# Approximation Schemes for k-Subset Sum Ratio and k-Way Number Partitioning Ratio

**Sotiris Kanellopoulos**[1]
National Technical University of Athens and Archimedes, Athena Research Center, Greece

**Giorgos Mitropoulos**[1]
National Technical University of Athens and Archimedes, Athena Research Center, Greece

**Antonis Antonopoulos**
National Technical University of Athens, Greece

**Nikos Leonardos**
National Technical University of Athens, Greece

**Aris Pagourtzis**
National Technical University of Athens and Archimedes, Athena Research Center, Greece

**Christos Pergaminelis**
National Technical University of Athens and Archimedes, Athena Research Center, Greece

**Stavros Petsalakis**
National Technical University of Athens, Greece

**Kanellos Tsitouras**
National Technical University of Athens, Greece

## Abstract

The Subset Sum Ratio problem (SSR) asks, given a multiset $A$ of positive integers, to find two disjoint subsets of $A$ such that the *largest-to-smallest* ratio of their sums is minimized. In this paper we study the $k$-version of SSR, namely $k$-Subset Sum Ratio ($k$-SSR), which asks to minimize the *largest-to-smallest* ratio of sums of $k$ disjoint subsets of $A$. We develop an approximation scheme for $k$-SSR running in $O(n^{2k}/\varepsilon^{k-1})$ time, where $n = |A|$ and $\varepsilon$ is the error parameter. To the best of our knowledge, this is the first FPTAS for $k$-SSR for fixed $k > 2$.

We also study the $k$-way Number Partitioning Ratio ($k$-PART) problem, which differs from $k$-SSR in that the $k$ subsets must constitute a partition of $A$; this problem in fact corresponds to the objective of minimizing the largest-to-smallest sum ratio in the family of Multiway Number Partitioning problems. We present a more involved FPTAS for $k$-PART, also achieving $O(n^{2k}/\varepsilon^{k-1})$ time complexity. Notably, $k$-PART is also equivalent to the Minimum Envy-Ratio problem with identical valuation functions, which has been studied in the context of fair division of indivisible goods. Thus, for the case of identical valuations, our FPTAS represents a significant improvement over the $O(n^{4k^2+1}/\varepsilon^{2k^2})$ bound obtained by Nguyen and Rothe's FPTAS [36] for Minimum Envy-Ratio with general additive valuations.

Lastly, we propose a second FPTAS for $k$-SSR, which employs carefully designed calls to the first one; the new scheme has a time complexity of $\widetilde{O}(n/\varepsilon^{3k-1})$, thus being much faster when $n \gg 1/\varepsilon$.

**2012 ACM Subject Classification** Theory of computation → Approximation algorithms analysis

**Keywords and phrases** Fully polynomial-time approximation schemes, Subset Sum Ratio, Number Partitioning, Fair division, Envy minimization, Pseudo-polynomial time algorithms

**Digital Object Identifier** 10.4230/LIPIcs.ISAAC.2025.44

**Related Version** *Full Version*: https://arxiv.org/abs/2503.18241 [24]

[1] Equal Contribution



36th International Symposium on Algorithms and Computation (ISAAC 2025).
Editors: Ho-Lin Chen, Wing-Kai Hon, and Meng-Tsung Tsai; Article No. 44; pp. 44:1–44:22

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

**Funding** Supported by project MIS 5154714 of the National Recovery and Resilience Plan Greece 2.0 funded by the European Union under the NextGenerationEU Program.

# 1 Introduction

The Subset Sum and Partition problems are fundamental in combinatorial optimization, with numerous applications in fair resource allocation, cryptography and scheduling. The Equal Subset Sum (ESS) problem [43] asks for two disjoint subsets of a given set with equal sums, while its optimization counterpart, Subset Sum Ratio (SSR) [43, 4], asks to minimize the ratio of sums of two disjoint subsets. These NP-hard problems arise in bioinformatics [15, 14], game theory and economics [30, 18], and cryptography [44], among others. A special case of ESS belongs to PPP [37], which is a subclass of TFNP, further adding to the ongoing interest for these problems.

In this paper we consider variations involving multiple subsets, namely $k$-Subset Sum Ratio ($k$-SSR) and $k$-way Number Partitioning Ratio ($k$-PART)[2]. In $k$-SSR, the goal is to find $k$ disjoint subsets such that the *largest-to-smallest* ratio of their sums is minimized, while $k$-PART further requires these subsets to form a partition of the input. Since these problems are NP-hard, we consider approximation algorithms. In particular, we present fully polynomial-time approximation schemes (FPTASs) for both problems.

Our results apply, among others, to scheduling tasks across $k$ processors (e.g. [39, 17, 22, 16]), as well as to *fair division of indivisible goods*, in particular the study of *envy* in discrete settings (e.g. [30, 36, 28] and more recently [2, 41, 45]). Specifically, $k$-PART is equivalent to the Minimum Envy-Ratio problem [30] with identical valuation functions, for which we achieve a significant improvement over (more general) earlier work [36].

## Related work

Since Bellman's seminal work on Subset Sum [5], no major improvement had occurred until a recent spike in interest culminated in various faster pseudo-polynomial algorithms [26, 7, 23, 12] and FPTASs [25, 20].

The ESS and SSR problems have seen recent advances, including FPTASs for SSR [35, 31, 1, 8] and an improved exact algorithm for ESS [33]. Notably, Bringmann [8] provided an SSR FPTAS running in time $O(n/\varepsilon^{0.9386})$, which is faster than known Subset Sum lower bounds. He achieved this in part by considering two cases based on input density. If the input is dense, a pigeonhole argument guarantees two subsets with sums within $1+\varepsilon$ and removing their intersection yields an approximate solution. If the input is sparse, keeping only the polylog$(1/\varepsilon)$ largest elements is proven to be sufficient for the approximation. Regarding variations of (Equal) Subset Sum involving $k \geq 2$ subsets, [3] extends the techniques of [26, 7] to provide both deterministic and randomized pseudo-polynomial algorithms.

For Partition, advances include a number of (very) recent FPTASs [34, 9, 11], with the latest work by Chen et al. [13] achieving an FPTAS running in time $\widetilde{O}(n+1/\varepsilon)$, which is near optimal under the Strong Exponential Time Hypothesis. In Multiway Number Partitioning [6, 40, 27, 38], different objective functions define various approaches to distributing elements among subsets, such as minimizing the maximum sum, maximizing the minimum sum, minimizing the *largest-to-smallest* difference of sums, or minimizing the *largest-to-smallest* ratio of sums. These objectives are not comparable: there are instances showing that the corresponding optimal solutions differ; we discuss this in detail in Subsection 2.1, along with applications of each variation.

[2] This problem is usually referred to as "multiway", but we call it "$k$-way" to emphasize that $k$ is fixed.

Bazgan et al. [4] provided the first FPTAS for SSR and demonstrated that Subset Sum Difference does not admit an FPTAS, unless P = NP. Similar arguments that apply to Partition have been further studied in [42]. Additionally, various related problems such as 3-Partition and Bin Packing are strongly NP-hard [19], implying that they do not admit an FPTAS or a pseudo-polynomial time algorithm unless P = NP. However, these problems involve a non-constant number of subsets. For our problems, we consider a *fixed* number $k$ of disjoint subsets. This restriction is quite natural in applications, as it is more likely to distribute a great amount of items to a small number of agents.

Sahni presents an FPTAS for Multiway Number Partitioning in [39] that minimizes the largest sum. However, this technique does not seem applicable to minimize the ratio. In [30], Lipton et al. show a PTAS for Minimum Envy-Ratio with identical valuations and non-constant number $k$ of agents, while claiming the existence of an FPTAS when $k$ is constant. Furthermore, in [36] Nguyen and Rothe show an FPTAS running in $O(n^{4k^2+1}/\varepsilon^{2k^2})$ time for the same problem with fixed $k$ and general additive valuation functions.

### Our contribution

In this paper, we present two FPTASs for $k$-SSR achieving running times[3] of $O(n^{2k}/\varepsilon^{k-1})$ and $\widetilde{O}(n/\varepsilon^{3k-1})$, and an FPTAS for $k$-PART running in time $O(n^{2k}/\varepsilon^{k-1})$. The latter represents a considerable improvement when compared to the FPTAS of [36], which is the best known bound for $k$-PART, although it concerns a more general setting. Note that the largest-to-smallest ratio of sums is the only objective function considered in the literature that both admits an FPTAS and is of interest to the study of *envy*. Our approach builds upon methods established for SSR [35, 31, 8], while introducing novel strategies to address the multi-subset setting and to comply with the partitioning constraint.

We present our first FPTAS for $k$-SSR in Section 3; to the best of our knowledge, this is the first ever FPTAS for this problem. To this end, we define a restricted version of $k$-SSR, for which we present a pseudo-polynomial time algorithm. Our approach relies on proving that the optimal solution of the restricted problem satisfies certain properties, while carefully handling edge cases involving large singleton sets.

Concerning $k$-PART, the partitioning constraint renders the application of our techniques more involved; we overcome this in Section 4 by identifying a *perfect* restriction parameter that ensures the existence of a *well-behaved* optimal solution. Interestingly, although this does not yield a pseudo-polynomial time algorithm for the respective restricted case, combining the aforementioned property with approximation techniques yields an FPTAS for $k$-PART.

Additionally, building on Bringmann's work [8], we find that applying density arguments to $k$-SSR encounters an obstacle in the dense case: obtaining $k$ subsets with approximately equal sums is unhelpful, since removing their intersection does not guarantee a feasible $k$-SSR solution. We resolve this in Section 5 by restricting our density argument to singletons and subsequently using our FPTAS from Section 3 as a subroutine on reduced instances.

## 2 Preliminaries

### 2.1 Objective functions for Multiway Number Partitioning

There are four objective functions commonly used for Multiway Number Partitioning, each one having applications in different fields.

[3] $\widetilde{O}$ hides polylog$(1/\varepsilon)$ factors.

1. Minimizing the largest sum. This is also known as makespan minimization and is used for the Minimum Finish Time problem in the context of identical machine scheduling [39, 22]. It is also used in problems related to bin packing [29].
2. Maximizing the smallest sum. This objective function has been studied in works related to scheduling [17] and bin packing [29]. It has also been studied in works related to the *maximin share* [21, 28, 10], which is a criterion for fair item allocation. Note that algorithms for objective function (2) can be modified to work for (1) and vice versa [27].
3. Minimizing the difference between the largest and the smallest sum. This objective function is less common, but it has been studied in works related to Multiway Number Partitioning, such as [32, 27]. No FPTAS exists for this objective [4, 42].
4. Minimizing the ratio between the largest and the smallest sum (e.g. [3]). Although this objective function is often overlooked in works regarding Multiway Number Partitioning (e.g. [27, 40]), it has been studied in the field of fair division, in the form of the Minimum Envy-Ratio problem [30, 36], as well as in the context of scheduling [16].

In this paper, we consider (4) as an objective function for $k$-way Number Partitioning. Observe that all four objectives are equivalent when $k = 2$; this does not hold for $k \geq 3$. This is well known for the first three objectives [27], but, for the sake of completeness, we will provide some counterexamples that prove that (4) is not equivalent to any of the other three, for $k \geq 3$.

It is easy to find counterexamples for the first two. Consider input $\{1, 2, 3, 10\}$ for $k = 3$. The partition $(\{1\}, \{2, 3\}, \{10\})$ minimizes the largest sum, but does not have optimal ratio. Similarly, consider $\{5, 5, 5, 10\}$ for $k = 3$. The partition $(\{5\}, \{5\}, \{5, 10\})$ maximizes the smallest sum, but does not have optimal ratio.

Finally, consider input $\{16, 16, 18, 20, 24, 27, 29, 40\}$ for $k = 4$. The following table compares the solution that minimizes the difference with the solution that minimizes the ratio.

**Table 1** Comparison of Multiway Number Partitioning solutions.

| Solution | Sums | Difference | Ratio |
|---|---|---|---|
| $\{40\}, \{16, 16, 18\}, \{24, 27\}, \{29, 20\}$ | {40,50,51,49} | **11** | 1.275 |
| $\{40, 16\}, \{24, 20\}, \{16, 29\}, \{18, 27\}$ | {56,44,45,45} | 12 | **1.27273** |

## 2.2 Notation and problem definitions

Let $[n] = \{1, \ldots, n\}$ for a positive integer $n$. We assume that the input $A = \{a_1, \ldots, a_n\}$ of our problems is sorted[4], i.e. $a_1 \leq \ldots \leq a_n$.

▶ **Definition 1** (Sum of a subset of indices). *Given a multiset $A = \{a_1, \ldots, a_n\}$ of positive integers and a set of indices $S \subseteq [n]$ we define:*

$$\Sigma(S, A) = \sum_{i \in S} a_i$$

[4] If the input is not sorted, an additional $O(n \log n)$ time would be required. This does not affect the time complexity of any of our algorithms, assuming that $\widetilde{O}$ also hides polylog$(n)$ factors.

▶ **Definition 2** (Max and Min of $k$ subsets). *Let $A = \{a_1, \ldots, a_n\}$ be a multiset of positive integers and $S_1, \ldots, S_k$ be $k$ disjoint subsets of $[n]$. We define the maximum and minimum sums obtained by these sets on $A$ as:*

$$M(S_1, \ldots, S_k, A) = \max_{i \in [k]} \Sigma(S_i, A) \quad \text{and} \quad m(S_1, \ldots, S_k, A) = \min_{i \in [k]} \Sigma(S_i, A)$$

▶ **Definition 3** (Largest-to-smallest ratio of $k$ subsets). *Let $A = \{a_1, \ldots, a_n\}$ be a multiset of positive integers and $S_1, \ldots, S_k$ be $k$ disjoint subsets of $[n]$. We define the* largest-to-smallest ratio *of these $k$ subsets on $A$ as:*

$$\mathcal{R}(S_1, \ldots, S_k, A) = \begin{cases} \frac{M(S_1,\ldots,S_k,A)}{m(S_1,\ldots,S_k,A)} & \text{if } m(S_1, \ldots, S_k, A) > 0 \\ +\infty & \text{if } m(S_1, \ldots, S_k, A) = 0 \end{cases}$$

Throughout the paper, we refer to the largest-to-smallest ratio of $k$ subsets simply as *ratio*.

We define the $k$-Subset Sum Ratio ($k$-SSR) problem as follows.

▶ **Definition 4** ($k$-SSR). *Given a multiset $A = \{a_1, \ldots, a_n\}$ of positive integers, find $k$ disjoint subsets $S_1, \ldots, S_k$ of $[n]$, such that $\mathcal{R}(S_1, \ldots, S_k, A)$ is minimized.*

▶ **Observation 5.** *Only the maximum and minimum sums affect the ratio function. If the remaining sets are altered without a sum becoming greater than the maximum or smaller than the minimum, the ratio remains unaffected. Additionally, if the minimum sum increases or the maximum sum decreases (while the other remains unchanged), the ratio decreases.*

Note that multisets are allowed as input, since they are not a trivial case for $k$-SSR (in contrast to regular SSR), unless there is a number with multiplicity $k$ or more. Throughout the paper, when referring to a solution $S = (S_1, \ldots, S_k)$, we will use the simplified notations $\mathcal{R}(S, A), M(S, A), m(S, A)$ to denote ratio, maximum and minimum sums respectively.

We similarly define the $k$-way Number Partitioning Ratio ($k$-PART) problem.

▶ **Definition 6** ($k$-PART). *Given a multiset $A = \{a_1, \ldots, a_n\}$ of positive integers, find $k$ disjoint subsets $S_1, \ldots, S_k$ of $[n]$ with $\bigcup_{i=1}^{k} S_i = [n]$, such that $\mathcal{R}(S_1, \ldots, S_k, A)$ is minimized.*

## 3 An FPTAS for $k$-Subset Sum Ratio

We define a restricted version of $k$-SSR, called $k$-SSR$_{\mathrm{R}}$, in which the largest element[5] of the first set ($S_1$) is forced to be the smallest among the maxima of all $k$ sets and equal to a given number $p$. This definition generalizes the *Semi-Restricted Subset-Sums Ratio* of [31].

▶ **Definition 7** ($k$-SSR$_{\mathrm{R}}$). *Given a sorted multiset $A = \{a_1, \ldots, a_n\}$ of positive integers and an integer $p, 1 \leq p \leq n - k + 1$, find $k$ disjoint subsets $S_1, \ldots, S_k$ of $[n]$ with $\max(S_1) = p$ and $\max(S_i) > p$ for $1 < i \leq k$, such that $\mathcal{R}(S_1, \ldots, S_k, A)$ is minimized.*[6]

Most of this section is dedicated to producing an FPTAS for the restricted version $k$-SSR$_{\mathrm{R}}$, which is subsequently used as a subroutine to obtain an FPTAS for $k$-SSR, by iterating over all possible values of $p$. Similar reductions to a version with restricted largest elements have been used in approximation schemes for Subset Sum Ratio [35, 31, 8]. In our paper, this technique is used to guarantee that each set $S_i$ contains a sufficiently large element, which is critical to ensure that the algorithm is an FPTAS.

[5] Throughout the paper, the term *element* of a set refers to the index $1 \leq i \leq n$ instead of the value $a_i$, unless explicitly stated otherwise.

[6] The case $p > n - k + 1$ would result in the instance having no feasible solution.

## 3.1 Properties of optimal solutions to $k$-SSR$_{\mathrm{R}}$ instances

First, we want to guarantee that there is always an optimal solution to $k$-SSR$_{\mathrm{R}}$ that satisfies a few important properties. Recall that the input is sorted. Define $Q = \sum_{i=1}^{p} a_i$ and $q = \max\{i \mid a_i \leq Q\}$. Since $Q = \Sigma([p], A)$ and $S_1 \subseteq [p]$, the following is immediate by Def. 2.

▶ **Observation 8.** *For all feasible solutions $(S_1, \ldots, S_k)$ to a $k$-SSR$_{\mathrm{R}}$ instance $(A, p)$*

$$m(S_1, \ldots, S_k, A) \leq \Sigma(S_1, A) \leq Q.$$

Observations 5 and 8 are used in the proof of the following theorem to transform solutions without increasing their ratio.

▶ **Theorem 9.** *For any $k$-SSR$_{\mathrm{R}}$ instance $(A, p)$ there exists an optimal solution whose sets satisfy the following:*

1. *For all sets $S_i$ containing only elements $j$ with $a_j \leq Q$ it holds that $\Sigma(S_i, A) < 2Q$.*
2. *Every set containing an element $j$ with $a_j > Q$ is a singleton.*
3. *The union of singleton sets $\{j\}$ s.t. $a_j > Q$ is $\bigcup_{i=1}^{x}\{q+i\}$, where $x \geq 0$ is the number of these singleton sets.*

**Proof.** Let $S = (S_1, \ldots, S_k)$ be an arbitrary feasible solution. If it violates property 1, i.e. there exists a set $S_i$ in $S$ that only contains elements $j$ with $a_j \leq Q$ and has sum $\Sigma(S_i, A) \geq 2Q$, we transform it as follows. For all $j \in S_i$, we have

$$m(S, A) \leq \Sigma(S_1, A) \leq Q \leq \Sigma(S_i \setminus \{j\}, A) < \Sigma(S_i, A) \leq M(S, A).$$

Thus, if we remove an element $j$ from $S_i$, the ratio cannot increase. We remove the smallest element from $S_i$. As long as the solution still has a set $S_i$ violating property 1, we can apply the same process repeatedly, until there are no more such sets. Since we are only removing elements, no new sets violating property 1 may appear during this process. Note that the largest element of every set remains intact, therefore the $k$-SSR$_{\mathrm{R}}$ restrictions $max(S_1) = p$ and $\max(S_i) > p$ for $1 < i \leq k$ are satisfied.

If the new solution $S'$ violates property 2, i.e. it contains a set $S_i$ with an element $j$ s.t. $a_j > Q$ and $|S_i| > 1$, we apply the following transformation. It holds that

$$m(S', A) \leq \Sigma(S_1, A) \leq Q < \Sigma(\{j\}, A) < \Sigma(S_i, A) \leq M(S', A).$$

As such, replacing set $S_i$ with set $\{j\}$ will result in equal or smaller ratio. We repeat this for every such set $S_i$, thus yielding a solution in which every such set is a singleton. Note that the derived solution is a feasible $k$-SSR$_{\mathrm{R}}$ solution and it still satisfies property 1, since we did not modify sets that contain only elements $j$ with $a_j \leq Q$.

Finally, if the new solution $S''$ violates property 3, i.e. it contains a singleton set $\{u\}$ s.t. $a_u > Q$ and there is an unselected[7] element $v$ such that $q < v < u$, we do the following. It holds that

$$m(S'', A) \leq \Sigma(S_1, A) \leq Q < \Sigma(\{v\}, A) \leq \Sigma(\{u\}, A) \leq M(S'', A).$$

Thus, selecting $v$ instead of $u$ does not increase the ratio. Repeating this for all $u$ and $v$ satisfying the above mentioned property forces the union of these singleton sets to contain the smallest indices possible, yielding a feasible solution that satisfies all three properties.

In conclusion, for any feasible solution we can find another one that satisfies all properties 1, 2, 3 and has equal or smaller ratio. Thus, applying this to an arbitrary optimal solution proves the theorem. ◀

[7] An element that is not in any set $S_i$ of the solution.

## 3.2 A pseudo-polynomial time algorithm for $k$-SSR$_{\mathrm{R}}$

In this subsection we present an exact algorithm for $k$-SSR$_{\mathrm{R}}$, which runs in $O(nQ^{k-1})$ time and returns an optimal solution satisfying the properties of Theorem 9. We present the algorithm in two parts for simplicity. Algorithm 1 picks certain singleton sets with large elements and calls Algorithm 2 to obtain candidate partial solutions for the remaining sets. We next describe both algorithms in detail; we refer the reader to the full version of this paper [24] for the respective pseudocode.

**Algorithm 1** `Exact_`$k$-SSR$_{\mathrm{R}}(A, p)$.

**Input:** A sorted multiset $A = \{a_1, \ldots, a_n\}, a_i \in \mathbb{Z}^+$ and an integer $p, 1 \leq p \leq n - k + 1$.
**Output:** Disjoint subsets $(S_1, \ldots, S_k)$ of $[n]$ with $\max(S_1) = p$, $\max(S_i) > p$ for $1 < i \leq k$, minimizing $\mathcal{R}(S_1, \ldots, S_k, A)$.

Algorithm 1 iterates over all possible values for the number $x$ of singleton sets $\{j\}$, with $a_j > Q$, specifically $x \in \{0, \ldots, \min\{k-1, n-q\}\}$. To construct these $x$ sets, it picks the smallest elements of $A$ that are greater than $Q$, in accordance to property 3 of Theorem 9. For each $x$, Algorithm 1 then calls Algorithm 2 as a subroutine; the latter uses dynamic programming (DP) to output partial solutions for $A' = \{a_i \in A \mid a_i \leq Q\}$ and $k' = k - x$. Note that Algorithm 2 is only called in cases where $p \leq |A'| - k' + 1 = q - k + x + 1$; otherwise, there is no feasible $k$-SSR$_{\mathrm{R}}$ solution for that value of $x$ (see Definition 7).

Algorithm 1 appends the $x$ precalculated singletons to each partial solution returned by Algorithm 2 and compares the ratio of each solution in order to find the best one. The $k$-SSR$_{\mathrm{R}}$ solution returned by Algorithm 1 is optimal, as will be shown in Theorem 12.

**Algorithm 2** `DP_`$k$-SSR$_{\mathrm{R}}(A, k, p)$.

**Input:** A sorted multiset $A = \{a_1, \ldots, a_q\}, a_i \in \mathbb{Z}^+$, an integer $k \geq 1$ and an integer $p, 1 \leq p \leq q - k + 1$, with the restriction $a_q \leq \sum_{i=1}^{p} a_i$.
**Output:** A set *solutions* containing tuples of $k$ disjoint subsets $(S_1, \ldots, S_k)$ of $[q]$, with $\max(S_1) = p$, $\max(S_i) > p$ for $1 < i \leq k$.

Algorithm 2 draws from techniques of [31] and [35], while employing a more involved DP formulation to address the multi-subset setting. A DP table $T$ is used to systematically construct partial solution sets, while ensuring that the constraints of $k$-SSR$_{\mathrm{R}}$ are satisfied.

Each cell of the DP table stores a tuple $(S_1, \ldots, S_k, sum_1)$, where $S_j$ $(1 \leq j \leq k)$ are the sets associated with said cell and $sum_1 = \Sigma(S_1, A)$. The coordinates of a cell $T_i[D][V]$ consist of the following components:

1. An index $i$, indicating the number of elements examined so far.
2. A *difference vector* $D = [d_2, d_3, \ldots, d_k]$ (sorted in nondecreasing order) that encodes the differences between the sum of $S_1$ and the sums of the other sets, i.e. $d_j = \Sigma(S_1, A) - \Sigma(S_j, A)$.
3. A Boolean *validity vector* $V = [v_2, v_3, \ldots, v_k]$, where each $v_j$ is `true` iff $\max(S_j) > p$.

The differences are used as coordinates in the DP table instead of the sums of the sets, in order to decrease the complexity of the algorithm by an order of $Q$ (see Lemma 13). We also introduce $V$ as a coordinate, which is crucial to ensure optimality without increasing the complexity of the algorithm; essentially, the ratio of two tuples can be compared only if all of their validity Booleans are equal (see Lemma 10).

We now describe the DP process in a bottom-up manner. All cells $T_i[D][V]$ are initialized to empty tuples, except the cell $T_0[a_p, \ldots, a_p][\texttt{false}, \ldots, \texttt{false}]$, which is initialized to $(\{p\}, \emptyset, \ldots, \emptyset, a_p)$, thus forcing $p$ to be contained in $S_1$. Every element $i \in [q]$ is processed in increasing order. Assuming row $T_{i-1}$ contains tuples constructed by using elements up to $i-1$ (along with $p$, which is always in $S_1$), the next row $T_i$ is filled as follows. For each non-empty tuple $C = (S_1, \ldots, S_k, sum_1)$ contained in some cell $T_{i-1}[D][V]$, $D = [d_2, \ldots, d_k]$, $V = [v_2, \ldots, v_k]$, we sequentially consider the cases below:

- Element $i$ is not added to any set of $C$. In this case, $C$ is copied into $T_i[D][V]$.
- Element $i$ is added to $S_1$ (only if $i < p$). This produces a new tuple $C' = (S_1 \cup \{i\}, \ldots, S_k, sum_1 + a_i)$ to be inserted into $T_i[D'][V]$, where $D' = [d'_2, \ldots, d'_k]$ is the appropriately updated difference vector, i.e. $d'_j = d_j + a_i$, $1 < j \leq k$.
- Element $i \neq p$ is added to each set $S_j$ $(1 < j \leq k)$, one at a time. This produces a new tuple $(S_1, \ldots, S_j \cup \{i\}, \ldots, S_k, sum_1)$, a difference vector $[d'_2, \ldots, d'_k]$ and a validity vector $[v'_2, \ldots, v'_k]$. Specifically, $v'_j$ is set to `true` if $i > p$ and $d'_j$ is set to $d_j - a_i$, while $v'_u = v_u$ and $d'_u = d_u$ for all $u \neq j$. This difference vector is then sorted in nondecreasing order, while the sets and the validity vector are rearranged accordingly, thus producing a tuple $C'$ and vectors $D', V'$. The tuple $C'$ is then inserted into $T_i[D'][V']$.

Algorithm 2 prevents the new tuple from being inserted into a cell in the following cases:

1. The cell already contains another tuple with equal or larger $sum_1$ (see Lemma 10).
2. The updated vector $D'$ contains a difference smaller than or equal to $-2Q$. Such a tuple can be ignored, due to property 1 of Theorem 9.

In the end, Algorithm 2 returns all non-empty tuples contained in cells $T_q[D][\texttt{true}, \ldots, \texttt{true}]$ (for all $D$), thus ensuring that every partial solution returned to Algorithm 1 complies with the $k$-SSR$_\mathrm{R}$ restrictions. This concludes the description of Algorithm 2.

We say that a *conflict* occurs when a tuple is to be stored in a cell $T_i[D][V]$ which is already occupied by another (non-empty) tuple. Note that conflicting tuples must have the same difference and validity vectors and only use elements up to $i$. Since the DP process must minimize the overall ratio, including potential singletons not participating in said process, conflict resolution becomes more involved for $k > 2$.

▶ **Lemma 10** (Conflict resolution). *Let $C_1 = (S_1, \ldots, S_k, sum_1)$ and $C_2 = (S'_1, \ldots, S'_k, sum'_1)$ be two tuples that result in a conflict in a cell $T_i[D][V]$ of the DP table of Algorithm 2, such that $sum_1 < sum'_1$. No optimal $k$-SSR$_\mathrm{R}$ solution may use*[8] *$C_1$.*

**Proof.** Let $S$ be a $k$-SSR$_\mathrm{R}$ solution. Split $S$ in two: $S_{small}$, containing the $k'$ sets returned by Algorithm 2, and $S_{large}$, containing the singletons $\{j\}$ s.t. $a_j > Q$. Assume $S_{small}$ uses $C_1$.

By assumption, $C_1$ and $C_2$ have identical $D$ and $V$ vectors and their sets involve only elements from $[i]$. This implies that any combination of elements larger than $i$ that can later be added to sets of $C_1$ to construct a partial[9] solution with vectors $D', V'$ can also be added to the corresponding sets of $C_2$ to construct a partial solution with the same vectors $D', V'$. Let $S'_{small}$ be the partial solution obtained by using $C_2$ instead of $C_1$ and adding the same combination of elements that would have been used to obtain $S_{small}$ from $C_1$. Consider the solution $S'$ that is obtained by appending $S_{large}$ to $S'_{small}$. Note that $S_{small}$ and $S'_{small}$ have the same validity vector, hence $S'$ is also a feasible $k$-SSR$_\mathrm{R}$ solution.

---

[8] In our dynamic programming framework, we say that a solution $S$ *uses* a tuple $C = (S_1, \ldots, S_k, sum_1)$ if $C$ appears in an intermediate step of the construction of $S$.

[9] We denote as *partial* a solution for which $S_{large}$ will be appended to obtain a feasible $k$-SSR$_\mathrm{R}$ solution.

Let $M_s, m_s$ be the maximum and minimum sum of $S_{small}$ respectively and $M'_s, m'_s$ those of $S'_{small}$. Since $sum_1 < sum'_1$ and the difference vector $D$ is the same for both solutions, it follows that $M_s < M'_s$, $m_s < m'_s$ and $M_s - m_s = M'_s - m'_s$. From these, we obtain

$$\frac{M_s}{m_s} > \frac{M'_s}{m'_s}. \tag{1}$$

Let $\{t\} \in S_{large}$ be the singleton set with the largest element. For all $\{i\} \in S_{large}$ such that $i \neq t$, we have

$$m_s \leq \Sigma(S_1, A) \leq Q < \Sigma(\{i\}, A) \leq \Sigma(\{t\}, A).$$

This implies that $\{i\}$ $(i \neq t)$ does not affect the ratio and $\{t\}$ only affects the ratio if $a_t > M_s$. Thus, all the sets in $S_{large}$ can be ignored when calculating $\mathcal{R}(S, A)$, except for $\{t\}$ if $a_t > M_s$. The same holds for $\mathcal{R}(S', A)$ and $M'_s$. Consider three cases:

1. $M_s < M'_s < a_t$. In this case, $\mathcal{R}(S, A) = a_t/m_s$ and $\mathcal{R}(S', A) = a_t/m'_s$. Since $m_s < m'_s$: $\mathcal{R}(S, A) > \mathcal{R}(S', A)$.
2. $M_s < a_t \leq M'_s$. In this case, $\mathcal{R}(S, A) = a_t/m_s > M_s/m_s$ and $\mathcal{R}(S', A) = M'_s/m'_s$. By inequality (1): $\mathcal{R}(S, A) > \mathcal{R}(S', A)$.
3. $a_t \leq M_s < M'_s$. In this case, $\mathcal{R}(S, A) = M_s/m_s$ and $\mathcal{R}(S', A) = M'_s/m'_s$. By inequality (1): $\mathcal{R}(S, A) > \mathcal{R}(S', A)$.

In all cases $\mathcal{R}(S, A) > \mathcal{R}(S', A)$, therefore $S$ cannot be optimal. ◀

▶ **Lemma 11** (Feasibility). *Every $k$-tuple of sets considered by Algorithm 1 is a feasible $k$-SSR$_{\mathrm{R}}$ solution.*

**Proof.** The $x$ singletons constructed by Algorithm 1 are disjoint and contain elements larger than $q$. In Algorithm 2, every element $i \leq q$ is processed in increasing order and added to at most one set at a time. As such, all sets are disjoint. Note that $p$ is contained in $S_1$ and $S_1$ cannot receive a larger element. Recall that $v_i$ is set to `true` when $S_i$ receives an element $j > p$ and Algorithm 2 only returns solutions with $V = [\texttt{true}, \ldots, \texttt{true}]$. Thus, all $k$-SSR$_{\mathrm{R}}$ restrictions regarding the largest element of each set are satisfied. ◀

The proof of the following theorem uses Lemmas 10 and 11. The main idea is to show that Algorithm 1 considers and therefore finds the optimal solution whose existence is guaranteed by Theorem 9.

▶ **Theorem 12** (Optimality). *Algorithm 1 returns an optimal solution for $k$-SSR$_{\mathrm{R}}$.*

**Proof.** Lemma 11 guarantees that Algorithm 1 returns a feasible $k$-SSR$_{\mathrm{R}}$ solution, so we only need to prove that its ratio is optimal. Let $S^*$ be the optimal $k$-SSR$_{\mathrm{R}}$ solution guaranteed by Theorem 9. We split $S^*$ in two: $S^*_{large}$, which contains the singleton sets $\{j\}$ s.t. $a_j > Q$, and $S^*_{small}$, which contains the rest of the sets. Note that the number $x$ of these singletons is bounded by $k-1$ or by the amount of sufficiently large elements, i.e. $n-q$. Thus, Algorithm 1 considers every $S_{large}$ that satisfies properties 2 and 3 of Theorem 9, including $S^*_{large}$.

Recall that Algorithm 1 uses Algorithm 2 as a subroutine in order to obtain candidate partial solutions, to which $S^*_{large}$ will be appended. We would like $S^*_{small}$ to be contained in the solutions returned by Algorithm 2.

Algorithm 2 prunes a solution if a difference $d_j$ would become lower than or equal to $-2Q$. Any $S_{small}$ satisfying property 1 of Theorem 9 will not be affected by this. This includes $S^*_{small}$.

When there is a conflict in a cell $T_i[D][V]$ between two tuples with different $sum_1$ values, $S^*_{small}$ cannot use the one with smaller $sum_1$ (by Lemma 10), since $S^*$ is a optimal. When there is a conflict in a cell $T_i[D][V]$ between two tuples $C_1, C_2$ with equal $sum_1$ values, observe that both tuples have exactly the same sums (but different sets), use only elements up to $i$ and have the same validity vector. This implies that any combination of elements greater than $i$ that can later be added to sets of $C_1$ can also be added to the same sets of $C_2$ and vice versa. Thus, any combination of sums that can be reached through $C_1$ can also be reached through $C_2$ and vice versa. This means that if one of these tuples leads to $S^*_{small}$, the other one leads to another partial solution $S^{**}_{small}$, which has exactly the same sums as $S^*_{small}$. Appending $S^*_{large}$ to $S^{**}_{small}$ yields a solution with the same sums as $S^*$, i.e. another optimal solution.

It follows directly from the description of Algorithm 2 that it constructs every possible combination of disjoint sets $S_1, \ldots, S_{k-x}$ with $\max(S_1) = p$, apart from the ones pruned as explained in the previous paragraphs.

Therefore, the partial solutions returned by Algorithm 2 contain $S^*_{small}$ or another partial solution with the same sums, which also leads to an optimal solution when appended with $S^*_{large}$. In either case, Algorithm 1 will find an optimal solution when iterating to find the best ratio. ◀

▶ **Lemma 13** (Complexity). *Algorithm* 1 *runs in time* $O(nQ^{k-1})$.

**Proof.** Algorithm 1 calls Algorithm 2 once for each value of $x$ in $\{0, \ldots, \min\{k-1, n-q\}\}$, where $x$ is the number of singleton sets $\{j\}$ s.t. $a_j > Q$. For a fixed $x$, Algorithm 2 is applied to an instance with $k' = k - x$ sets and $q = O(n)$ elements.

Due to the pruning done in Algorithm 2, it holds that $\forall j \in \{2, \ldots, k'\}$: $-2Q < d_j \leq Q$, where the second inequality holds because $\Sigma(S_1, A) \leq Q$. Since the algorithm stores $D$ in non-decreasing order (i.e., $d_2 \leq d_3 \leq \cdots \leq d_{k'}$), there are $O((3Q)^{k'-1}/(k'-1)!)$ distinct difference vectors. Taking into account the validity vector $V$ and all $T_i$'s, we obtain the following bound for the amount of DP cells: $O(n(6Q)^{k'-1}/(k'-1)!)$.

For constant $k$, this bound becomes $O(nQ^{k'-1})$. Note that all operations of Algorithm 2 on DP cells (such as sorting the difference vector after adding an element) run in time $O(k')$, so the time complexity of Algorithm 2 is $O(nQ^{k'-1}) = O(nQ^{k-x-1})$. Hence, the time complexity of Algorithm 1 is bounded by:

$$\sum_{x=0}^{k-1} nQ^{k-x-1} = O(nQ^{k-1})$$ ◀

## 3.3 FPTASs for $k$-$\mathrm{SSR}_\mathrm{R}$ and $k$-SSR

By Theorem 12 and Lemma 13, Algorithm 1 solves $k$-$\mathrm{SSR}_\mathrm{R}$ in pseudo-polynomial time. To obtain an FPTAS for $k$-$\mathrm{SSR}_\mathrm{R}$, we scale (and round down) the input set by a factor $\delta = \frac{\varepsilon \cdot a_p}{3 \cdot n}$ (cf. [31, 35]).

This leads to Algorithm 3, which calls Algorithm 1 in order to find an optimal solution for the rounded input $A_r$, yielding a $(1+\varepsilon)$-approximation of the (largest-to-smallest) ratio of an optimal solution $(S^*_1, \ldots, S^*_k)$ of the original $k$-$\mathrm{SSR}_\mathrm{R}$ instance.

**Algorithm 3** FPTAS_$k$-SSR$_{\mathrm{R}}(A, p, \varepsilon)$.

**Input:** A sorted multiset $A = \{a_1, \ldots, a_n\}, a_i \in \mathbb{Z}^+$, an integer $p$ such that $1 \leq p \leq n-k+1$ and an error parameter $\varepsilon \in (0, 1)$.

**Output:** Disjoint subsets $(S_1, \ldots, S_k)$ of $[n]$ with $\max(S_1) = p$, $\max(S_i) > p$ for $1 < i \leq k$ and $\mathcal{R}(S_1, \ldots, S_k, A) \leq (1+\varepsilon) \cdot \mathcal{R}(S_1^*, \ldots, S_k^*, A)$.

1: $\delta \leftarrow \frac{\varepsilon \cdot a_p}{3 \cdot n}$, $A_r \leftarrow \emptyset$
2: **for** $i \leftarrow 1, \ldots, n$ **do** $a_i^r \leftarrow \lfloor \frac{a_i}{\delta} \rfloor$, $A_r \leftarrow A_r \cup \{a_i^r\}$
3: $(S_1, \ldots, S_k) \leftarrow$ Exact_$k$-SSR$_{\mathrm{R}}(A_r, p)$ ▷ Call Alg. 1 for rounded instance
4: **return** $S_1, \ldots S_k$

▶ **Theorem 14** ($k$-SSR$_{\mathrm{R}}$ approximation). *Let $(S_1, \ldots, S_k)$ be the sets returned by Algorithm 3 for a $k$-SSR$_{\mathrm{R}}$ instance $(A = \{a_1, \ldots, a_n\}, p)$ with error parameter $\varepsilon$. Let $(S_1^*, \ldots, S_k^*)$ be an optimal solution for the same $k$-SSR$_{\mathrm{R}}$ instance. Then:*

$$\mathcal{R}(S_1, \ldots, S_k, A) \leq (1+\varepsilon) \cdot \mathcal{R}(S_1^*, \ldots, S_k^*, A)$$

The proof of Theorem 14 closely follows the proofs presented in section 5 of [31]. We include the proof in the full version of this paper [24].

By Lemma 13, Algorithm 3 solves $k$-SSR$_{\mathrm{R}}$ in $O(nQ^{k-1})$ (where $Q = \sum_{i=1}^{p} a_i^r$). Since we scaled the input by $\delta$, the values of Q are bounded as follows:

$$Q^{k-1} = \left(\sum_{i=1}^{p} a_i^r\right)^{k-1} \leq \left(n \cdot a_p^r\right)^{k-1} \leq \left(\frac{n \cdot a_p}{\delta}\right)^{k-1} = \left(\frac{3 \cdot n^2}{\varepsilon}\right)^{k-1} = \frac{3^{k-1} \cdot n^{2k-2}}{\varepsilon^{k-1}}$$

Therefore, Algorithm 3 runs in time $O(n^{2k-1}/\varepsilon^{k-1})$. To obtain an FPTAS for the (unrestricted) $k$-SSR problem, we run Algorithm 3 once for each possible value of $p$ ($1 \leq p \leq n-k+1$) and pick the solution with the best ratio. Thus, we obtain the main theorem of this section.

▶ **Theorem 15.** *There is an FPTAS for $k$-SSR that runs in $O(n^{2k}/\varepsilon^{k-1})$ time.*

## 4 An FPTAS for k-way Number Partitioning Ratio

In this section we present an FPTAS for $k$-way Number Partitioning Ratio ($k$-PART) by extending and refining the techniques of Section 3. Recall that in the case of $k$-PART, every element needs to be assigned to a set. We define the following restricted version of $k$-PART, which is analogous to $k$-SSR$_{\mathrm{R}}$.

▶ **Definition 16** ($k$-PART$_{\mathrm{R}}$). *Given a sorted multiset $A = \{a_1, \ldots, a_n\}$ of positive integers and an integer $p, 1 \leq p \leq n-k+1$, find $k$ disjoint subsets $S_1, \ldots, S_k$ of $[n]$ with $\bigcup_{i=1}^{k} S_i = [n]$, $\max(S_1) = p$ and $\max(S_i) > p$ for $1 < i \leq k$, such that $\mathcal{R}(S_1, \ldots, S_k, A)$ is minimized.*

The main challenge in expanding our technique to $k$-PART stems from the fact that there exist instances $(A, p)$ of $k$-PART$_{\mathrm{R}}$ where all optimal solutions violate one or more of the properties of Theorem 9. This is a problem, since the time complexity of our $k$-SSR$_{\mathrm{R}}$ algorithm relies heavily on Theorem 9.

We overcome this by showing that there exists a $p^*$ such that the corresponding optimal solution $S^*$ to $(A, p^*)$ is well-behaved (see Theorem 18). It suffices to guarantee that for $p^*$, the algorithm will consider $S^*$ (or another equivalent solution) as a candidate solution for the rounded $k$-PART$_{\mathrm{R}}$ instance. Note that since $S^*$ is not guaranteed to be optimal for the rounded $k$-PART$_{\mathrm{R}}$ instance, we obtain *neither an exact algorithm nor an FPTAS* for the restricted problem $k$-PART$_{\mathrm{R}}$.

## 4.1 Properties of $k$-PART$_{\mathrm{R}}$ instances

▶ **Definition 17** (Perfect $p$)**.** *Let $A = \{a_1, \ldots, a_n\}$ be the input to the $k$-PART problem. We call a number $p : 1 \leq p \leq n - k + 1$* perfect *for $A$ if the optimal solution(s) for the $k$-PART$_{\mathrm{R}}$ instance $(A, p)$ have the same (largest-to-smallest) ratio as the optimal solution(s) for the $k$-PART instance $A$.*

We define $Q = \sum_{i=1}^{p} a_i$ and $q = \max\{i \mid a_i \leq Q\}$, in the same manner as we did for $k$-SSR$_{\mathrm{R}}$. The next theorem is analogous to Theorem 9, with a key difference: its properties are only guaranteed for some *perfect* $p$. Its proof uses arguments similar in spirit to the proof of Theorem 9. Special attention is required since elements that violate a property are reassigned to other sets rather than being removed entirely from the solution. This may cause some $k$-PART$_{\mathrm{R}}$ restriction to be violated, thus rendering the proof significantly more involved.

▶ **Theorem 18.** *Given a $k$-PART instance $A$, there exists a perfect $p$ for $A$, for which there is an optimal solution for the $k$-PART$_{\mathrm{R}}$ instance $(A, p)$ satisfying the following properties:*
1. *For all sets $S_i$ containing only elements $j \leq q$ it holds that $\Sigma(S_i, A) < 2Q$.*
2. *All elements $j > q$ are contained in singleton sets.*

**Proof.** For some arbitrary value of $p$, let $S = (S_1, \ldots, S_k)$ be an arbitrary feasible solution for the $k$-PART$_{\mathrm{R}}$ instance $(A, p)$. If $S$ breaks property 2, i.e. there exists a set $S_i$ in $S$ with $|S_i| > 1$ containing an element $j > q$, we do the following. Let $S_m$ be a set with minimum sum[10] and $u$ the smallest element of $S_i$. The following inequalities hold.

$$m(S, A) = \Sigma(S_m, A) \leq Q < \Sigma(S_i \setminus \{u\}, A)$$

$$\Sigma(S_m \cup \{u\}, A) \leq Q + a_u < \Sigma(S_i, A) \leq M(S, A)$$

From these we can infer that, by moving $u$ from $S_i$ to $S_m$, the minimum sum cannot decrease and the maximum cannot increase. Thus, we move $u$ as described, yielding a solution $S'$ with $\mathcal{R}(S', A) \leq \mathcal{R}(S, A)$. However, it might be the case that $S_m$ is $S_1$ and adding $u$ to it breaks the restrictions of $k$-PART$_{\mathrm{R}}$, so $S'$ might not be a feasible solution for $(A, p)$. For the new solution $S'$, define $p'$ as the minimum among the maxima of its $k$ sets and call $S_1$ the set that contains $p'$. Note that $S'$ is a feasible solution for the $k$-PART$_{\mathrm{R}}$ instance $(A, p')$.

If $S'$ still breaks property 2 for $Q' = \sum_{i=1}^{p'} a_i$ and $q' = \max\{i \mid a_i \leq Q'\}$, apply the same process. By doing this repeatedly, $p$ cannot decrease and the ratio of the solution cannot increase. Note that $p$ cannot exceed $n - k + 1$ with this process, so at some point $p$ will stop increasing. Let $p_f$ be this final value and define $Q_f, q_f$ accordingly for $p_f$. Repeating the process will at some point yield a solution that satisfies property 2 for the $k$-PART$_{\mathrm{R}}$ instance $(A, p_f)$.

Let $S''$ be the feasible solution for the $k$-PART$_{\mathrm{R}}$ instance $(A, p_f)$, derived from the process described in the previous paragraphs. $S''$ satisfies property 2 for values $Q_f, q_f$. If $S''$ breaks property 1, i.e. it contains a set $S_i$ consisting only of elements $j \leq q_f$ and having sum $\Sigma(S_i, A) \geq 2Q_f$, we apply the following transformation. Let $S_m$ be a set with minimum sum. Take the smallest element $j$ from $S_i$ and move it to $S_m$. Observe that $a_j \leq Q_f$. Hence, the following inequalities hold.

$$m(S'', A) = \Sigma(S_m, A) \leq Q_f \leq \Sigma(S_i \setminus \{j\}, A)$$

$$\Sigma(S_m \cup \{j\}, A) \leq 2Q_f \leq \Sigma(S_i, A) \leq M(S'', A)$$

[10] There could be multiple sets with the same (minimum) sum.

From these we can infer that, by moving $j$ from $S_i$ to $S_m$, the minimum sum cannot decrease and the maximum cannot increase. We repeatedly apply this process until we end up with a solution satisfying property 1, just like we did in the previous paragraphs for property 2, since the same arguments hold regarding the increment of $p$ and the ratio not increasing. Furthermore, this transformation preserves property 2, as $p$ cannot decrease through this process and we only add elements to the set with the smallest sum (which cannot be a set containing an element $j > q$).

In conclusion, for any feasible solution for a $k$-PART$_{\mathrm{R}}$ instance $(A, p)$ that violates some property, we can find a feasible solution for another $k$-PART$_{\mathrm{R}}$ instance $(A, p_{new})$ that satisfies both properties and has equal or smaller ratio. Suppose we apply this to an optimal solution for a $k$-PART$_{\mathrm{R}}$ instance $(A, p)$, with $p$ being *perfect*[11] for the $k$-PART instance $A$. Since the ratio of the solution does not increase throughout the constructions described in this proof, any new $p'$ obtained by the constructions must also be *perfect* and the solution obtained must be optimal for $(A, p')$. ◀

Let $p^*$ be one of the *perfect* elements guaranteed to exist by Theorem 18 for $A$ and $S^*$ be a respective optimal solution for $(A, p^*)$ satisfying properties 1 and 2. The following lemmas indicate that we can prune certain values of $p$, without missing $S^*$. We define $x = n - q$. Intuitively, for $(A, p^*)$, $x$ is the number of large elements that must be contained in singleton sets in $S^*$, according to Theorem 18.

▶ **Lemma 19.** *If for a $k$-PART$_{\mathrm{R}}$ instance $(A, p)$ we have $x > k - 1$, then $p \neq p^*$.*

**Proof.** Since $S_1$ cannot contain elements $j > q$, there are $k - 1$ sets that can contain such elements. The number of these elements is $x$. If $x > k - 1$, by the pigeonhole principle, every feasible solution contains a set with two or more of these elements. This contradicts property 2 of Theorem 18, therefore $p \neq p^*$. ◀

▶ **Lemma 20.** *If for a $k$-PART$_{\mathrm{R}}$ instance $(A, p)$ we have $x = k - 1$ and $p < q$, then $p \neq p^*$.*

**Proof.** Because $p < q$, $S_1$ cannot contain $q$. Thus, there are at least $x + 1 = k$ elements that cannot be contained in $S_1$, with $k - 1$ of them being greater than $q$. By the pigeonhole principle, every feasible solution contains a set with two or more elements, one of which is greater than $q$. This contradicts property 2 of Theorem 18, therefore $p \neq p^*$. ◀

## 4.2 Obtaining an FPTAS for *k*-PART

We now present the main algorithm for $k$-PART. Its design is analogous to that of Algorithms 1 and 3, with two important differences:

1. There is no iteration for different amounts of singletons with elements $j > q$. Instead, we prune some values of $p$ according to Lemmas 19 and 20 and fix $x$ singletons for the rest.
2. In contrast to $k$-SSR$_{\mathrm{R}}$, we do not necessarily obtain the optimal solution to each rounded $k$-PART$_{\mathrm{R}}$ instance $(A_r, p)$. Interestingly, we will later prove that the obtained solution is sufficient for the algorithm to be an FPTAS for $k$-PART.

[11] Such a $p$ always exists, by definition.

**Algorithm 4** FPTAS_$k$-PART$(A, \varepsilon)$.

**Input:** A sorted multiset $A = \{a_1, \ldots, a_n\}, a_i \in \mathbb{Z}^+$ and an error parameter $\varepsilon \in (0, 1)$.
**Output:** Disjoint subsets $(S_1, \ldots, S_k)$ of $[n]$, such that $\bigcup_{i=1}^{k} S_i = [n]$ and their ratio satisfies $\mathcal{R}(S_1, \ldots, S_k, A) \leq (1+\varepsilon)\mathcal{R}(S_1^*, \ldots, S_k^*, A)$.

```
for p ← 1, ..., n − k + 1 do
    best_ratio[p] ← ∞, best_solution[p] ← (∅, ..., ∅)
for p ← 1, ..., n − k + 1 do
    Q ← Σ_{i=1}^{p} a_i, q ← max{i | a_i ≤ Q}, x ← n − q
    if x > k − 1 ∨ (x = k − 1 ∧ p < q) then                ▷ Lemmas 19 and 20
        continue to next p
    for y ← 1, ..., x do S_{k−x+y} ← {q + y}               ▷ x singletons
    δ ← ε·a_p / (3·n), A_r ← ∅, k' ← k − x
    for i ← 1, ..., n do a_i^r ← ⌊a_i / δ⌋, A_r ← A_r ∪ {a_i^r}
    A'_r ← {a_1^r, ..., a_q^r}                             ▷ q items for DP
    DP_solutions ← DP_k-PART_R(A'_r, k', p, Q/δ)
    for all (S_1, ..., S_{k'}) in DP_solutions do
        current_ratio ← R(S_1, ..., S_k, A_r)              ▷ k-PART_R solution
        if current_ratio < best_ratio[p] then
            best_solution[p] ← (S_1, ..., S_k)             ▷ Best solution for each A_r
            best_ratio[p] ← current_ratio                  ▷ and its ratio
final_ratio ← ∞, final_solution ← 0
for p ← 1, ..., n − k + 1 do                               ▷ Iterate for all p to find best sol. for A
    current_ratio ← R(best_solution[p], A)
    if current_ratio < final_ratio then
        final_solution ← best_solution[p]
        final_ratio ← current_ratio
return final_solution
```

Algorithm 4 calls a dynamic programming subroutine for each value of $p$, in order to find candidate partial solutions for elements $j \leq q$. This DP subroutine, DP_$k$-PART$_{\mathrm{R}}(A'_r, k', p, Q)$, is a direct extension of Algorithm 2, with the following three differences.

1. The case of an element not being added to any set is skipped.
2. When a conflict occurs in a DP cell $T_i[D][V]$, we do not use $sum_1$ to resolve it. For $k$-PART$_{\mathrm{R}}$ every element is included in some set, thus conflicts can only occur between tuples with identical sums. Hence, it does not matter which tuple is preferred (as proven in Theorem 12 for conflicts with equal sums) and there is no need to ever consider $sum_1$.
3. The bound for pruning large negative differences is chosen as $-2Q/\delta$, where $Q = \sum_{i=1}^{p} a_i$ is calculated using the *initial* values $a_i$ instead of the scaled and rounded values $a_i^r$. This is necessary to avoid pruning edge case solutions. We will expand on this in Lemma 22.

The respective pseudocode is included in the full version of this work [24]. We now present the following lemma, whose proof is essentially identical to that of Lemma 11.

▶ **Lemma 21** (Feasibility). *Every $k$-tuple of sets whose $\mathcal{R}(S_1, \ldots, S_k, A_r)$ value is considered by Algorithm 4 is a feasible solution for the $k$-PART$_{\mathrm{R}}$ instance $(A_r, p)$.*

Recall that we defined $p^*$ and $S^*$ as a *perfect* $p$ and a respective optimal solution for $(A, p^*)$ whose existence is guaranteed by Theorem 18.

▶ **Lemma 22** (Near-optimality). *When Algorithm 4 iterates to compare the ratio of solutions for the $k$-PART$_{\mathrm{R}}$ instance $(A_r, p^*)$, it will consider either $S^*$ or another solution $S = (S_1, \ldots, S_k)$ with $\Sigma(S_i, A_r) = \Sigma(S_i^*, A_r)$, $\forall i \in [k]$.*

**Proof.** According to Lemmas 19 and 20, we cannot have $x > k-1$ or $x = k-1$ and $p < q$ for $p = p^*$. The singleton sets enforced by Algorithm 4 are exactly the singleton sets contained in $S^*$, according to property 2 of Theorem 18.

Recall that a solution in the DP subroutine is pruned if it has some difference $d_j \leq -2Q/\delta$. By property 1 of Theorem 18, it holds that $\Sigma(S_i^*, A) < 2Q$, $\forall i \in [k-x]$. Since $A_r$ contains elements scaled by $\delta$ and *rounded down*, we infer that $\forall i \in [k-x]$

$$\Sigma(S_i^*, A_r) \leq \Sigma(S_i^*, A)/\delta < 2Q/\delta.$$

This implies that for $S^*$, there will be no $d_j \leq -2Q/\delta$ (at any point in its dynamic programming construction).

As already explained, DP conflicts for $k$-PART$_{\mathrm{R}}$ occur only between tuples with identical sums. Recall that two conflicting tuples have the same vectors $D, V$ and only use elements up to some element $i$, so any combination of elements that can be added to sets of one tuple can also be added to the respective sets of the other tuple. This is explained in more detail in the proof of Theorem 12. We infer that it is possible for $S^*$ to be overwritten by another solution $S = (S_1, \ldots, S_k)$ with $\Sigma(S_i, A_r) = \Sigma(S_i^*, A_r)$, $\forall i \in [k]$.

The DP subroutine constructs every possible combination of disjoint sets $S_1, \ldots, S_{k-x}$ with $\max(S_1) = p$ and $\bigcup_{i=1}^{k} S_i = [q]$, apart from the ones pruned by the cases mentioned in the previous paragraph. Thus, the lemma follows. ◀

▶ **Theorem 23.** *Algorithm 4 is an FPTAS for $k$-PART that runs in time $O(n^{2k}/\varepsilon^{k-1})$.*

**Proof.** We rely upon the observation that the proof of Theorem 14 does not necessarily require an optimal solution for the rounded $k$-SSR$_{\mathrm{R}}$ instance $A_r$; it suffices for the algorithm to consider a $k$-SSR$_{\mathrm{R}}$ solution $S$ with $\mathcal{R}(S, A_r) \leq \mathcal{R}(S_{opt}, A_r)$, where $S_{opt}$ is an optimal solution for the $k$-SSR instance $A$ (prior to rounding).

First, consider the case $p \neq p^*$. If the conditions of Lemma 19 or 20 are met, this $p$ will be skipped. Otherwise, a $k$-tuple of sets will be found, which is a feasible solution for the $k$-PART$_{\mathrm{R}}$ instance $(A_r, p)$, according to Lemma 21.

Second, consider the case $p = p^*$. By Lemma 22, Algorithm 4 will consider $S^*$ or another solution $S$ with $\mathcal{R}(S, A_r) = \mathcal{R}(S^*, A_r)$ as a solution for the instance $(A_r, p^*)$. Therefore, for the solution $S_{alg}$ found by the algorithm in this iteration, it holds that $\mathcal{R}(S_{alg}, A_r) \leq \mathcal{R}(S^*, A_r)$. With the aforementioned observation, the proof of the following inequality is identical to the proof of Theorem 14.

$$\mathcal{R}(S_{alg}, A) \leq (1+\varepsilon)\mathcal{R}(S^*, A)$$

Taking into account both cases, the final solution returned by the algorithm is a feasible $k$-PART solution for $A$ and has ratio smaller than or equal to that of the solution found in the $p^*$-th iteration. It follows that Algorithm 4 is a $(1+\varepsilon)$-approximation for $k$-PART.

We now analyze the complexity of the algorithm. The DP subroutine runs in time $O(n(Q/\delta)^{k-1})$, by the same reasoning as in the proof of Lemma 13 for $x = 0$ (which is the worst case). Taking into account the iteration for all possible values of $p$ and using a bound for $Q/\delta$ just like we did for $k$-SSR, we infer that Algorithm 4 runs in $O(n^{2k}/\varepsilon^{k-1})$. ◀

## 5 An FPTAS for $k$-SSR with linear dependence on $n$

In this section, we provide an alternative FPTAS for $k$-SSR, using techniques inspired from [8] to reduce the problem to smaller instances. We then use the $k$-SSR FPTAS of Section 3 to solve them.

We define an alternative restricted version of $k$-SSR, called $k$-SSR$_{\mathrm{L}}$, in which the solution is forced to contain the largest element of the input. This definition is based on the SSR$_{\mathrm{L}}$ problem defined in [8].

▶ **Definition 24** ($k$-SSR$_{\mathrm{L}}$). *Given a sorted multiset $A = \{a_1, \ldots, a_n\}$ of positive integers, find $k$ disjoint subsets $S_1, \ldots, S_k$ of $[n]$ with $n \in \bigcup_{i=1}^{k} S_i$, such that $\mathcal{R}(S_1, \ldots, S_k, A)$ is minimized.*

Let $A = \{a_1, \ldots, a_n\}$ be a sorted multiset of $n$ positive integers. We denote by $A[l, r]$ the subset of $A$ consisting of all items $a_i$, such that $l \leq i \leq r$. We also use the notation $\mathrm{sum}(A) = \sum_{a \in A} a$.

For a multiset $A = \{a_1, \ldots, a_n\}$, we denote the optimal ratio of the $k$-SSR instance $A$ as $\mathrm{OPT}(A)$ and the optimal ratio of the $k$-SSR$_{\mathrm{L}}$ instance $A$ as $\mathrm{OPT}_{\mathrm{L}}(A)$. By definition, it holds that $1 \leq \mathrm{OPT}(A) \leq \mathrm{OPT}_{\mathrm{L}}(A)$.

▶ **Lemma 25.** *Let $A = \{a_1, \ldots, a_n\}$ be a sorted multiset of positive integers. It holds that*

$$\mathrm{OPT}(A) = \min_{k \leq j \leq n} \mathrm{OPT}_{\mathrm{L}}\left(A[1, j]\right).$$

**Proof.** Let $t \in \{k, \ldots, n\}$ be the maximum element[12] contained in some optimal solution $S$ of the $k$-SSR instance $A$. By Definition 24, $S$ is a feasible solution for the $k$-SSR$_{\mathrm{L}}$ instance $A[1, t]$. This implies $\mathrm{OPT}(A) \geq \min_{k \leq j \leq n} \mathrm{OPT}_{\mathrm{L}}\left(A[1, j]\right)$.

Note that any feasible solution for a $k$-SSR$_{\mathrm{L}}$ instance $A[1, j]$ (for any $j \in \{k, \ldots, n\}$) is a feasible solution for the $k$-SSR instance $A$. Thus, it also holds that $\mathrm{OPT}(A) \leq \min_{k \leq j \leq n} \mathrm{OPT}_{\mathrm{L}}\left(A[1, j]\right)$. ◀

Lemma 26 provides the key structural insight of our algorithm. Intuitively, the fact that a $k$-SSR$_{\mathrm{L}}$ solution contains the largest element allows us to consider only the largest $O\big((1/\varepsilon)\ln(1/\varepsilon)\big)$ elements for each $k$-SSR$_{\mathrm{L}}$ instance $A[1, j]$, in order to obtain a $(1+\varepsilon)$-approximation of the optimal $k$-SSR solution.

▶ **Lemma 26** (Largest elements). *For any sorted multiset $A = \{a_1, \ldots, a_n\}$ of positive integers and any $\varepsilon \in (0, 1)$, the following holds:*

$$\mathrm{OPT}(A) \leq \min_{k \leq j \leq n} \mathrm{OPT}_{\mathrm{L}}\left(A\big[j - C + 1, j\big]\right) \leq (1+\varepsilon)\mathrm{OPT}(A), \tag{2}$$

*where $C = (c+1)(k-1)$ and $c = 1 + \left\lceil \left(1 + \frac{1}{\varepsilon}\right) \ln \frac{2(k-1)}{\varepsilon^2} \right\rceil$.*

**Proof.** Since $A\big[j - C + 1, j\big]$ is a subset of $A$ and $k$-SSR$_{\mathrm{L}}$ is (by definition) a restricted version of $k$-SSR, it follows directly that

$$\mathrm{OPT}(A) \leq \mathrm{OPT}\left(A\big[j - C + 1, j\big]\right) \leq \mathrm{OPT}_{\mathrm{L}}\left(A\big[j - C + 1, j\big]\right).$$

Note that the above is true for any $j \in [n]$, therefore it is also true for the minimum. Thus, the first inequality of (2) holds. As for the second inequality of (2), we distinguish between two cases.

[12] If $t < k$, all feasible solutions for both problems have infinite ratio.

*Case 1 (Dense case):* There exists some $v \in \{k, \ldots, n\}$ s.t. $a_v \leq (1+\varepsilon)a_{v-k+1}$. Since $C \geq k$, by setting $S_i = \{v-k+i\}$ for each $i \in [k]$, we obtain

$$\min_{k \leq j \leq n} \mathrm{OPT}_{\mathrm{L}}(A[j-C+1, j]) \leq \mathrm{OPT}_{\mathrm{L}}(A[v-C+1, v]) \leq (1+\varepsilon) \leq (1+\varepsilon)\mathrm{OPT}(A).$$

*Case 2 (Sparse case):* For all $v \in \{k, \ldots, n\}$ it holds that $a_v > (1+\varepsilon)a_{v-k+1}$. By repeatedly applying this inequality, it follows that for all $v \in \{k, \ldots, n\}$ and all $i \geq 1$ such that $v - i(k-1) \geq 1$,

$$a_{v-i(k-1)} < (1+\varepsilon)^{-i} a_v. \tag{3}$$

Thus, for all such $v, i$ it holds that

$$\begin{aligned} \mathrm{sum}\Big(A\big[v-(i+1)(k-1)+1,\ v-i(k-1)\big]\Big) &\leq (k-1)a_{v-i(k-1)} \\ &< (k-1)(1+\varepsilon)^{-i}a_v. \quad \text{[by Ineq. (3)]} \end{aligned} \tag{4}$$

Consider the set $A\big[1, v-c(k-1)\big]$. Splitting this set into subsets of size (at most) $k-1$ and using (4) for each of them yields

$$\mathrm{sum}\Big(A\big[1, v-c(k-1)\big]\Big) \leq (k-1)\sum_{i=0}^{\infty}(1+\varepsilon)^{-(c+i)}a_v = \frac{(k-1)a_v}{\varepsilon(1+\varepsilon)^{c-1}}, \tag{5}$$

where the last step is calculated as the sum of a geometric series. Note that (5) holds even if $v - c(k-1) < 1$, since then it would be $\mathrm{sum}\Big(A\big[1, v-c(k-1)\big]\Big) = 0$. Substituting $c$ into (5) and using $(1+\varepsilon)^{1+1/\varepsilon} > e$ yields

$$\mathrm{sum}\Big(A\big[1, v-c(k-1)\big]\Big) < \frac{\varepsilon}{2} \cdot a_v.$$

For any $k \leq j \leq n$, we choose $v = j-k+1$ to obtain

$$\mathrm{sum}\Big(A\big[1, j-C\big]\Big) < \frac{\varepsilon}{2} \cdot a_{j-k+1}. \tag{6}$$

Intuitively, inequality (6) is a bound for the sum of the $j-C$ *smallest* elements of $A$. This inequality will be used to obtain a $(1+\varepsilon)$-approximation by removing these elements from an optimal $k$-$\mathrm{SSR}_{\mathrm{L}}$ solution.

Let $S = (S_1, \ldots, S_k)$ be an optimal solution for the $k$-$\mathrm{SSR}_{\mathrm{L}}$ instance $A_j = A[1, j]$. We define

$$S_M = \arg\max_{S_i \in S} \Sigma(S_i, A_j) \quad \text{and} \quad S_m = \arg\min_{S_i \in S} \Sigma(S_i, A_j).$$

Now consider the $k$-$\mathrm{SSR}_{\mathrm{L}}$ instance $A'_j = A\big[j-C+1, j\big]$ (i.e. $A'_j$ contains the $C$ *largest* elements of $A_j$). For each $S_i \in S$, we define $S'_i = S_i \cap [j-C+1, j]$, i.e. the subset of $S_i$ that contains elements $i$ such that $a_i \in A'_j$. We also define the respective $k$-tuple of sets, $S' = (S'_1, \ldots, S'_k)$, and the following sets,

$$S_{\mathcal{M}} = \arg\max_{S_i \in S} \Sigma(S'_i, A'_j) \quad \text{and} \quad S_{\mu} = \arg\min_{S_i \in S} \Sigma(S'_i, A'_j).$$

Observe that, by the above definitions, $S'_{\mathcal{M}}$ is a set with maximum sum among all sets in $S'$ and $S'_{\mu}$ is a set with minimum sum among all sets in $S'$. By (6), for all $i \in [k]$ it holds that

$$\Sigma(S_i, A_j) - \Sigma(S'_i, A'_j) \leq \mathrm{sum}\Big(A\big[1, j-C\big]\Big) < \frac{\varepsilon}{2} \cdot a_{j-k+1}.$$

Therefore, we obtain

$$\Sigma(S'_i, A'_j) \leq \Sigma(S_i, A_j) \leq \Sigma(S'_i, A'_j) + (\varepsilon/2) \cdot a_{j-k+1},\ \forall i \in [k]. \tag{7}$$

We now obtain a bound for the ratio $\mathcal{R}(S', A'_j)$ as follows.

$$\begin{aligned}\mathcal{R}(S', A'_j) = \frac{\Sigma\left(S'_{\mathcal{M}}, A'_j\right)}{\Sigma\left(S'_{\mu}, A'_j\right)} &\leq \frac{\Sigma\left(S_{\mathcal{M}}, A_j\right)}{\Sigma\left(S_{\mu}, A_j\right) - \frac{\varepsilon}{2} \cdot a_{j-k+1}} \qquad \text{[by Ineq. (7)]} \\ &\leq \frac{\Sigma\left(S_M, A_j\right)}{\Sigma\left(S_m, A_j\right) - \frac{\varepsilon}{2} \cdot a_{j-k+1}}. \end{aligned} \tag{8}$$

Recall that $j \in \bigcup_{i=1}^{k} S_i$ (by Definition 24), which implies that $\Sigma(S_M, A_j) \geq a_j$. Let $S^s = \{S_1^s, \ldots, S_k^s\}$ be a solution consisting of $k$ singleton sets that contain the $k$ largest elements of $A_j$, i.e. $S_i^s = \{j - i + 1\}$, $\forall i \in [k]$. Since $S^s$ is a feasible solution for $k$-SSR$_{\mathrm{L}}$ with ratio $a_j/a_{j-k+1}$, it holds that $\mathcal{R}(S, A_j) = \Sigma(S_M, A_j)/\Sigma(S_m, A_j) \leq a_j/a_{j-k+1}$. Thus, we have

$$\Sigma(S_m, A_j) \geq \Sigma(S_M, A_j) \cdot \frac{a_{j-k+1}}{a_j} \geq a_{j-k+1}.$$

Combining this with (8), we obtain

$$\mathcal{R}(S', A'_j) \leq \frac{\Sigma(S_M, A_j)}{\Sigma(S_m, A_j)\left(1 - \frac{\varepsilon}{2}\right)} = \frac{2}{2 - \varepsilon} \cdot \mathcal{R}(S, A_j).$$

Assuming $\varepsilon \in (0, 1)$, this becomes $\mathcal{R}(S', A'_j) \leq (1 + \varepsilon)\mathcal{R}(S, A_j)$. Note that some set $S'_i$ contains $j$, so $S'$ is a feasible solution for the $k$-SSR$_{\mathrm{L}}$ instance $A'_j$, therefore

$$\mathrm{OPT}_{\mathrm{L}}(A'_j) \leq \mathcal{R}(S', A'_j) \leq (1 + \varepsilon)\mathcal{R}(S, A_j) = (1 + \varepsilon)\mathrm{OPT}_{\mathrm{L}}(A_j).$$

By the definitions of $A_j, A'_j$, we have the following for all $j \in \{k, \ldots, n\}$:

$$\mathrm{OPT}_{\mathrm{L}}\left(A\big[j - C + 1, j\big]\right) \leq (1 + \varepsilon)\mathrm{OPT}_{\mathrm{L}}(A[1, j]).$$

Taking the minimum over all $j \in \{k, \ldots, n\}$ and using Lemma 25, we have

$$\min_{k \leq j \leq n} \mathrm{OPT}_{\mathrm{L}}\left(A\big[j - C + 1, j\big]\right) \leq (1 + \varepsilon)\mathrm{OPT}(A). \qquad \blacktriangleleft$$

▶ **Lemma 27** (Reduction). *If there is a $(1+\varepsilon)$-approximation algorithm for $k$-SSR$_{\mathrm{L}}$ running in time $T_L(n, \varepsilon)$, then there is a $(1+\varepsilon)$-approximation algorithm for $k$-SSR running in time*

$$O\left(nT_L\left(\frac{9k}{\varepsilon}\ln\frac{k}{\varepsilon}, \frac{\varepsilon}{3}\right)\right).$$

**Proof.** Let $A$ be a multiset of $n$ positive integers $a_1 \leq \ldots \leq a_n$. For each $j \in \{k, \ldots, n\}$, consider $A'_j = A\big[j - C + 1, j\big]$, where

$$C = (c + 1)(k - 1) \qquad \text{and} \qquad c = 1 + \left\lceil \left(1 + \frac{1}{\varepsilon}\right) \ln \frac{2(k-1)}{\varepsilon^2} \right\rceil.$$

Using the given $(1+\varepsilon)$-approximation algorithm for each of the $k$-SSR$_{\mathrm{L}}$ instances $A'_j$, we receive a $k$-tuple of sets $S^j = (S_1^j, \ldots, S_k^j)$ for which $\mathcal{R}\left(S^j, A'_j\right) \leq (1 + \varepsilon)\mathrm{OPT}_{\mathrm{L}}\left(A'_j\right)$. Thus,

$$\min_{k \leq j \leq n} \mathcal{R}\left(S^j, A'_j\right) \leq (1 + \varepsilon) \min_{k \leq j \leq n} \mathrm{OPT}_{\mathrm{L}}\left(A'_j\right) \leq (1 + \varepsilon)^2 \mathrm{OPT}(A). \qquad \text{[by Lemma 26]}$$

For all $j \in \{k, \ldots, n\}$, the set $A'_j$ contains at most $C$ elements, with

$$C = (k-1)\left(2 + \left\lceil \left(1+\frac{1}{\varepsilon}\right)\ln\frac{2(k-1)}{\varepsilon^2}\right\rceil\right) < 3k + k\left(1+\frac{1}{\varepsilon}\right)\ln\frac{2k}{\varepsilon^2} < \frac{9k}{\varepsilon}\ln\frac{k}{\varepsilon},$$

assuming $k \geq 2$ and $\varepsilon \in (0,1)$. As such, we have to run the given algorithm on $O(n)$ $k$-SSR$_{\mathrm{L}}$ instances of size bounded by $(9k/\varepsilon)\ln(k/\varepsilon)$ in order to obtain a $(1+\varepsilon)^2$-approximation for $\mathrm{OPT}(A)$. For all $\varepsilon \in (0,1)$ it holds that $(1+\varepsilon)^2 \leq 1+3\varepsilon$. Therefore, by setting the error parameter to $\varepsilon/3$ instead of $\varepsilon$ in the approximation algorithm for $k$-SSR$_{\mathrm{L}}$, we obtain a $(1+\varepsilon)$-approximation for $\mathrm{OPT}(A)$ running in time

$$O\left(nT_L\left(\frac{9k}{\varepsilon}\ln\frac{k}{\varepsilon}, \frac{\varepsilon}{3}\right)\right). \qquad \blacktriangleleft$$

We now present the main theorem of this section, which follows by using the FPTAS of Theorem 15 as the $(1+\varepsilon)$-approximation algorithm for $k$-SSR$_{\mathrm{L}}$ in Lemma 27.

▶ **Theorem 28.** *There is an FPTAS for* $k$-SSR *that runs in* $\widetilde{O}(n/\varepsilon^{3k-1})$ *time, where* $\widetilde{O}$ *hides* $\mathrm{polylog}(1/\varepsilon)$ *factors.*

**Proof.** Let $A$ be a $k$-SSR instance. Suppose we use our $k$-SSR FPTAS running in time $T_L(n,\varepsilon) = O(n^{2k}/\varepsilon^{k-1})$ (see Theorem 15) as a subroutine to solve the $k$-SSR$_{\mathrm{L}}$ instances $A'_j = A\big[j-C+1, j\big]$, $k \leq j \leq n$. This subroutine serves as a $(1+\varepsilon)$-approximation for $k$-SSR$_{\mathrm{L}}$, since for the solution $S^j = (S_1^j, \ldots, S_k^j)$ returned it holds that

$$\mathcal{R}\left(S^j, A'_j\right) \leq (1+\varepsilon)\mathrm{OPT}(A'_j) \leq (1+\varepsilon)\mathrm{OPT}_{\mathrm{L}}(A'_j).$$

Note that this subroutine does not take into account the $k$-SSR$_{\mathrm{L}}$ restriction $j \in \bigcup_{i=1}^{k} S_i^j$, therefore it might return an invalid $k$-SSR$_{\mathrm{L}}$ solution for $A'_j$. However, all solutions returned are feasible for the (unrestricted) $k$-SSR instance $A$, therefore the solution which yields $\min_{k\leq j\leq n}\mathcal{R}\left(S^j, A'_j\right)$ is a feasible one. As such, by Lemma 27 we obtain another FPTAS for $k$-SSR running in time

$$O\left(nT_L\left(\frac{9k}{\varepsilon}\ln\frac{k}{\varepsilon}, \frac{\varepsilon}{3}\right)\right) = \widetilde{O}\left(\frac{n}{\varepsilon^{3k-1}}\right). \qquad \blacktriangleleft$$

## References


1. Giannis Alonistiotis, Antonis Antonopoulos, Nikolaos Melissinos, Aris Pagourtzis, Stavros Petsalakis, and Manolis Vasilakis. Approximating subset sum ratio via partition computations. *Acta Informatica*, 61(2):101–113, 2024. doi:10.1007/S00236-023-00451-7.
2. Georgios Amanatidis, Aris Filos-Ratsikas, and Alkmini Sgouritsa. Pushing the frontier on approximate EFX allocations. In *Proceedings of the 25th ACM Conference on Economics and Computation*, EC '24, pages 1268–1286, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3670865.3673582.
3. Antonis Antonopoulos, Aris Pagourtzis, Stavros Petsalakis, and Manolis Vasilakis. Faster algorithms for k-subset sum and variations. *J. Comb. Optim.*, 45(1):24, 2023. doi:10.1007/S10878-022-00928-0.
4. Cristina Bazgan, Miklos Santha, and Zsolt Tuza. Efficient approximation algorithms for the SUBSET-SUMS EQUALITY problem. *J. Comput. Syst. Sci.*, 64(2):160–170, 2002. doi:10.1006/JCSS.2001.1784.
5. Richard E. Bellman. *Dynamic programming*. Princeton University Press, Princeton, NJ, 1957.

6 Samuel Bismuth, Vladislav Makarov, Erel Segal-Halevi, and Dana Shapira. Partitioning problems with splittings and interval targets. In Julián Mestre and Anthony Wirth, editors, *35th International Symposium on Algorithms and Computation, ISAAC 2024, December 8-11, 2024, Sydney, Australia*, volume 322 of *LIPIcs*, pages 12:1–12:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ISAAC.2024.12.

7 Karl Bringmann. A near-linear pseudopolynomial time algorithm for subset sum. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1073–1084. SIAM, 2017. doi:10.1137/1.9781611974782.69.

8 Karl Bringmann. Approximating subset sum ratio faster than subset sum. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 1260–1277. SIAM, 2024. doi:10.1137/1.9781611977912.50.

9 Karl Bringmann and Vasileios Nakos. A fine-grained perspective on approximating subset sum and partition. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1797–1815. SIAM, 2021. doi:10.1137/1.9781611976465.108.

10 Eric Budish. The combinatorial assignment problem: approximate competitive equilibrium from equal incomes. In Moshe Dror and Greys Sosic, editors, *Proceedings of the Behavioral and Quantitative Game Theory - Conference on Future Directions, BQGT '10, Newport Beach, California, USA, May 14-16, 2010*, page 74:1. ACM, 2010. doi:10.1145/1807406.1807480.

11 Lin Chen, Jiayi Lian, Yuchen Mao, and Guochuan Zhang. Approximating partition in near-linear time. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC 2024, pages 307–318, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3618260.3649727.

12 Lin Chen, Jiayi Lian, Yuchen Mao, and Guochuan Zhang. An improved pseudopolynomial time algorithm for subset sum. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024, Chicago, IL, USA, October 27-30, 2024*, pages 2202–2216. IEEE, 2024. doi:10.1109/FOCS61266.2024.00129.

13 Lin Chen, Jiayi Lian, Yuchen Mao, and Guochuan Zhang. A note on deterministic FPTAS for partition, 2025. doi:10.48550/arXiv.2501.12848.

14 Mark Cieliebak and Stephan J. Eidenbenz. Measurement errors make the partial digest problem NP-hard. In Martin Farach-Colton, editor, *LATIN 2004: Theoretical Informatics, 6th Latin American Symposium, Buenos Aires, Argentina, April 5-8, 2004, Proceedings*, volume 2976 of *Lecture Notes in Computer Science*, pages 379–390. Springer, 2004. doi:10.1007/978-3-540-24698-5_42.

15 Mark Cieliebak, Stephan J. Eidenbenz, and Paolo Penna. Noisy data make the partial digest problem NP-hard. In Gary Benson and Roderic D. M. Page, editors, *Algorithms in Bioinformatics, Third International Workshop, WABI 2003, Budapest, Hungary, September 15-20, 2003, Proceedings*, volume 2812 of *Lecture Notes in Computer Science*, pages 111–123. Springer, 2003. doi:10.1007/978-3-540-39763-2_9.

16 Ed Coffman and Michael Langston. A performance guarantee for the greedy set-partitioning algorithm. *Acta Informatica*, 21:409–415, November 1984. doi:10.1007/BF00264618.

17 Bryan Deuermeyer, Donald Friesen, and Michael Langston. Scheduling to maximize the minimum processor finish time in a multiprocessor system. *SIAM Journal on Algebraic and Discrete Methods*, 3, June 1982. doi:10.1137/0603019.

18 Paul Duetting, Michal Feldman, and Yarden Rashti. Succinct ambiguous contracts, 2025. doi:10.48550/arXiv.2503.02592.

19 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

20 George Gens and Eugene Levner. A fast approximation algorithm for the subset-sum problem. *INFOR: Information Systems and Operational Research*, 32(3):143–148, 1994. doi:10.1080/03155986.1994.11732245.

**21** Theodore P. Hill. Partitioning General Probability Measures. *The Annals of Probability*, 15(2):804–813, 1987. doi:10.1214/aop/1176992173.

**22** Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *J. ACM*, 34(1):144–162, January 1987. doi:10.1145/7531.7535.

**23** Ce Jin and Hongxun Wu. A simple near-linear pseudopolynomial time randomized algorithm for subset sum. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San Diego, CA, USA*, volume 69 of *OASIcs*, pages 17:1–17:6. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/OASICS.SOSA.2019.17.

**24** Sotiris Kanellopoulos, Giorgos Mitropoulos, Antonis Antonopoulos, Nikos Leonardos, Aris Pagourtzis, Christos Pergaminelis, Stavros Petsalakis, and Kanellos Tsitouras. Approximation schemes for k-subset sum ratio and k-way number partitioning ratio, 2025. doi:10.48550/arXiv.2503.18241.

**25** Hans Kellerer, Renata Mansini, Ulrich Pferschy, and Maria Grazia Speranza. An efficient fully polynomial approximation scheme for the subset-sum problem. *J. Comput. Syst. Sci.*, 66(2):349–370, 2003. doi:10.1016/S0022-0000(03)00006-0.

**26** Konstantinos Koiliaris and Chao Xu. Faster pseudopolynomial time algorithms for subset sum. *ACM Trans. Algorithms*, 15(3):40:1–40:20, 2019. doi:10.1145/3329863.

**27** Richard E. Korf. Objective functions for multi-way number partitioning. In Ariel Felner and Nathan R. Sturtevant, editors, *Proceedings of the Third Annual Symposium on Combinatorial Search, SOCS 2010, Stone Mountain, Atlanta, Georgia, USA, July 8-10, 2010*, pages 71–72. AAAI Press, 2010. doi:10.1609/SOCS.V1I1.18172.

**28** David Kurokawa, Ariel D. Procaccia, and Junxing Wang. Fair enough: Guaranteeing approximate maximin shares. *J. ACM*, 65(2):8:1–8:27, 2018. doi:10.1145/3140756.

**29** Joseph Y-T. Leung. Bin packing with restricted piece sizes. *Information Processing Letters*, 31(3):145–149, 1989. doi:10.1016/0020-0190(89)90223-8.

**30** Richard J. Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. On approximately fair allocations of indivisible goods. In Jack S. Breese, Joan Feigenbaum, and Margo I. Seltzer, editors, *Proceedings 5th ACM Conference on Electronic Commerce (EC-2004), New York, NY, USA, May 17-20, 2004*, pages 125–131. ACM, 2004. doi:10.1145/988772.988792.

**31** Nikolaos Melissinos and Aris Pagourtzis. A faster FPTAS for the subset-sums ratio problem. In Lusheng Wang and Daming Zhu, editors, *Computing and Combinatorics - 24th International Conference, COCOON 2018, Qing Dao, China, July 2-4, 2018, Proceedings*, volume 10976 of *Lecture Notes in Computer Science*, pages 602–614. Springer, 2018. doi:10.1007/978-3-319-94776-1_50.

**32** Stephan Mertens. The easiest hard problem: Number partitioning. In Allon G. Percus, Gabriel Istrate, and Cristopher Moore, editors, *Computational Complexity and Statistical Physics*, Santa Fe Institute Studies in the Sciences of Complexity, pages 125–140. Oxford University Press, 2006.

**33** Marcin Mucha, Jesper Nederlof, Jakub Pawlewicz, and Karol Wegrzycki. Equal-subset-sum faster than the meet-in-the-middle. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, volume 144 of *LIPIcs*, pages 73:1–73:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.ESA.2019.73.

**34** Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. A subquadratic approximation scheme for partition. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 70–88. SIAM, 2019. doi:10.1137/1.9781611975482.5.

**35** Danupon Nanongkai. Simple FPTAS for the subset-sums ratio problem. *Inf. Process. Lett.*, 113(19-21):750–753, 2013. doi:10.1016/J.IPL.2013.07.009.

**36** Trung Thanh Nguyen and Jörg Rothe. Minimizing envy and maximizing average nash social welfare in the allocation of indivisible goods. *Discret. Appl. Math.*, 179:54–68, 2014. doi:10.1016/J.DAM.2014.09.010.

**37** Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994. doi:10.1016/S0022-0000(05)80063-7.

**38** Diego Recalde, Daniel Severín, Ramiro Torres, and Polo Vaca. An exact approach for the balanced k-way partitioning problem with weight constraints and its application to sports team realignment. *J. Comb. Optim.*, 36(3):916–936, 2018. doi:10.1007/S10878-018-0254-1.

**39** Sartaj Sahni. Algorithms for scheduling independent tasks. *J. ACM*, 23(1):116–127, 1976. doi:10.1145/321921.321934.

**40** Ethan L. Schreiber, Richard E. Korf, and Michael D. Moffitt. Optimal multi-way number partitioning. *J. ACM*, 65(4):24:1–24:61, 2018. doi:10.1145/3184400.

**41** Max Springer, Mohammad Taghi Hajiaghayi, and Hadi Yami. Almost envy-free allocations of indivisible goods or chores with entitlements. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. doi:10.1609/aaai.v38i9.28851.

**42** 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.

**43** Gerhard J. Woeginger and Zhongliang Yu. On the equal-subset-sum problem. *Inf. Process. Lett.*, 42(6):299–302, 1992. doi:10.1016/0020-0190(92)90226-L.

**44** Yair Zadok, Nadav Voloch, Noa Voloch-Bloch, and Maor Meir Hajaj. Multiple subset problem as an encryption scheme for communication. *CoRR*, abs/2401.09221, 2024. doi:10.48550/arXiv.2401.09221.

**45** Houyu Zhou, Tianze Wei, Biaoshuai Tao, and Minming Li. Fair allocation of items in multiple regions. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. doi:10.1609/aaai.v38i9.28861.