otherwise. Our positive results for the optimization problems consider both min-max and max-min versions.

Using the same reduction, we provide a fully polynomial-time approximation scheme for the case where the number of split items is lower than  $n - 2$ .

**2012 ACM Subject Classification** Mathematics of computing  $\rightarrow$  Combinatorial algorithms

**Keywords and phrases** Number Partitioning, Fair Division, Identical Machine Scheduling

**Digital Object Identifier** 10.4230/LIPIcs.ISAAC.2024.12

**Related Version** *Full Version*: <https://arxiv.org/pdf/2204.11753> [4]

**Funding** *Samuel Bismuth*: Israel Science Foundation grant no. 712/20.

*Erel Segal-Halevi*: Israel Science Foundation grant no. 712/20.

**Acknowledgements** The paper started from discussions in the stack exchange network:

(1) <https://cstheory.stackexchange.com/q/42275>;

(2) <https://cs.stackexchange.com/a/141322>.

Dec-SPLITITEM[3, 1]( $\mathcal{X}$ ) was first solved by Mikhail Rudoy using case analysis. The relation to FPTAS was raised by Chao Xu. We are also grateful to John L. <https://cs.stackexchange.com/a/149567>.



© Samuel Bismuth, Vladislav Makarov, Erel Segal-Halevi, and Dana Shapira; licensed under Creative Commons License CC-BY 4.0

35th International Symposium on Algorithms and Computation (ISAAC 2024).

Editors: Julián Mestre and Anthony Wirth; Article No. 12; pp. 12:1–12:15

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

In the classic setting of the  $n$ -way number partitioning problem, the inputs are a list  $\mathcal{X} = (x_1, \dots, x_m)$  of  $m$  non-negative integers and a number of bins  $n$ , and the required output is an  $n$ -way partition (a partition of the integers into  $n$  bins) that attains some pre-determined objective. In the decision version of the problem, the objective is to decide whether there exists an  $n$ -way partition of  $\mathcal{X}$  such that every bin sum is exactly equal to  $\sum_{x_i \in \mathcal{X}} x_i/n$  (we call it a *perfect partition*). In the min-max optimization version, the objective is to find an  $n$ -way partition of  $\mathcal{X}$  such that the maximum bin sum is minimized, while in the max-min optimization version, the goal is to maximize the smallest bin sum.

For each problem of this paper, the problem objective is mentioned first, the fixed parameters in square brackets, and the problem input in parenthesis. Let us formally define the min-max version of the  $n$ -way number partitioning problem, where  $n$  is a fixed parameter:

*MinMax-PART* $[n](\mathcal{X})$ : Minimize  $\max(b_1, \dots, b_n)$ , where  $b_1, \dots, b_n$  are sums of bins in an  $n$ -way partition of  $\mathcal{X}$ .

When  $n$  is unbounded, *Dec-PART* $(n, \mathcal{X})$  (the decision version of the  $n$ -way number partitioning problem) is known to be strongly NP-hard (it is equivalent to 3-partition) [8]. And for every fixed  $n \geq 2$  *Dec-PART* $[n](\mathcal{X})$  is known to be NP-hard [9]. In addition, many instances of the decision version are negative (there is no perfect partition). The latter reasons give us a good motivation to investigate variants of the  $n$ -way number partitioning problem, for which the running time complexity is better, and the number of positive instances (admitting a perfect partition) will significantly grow. We present three variants, that relax the initial problem to solve our concerns. The first two variants allow “divisible” items, bounded by some natural numbers  $s$  and  $t$ . We define the decision and the min-max versions as follows:

*Dec-SPLITITEM* $[n, s](\mathcal{X})$ : Decide if there exists a partition of  $\mathcal{X}$  among  $n$  bins with at most  $s$  split items, such that  $\max(b_1, \dots, b_n) \leq S$ .

*MinMax-SPLITITEM* $[n, s](\mathcal{X})$ : Minimize  $\max(b_1, \dots, b_n)$ , where  $b_1, \dots, b_n$  are sums of bins in an  $n$ -way partition of  $\mathcal{X}$  in which at most  $s$  items are split.

*Dec-SPLITTING* $[n, t](\mathcal{X})$ : Decide if there exists a partition of  $\mathcal{X}$  among  $n$  bins with at most  $t$  splittings, such that  $\max(b_1, \dots, b_n) \leq S$ .

*MinMax-SPLITTING* $[n, t](\mathcal{X})$ : Minimize  $\max(b_1, \dots, b_n)$ , where  $b_1, \dots, b_n$  are sums of bins in an  $n$ -way partition of  $\mathcal{X}$  in which at most  $t$  splittings are allowed.

The number of splittings is at least the number of split items but might be larger. For example, a single item split into 10 different bins counts as 9 splittings. Note that the problem definitions do not determine in advance which items will be split, but only bound their number, or bound the number of splittings. The solver may decide which items to split after receiving the input.

Our motivating application for the variants comes from fair division and machine scheduling. For fair division, some  $m$  items with market values  $x_1, \dots, x_m$  have to be divided among  $n$  agents. A *perfect partition* is one in which each partner receives a total value of exactly  $\sum_i x_i/n$ . When the original instance does not admit a perfect partition, we may want to *split* one or more items among two or more agents. Or, we can allow some *splittings*. Divisible items are widespread in fair division applications – the ownership of one or more items may be split to attain a perfectly fair partition. However, divisible items may be inconvenient or expensive. Therefore, the number of split items or splittings should be bounded. The same is true for machine scheduling, in which agents are considered as machines, and items as jobs. It is possible for a job to be divided and be processed by two machines simultaneously. The following examples tackle real-life fair division or machine scheduling problems.

(1) Consider  $n = 2$  heirs who inherited  $m = 3$  houses and have to divide them fairly. The house values are  $\mathcal{X} = (100, 200, 400)$ . If all houses are considered discrete, then an equal division is not possible. If all houses can be split, then an equal division is easy to attain by giving each heir 50% of every house, but it is inconvenient since it requires all houses to be jointly managed. A solution often used in practice is to decide in advance that a *single* house can be split (or only one splitting is allowed). In this case, after receiving the input, we can determine that splitting the house with a value of 400 lets us attain a division in which each heir receives the same value of 350.<sup>1</sup>

(2) Consider a food factory with  $n = 3$  identical chopping machines, who has to cut  $m = 4$  vegetables with processing times  $\mathcal{X} = (10, 7, 5, 5)$  minutes. Each job is divisible as one vegetable may be cut in different machines, but splitting a job is inconvenient since it requires washing more dishes. Without splitting, the minimum total processing time is 10 minutes: (10), (7), (5, 5). By splitting the vegetable with processing time 10 into three different machines, the processing time is 9 minutes: (7, 2), (5, 4), (5, 4).

The third variant only admits a decision version, parameterized by a rational number  $u \geq 0$ :

*Dec-INTER* $[n, u](\mathcal{X})$ : Decide if there exists a partition of  $\mathcal{X}$  into  $n$  bins with sums  $b_1, \dots, b_n$  such that  $S \leq \max(b_1, \dots, b_n) \leq S + u \cdot M$ , where  $S := (\sum_i x_i)/n$  and  $M := (\max_i x_i)/n$ .

We will also use another definition of this variant, parameterized by a rational number  $v \geq 0$ :

*Dec-INTER* $[n, v](\mathcal{X})$ : Decide if there exists a partition of  $\mathcal{X}$  into  $n$  bins with sums  $b_1, \dots, b_n$  such that  $S \leq \max(b_1, \dots, b_n) \leq (1 + v) \cdot S$ , where  $S := (\sum_i x_i)/n$ .

Note that when  $u = vS/M$ , both definitions are the same. In general, the runtime complexity of this problem depends on the size of the allowed interval (i.e., the interval  $[S, S + u \cdot M]$ ): the problem is NP-complete when the interval is “small” and in P when the interval is “large”. Specifically, when  $n = 2$ , the runtime complexity depends on the ratio of the allowed interval to the *bin sum*, while when  $n \geq 3$  it depends on the ratio of the allowed interval to the *largest item*. We notice that, if we can solve INTER for any interval length in polynomial-time, then by binary search we can solve PART in polynomial-time; which is not possible unless P=NP. So in INTER, we look for the smallest interval for which we can decide in polynomial-time whether it contains a solution.

As an application example, consider the fair allocation of indivisible items among two agents. Suppose there is a small amount of money, that can be used to compensate for a small deviation from equality in the allocation. But if the deviation is too big, the agents prefer to find another solution. We can check the feasibility using INTER such that the interval is the amount of money available.

The INTER variant is similar to the *Fully Polynomial-Time Approximation Scheme* (FPTAS) definition:

► **Definition 1.** An FPTAS for *MinMax-PART* $[n](\mathcal{X})$  is an algorithm that finds, for each rational  $\epsilon > 0$ , an  $n$ -way partition of  $\mathcal{X}$  with  $OPT \leq \max(b_1, \dots, b_n) \leq (1 + \epsilon) \cdot OPT$ , where  $OPT$  is the smallest possible value of  $\max(b_1, \dots, b_n)$  in the given instance, in time  $O(\text{poly}(m, 1/\epsilon, \log S))$ .

An FPTAS finds a solution for which the relative deviation from optimality depends on the optimal *integral* solution. In contrast, in the *Dec-INTER* $[n, u](\mathcal{X})$  problem, we look for a solution for which the relative deviation from optimality depends on the optimal *fractional* solution.

<sup>1</sup> Split the house worth 400 such that one heir gets 7/8 of it, and the other gets 1/8 of it plus both the 100 and 200 houses.

■ **Table 1** Run-time complexity of the  $n$ -way number partitioning variants. In SPLITITEM,  $s$  (an integer) is the number of items the algorithm is allowed to split. In SPLITTING,  $t$  (an integer) is the number of splittings the algorithm is allowed to make. In INTER,  $u$  (a rational number) is the ratio between the allowed interval length and  $M$ .

Problem	Objective	Num of bins	Bound	Run-time complexity
PART	Dec	Unbounded	$s = t$	Strongly NP-hard [[8]]
		Constant $n$	$= u = 0$	NP-complete [[9]]
SPLITITEM	Dec	Unbounded	$s$ (any)	Strongly NP-hard [[4]Corollary 19]
		Constant $n \geq 3$	$s < n - 2$	NP-complete [Theorem 14]
	MinMax	$s \geq n - 2$	$O(\text{poly}(m, \log S))$ [Theorem 16]	
	MaxMin	$s \geq n - 2$	$O(\text{poly}(m, \log S))$ [[4]Theorem 22]	
	All	Constant $n \geq 2$	$s \geq n - 1$	$O(m + n)$ [cut-the-line]
SPLITTING	Dec	Unbounded	$t$ (any)	Strongly NP-hard [[4]Theorem 20]
		Constant $n$	$t < n - 1$	NP-complete [Theorem 17]
	All		$t \geq n - 1$	$O(m + n)$ [cut-the-line]
INTER	Dec	Unbounded	$u$ (any)	Strongly NP-hard [[4]Theorem 17]
		Constant $n \geq 3$	$u < n - 2$	NP-complete [Theorem 11]
	Dec		$u \geq n - 2$	$O(\text{poly}(m, \log S))$ [Theorem 10]
		Constant $n \geq 2$	$u \geq n - 1$	$O(m + n)$ [cut-the-line+Thm 13]
		$n = 2$	$u > 0$	$O(\text{poly}(m, \log S, 1/u))$ [Theorem 6]

### Contribution

When  $s, t, u = 0$ , SPLITITEM, SPLITTING and INTER decision versions are equivalent to the NP-hard PART decision version. In contrast, when  $s, t \geq n - 1$  the problem is easily solvable by the following algorithm: put the items on a line, cut the line into  $n$  pieces with an equal total value, and put each piece in a bin. Since  $n - 1$  cuts are made, at most  $n - 1$  items need to be split. So for  $n = 2$ , the runtime complexity of the Dec-SPLITITEM $[n, s](\mathcal{X})$  and Dec-SPLITTING $[n, t](\mathcal{X})$  problem is well-understood (assuming  $P \neq NP$ ): it is polynomial-time solvable if and only if  $s, t \geq 1$ . The case for INTER is slightly different since  $u, v$  are rational numbers. We summarize all our results in Table 1.

In Section 4 we show a two-way polynomial-time reduction between problems SPLITITEM and INTER. This reduction is the key for many of our results. We use it to handle the case where the number of split items is smaller than  $n - 2$ . First, we design an FPTAS in [4][Appendix A.3]. Second, we develop a practical (not polynomial-time) algorithm, for solving MinMax-SPLITITEM $[n, s](\mathcal{X})$  for any  $s \geq 0$ . The algorithm can use any practical algorithm for solving the MinMax-PART $[n](\mathcal{X})$  problem. The latest helps us in [4][Appendix A.4] to conduct some experiences to various randomly generated instances and analyze the effect of  $s$  on the quality of the attained solution. The supplement provides complementary results and technical proof details omitted from the main text.

## 2 Related Work

In most combinatorial optimization problems, there is a clear distinction between discrete and continuous variables. E.g., when a problem is modeled by a mixed-integer program, each variable in the program is determined in advance to be either discrete (must get an integer value) or continuous (can get any real value). The problems we study belong to a much smaller class of problems, in which all variables are potentially continuous, but there is an upper bound on the number of variables that can be non-discrete. We describe some such problems below.

**Bounded splitting in fair division.** The idea of finding fair allocations with a bounded number of split items originated from [5, 6]. They presented the *Adjusted Winner* (AW) procedure for allocating items among two agents with possibly different valuations. AW finds an allocation that is *envy-free* (no agent prefers the bundle of another agent), *equitable* (both agents receive the same subjective value), and *Pareto-optimal* (there is no other allocation where some agent gains and no agent loses), and in addition, at most a single item is split between the agents. Hence, AW solves a problem that is similar to  $\text{Dec-SPLITITEM}[n = 2, s = 1](\mathcal{X})$  but more general, since AW allows the agents to have different valuations to the same items.

Most similar to our paper is the recent work of [15]. Their goal is to find an allocation among  $n$  agents with different valuations, which is both fair and *fractionally Pareto-optimal* (fPO), a property stronger than Pareto-optimality (there is no other discrete or fractional allocation where some agent gains and no agent loses). This is a very strong requirement: when  $n$  is fixed, and the valuations are *non-degenerate* (i.e., for every two agents, no two items have the same value-ratio), the number of fPO allocations is polynomial in  $m$ , and it is possible to enumerate all such allocations in polynomial time. Based on this observation, they present an algorithm that finds an allocation with the smallest number of split items, among all allocations that are fair and fPO. In contrast, in our paper, we do not require fPO, which may allow allocations with fewer split items or splittings. However, the number of potential allocations becomes exponential, so enumerating them all is no longer feasible.

Another paper [3] studies the same problems, but, whereas our paper focuses on identical valuations, they give new results on binary valuations (i.e., each agent values each item as 0 or 1), generalized binary valuations (i.e., each agent values each item as 0 or  $x_i$ , which can be considered as the price of the item) and negative results on non-degenerate valuations, complementing the results given by [15].

Recently, [1, 2] studied an allocation problem where some items are divisible and some are indivisible. In contrast to our setting, in [1] the distinction between divisible and indivisible items is given in advance, that is, the algorithm can only divide items that are pre-determined as divisible. In [2], for each good, some agents may regard it as indivisible, while other agents may regard the good as divisible. In our setting, only the *number* of divisible items (splittings) is given in advance, but the algorithm is free to choose *which* items to split after receiving the input.

**Splitting in job scheduling.** There are several variants of job scheduling problems in which it is allowed to break jobs apart. They can be broadly classified into *preemption* and *splitting*. In the preemption variants, different parts of a job must be processed at different times. In the three-field notation, they are denoted by “pmtn” and were first studied by [13]. In the splitting variants, different parts of a job may be processed simultaneously on different machines. They are denoted by “split” and were introduced by [18].

Various problems have been studied in job scheduling with preemption. The most closely related to our problem is the generalized multiprocessor scheduling (GMS). It has two variants. In the first variant, the total number of preemptions is bounded by some fixed integer. In the second variant, each job  $j$  has an associated parameter that bounds the number of times  $j$  can be preempted. In both variants, the goal is to find a schedule that minimizes the makespan subject to the preemption constraints. For identical machines, [16] prove that with the bound of  $n - 2$  on the total number of preemptions, the problem is NP-hard, whereas [13] shows a linear-time algorithm with  $n - 1$  preemptions. In Theorem 17 we prove an analogous result where the bound is on the total number of splittings.

In all the works we surveyed, there is no global bound on the number of splitting jobs. As far as we know, bounding the number of splittings or split jobs was not studied before.

**Fractional bin-packing.** Another problem, in which splitting was studied, is the classical bin-packing problem. Bin-packing with fragmented items are first introduced by [12]. They called the problem fragmentable object bin-packing problem and prove that the problem is NP-hard. It is later split into two variants. In the first variant called bin-packing with size-increasing fragmentation (BP-SIF), each item may be fragmented; overhead units are added to the size of every fragment. In the second variant called bin-packing with size-preserving fragmentation (BP-SPF) each item has a size and a cost; fragmenting an item increases its cost but does not change its size. Menakerman and Rom [14] show that BP-SIF and BP-SPF are NP-hard in the strong sense. Despite the hardness, they present several algorithms and investigate their performance. Their algorithms use classic algorithms for bin-packing, like next-fit and first-fit decreasing, as a base for their algorithms.

Finally, the fractional knapsack problem with penalties is recently introduced by [11]. They develop an FPTAS and a dynamic program for the problem, and they show an extensive computational study comparing the performance of their models.

### 3 Partition with Interval Target

In this section we analyze the problems  $\text{Dec-INTER}[n, v](\mathcal{X})$  and  $\text{Dec-INTER}[n, u](\mathcal{X})$ .

#### 3.1 The $\text{Dec-Inter}[n, v](\mathcal{X})$ problem

Given an instance of  $\text{Dec-INTER}[n, v](\mathcal{X})$ , we say that a partition of  $\mathcal{X}$  is  $v$ -feasible if  $S \leq \max(b_1, \dots, b_n) \leq (1+v) \cdot S$ , where  $b_1, \dots, b_n$  are the bin sums and  $S$  is the sum of items divided by  $n$ . The  $\text{Dec-INTER}[n, v](\mathcal{X})$  problem is to decide whether a  $v$ -feasible partition exists.

► **Lemma 2.** *When  $v \geq 1$ , the problem  $\text{Dec-INTER}[n, v](\mathcal{X})$  can be decided in linear time by a greedy algorithm.*

The proof is in [4][Appendix B.3]. We focus below on the case  $v < 1$ .

► **Definition 3.** *Given an instance of  $\text{Dec-INTER}[n, v](\mathcal{X})$ , a rational number  $\epsilon > 0$ , and a partition of  $\mathcal{X}$  among  $n$  bins, an almost-full bin is a bin with sum larger than  $(1+v) \cdot S / (1+\epsilon)$ .*

A known FPTAS for the  $\text{MinMax-PART}[n](\mathcal{X})$  problem [17] gives us valuable information since we can easily verify that if the output of this FPTAS is smaller than  $(1+v) \cdot S$  then in any  $n$ -way partition of  $\mathcal{X}$ , at least one bin is almost-full. To gain more information on the instance, we apply an FPTAS for a constrained variant of  $\text{PART}$ , with a *Critical Coordinate*. For an integer  $n \geq 2$ , a list  $\mathcal{X}$ , and a rational number  $v > 0$ , we define the following problem:

$\text{MinMax-PART}[n, v, i](\mathcal{X})^2$ : *Minimize  $\max(b_1, \dots, b_{i-1}, b_{i+1}, \dots, b_n)$  subject to  $b_i \leq (1+v) \cdot S$  where  $b_1, \dots, b_n$  are bin sums in an  $n$ -way partition of  $\mathcal{X}$ .*

The general technique developed by [17] for converting a dynamic program to an FPTAS can be used to design an FPTAS for  $\text{MinMax-PART}[n, v, i](\mathcal{X})$ ; we give the details in [4][Appendix C.1]. We denote by  $\text{FPTAS}(\text{MinMax-PART}[n, v, i](\mathcal{X}), \epsilon)$  the largest bin sum in the solution obtained by the FPTAS.

► **Lemma 4.** *For any  $n \geq 2, v > 0, \epsilon > 0$ , if, for all  $i \in [n]$ ,  $\text{FPTAS}(\text{MinMax-PART}[n, v, i](\mathcal{X}), \epsilon) > (1+v) \cdot S$ , then in any  $v$ -feasible  $n$ -way partition of  $\mathcal{X}$ , at least two bins are almost-full.*

<sup>2</sup> The critical coordinate is parameterized by  $i$ . In this work, we do not use this parameter, but other results may iterate over every critical coordinate possible.



**Proof.** Suppose by contradiction that there exists a  $v$ -feasible partition of  $\mathcal{X}$  with at most one almost-full bin. Let  $i$  be the index of the bin with the largest sum in that partition. Since bin  $i$  has the largest sum, if there is one almost-full bin, it must be bin  $i$ . Hence, bins that are not  $i$  are not almost-full, so  $\max(b_1, \dots, b_{i-1}, b_{i+1}, \dots, b_n) \leq (1+v) \cdot S / (1+\epsilon)$ . Moreover,  $b_i \leq (1+v) \cdot S$  since the partition is  $v$ -feasible. Therefore,  $\text{FPTAS}(\text{MinMax-PART}[n, v, i](\mathcal{X}), \epsilon) \leq (1+v) \cdot S$  by the definition of FPTAS. This contradicts the lemma assumption.  $\blacktriangleleft$

Using Theorem 4, we can now derive a complete algorithm for  $\text{Dec-INTER}[n = 2, v](\mathcal{X})$ .

■ **Algorithm 1**  $\text{Dec-INTER}[n = 2, v](\mathcal{X})$ .

- 
- 1:  $b_2 \leftarrow \text{FPTAS}(\text{MinMax-PART}[n = 2, v, i](\mathcal{X}), \epsilon = v/2)$ .
  - 2: If  $b_2 \leq (1+v) \cdot S$ , return “yes”.
  - 3: Else, return “no”.
- 

► **Theorem 5.** For any rational  $v > 0$ , Algorithm 1 solves the  $\text{Dec-INTER}[n = 2, v](\mathcal{X})$  problem in time  $O(\text{poly}(m, \log S, 1/v))$ , where  $m$  is the number of items in  $\mathcal{X}$  and  $S = (\sum_i x_i)/n$  is the perfect bin sum.

**Proof.** The run-time of Algorithm 1 is dominated by the run-time of the FPTAS for  $\text{MinMax-PART}[n = 2, v, i](\mathcal{X})$ , which is  $O(\text{poly}(m, \log S, 1/\epsilon)) = O(\text{poly}(m, \log S, 1/v))$  (we show in [4][Appendix C.3.1] that the exact run-time is  $O(\frac{m}{v} \log S)$ ). It remains to prove that Algorithm 1 indeed solves  $\text{Dec-INTER}[n = 2, v](\mathcal{X})$  correctly.

If  $b_2$ , the returned bin sum of  $\text{FPTAS}(\text{MinMax-PART}[n = 2, v, i](\mathcal{X}), \epsilon = v/2)$ , is at most  $(1+v) \cdot S$ , then the partition found by the FPTAS is  $v$ -feasible, so Algorithm 1 answers “yes” correctly. Otherwise, by Theorem 4, in any  $v$ -feasible partition of  $\mathcal{X}$  into two bins, both bins are almost-full. This means that, in any  $v$ -feasible partition, both bin sums  $b_1$  and  $b_2$  are larger than  $(1+v) \cdot S / (1+\epsilon)$ , which is larger than  $S$  since  $\epsilon = v/2$ . So  $b_1 + b_2 > 2S$ . But this is impossible since the sum of the items is  $2S$  by assumption. Hence, no  $v$ -feasible partition exists, and Algorithm 1 answers “no” correctly.  $\blacktriangleleft$

► **Corollary 6.** For any rational  $u > 0$ , Algorithm 1 solves the  $\text{Dec-INTER}[n = 2, u](\mathcal{X})$  problem in time  $O(\text{poly}(m, \log S, 1/u))$ .

**Proof.** For any rational  $u > 0$ , let  $v := uM/S$ , that is  $v > 0$ . Algorithm 1 solves the  $\text{Dec-INTER}[n = 2, v = uM/S](\mathcal{X})$  problem in time  $O(\text{poly}(m, \log S, S/uM))$ , where  $m$  is the number of items in  $\mathcal{X}$  and  $S = (\sum_i x_i)/n$  is the perfect bin sum. Since  $S \leq mM$ , the algorithm runs in time  $O(\text{poly}(m, \log S, 1/u))$  for any  $u > 0$ .  $\blacktriangleleft$

► **Remark 7.** The reader may wonder why we cannot use a similar algorithm for  $n \geq 3$ . For example, we could have considered a variant of  $\text{MinMax-PART}[n, v, i](\mathcal{X})$  with two critical coordinates:

*Minimize*  $\max(b_3, \dots, b_n)$  *subject to*  $b_1 \leq (1+v) \cdot S$  *and*  $b_2 \leq (1+v) \cdot S$ , *where*  
 $b_1, b_2, b_3, \dots, b_n$  *are bin sums in an  $n$ -way partition of*  $\mathcal{X}$ .

---

<sup>3</sup> Instead of an FPTAS for  $\text{MinMax-PART}[n = 2, v, i](\mathcal{X})$ , we could use an FPTAS for the Subset Sum problem [10], using the same arguments. The critical coordinate is not needed in the Subset Sum FPTAS, since the output is always smaller than the target. We prefer to use the FPTAS for  $\text{MinMax-PART}[n = 2, v, i](\mathcal{X})$ , since it is based on the general technique of [17], that we use later for solving other problems.

If the FPTAS for this problem does not find a  $v$ -feasible partition, then any  $v$ -feasible partition must have at least three almost-full bins. Since not all bins can be almost-full, one could have concluded that there is no  $v$ -feasible partition into  $n = 3$  bins.

Unfortunately, the problem with two critical coordinates probably does not have an FPTAS even for  $n = 3$ , since it is equivalent to the Multiple Subset Sum problem, which does not have an FPTAS unless  $P=NP$  [7]. In the next subsection we handle the case  $n \geq 3$  in a different way.

### 3.2 Dec-Inter $[n, u](\mathcal{X})$ : an algorithm for $n \geq 3$ and $u \geq n - 2$

The case when  $u \geq n - 1$  is solved by the cut-the-line algorithm combined with Theorem 13. Here, we prove a more general case where  $u \geq n - 2$ . Given an instance of Dec-INTER $[n, u](\mathcal{X})$ , where the sum of the items is  $n \cdot S$  and the largest item is  $n \cdot M$ , where  $S, M \in \mathbb{Q}$ , we say that a partition of  $\mathcal{X}$  is  $u$ -possible if  $S \leq \max(b_1, \dots, b_n) \leq S + u \cdot M$ , where  $b_1, \dots, b_n$  are the bin sums. The Dec-INTER $[n, u](\mathcal{X})$  problem is to decide whether a  $u$ -possible partition exists. Given an instance of Dec-INTER $[n, u](\mathcal{X})$ , we let  $v := uM/S$ , so that a partition is  $u$ -possible if and only if it is  $v$ -feasible.

The algorithm starts by running FPTAS(MinMax-PART $[n, v, i](\mathcal{X})$ ,  $\epsilon = v/(4m^2)$ ). If the FPTAS find a  $v$ -feasible partition, we return “yes”. Otherwise, by Theorem 4, any  $v$ -feasible partition must have at least two almost-full bins.

We take a detour from the algorithm and prove some existential results about partitions with two or more almost-full bins. We assume that there are more items than bins, that is,  $m > n$ . This assumption is because if  $m \leq n$ , one can compute all the combinations using brute force (note that the running time is polynomial since  $2^m \leq 2^n = O(1)$  since  $n$  is a fixed parameter).

#### 3.2.1 Structure of partitions with two or more almost-full bins

We distinguish between big, medium, and small items defined as follows. A *small item* is an item with length smaller than  $2\epsilon nS$ ; a *big item* is an item with length greater than  $(\frac{v}{n-2} - 2\epsilon)nS$ . All other items are called *medium items*. Our main structural Lemma is the following.

► **Lemma 8.** *Suppose that  $u \geq n - 2$ ,  $v = uM/S < 1$ ,  $\epsilon = v/4m^2$  and the following properties hold.*

- (1) *There is no  $v$ -feasible partition with at most 1 almost-full bin;*
- (2) *There is a  $v$ -feasible partition with at least 2 almost-full bins.*

*Then, there is a  $v$ -feasible partition with the following properties.*

- (a) *Exactly two bins (w.l.o.g. bins 1 and 2) are almost-full.*
- (b) *The sum of every not-almost-full bin  $i \in \{3, \dots, n\}$  satisfies*

$$\left(1 - \frac{2}{n-2}v - 2\epsilon\right) \cdot S \leq b_i \leq \left(1 - \frac{2}{n-2}v + (n-1)2\epsilon\right) \cdot S.$$

- (c) *Every item in an almost-full bin is a big-item.*
- (d) *Every item in a not-almost-full bin is either a small-item, or a big-item larger or equal to every item in bins 1,2.*
- (e) *There are no medium-items at all.*
- (f) *Every not-almost-full bin contains the same number of big-items, say  $\ell$ , where  $\ell$  is an integer (it may contain, in addition, any number of small-items).*
- (g) *Every almost-full bin contains  $\ell + 1$  big-items (and no small-items).*



As an example of this situation, consider an instance with 7 items, all of which have size 1, with  $n = 5$  and  $u = 3$ . Then, there is a  $u$ -possible partition with two almost-full bins:  $(1, 1), (1, 1), (1), (1), (1)$ , and no  $u$ -possible partition with 1 or 0 almost-full bins. See [4][Appendix B.4.1] for details. A full proof ([4][Appendix B.4.2]) of the Lemma appears in the appendix, here we provide a sketch proof.

**Proof Sketch.** We start with an arbitrary  $v$ -feasible partition with some  $r \geq 2$  almost-full bins  $1, \dots, r$ , and convert it using a sequence of transformations to another  $v$ -feasible partition satisfying properties (a)–(g), as explained below. Note that the transformations are not part of our algorithm and are only used to prove the lemma. First, we note that there must be at least one bin that is not almost-full, since the sum of an almost-full bin is larger than  $S$  whereas the sum of all  $n$  bins is  $n \cdot S$ .

**For (a)**, if there are  $r \geq 3$  almost-full bins, we move any item from one of the almost-full bins  $3, \dots, r$  to some not-almost-full bin. We prove that, as long as  $r \geq 3$ , the target bin remains not-almost-full. This transformation is repeated until  $r = 2$  and only bins 1 and 2 remain almost-full.

**For (b)**, for the lower bound, if there is  $i \in \{3, \dots, n\}$  for which  $b_i$  is smaller than the lower bound, we move an item from bins 1, 2 to bin  $i$ . We prove that bin  $i$  remains not-almost-full, so by assumption (1), bins 1, 2 must remain almost-full. We repeat until  $b_i$  satisfies the lower bound. Once all bins satisfy the lower bound, we prove that the upper bound is satisfied too.

**For (c)**, if bin 1 or 2 contains an item that is not big, we move it to some bin  $i \in \{3, \dots, n\}$ . We prove that bin  $i$  remains not-almost-full, so by assumption (1), bins 1, 2 must remain almost-full. We repeat until bins 1 and 2 contain only big-items.

**For (d)**, if some bin  $i \in \{3, \dots, n\}$  contains an item bigger than  $2nS\epsilon$  and smaller than any item in bin 1 or bin 2, we exchange it with an item from bin 1 or 2. We prove that, after the exchange,  $b_i$  remains not-almost-full, so bins 1, 2 must remain almost-full. We repeat until bins 1, 2 contain only the smallest big-items. Note that transformations (b), (c), (d) increase the sum in the not-almost-full bins  $3, \dots, n$ , so the process must end.

**For (e)**, it follows logically from properties (d) and (c): if bins 1, 2 contain only big items and the other bins contain only big and small items, then the instance cannot contain any medium items (that are neither big nor small). For clarity and verification, we provide a stand-alone proof.

**For (f)**, we use the fact that the difference between two not-almost-full bins is at most  $2nS\epsilon$  by property (b), and show that it is too small to allow a difference of a whole big-item.

**For (g)**, because by (d) bins 1 and 2 contain the smallest big-items, whereas their sum is larger than bins  $3, \dots, n$ , they must contain at least  $\ell + 1$  big-items. We prove that, if they contain  $\ell + 2$  big-items, then their sum is larger than  $(1 + v)S$ , which contradicts  $v$ -feasibility.  $\blacktriangleleft$

Properties (f) and (g) imply:

► **Corollary 9.** *Suppose that  $u \geq n - 2$ ,  $v = uM/S$  and  $\epsilon = v/4m^2$ . Let  $B \subseteq \mathcal{X}$  be the set of big items in  $\mathcal{X}$ . If there is a  $v$ -feasible partition with at least two almost-full bins, and no  $v$ -feasible partition with at most one almost-full bin, then  $|B| = n\ell + 2$  for  $\ell \in \mathbb{N}$ .*

### 3.2.2 Back to the algorithm

We have left the algorithm at the point when  $\text{FPTAS}(\text{MinMax-PART}[n = 2, v, i](\mathcal{X}), \epsilon = v/4m^2) > (1 + v) \cdot S$  that is the FPTAS did not return a  $v$ -feasible partition. Theorem 4 implies that if a  $v$ -feasible partition exists, then there exists a  $v$ -feasible partition satisfying all properties of Theorem 8 and Theorem 9. We can find such a partition (if it exists) in two steps:

- **For bins 1, 2:** Find a  $v$ -feasible partition of the  $2\ell + 2$  smallest items in  $B$  into two bins with  $\ell + 1$  items in each bin.
- **For bins 3,  $\dots$ ,  $n$ :** Find a  $v$ -feasible partition of the remaining items in  $\mathcal{X}$  into  $n - 2$  bins.

For bins 3,  $\dots$ ,  $n$ , we use the FPTAS for the problem  $\text{MinMax-PART}[n = n - 2](\mathcal{X})$ . If it returns a  $v$ -feasible partition, we are done. Otherwise, by FPTAS definition, every partition into  $(n - 2)$  bins must have at least one almost-full bin. But by Theorem 8(a), all bins 3,  $\dots$ ,  $n$  are not almost-full which is a contradiction. Therefore, if the FPTAS does not find a  $v$ -feasible partition, we answer “no”. Bins 1 and 2 require a more complicated algorithm that is explained in [4][Appendix B.2]. We are now ready to present the complete algorithm for  $\text{Dec-INTER}[n, u](\mathcal{X})$ , presented in Algorithm 2.

■ **Algorithm 2**  $\text{Dec-INTER}[n, u](\mathcal{X})$  (complete algorithm).

- 
- 1:  $v \leftarrow uM/S$  and  $\epsilon \leftarrow v/(4m^2)$ .
  - 2: If  $\text{FPTAS}(\text{MinMax-PART}[n, v, i](\mathcal{X}), \epsilon) \leq (1 + v) \cdot S$ , return “yes”.
  - 3:  $B \leftarrow \{x_i \in \mathcal{X} \mid x_i > nS(\frac{v}{n-2} - 2\epsilon)\}$  ▷ big items
  - 4: If  $|B|$  is not of the form  $n\ell + 2$  for some integer  $\ell$ , return “no”.
  - 5:  $B_{1:2} \leftarrow$  the  $2\ell + 2$  smallest items in  $B$ . ▷ break ties arbitrarily
  - 6:  $B_{3..n} \leftarrow \mathcal{X} \setminus B_{1:2}$ . ▷ big and small items
  - 7:  $b_3 \leftarrow \text{FPTAS}(\text{MinMax-PART}[n = n - 2](B_{3..n}), \epsilon)$  ▷ Computes an approximately-optimal  $(n - 2)$ -way partition of  $B_{3..n}$  and returns the maximum bin sum in the partition.
  - 8: If  $b_3 > (1 + v)S$ , return “no”. ▷ The FPTAS did not find a  $v$ -feasible partition.
  - 9:  $\overline{B}_{1:2} \leftarrow \{nM_{1:2} - x \mid x \in B_{1:2}\}$  and  $\bar{v} \leftarrow (S + vS - S_{1:2})/\overline{S}_{1:2}$ .
  - 10: Look for a  $\bar{v}$ -feasible partition of  $\overline{B}_{1:2}$  into two subsets of  $\ell + 1$  items (see [4][Appendix B.2]).
  - 11: If a  $\bar{v}$ -feasible partition is found, return “yes”. Else, return “no”.
- 

► **Theorem 10.** For any fixed integer  $n \geq 3$  and rational number  $u \geq n - 2$ , Algorithm 2 solves  $\text{Dec-INTER}[n, u](\mathcal{X})$  in  $O(\text{poly}(m, \log S))$  time, where  $m$  is the number of items in  $\mathcal{X}$ , and  $S$  is the average bin size.

**Proof.** If Algorithm 2 answers “yes”, then clearly a  $v$ -feasible partition exists. To complete the correctness proof, we have to show that the opposite is true as well.

Suppose there exists a  $v$ -feasible partition. If the partition has at most one almost-full bin, then by Theorem 4, it is found by the FPTAS in step 2. Otherwise, the partition must have at least two almost-full bins, and there exists a  $v$ -feasible partition satisfying the properties of Theorem 8. By Theorem 9, the algorithm does not return “no” in step 4. By properties (a) and (b), there exists a partition of  $B_{3..n}$  into  $n - 2$  bins 3,  $\dots$ ,  $n$  which are not almost-full. By definition, the FPTAS in step 7 finds a partition with  $\max(b_3, \dots, b_n) \leq (1 + v)S$ . The final steps, regarding the partition of  $B_{1:2}$ , are justified by the discussion at [4][Appendix B.2]. The complete running time  $O(\text{poly}(m, \log S))$  of Algorithm 2 is justified by the running time of the FPTAS for  $\text{FPTAS}(\text{MinMax-PART}[n = 2, v, i](\mathcal{X}), \epsilon)$

and for FPTAS(MinMax-PART $[n = n - 2](B_{3..n}), \epsilon)$ . Note that  $1/v$  is polynomial in  $m$  since  $1/v = S/uM \leq mM/uM = m/u = O(m)$  since  $u$  is fixed. The exact running time,  $O(m^4 \log S)$ , is detailed in [4][Appendix C.3.2].  $\blacktriangleleft$

### 3.3 Hardness for $n \geq 3$ bins and $u < n - 2$

The following theorem complements the previous subsection.

► **Theorem 11.** *Given a fixed integer  $n \geq 3$  and a positive rational number  $u < n - 2$ , the problem Dec-INTER $[n, u](\mathcal{X})$  is NP-complete.*

**Proof.** Given an  $n$ -way partition of  $m$  items, summing the sizes of all elements in each bin allows us to check whether the partition is  $u$ -possible in linear time. So, the problem is in NP. To prove that Dec-INTER $[n, u](\mathcal{X})$  is NP-Hard, we reduce from the equal-cardinality partition problem, proved to be NP-hard in [9]: given a list with an even number of integers, decide if they can be partitioned into two subsets with the same sum and the same cardinality.

Given an instance  $\mathcal{X}_1$  of equal-cardinality partition, denote the number of items in  $\mathcal{X}_1$  by  $2m'$ . Define  $M$  to be the sum of numbers in  $\mathcal{X}_1$  divided by  $2n(1 - \frac{u}{n-2})$ , so that the sum of items in  $\mathcal{X}_1$  is  $2n(1 - \frac{u}{n-2})M$  (where  $n$  and  $u$  are the parameters in the theorem statement). We can assume w.l.o.g. that all items in  $\mathcal{X}_1$  are at most  $n(1 - \frac{u}{n-2})M$ , since if some item is larger than half of the sum, the answer is necessarily “no”.

Construct an instance  $\mathcal{X}_2$  of the equal-cardinality partition problem by replacing each item  $x$  in  $\mathcal{X}_1$  by  $nM - x$ . So  $\mathcal{X}_2$  contains  $2m'$  items between  $n(\frac{u}{n-2})M$  and  $nM$ . Their sum, which we denote by  $2S'$ , satisfies  $2S' = 2m' \cdot nM - 2n(1 - \frac{u}{n-2})M = 2n(m' - 1 + \frac{u}{n-2})M$ . Clearly,  $\mathcal{X}_1$  has an equal-sum equal-cardinality partition (with bin sums  $n(1 - \frac{u}{n-2})M$ ) iff  $\mathcal{X}_2$  has an equal-sum equal-cardinality partition (with bin sums  $S' = n(m' - 1 + \frac{u}{n-2})M$ ).

Construct an instance  $(\mathcal{X}_3, u)$  of Dec-INTER $[n, u](\mathcal{X})$  by adding  $(n - 2)(m' - 1)$  items of size  $nM$ . Note that  $nM$  is indeed the largest item size in  $\mathcal{X}_3$ . Denote the sum of item sizes in  $\mathcal{X}_3$  by  $nS$ . Then

$$\begin{aligned} nS &= 2S' + (n - 2)(m' - 1) \cdot nM = n \left( 2(m' - 1) + \frac{2u}{n - 2} + (n - 2)(m' - 1) \right) \cdot M \\ &= n \left( n(m' - 1) + \frac{2u}{n - 2} \right) M; \end{aligned}$$

$$S + uM = \left( n(m' - 1) + \frac{2u}{n - 2} + u \right) M = \left( n(m' - 1) + \frac{nu}{n - 2} \right) M = S',$$

so a partition of  $\mathcal{X}_3$  is  $u$ -possible if and only if the sum of each of the  $n$  bins in the partition is at most  $S + uM = S'$ .

We now prove that if  $\mathcal{X}_2$  has an equal-sum equal-cardinality partition, then the instance  $(\mathcal{X}_3, u)$  has a  $u$ -possible partition, and vice versa. If  $\mathcal{X}_2$  has an equal-sum partition, then the items of  $\mathcal{X}_2$  can be partitioned into two bins of sum  $S'$ , and the additional  $(n - 2)(m' - 1)$  items can be divided into  $n - 2$  bins of  $m' - 1$  items each. The sum of these items is

$$(m' - 1) \cdot nM = n(m' - 1)M = S - \frac{2}{n - 2}uM < S + uM = S', \quad (1)$$

so the resulting partition is a  $u$ -possible partition of  $\mathcal{X}_3$ . Conversely, suppose  $\mathcal{X}_3$  has a  $u$ -possible partition. Let us analyze its structure.

## 12:12 Partitioning Problems with Splittings and Interval Targets

Since the partition is  $u$ -possible, the sum of every two bins is at most  $2(S + uM)$ . So the sum of every  $n - 2$  bins is at least  $nS - 2(S + uM) = (n - 2)S - 2uM$ . Since the largest  $(n - 2)(m' - 1)$  items in  $\mathcal{X}_3$  sum up to exactly  $(n - 2)S - 2uM$  by (1), every  $n - 2$  bins must contain at least  $(n - 2)(m' - 1)$  items. Since  $\mathcal{X}_3$  has  $(n - 2)(m' - 1) + 2m'$  items overall,  $n - 2$  bins must contain exactly  $(n - 2)(m' - 1)$  items, such that each item size must be  $nM$ , and their sum must be  $(n - 2)S - 2uM$ . The other two bins contain together  $2m'$  items with a sum of  $2(S + uM)$ , so each of these bins must have a sum of exactly  $S + uM$ . Since  $(m' - 1) \cdot nM < S + uM$  by (1), each of these two bins must contain exactly  $m'$  items. These latter two bins are an equal-sum equal-cardinality partition for  $\mathcal{X}_2$ . This construction is done in polynomial time, completing the reduction. ◀

### 4 Partition with Split Items

We now deal with the problem SPLITITEM. We redefine the Dec-SPLITITEM $[n, s](\mathcal{X})$  problem. For a fixed number  $n \geq 2$  of bins, given a list  $\mathcal{X}$ , the number of split items  $s \in \{0, \dots, m\}$  and a rational number  $v \geq 0$ , define:

*Dec-SPLITITEM $[n, s, v](\mathcal{X})$ : Decide if there exists a partition of  $\mathcal{X}$  among  $n$  bins with at most  $s$  split items, such that  $\max(b_1, \dots, b_n) \leq (1 + v)S$ .*

The special case  $v = 0$  corresponds to the Dec-SPLITITEM $[n, s](\mathcal{X})$  problem. The following Lemma shows that, w.l.o.g., we can consider only the longest items for splitting.

► **Lemma 12.** *For every partition with  $s \in \mathbb{N}$  split items and bin sums  $b_1, \dots, b_n$ , there exists a partition with the same bin sums  $b_1, \dots, b_n$  in which only the  $s$  largest items are split.*

**Proof.** Consider a partition in which some item with length  $x$  is split between two or more bins, whereas some item with length  $y > x$  is allocated entirely to some bin  $i$ . Construct a new partition as follows: first move item  $x$  to bin  $i$ ; second remove from bin  $i$ , a fraction  $\frac{x}{y}$  of item  $y$ ; and finally split that fraction of item  $y$  among the other bins, in the same proportions as the previous split of item  $x$ . All bin sums remain the same. Repeat the argument until only the longest items are split. ◀

► **Theorem 13.** *For any fixed integers  $n \geq 2$  and  $u \geq 0$ , there is a polynomial-time reduction from Dec-INTER $[n, u](\mathcal{X})$  to Dec-SPLITITEM $[n, s = u, v = 0](\mathcal{X})$ .*

**Proof.** Given an instance  $\mathcal{X}$  of Dec-INTER $[n, u](\mathcal{X})$ , we add  $u$  items of size  $nM$ , where  $nM$  is the size of the biggest item in  $\mathcal{X}$  to construct an instance  $\mathcal{X}'$  of Dec-SPLITITEM $[n, s = u, v = 0](\mathcal{X}')$ .

First, assume that  $\mathcal{X}$  has a  $u$ -possible partition. Then there are  $n$  bins with a sum at most  $S + uM$ . Take the  $u$  added items of size  $nM$  and add them to the bins, possibly splitting some items between bins, such that the sum of each bin becomes exactly  $S + uM$ . This is possible because the sum of the items in  $\mathcal{X}'$  is  $nS + unM = n(S + uM)$ . The result is a 0-feasible partition of  $\mathcal{X}'$  with at most  $u$  split items.

Second, assume that  $\mathcal{X}'$  has a 0-feasible partition with at most  $u$  split items. Then there are  $n$  bins with a sum of exactly  $S + uM$ . By Theorem 12, we can assume the split items are the largest ones, which are the  $u$  added items of size  $nM$ . Remove these items to get a partition of  $\mathcal{X}$ . The sum in each bin is now at most  $S + uM$ , so the partition is  $u$ -possible. This construction is done in polynomial time, which completes the proof. ◀

► **Corollary 14.** *For every fixed integers  $n \geq 3$  and  $s \in \{0, \dots, n - 3\}$ , the problem Dec-SPLITITEM $[n, s](\mathcal{X})$  is NP-complete.*

**Proof.** Theorem 11 and Theorem 13 imply that  $\text{Dec-SPLITITEM}[n, s](\mathcal{X})$  is NP-hard. The problem  $\text{Dec-SPLITITEM}[n, s](\mathcal{X})$  is in NP since given a partition, summing the sizes of the items (or items fractions) in each bin let us check in linear time whether the partition has equal bin sums.  $\blacktriangleleft$

► **Theorem 15.** *For any fixed integers  $n \geq 2, s \geq 0$  and rational  $v \geq 0$ , there is a polynomial-time reduction from  $\text{Dec-SPLITITEM}[n, s, v](\mathcal{X})$ , to  $\text{Dec-INTER}[n, u](\mathcal{X})$  for some rational number  $u \geq s$ .*

**Proof.** Given an instance  $\mathcal{X}$  of  $\text{Dec-SPLITITEM}[n, s, v](\mathcal{X})$ , denote the sum of all items in  $\mathcal{X}$  by  $nS$  and the largest item size by  $nM$  where  $S, M \in \mathbb{Q}$ . Construct an instance  $\mathcal{X}'$  of  $\text{Dec-INTER}[n, u](\mathcal{X}')$  by removing the  $s$  largest items from  $\mathcal{X}$ . Denote the sum of remaining items by  $nS'$  for some  $S' \leq S$ , and the largest remaining item size by  $nM'$  for some  $M' \leq M$ . Note that the size of every removed item is between  $nM'$  and  $nM$ , so  $sM' \leq S - S' \leq sM$ . Set  $u := (S + vS - S')/M'$ , so  $S' + uM' = S + vS$ . Note that  $u \geq (S - S')/M' \geq s$ .

First, assume that  $\mathcal{X}$  has a  $v$ -feasible partition with  $s$  split items. By Theorem 12, we can assume that only the  $s$  largest items are split. Therefore, removing the  $s$  largest items results in a partition of  $\mathcal{X}'$  with no split items, where the sum in each bin is at most  $S + vS = S' + uM'$ . This is a  $u$ -possible partition of  $\mathcal{X}'$ .

Second, assume that  $\mathcal{X}'$  has a  $u$ -possible partition. In this partition, each bin sum is at most  $S' + uM' = S + vS$ , so it is a  $v$ -feasible partition of  $\mathcal{X}'$ . To get a  $v$ -feasible partition of  $\mathcal{X}$ , take the  $s$  previously removed items and add them to the bins, possibly splitting some items between bins, such that the sum in each bin remains at most  $S + vS$ . This is possible since the items sum is  $nS \leq n(S + vS)$ . This construction is done in polynomial time.  $\blacktriangleleft$

Combining Theorem 15 with Theorem 10 provides a polynomial time algorithm to solve  $\text{Dec-SPLITITEM}[n, s, v](\mathcal{X})$  for any fixed  $n \geq 3, s \geq n - 2$  and rational  $v \geq 0$ . The latter is used to solve the  $\text{MinMax-SPLITITEM}[n, s](\mathcal{X})$  optimization problem by using binary search on the parameter  $v$  of the  $\text{Dec-SPLITITEM}[n, s, v](\mathcal{X})$  problem. The details are given in [4][Appendix B.1]. The binary search procedure needs to solve at most  $\log_2(nS)$  instances of  $\text{Dec-SPLITITEM}[n, s, v](\mathcal{X})$ .

► **Corollary 16.** *For any fixed integers  $n \geq 3$  and  $s \geq n - 2$ ,  $\text{MinMax-SPLITITEM}[n, s](\mathcal{X})$  can be solved in  $O(\text{poly}(m, \log S))$  time.*

We complete this result by providing a polynomial-time algorithm for the max-min version:  $\text{MaxMin-SPLITITEM}[n, s](\mathcal{X})$  for  $s \geq n - 2$  in [4][Appendix A.1].

## 5 Partition with Splittings

In this section, we analyze the SPLITTING variant.

► **Theorem 17.** *For any fixed integer  $n \geq 2$  and fixed  $t \in \mathbb{N}$  such that  $t \leq n - 2$ , the problem  $\text{Dec-SPLITTING}[n, t](\mathcal{X})$  is NP-complete.*

**Proof.** Given a partition with  $n$  bins,  $m$  items, and  $t$  splittings, summing the size of each item (or fraction of item) in each bin allows us to check whether or not the partition is perfect in linear time. So, the problem is in NP.

To prove that  $\text{Dec-SPLITTING}[n, t](\mathcal{X})$  is NP-Hard, we apply a reduction from the Subset Sum problem. We are given an instance  $\mathcal{X}_1$  of Subset Sum with  $m$  items summing up to  $S$  and target sum  $T < S$ . We build an instance  $\mathcal{X}_2$  of  $\text{Dec-SPLITTING}[n, t](\mathcal{X}_2)$  by adding two items,  $x_1, x_2$ , such that  $x_1 = S + T$  and  $x_2 = 2S(t + 1) - T$  and  $n - 2 - t$  auxiliary items of size  $2S$ . Notice that the sum of the items in  $\mathcal{X}_2$  equals

$$\begin{aligned}
S + (S + T) + 2S(t + 1) - T + 2S(n - 2 - t) &= 2S + 2S(t + 1) + 2S(n - 2 - t) \\
&= 2S \cdot (1 + t + 1 + n - 2 - t) = 2Sn.
\end{aligned}$$

The goal is to partition items into  $n$  bins with a sum of  $2S$  per bin, and at most  $t$  splittings.

First, assume that there is a subset of items  $W_1$  in  $\mathcal{X}_1$  with a sum equal to  $T$ . Define a set,  $W_2$ , of items that contains all items in  $\mathcal{X}_1$  that are not in  $W_1$ , plus  $x_1$ . The sum of  $W_2$  is  $(S - T) + x_1 = S + T + S - T = 2S$ . Assign the items of  $W_2$  to the first bin. Assign each auxiliary item to a different bin. There are  $n - (n - 2 - t + 1) = t + 1$  bins left. The sum of the remaining items is  $2S(t + 1)$ . Using the “cut-the-line” algorithm described in the introduction, these items can be partitioned into  $t + 1$  bins of equal sum  $2S$ , with at most  $t$  splittings. All in all, there are  $n$  bins with a sum of  $2S$  per bin, and the total number of splittings is at most  $t$ .

Second, assume that there exists an equal partition for  $n$  bins with  $t$  splittings. Since  $x_2 = 2S(t + 1) - T = 2S \cdot t + (2S - T) > 2S \cdot t$ , this item must be split between  $t + 1$  bins, which makes the total number of splittings at least  $t$ . Also, the auxiliary items must be assigned without splittings into  $n - 2 - t$  different bins. There is  $n - t - 1 - n + 2 + t = 1$  bin remaining, say bin  $i$ , containing only whole items, not containing any part of  $x_2$ , and not containing any auxiliary item. Bin  $i$  must contain  $x_1$ , otherwise its sum is at most  $S$  (sum of items in  $\mathcal{X}_1$ ). Let  $W_1$  be the items of  $\mathcal{X}_1$  that are not in bin  $i$ . The sum of  $W_1$  is  $S - (2S - x_1) = x_1 - S = T$ , so it is a solution to  $\mathcal{X}_1$ . ◀

## 6 Conclusion and Future Directions

We presented three variants of the  $n$ -way number partitioning problem.

In the language of fair item allocation, we have solved the problem of finding a fair allocation among  $n$  agents with identical valuations, when the ownership of some  $s$  items may be split between agents. When agents may have different valuations, there are various fairness notions, such as *proportionality*, *envy-freeness* or *equitability*. A future research direction is to develop algorithms for finding such allocations with a bounded number of shared items. We already have preliminary results for proportional allocation among three agents with different valuations, which are based on the algorithms in the present paper.

In the language of machine scheduling,  $\text{MinMax-SPLITITEM}[n, s](\mathcal{X})$  corresponds to finding a schedule that minimizes the makespan on  $n$  identical machines when  $s$  jobs can be split between the machines;  $\text{Dec-INTER}[n, u](\mathcal{X})$  corresponds to find a schedule in which the makespan is in a given interval. In a separate technical report, we have replicated the results in the present paper for the more general case of *uniform machines* in which machines may have different speeds  $r_j$ , such that a job with length  $x_i$  runs on machine  $j$  in time  $x_i/r_j$ . It may be interesting to study the more general setting of *unrelated machines*.

Our analysis shows the similarities and differences between these variants and the more common notion of FPTAS. One may view our results as introducing a new kind of approximation that approximates a decision problem by returning “yes” if and only if there exists a solution between  $PER$  and  $(1 + v) \cdot PER$ , where  $PER$  represents the value of a perfect solution. For the  $n$ -way number partitioning problem, a perfect solution is easy to define: it is a partition with equal bin sums. A more general definition of  $PER$  could be the solution to the fractional relaxation of an integer linear program representing the problem. As shown, NP-hard decision problems may become tractable when  $v$  is sufficiently large.



## References

- 1 Xiaohui Bei, Zihao Li, Jinyan Liu, Shengxin Liu, and Xinhang Lu. Fair division of mixed divisible and indivisible goods. *Artif. Intell.*, 293:103436, 2021. doi:10.1016/J.ARTINT.2020.103436.
- 2 Xiaohui Bei, Shengxin Liu, and Xinhang Lu. Fair division with subjective divisibility. *CoRR*, abs/2310.00976, 2023. doi:10.48550/arXiv.2310.00976.
- 3 Samuel Bismuth, Ivan Bliznets, and Erel Segal-Halevi. Fair division with bounded sharing: Binary and non-degenerate valuations. In Guido Schäfer and Carmine Ventre, editors, *Algorithmic Game Theory*, pages 89–107, Cham, 2024. Springer Nature Switzerland. doi:10.1007/978-3-031-71033-9\_6.
- 4 Samuel Bismuth, Vladislav Makarov, Erel Segal-Halevi, and Dana Shapira. Partitioning problems with splittings and interval targets, 2024. arXiv:2204.11753.
- 5 Steven J. Brams and Alan D. Taylor. *Fair Division: From Cake Cutting to Dispute Resolution*. Cambridge University Press, Cambridge UK, February 1996.
- 6 Steven J. Brams and Alan D. Taylor. *The win-win solution - guaranteeing fair shares to everybody*. W. W. Norton & Company, New York, 2000.
- 7 Alberto Caprara, Hans Kellerer, and Ulrich Pferschy. The multiple subset sum problem. *SIAM Journal on Optimization*, 11(2):308–319, 2000. doi:10.1137/S1052623498348481.
- 8 M. R. Garey and David S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM J. Comput.*, 4(4):397–411, 1975. doi:10.1137/0204035.
- 9 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.
- 10 Hans Kellerer, Renata Mansini, Ulrich Pferschy, and Maria Grazia Speranza. An efficient fully polynomial approximation scheme for the subset-sum problem. *Journal of Computer and System Sciences*, 66(2):349–370, 2003. doi:10.1016/S0022-0000(03)00006-0.
- 11 Enrico Malaguti, Michele Monaci, Paolo Paronuzzi, and Ulrich Pferschy. Integer optimization with penalized fractional values: The knapsack case. *Eur. J. Oper. Res.*, 273(3):874–888, 2019. doi:10.1016/j.ejor.2018.09.020.
- 12 C.A. Mandal, P.P. Chakrabarti, and S. Ghose. Complexity of fragmentable object bin packing and an application. *Computers and Mathematics with Applications*, 35(11):91–97, 1998. doi:10.1016/S0898-1221(98)00087-X.
- 13 Robert McNaughton. Scheduling with deadlines and loss functions. *Management Science*, 6(1):1–12, 1959. URL: <http://www.jstor.org/stable/2627472>.
- 14 Nir Menakerman and Raphael Rom. Bin packing with item fragmentation. In Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Roberto Tamassia, editors, *Algorithms and Data Structures, 7th International Workshop, WADS 2001, Providence, RI, USA, August 8-10, 2001, Proceedings*, volume 2125 of *Lecture Notes in Computer Science*, pages 313–324. Springer, 2001. doi:10.1007/3-540-44634-6\_29.
- 15 Fedor Sandomirskiy and Erel Segal-Halevi. Efficient fair division with minimal sharing. *Oper. Res.*, 70(3):1762–1782, 2022. doi:10.1287/opre.2022.2279.
- 16 Evgeny Shchepin and Nodari Vakhania. New tight np-hardness of preemptive multiprocessor and open-shop scheduling. In *Proceedings of 2nd multidisciplinary international conference on scheduling: theory and applications MISTA 2005*, pages 606–629, 2005.
- 17 Gerhard J. Woeginger. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? *INFORMS J. Comput.*, 12(1):57–74, 2000. doi:10.1287/IJOC.12.1.57.11901.
- 18 Wenxun Xing and Jiawei Zhang. Parallel machine scheduling with splitting jobs. *Discret. Appl. Math.*, 103(1-3):259–269, 2000. doi:10.1016/S0166-218X(00)00176-1.