Approximating Subset Sum Ratio via Subset Sum Computations

Giannis Alonistiotis[1], Antonis Antonopoulos[1], Nikolaos Melissinos[2], Aris Pagourtzis[1], Stavros Petsalakis[1], and Manolis Vasilakis[1](✉)

[1] School of Electrical and Computer Engineering, National Technical University of Athens, Polytechnioupoli, 15780 Zografou, Athens, Greece
{ialonistiotis,aanton,spetsalakis,mvasilakis}@corelab.ntua.gr, pagour@cs.ntua.gr

[2] Université Paris-Dauphine, PSL University, CNRS, LAMSADE, 75016 Paris, France
nikolaos.melissinos@dauphine.eu

Abstract. We present a new FPTAS for the Subset Sum Ratio problem, which, given a set of integers, asks for two disjoint subsets such that the ratio of their sums is as close to 1 as possible. Our scheme makes use of exact and approximate algorithms for the closely related Subset Sum problem, hence any progress over those—such as the recent improvement due to Bringmann and Nakos [SODA 2021]—carries over to our FPTAS. Depending on the relationship between the size of the input set n and the error margin ε, we improve upon the best currently known algorithm of Melissinos and Pagourtzis [COCOON 2018] of complexity $\mathcal{O}(n^4/\varepsilon)$. In particular, the exponent of n in our proposed scheme may decrease down to 2, depending on the Subset Sum algorithm used. Furthermore, while the aforementioned state of the art complexity, expressed in the form $\mathcal{O}((n + 1/\varepsilon)^c)$, has constant $c = 5$, our results establish that $c < 5$.

Keywords: Approximation scheme · Combinatorial optimization · Knapsack problems · Subset Sum · Subset Sum Ratio

1 Introduction

One of Karp's 21 NP-complete problems [18], Subset Sum has seen astounding progress over the last few years. Koiliaris and Xu [21], Bringmann [7] and Jin and Wu [17] have presented pseudopolynomial algorithms resulting in substantial improvements over the long-standing standard approach of Bellman [6], and the improvement by Pisinger [29]. Moreover, the latter two algorithms [7,17] match the SETH-based lower bounds proved in [1]. Additionally, recently there has been progress in the approximation scheme of Subset Sum, the first such improvement in over 20 years, with a new algorithm introduced by Bringmann

Aris Pagourtzis and Stavros Petsalakis were supported in part by the PEVE 2020 basic research support program of the National Technical University of Athens.

and Nakos [9], as well as corresponding lower bounds obtained through the lens of fine-grained complexity.

The Equal Subset Sum problem, which, given an input set, asks for two disjoint subsets of equal sum, is closely related to Subset Sum. It finds applications in multiple different fields, ranging from computational biology [10,13] and computational social choice [22], to cryptography [30], to name a few. In addition, it is related to important theoretical concepts such as the complexity of search problems in the class TFNP [28].

The centerpiece of this paper is the Subset Sum Ratio problem, the optimization version of Equal Subset Sum, which asks, given an input set $S \subseteq \mathbb{N}$, for two disjoint subsets $S_1, S_2 \subseteq S$, such that the following ratio is minimized

$$\frac{\max\left\{\sum_{s_i \in S_1} s_i, \sum_{s_j \in S_2} s_j\right\}}{\min\left\{\sum_{s_i \in S_1} s_i, \sum_{s_j \in S_2} s_j\right\}}$$

We present a new approximation scheme for this problem, highlighting its close relationship with the classical Subset Sum problem. Our proposed algorithm is the first to associate these closely related problems and, depending on the relationship of the cardinality of the input set n and the value of the error margin ε, achieves better asymptotic bounds than the current state of the art [23]. Moreover, while the complexity of the current state of the art approximation scheme expressed in the form $\mathcal{O}((n + 1/\varepsilon)^c)$ has an exponent $c = 5$, we present an FPTAS with constant $c < 5$.

1.1 Related Work

Equal Subset Sum as well as its optimization version called Subset Sum Ratio [5] are closely related to problems appearing in many scientific areas. Some examples include the Partial Digest problem, which comes from computational biology [10,13], the allocation of individual goods [22], tournament construction [20], and a variation of Subset Sum, called Multiple Integrated Sets SSP, which finds applications in the field of cryptography [30]. Furthermore, it is related to important concepts in theoretical computer science; for example, a restricted version of Equal Subset Sum lies in a subclass of the complexity class TFNP, namely in PPP [28], a class consisting of search problems that always have a solution due to some pigeonhole argument, and no polynomial time algorithm is known for this restricted version.

Equal Subset Sum has been proven NP-hard by Woeginger and Yu [31] (see also the full version of [25] for an alternative proof) and several variations have been proven NP-hard by Cieliebak *et al.* in [11,12]. A 1.324-approximation algorithm has been proposed for Subset Sum Ratio in [31] and several FPTASs appeared in [5,23,27], the fastest so far being the one in [23] of complexity $\mathcal{O}(n^4/\varepsilon)$, the complexity of which seems to also apply to various meaningful special cases, as shown in [24].

As far as exact algorithms are concerned, recent progress has shown that Equal Subset Sum can be solved probabilistically in[1] $\mathcal{O}^*(1.7088^n)$ time [25], faster than a standard "meet-in-the-middle" approach yielding an $\mathcal{O}^*(3^{n/2}) \leq \mathcal{O}^*(1.7321^n)$ time algorithm.

These problems are tightly connected to Subset Sum, which has seen impressive advances recently, due to Koiliaris and Xu [21] who gave a deterministic $\tilde{\mathcal{O}}(\sqrt{n}t)$ algorithm, where n is the number of input elements and t is the target, and by Bringmann [7] who gave a $\tilde{\mathcal{O}}(n+t)$ randomized algorithm, which is essentially optimal under SETH [1]. See also [2] for an extension of these algorithms to a more general setting. Jin and Wu subsequently proposed a simpler randomized algorithm [17] achieving the same bounds as [7], which however seems to only solve the decision version of the problem. Recently, Bringmann and Nakos [8] have presented an $\mathcal{O}\big(|\mathcal{S}_t(Z)|^{4/3}\text{poly}(\log t)\big)$ algorithm, where $\mathcal{S}_t(Z)$ is the set of all subset sums of the input set Z that are smaller than t, based on top-k convolution.

Regarding approximation schemes for Subset Sum, recently Bringmann and Nakos [9] presented the first improvement in over 20 years, since the scheme of [19] had remained the state of the art. Making use of modern techniques, they additionally provide lower bounds based on the popular *min-plus convolution* conjecture [14]. Furthermore, they present a new FPTAS for the closely related Partition problem, by observing that their techniques can be used to approximate a slightly more *relaxed* version of the Subset Sum problem, firstly studied in [25].

1.2 Our Contribution

We present a novel approximation scheme for the Subset Sum Ratio problem. Our algorithm makes use of exact and approximation algorithms for Subset Sum, thus, any improvement over those carries over to our proposed scheme. Additionally, depending on the relationship between n and ε, our algorithm improves upon the best existing approximation scheme of [23].

We start by presenting some necessary background in Sect. 2. Afterwards, in Sect. 3 we introduce an FPTAS for a restricted version of the problem. In the following Sect. 4, we explain how to make use of the algorithm presented in the previous section, in order to obtain an approximation scheme for the Subset Sum Ratio problem. The complexity of the final scheme is thoroughly analyzed in Sect. 5, followed by some possible directions for future research in Sect. 6.

2 Preliminaries

Let, for $x \in \mathbb{N}$, $[x] = \{0, \dots, x\}$ denote the set of integers in the interval $[0, x]$. Given a set $S \subseteq \mathbb{N}$, denote its *largest element* by $\max(S)$ and the sum of its

[1] Standard $\mathcal{O}^*$ notation is used to hide polynomial and $\tilde{\mathcal{O}}$ to hide polylogarithmic factors.

elements by $\Sigma(S) = \sum_{s \in S} s$. If we are additionally given a value $\varepsilon \in (0,1)$, define the following *partition* of its elements:

- The set of its *large* elements as $L(S,\varepsilon) = \{s \in S \mid s \geq \varepsilon \cdot \max(S)\}$. Note that $\max(S) \in L(S,\varepsilon)$, for any $\varepsilon \in (0,1)$.
- The set of its *small* elements as $M(S,\varepsilon) = \{s \in S \mid s < \varepsilon \cdot \max(S)\}$.

In the following, since the values of the associated parameters will be clear from the context, they will be omitted and we will refer to these sets simply as L and M.

The following definitions will be useful for the rest of this paper.

Definition 1 (Closest Set and pair). *Given a set $S_i \subseteq A$, we define as its* closest set *a set $S_{i,\mathrm{opt}}$ such that $S_{i,\mathrm{opt}} \subseteq A \backslash S_i$ and $\Sigma(S_i) \geq \Sigma(S_{i,\mathrm{opt}}) \geq \Sigma(S')$ for all $S' \subseteq A \backslash S_i$. The pair $(S_i, S_{i,\mathrm{opt}})$ is called* closest pair*.*

Definition 2 (ε-close set and pair). *Given a set $S_i \subseteq A$, we define as its ε-*close set *a set $S_{i,\varepsilon}$ such that $S_{i,\varepsilon} \subseteq A \backslash S_i$ and $\Sigma(S_i) \geq \Sigma(S_{i,\varepsilon}) \geq (1-\varepsilon) \cdot \Sigma(S_{i,\mathrm{opt}})$. The pair $(S_i, S_{i,\varepsilon})$ is called ε-*close pair*.*

Remark 1. Note that $S_{i,\mathrm{opt}}$ is also an ε-close set of S_i for any $\varepsilon \in (0,1)$.

Definition 3 (Subset Sum). *Given a set X and target t, compute a subset $Y \subseteq X$, such that $\Sigma(Y) = \max\{\Sigma(Z) \mid Z \subseteq X, \Sigma(Z) \leq t\}$.*

Definition 4 (Approximate Subset Sum). *Given a set X, target t and error margin ε, compute a subset $Y \subseteq X$ such that $(1-\varepsilon) \cdot OPT \leq \Sigma(Y) \leq OPT$, where $OPT = \max\{\Sigma(Z) \mid Z \subseteq X, \Sigma(Z) \leq t\}$.*

By solving Subset Sum or its approximate version, one can compute an ε-close set for a given subset $S_i \subseteq A$ as follows.

1. **Closest set ($S_{i,\mathrm{opt}}$) computation**
   Compute the subset sums of set $A \backslash S_i$ with target $\Sigma(S_i)$ and keep the largest non exceeding. This can be achieved by a standard meet in the middle [16] algorithm.
2. **ε-close set ($S_{i,\varepsilon}$) computation**
   Run an approximate Subset Sum algorithm [9,19] with error margin ε on set $A \backslash S_i$ with target $\Sigma(S_i)$.

3 Approximation Scheme for a Restricted Version

In this section, we present an FPTAS for the constrained version of the Subset Sum Ratio problem where we are only interested in approximating solutions that involve large subset sums. By this, we mean that for at least one of the subsets of the optimal solution, the sum of its large elements must be no less than $\max(A) = a_n$ (assuming that $A = \{a_1, \ldots, a_n\}$ is the *sorted* input set); let r_{opt} denote the subset sum ratio of such an optimal solution. Our FPTAS will return a solution of ratio r, such that $1 \leq r \leq (1+\varepsilon) \cdot r_{\mathrm{opt}}$, for a given error margin $\varepsilon \in (0,1)$; however, we allow that the sets of the returned solution do not necessarily satisfy the aforementioned constraint (i.e. the sum of their large elements may be less than a_n).

3.1 Outline of the Algorithm

We now present a rough outline of the algorithm, along with its respective pseudocode:

- At first, we search for approximate solutions involving exclusively large elements from $L(A, \varepsilon)$.
- To this end, we produce the subset sums formed by these large elements. If their number exceeds n/ε^2, then we can easily find an approximate solution.
- Otherwise, for each of the produced subsets, we find its corresponding ε'-close set, for some appropriate ε' defined later.
- Then, it suffices to consider only these pairs of subsets when searching for an approximate solution.
- In the case that the optimal solution involves small elements, we can approximate it by adding elements of $M(A, \varepsilon)$ in a greedy way.

Algorithm 1. ConstrainedSSR(A, ε, T)

```
Input : Sorted set $A = \{a_1, \ldots, a_n\}$, error margin $\varepsilon$ and table of partial sums $T$.
Output : $(1+\varepsilon)$-approximation of the optimal solution respecting the constraint.
 1: Partition $A$ to $M = \{a_i \in A \mid a_i < \varepsilon \cdot a_n\}$ and $L = \{a_i \in A \mid a_i \geq \varepsilon \cdot a_n\}$.
 2: Split interval $[0, n \cdot a_n]$ to $n/\varepsilon^2$ bins of size $\varepsilon^2 \cdot a_n$.
 3: while filling the bins with the subset sums of $L$ do
 4:     if two subset sums correspond to the same bin then
 5:         return an approximation solution based on these.   ▷ $\mathcal{O}(n/\varepsilon^2)$ complexity.
 6:     end if
 7: end while
 8: $2^{|L|} \leq n/\varepsilon^2 \iff |L| \leq \log(n/\varepsilon^2)$.
 9: for each subset in a bin do                                ▷ $\mathcal{O}(n/\varepsilon^2)$ subsets.
10:     Find its $\varepsilon'$-close set.                      ▷ Complexity analysis in Section 5.
11:     Add small elements accordingly.     ▷ $\mathcal{O}(\log n)$ complexity, see Subsection 3.3.
12: end for
```

3.2 Regarding only Large Elements

We firstly search for an $(1+\varepsilon)$-approximate solution with $\varepsilon \in (0, 1)$, without involving any of the elements that are smaller than $\varepsilon \cdot a_n$. Let $M = \{a_i \in A \mid a_i < \varepsilon \cdot a_n\}$ be the set of small elements and $L = A \backslash M = \{a_i \in A \mid a_i \geq \varepsilon \cdot a_n\}$ be the set of large elements.

After partitioning the input set, we split the interval $[0, n \cdot a_n]$ into smaller intervals, called bins, of size $l = \varepsilon^2 \cdot a_n$ each, as depicted in Fig. 1.

Thus, there are a total of $B = n/\varepsilon^2$ bins. Notice that each possible subset of the input set will belong to a respective bin constructed this way, depending on its sum. Additionally, if two sets correspond to the same bin, then the difference of their subset sums will be at most l.

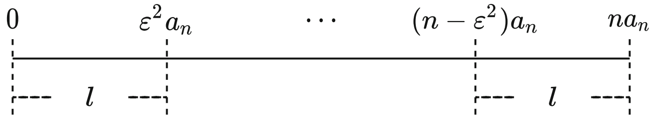

Fig. 1. Split of the interval $[0, n \cdot a_n]$ to bins of size l.

The next step of our algorithm is to generate all the possible subset sums, occurring from the set of large elements L. The complexity of this procedure is $\mathcal{O}(2^{|L|})$, where $|L|$ is the cardinality of set L. Notice however, that it is possible to bound the number of the produced subset sums by the number of bins B, since if two sums belong to the same bin they constitute a solution, as shown in Lemma 1, in which case the algorithm terminates in time $\mathcal{O}(n/\varepsilon^2)$.

Lemma 1. *If two subsets correspond to the same bin, we can find an $(1+\varepsilon)$-approximation solution.*

Proof. Suppose there exist two sets $L_1, L_2 \subseteq L$ whose sums correspond to the same bin, with $\varSigma(L_1) \leq \varSigma(L_2)$. Notice that there is no guarantee regarding the disjointness of said subsets, thus consider $L'_1 = L_1 \backslash L_2$ and $L'_2 = L_2 \backslash L_1$, for which it is obvious that $\varSigma(L'_1) \leq \varSigma(L'_2)$.

Additionally, assume that $L'_1 \neq \emptyset$. Then it holds that

$$\varSigma(L'_2) - \varSigma(L'_1) = \varSigma(L_2) - \varSigma(L_1) \leq l$$

Therefore, the sets L'_1 and L'_2 constitute an $(1+\varepsilon)$-approximation solution, since

$$\begin{aligned}\frac{\varSigma(L'_2)}{\varSigma(L'_1)} &\leq \frac{\varSigma(L'_1) + l}{\varSigma(L'_1)} = 1 + \frac{l}{\varSigma(L'_1)} \\ &\leq 1 + \frac{\varepsilon^2 \cdot a_n}{\varepsilon \cdot a_n} = 1 + \varepsilon\end{aligned}$$

where the last inequality is due to the fact that $L'_1 \subseteq L$ is composed of elements $\geq \varepsilon \cdot a_n$, thus $\varSigma(L'_1) \geq \varepsilon \cdot a_n$.

It remains to show that $L'_1 \neq \emptyset$. Assume that $L'_1 = \emptyset$. This implies that $L_1 \subseteq L_2$ and since we consider each subset of L only once and the input is a set and not a multiset, it holds that $L_1 \subset L_2 \implies L'_2 \neq \emptyset$. Since L_1 and L_2 correspond to the same bin, it holds that

$$\varSigma(L_2) - \varSigma(L_1) \leq l \implies \varSigma(L'_2) - \varSigma(L'_1) \leq l \implies \varSigma(L'_2) \leq l$$

which is a contradiction, since L'_2 is a non empty subset of L, which is comprised of elements greater than or equal to $\varepsilon \cdot a_n$, hence $\varSigma(L'_2) \geq \varepsilon \cdot a_n > \varepsilon^2 \cdot a_n = l$, since $\varepsilon < 1$.

Consider an ε' such that $1/(1-\varepsilon') \leq 1+\varepsilon$ for all $\varepsilon \in (0,1)$, for instance $\varepsilon' = \varepsilon/2$ (the exact value of ε' will be computed in Sect. 5). If every produced subset sum of the previous step belongs to a distinct bin, then, we compute their respective ε'-close sets, as described in Sect. 2. We can approximate an optimal solution that involves exclusively large elements using these pairs.

Before we prove the previous statement, observe that, if the optimal solution involves sets $L_1, L_2 \subseteq L$ composed only of large elements, where $\Sigma(L_1) \leq \Sigma(L_2)$, then $\Sigma(L_1) = \Sigma(L_{2,\mathrm{opt}})$, where $L_{2,\mathrm{opt}}$ is a closest set of L_2, with respect to the set $L \backslash L_2$.

Lemma 2. *If the optimal ratio r_{opt} involves sets consisting of only large elements, then there exists an ε'-close pair with ratio $r \leq (1+\varepsilon) \cdot r_{\mathrm{opt}}$.*

Proof. Assume that the sets $S_1^*, S_2^* \subseteq L$ form the optimal solution (S_2^*, S_1^*) and $\frac{\Sigma(S_2^*)}{\Sigma(S_1^*)} = r_{\mathrm{opt}} \geq 1$ is the optimal ratio. Then, as mentioned, it holds that $\Sigma(S_1^*) = \Sigma(S_{2,\mathrm{opt}}^*)$. For each set of large elements, there exists an ε'-close set and a corresponding ε'-close pair; let $(S_2^*, S_{2,\varepsilon'}^*)$ be this pair for set S_2^*. Then,

$$\Sigma(S_2^*) \geq \Sigma(S_1^*) = \Sigma(S_{2,\mathrm{opt}}^*) \geq \Sigma(S_{2,\varepsilon'}^*) \geq (1-\varepsilon') \cdot \Sigma(S_1^*)$$

Thus, it holds that

$$1 \leq \frac{\Sigma(S_2^*)}{\Sigma(S_{2,\varepsilon'}^*)} \leq \frac{1}{(1-\varepsilon')} \cdot \frac{\Sigma(S_2^*)}{\Sigma(S_1^*)} \leq (1+\varepsilon) \cdot r_{\mathrm{opt}}$$

Therefore, we have proved that in the case where the optimal solution consists of sets comprised of only large elements, it is possible to find an $(1+\varepsilon)$-approximation solution. This is achieved by computing an ε'-close set for each subset $L_i \subseteq L$ belonging in some bin, using the algorithms described in the preliminaries, with respect to set $L \backslash L_i$ and target $\Sigma(L_i)$. The total cost of these algorithms will be thoroughly analyzed in Sect. 5 and depends on the algorithm used.

It is important to note that by utilizing an (exact or approximation) algorithm for Subset Sum, we establish a connection between the complexities of Subset Sum and approximating Subset Sum Ratio in a way that any future improvement in the first carries over to the second.

3.3 General $(1+\varepsilon)$-Approximation Solutions

Whereas we previously considered optimal solutions involving exclusively large elements, here we will search for approximations for those optimal solutions that use all the elements of the input set, hence include small elements, and satisfy our constraint. We will prove that in order to approximate those optimal solutions, it suffices to consider only the ε'-close pairs corresponding to each distinct bin and add small elements to them. In other words, instead of considering any two random disjoint subsets consisting of large elements[2] and subsequently adding

[2] Note that the number of these random pairs is $3^{|L|}$.

to these the small elements, we can instead consider only the pairs computed in the previous step, the number of which is bounded by the number of bins $B = n/\varepsilon^2$. Moreover, we will prove that it suffices to add the small elements to our solution in a greedy way.

Since the algorithm has not detected a solution so far, due to Lemma 1 every computed subset sum of set L belongs to a different bin. Thus, their total number is bounded by the number of bins B, i.e.

$$2^{|L|} \leq \left(\frac{n}{\varepsilon^2}\right) \iff |L| \leq \log\left(\frac{n}{\varepsilon^2}\right).$$

We proceed by involving small elements in order to reduce the difference between the sums of ε'-close pairs, thus reducing their ratio.

Lemma 3. *Given the ε'-close pairs, one can find an $(1+\varepsilon)$-approximation solution for the constrained version of* Subset Sum Ratio*, in the case that the optimal solution involves small elements.*

Proof (sketch). Due to page limitations, we only give a short sketch of the proof here; the complete proof is included in the full version of the paper.

Let $S_1^* = L_1^* \cup M_1^*$ and $S_2^* = L_2^* \cup M_2^*$ be disjoint subsets that form an optimal solution, where $\Sigma(S_1^*) \leq \Sigma(S_2^*)$, $L_1^*, L_2^* \subseteq L$ and $M_1^*, M_2^* \subseteq M$.

For $\Sigma(L_1^*) < \Sigma(L_2^*)$ (respectively $\Sigma(L_2^*) < \Sigma(L_1^*)$), we show that is suffices to add an appropriate subset $M_k \subseteq M$ to $L_{2,\varepsilon'}^*$ (respectively $L_{1,\varepsilon'}^*$) in order to approximate the optimal solution $r_{\text{opt}} = \frac{\Sigma(S_2^*)}{\Sigma(S_1^*)}$, where $M_k = \{a_i \in M \mid i \in [k]\}$ and $k \leq |M|$.

Therefore, by adding in a greedy way small elements to an ε'-close set of the set with the largest sum among L_1^* and L_2^*, we can successfully approximate the optimal solution.

Adding Small Elements Efficiently. Here, we will describe a method to efficiently add small elements to our sets. As a reminder, up to this point the algorithm has detected an ε'-close pair (L_2, L_1), such that $L_1, L_2 \subseteq L$ with $\Sigma(L_1) < \Sigma(L_2)$. Thus, we search for some k such that $\Sigma(L_1 \cup M_k) \leq \Sigma(L_2) + \varepsilon \cdot a_n$, where $M_k = \{a_i \in M \mid i \in [k]\}$. Notice that if $\Sigma(M) \geq \Sigma(L_2) - \Sigma(L_1)$, there always exists such a set M_k, since by definition, each element of set M is smaller than $\varepsilon \cdot a_n$. In order to determine M_k, we make use of an array of partial sums $T[k] = \Sigma(M_k)$, where $k \leq |M|$. Notice that T is sorted; therefore, since T is already available (see Algorithm 2), each time we need to compute a subset with the desired property, this can be done in $\mathcal{O}(\log k) = \mathcal{O}(\log n)$ time.

4 Final Algorithm

The algorithm presented in the previous section constitutes an approximation scheme for Subset Sum Ratio, in the case where at least one of the solution subsets has sum of its large elements greater than, or equal to the max element of the input set. Thus, in order to solve the Subset Sum Ratio problem, it suffices to run the previous algorithm n times, where n depicts the cardinality of the input set A, while each time removing the max element of A.

In particular, suppose that the optimal solution involves disjoint sets S_1^* and S_2^*, where $a_k = \max\{S_1^* \cup S_2^*\}$. There exists an iteration for which the algorithm considers as input the set $A_k = \{a_1, \ldots, a_k\}$. In this iteration, the element a_k is the largest element and the algorithm searches for a solution where the sum of the large elements of one of the two subsets is at least a_k. The optimal solution has this property so the ratio of the approximate solution that the algorithm of the previous section returns is at most $(1 + \varepsilon)$ times the optimal.

Consequently, n repetitions of the algorithm suffice to construct an FPTAS for Subset Sum Ratio.

Notice that if at some repetition, the sets returned due to the algorithm of Sect. 3 have ratio at most $1 + \varepsilon$, then this ratio successfully approximates the optimal ratio $r_{\text{opt}} \geq 1$, since $1 + \varepsilon \leq (1 + \varepsilon) \cdot r_{\text{opt}}$, therefore they constitute an approximation solution.

Algorithm 2. SSR(A, ε)

```
Input : Sorted set $A = \{a_1, \ldots, a_n\}$ and error margin $\varepsilon$.
Output : $(1 + \varepsilon)$-approximation of the optimal solution for Subset Sum Ratio.
1: Create array $T$ such that $T[k] = \sum_{i=1}^{k} a_i$.        ▷ $\Theta(n)$ time.
2: for $i = n, \ldots, 1$ do
3:     ConstrainedSSR$(\{a_1, \ldots, a_i\}, \varepsilon, T)$
4: end for
```

5 Complexity

The total complexity of the final algorithm is determined by three distinct operations, over the n iterations of the algorithm:

1. The cost to compute all the possible subset sums occurring from large elements. It suffices to consider the case where this is bounded by the number of bins $B = n/\varepsilon^2$, due to Lemma 1.
2. The cost to find the ε'-close pair for each subset in a distinct bin. The cost of this operation will be analyzed in the following subsection.
3. The cost to include small elements to the ε'-close pairs. There are B ε'-close pairs, and each requires $\mathcal{O}(\log n)$ time, thus the total time required is $\mathcal{O}\left(\frac{n}{\varepsilon^2} \cdot \log n\right)$.

5.1 Complexity to Find the ε'-Close Pairs

Using Exact Subset Sum Computations. The first algorithm we mentioned is a standard meet in the middle algorithm. Here we will analyze its complexity.

Let subset $L' \subseteq L$ such that $|L'| = k$. The meet in the middle algorithm on the set $L \backslash L'$ costs time

$$\mathcal{O}\left(\frac{|L \backslash L'|}{2} \cdot 2^{\frac{|L \backslash L'|}{2}}\right).$$

Notice that the number of subsets of L of cardinality k is $\binom{|L|}{k}$ and that $|L| \leq \log(n/\varepsilon^2)$. Additionally,

$$\begin{aligned}\sum_{k=0}^{|L|} \binom{|L|}{k} \cdot 2^{\frac{|L|-k}{2}} \cdot \frac{|L|-k}{2} &= 2^{|L|/2} \cdot \sum_{k=0}^{|L|} \binom{|L|}{k} \cdot 2^{-k/2} \cdot \frac{|L|-k}{2} \\ &\leq 2^{|L|/2} \cdot \frac{|L|}{2} \cdot \sum_{k=0}^{|L|} \binom{|L|}{k} \cdot 2^{-k/2}\end{aligned}$$

Furthermore, let $c = (1 + 2^{-1/2})$, where $\log c = 0.7715... < 0.8$. Due to Binomial Theorem, it holds that

$$\sum_{k=0}^{|L|} \binom{|L|}{k} \cdot 2^{-k/2} = (1 + 2^{-1/2})^{|L|} = c^{|L|} \leq c^{\log(n/\varepsilon^2)} = (n/\varepsilon^2)^{\log c}$$

Consequently, the complexity to find a closest set for every subset in a bin is

$$\begin{aligned}\mathcal{O}\Big(2^{|L|/2} \cdot \frac{|L|}{2} \cdot (n/\varepsilon^2)^{\log c}\Big) &= \mathcal{O}\Big((n/\varepsilon^2)^{1/2} \cdot \log(n/\varepsilon^2) \cdot (n/\varepsilon^2)^{\log c}\Big) \\ &= \mathcal{O}\Big(\frac{n^{1.3}}{\varepsilon^{2.6}} \cdot \log(n/\varepsilon^2)\Big)\end{aligned}$$

Using approximate Subset Sum computations. Here we will analyze the complexity in the case we run an approximate Subset Sum algorithm in order to compute the ε'-close pairs.

For subset $L_i \subseteq L$ of sum $\Sigma(L_i)$, we run an approximate Subset Sum algorithm ([9,19]), with error margin ε' such that

$$\frac{1}{1-\varepsilon'} \leq 1 + \varepsilon \iff \varepsilon' \leq \frac{\varepsilon}{1+\varepsilon}$$

By choosing the maximum such ε', we have that

$$\varepsilon' = \frac{\varepsilon}{1+\varepsilon} \implies \frac{1}{\varepsilon'} = \frac{1+\varepsilon}{\varepsilon} = \frac{1}{\varepsilon} + 1 \implies \frac{1}{\varepsilon'} = \mathcal{O}\Big(\frac{1}{\varepsilon}\Big)$$

Thus, if we use for instance the approximation algorithm[3] presented at [19], the complexity of finding all the ε'-close sets (one for every subset in a bin, for a total of a maximum of $B = n/\varepsilon^2$ subsets) is

[3] Of complexity $\mathcal{O}\big(\min\{\frac{n}{\varepsilon}, n + \frac{1}{\varepsilon^2} \cdot \log(1/\varepsilon)\}\big)$ for n elements and error margin ε.

$$\mathcal{O}\left(\frac{n}{\varepsilon^2} \cdot \min\left\{\frac{|L|}{\varepsilon'}, |L| + \frac{1}{(\varepsilon')^2} \cdot \log(1/\varepsilon')\right\}\right) =$$
$$\mathcal{O}\left(\frac{n}{\varepsilon^2} \cdot \min\left\{\frac{|L|}{\varepsilon}, |L| + \frac{1}{\varepsilon^2} \cdot \log(1/\varepsilon)\right\}\right) =$$
$$\mathcal{O}\left(\frac{n}{\varepsilon^2} \cdot \min\left\{\frac{\log(n/\varepsilon^2)}{\varepsilon}, \log(n/\varepsilon^2) + \frac{1}{\varepsilon^2} \cdot \log(1/\varepsilon)\right\}\right)$$

5.2 Total Complexity

The total complexity of the algorithm occurs from the n distinct iterations required and depends on the algorithm chosen to find the ε'-close pairs, since both of the presented algorithms dominate the time of the rest of the operations. Thus, by choosing the fastest one (depending on the relationship between n and ε), the final complexity is

$$\mathcal{O}\left(\min\left\{\frac{n^{2.3}}{\varepsilon^{2.6}} \cdot \log(n/\varepsilon^2), \frac{n^2}{\varepsilon^3} \cdot \log(n/\varepsilon^2), \frac{n^2}{\varepsilon^2}\left(\log(n/\varepsilon^2) + \frac{1}{\varepsilon^2} \cdot \log(1/\varepsilon)\right)\right\}\right)$$

6 Conclusion and Future Work

The main contribution of this paper, apart from the introduction of a new FPTAS for the Subset Sum Ratio problem, is the establishment of a connection between Subset Sum and approximating Subset Sum Ratio. In particular, we showed that any improvement over the classic meet in the middle algorithm [16] for Subset Sum, or over the approximation scheme for Subset Sum will result in an improved FPTAS for Subset Sum Ratio.

Additionally, we establish that the complexity of approximating Subset Sum Ratio, expressed in the form $\mathcal{O}((n + 1/\varepsilon)^c)$ has an exponent $c < 5$, which is an improvement over all the previously presented FPTASs for the problem.

It is important to note however, that there is a distinct limit to the complexity that one may achieve for the Subset Sum Ratio problem using the techniques discussed in this paper.

As a direction for future research, we consider the notion of the *weak* approximation of Subset Sum, as discussed in [9,26], which was used in order to approximate the slightly easier Partition problem, and may be able to replace the approximate Subset Sum algorithm in the computation of the ε'-close sets.

Another possible direction could be the use of exact Subset Sum algorithms parameterized by a *concentration* parameter β, as described in [3,4], where they solve the decision version of Subset Sum. See also [15] for a use of this parameter under a pseudopolynomial setting. It would be interesting to investigate whether analogous arguments could be used to solve the optimization version.

References

1. Abboud, A., Bringmann, K., Hermelin, D., Shabtay, D.: Seth-based lower bounds for subset sum and bicriteria path. In: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, 6–9 January 2019, pp. 41–57. SIAM (2019). https://doi.org/10.1137/1.9781611975482.3
2. Antonopoulos, A., Pagourtzis, A., Petsalakis, S., Vasilakis, M.: Faster algorithms for k-subset sum and variations. In: J., Chen, Li, M., Zhang, G. (eds.): Frontiers of Algorithmics - International Joint Conference, IJTCS-FAW 2021, Beijing, 16–19 August 2021, Proceedings. LNCS, vol. 12874, pp. 37–52. Springer (2021). https://doi.org/10.1007/978-3-030-97099-4_3
3. Austrin, P., Kaski, P., Koivisto, M., Nederlof, J.: Subset sum in the absence of concentration. In: 32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, 4–7 March 2015, Garching. LIPIcs, vol. 30, pp. 48–61. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2015). https://doi.org/10.4230/LIPIcs.STACS.2015.48
4. Austrin, P., Kaski, P., Koivisto, M., Nederlof, J.: Dense subset sum may be the hardest. In: 33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, 17–20 February 2016. https://doi.org/10.4230/LIPIcs.STACS.2016.13
5. Bazgan, C., Santha, M., Tuza, Z.: Efficient approximation algorithms for the SUBSET-SUMS EQUALITY problem. J. Comput. Syst. Sci. **64**(2), 160–170 (2002). https://doi.org/10.1006/jcss.2001.1784
6. Bellman, R.E.: Dynamic Programming. Princeton University Press, Princeton (1957)
7. Bringmann, K.: A near-linear pseudopolynomial time algorithm for subset sum. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017. pp. 1073–1084. SIAM, Philadelphia (2017). https://doi.org/10.1137/1.9781611974782.69
8. Bringmann, K., Nakos, V.: Top-k-convolution and the quest for near-linear output-sensitive subset sum. In: Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, pp. 982–995. ACM, New York (2020). https://doi.org/10.1145/3357713.3384308
9. Bringmann, K., Nakos, V.: A fine-grained perspective on approximating subset sum and partition. In: Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, 10–13 January 2021, pp. 1797–1815. SIAM (2021). https://doi.org/10.1137/1.9781611976465.108
10. Cieliebak, M., Eidenbenz, S.J.: Measurement errors make the partial digest problem np-hard. In: LATIN 2004 Theoretical Informatics. 6th Latin American Symposium Lecture Notes in Computer Science, vol. 2976, pp. 379–390. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24698-5_42
11. Cieliebak, M., Eidenbenz, S.J., Pagourtzis, A.: Composing equipotent teams. In: Fundamentals of Computation Theory. 14th International Symposium, FCT 2003 Lecture Notes in Computer Science, vol. 2751, pp. 98–108. Springer, Berlin, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45077-1_10
12. Cieliebak, M., Eidenbenz, S.J., Pagourtzis, A., Schlude, K.: On the complexity of variations of equal sum subsets. Nord. J. Comput. **14**(3), 151–172 (2008)
13. Cieliebak, M., Eidenbenz, S.J., Penna, P.: Noisy data make the partial digest problem NP-hard. In: Algorithms in Bioinformatics, Third International Workshop, WABI 2003. Lecture Notes in Computer Science, vol. 2812, pp. 111–123. Springer, Berlin, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39763-2_9

14. Cygan, M., Mucha, M., Wegrzycki, K., Wlodarczyk, M.: On problems equivalent to (min, +)-convolution. ACM Trans. Algorithms **15**(1), 14:1–14:25 (2019). https://doi.org/10.1145/3293465
15. Dutta, P., Rajasree, M.S.: Efficient reductions and algorithms for variants of subset sum. CoRR abs/2112.11020 (2021). https://arxiv.org/abs/2112.11020
16. Horowitz, E., Sahni, S.: Computing partitions with applications to the knapsack problem. J. ACM **21**(2), 277–292 (1974). https://doi.org/10.1145/321812.321823
17. Jin, C., Wu, H.: A simple near-linear pseudopolynomial time randomized algorithm for subset sum. In: 2nd Symposium on Simplicity in Algorithms (SOSA 2019), vol. 69, pp. 17:1–17:6 (2018). https://doi.org/10.4230/OASIcs.SOSA.2019.17
18. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of Computer Computations. The IBM Research Symposia Series, pp. 85–103. Springer, Boston (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
19. Kellerer, H., Mansini, R., Pferschy, U., Speranza, M.G.: An efficient fully polynomial approximation scheme for the subset-sum problem. J. Comput. Syst. Sci. **66**(2), 349–370 (2003). https://doi.org/10.1016/S0022-0000(03)00006-0
20. Khan, M.A.: Some problems on graphs and arrangements of convex bodies (2017). https://doi.org/10.11575/PRISM/10182
21. Koiliaris, K., Xu, C.: Faster pseudopolynomial time algorithms for subset sum. ACM Trans. Algorithms **15**(3), 401–4020 (2019). https://doi.org/10.1145/3329863
22. Lipton, R.J., Markakis, E., Mossel, E., Saberi, A.: On approximately fair allocations of indivisible goods. In: Proceedings of the 5th ACM Conference on Electronic Commerce (EC-2004). pp. 125–131. ACM, New York (2004). https://doi.org/10.1145/988772.988792
23. Melissinos, N., Pagourtzis, A.: A faster FPTAS for the subset-sums ratio problem. In: Computing and Combinatorics - 24th International Conference, COCOON 2018. Lecture Notes in Computer Science, vol. 10976, pp. 602–614. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94776-1_50
24. Melissinos, N., Pagourtzis, A., Triommatis, T.: Approximation schemes for subset sum ratio problems. In: Li, M. (ed.) FAW 2020. LNCS, vol. 12340, pp. 96–107. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59901-0_9
25. Mucha, M., Nederlof, J., Pawlewicz, J., Wegrzycki, K.: Equal-subset-sum faster than the meet-in-the-middle. In: 27th Annual European Symposium on Algorithms, ESA 2019. LIPIcs, vol. 144, pp. 73:1–73:16 (2019). https://doi.org/10.4230/LIPIcs.ESA.2019.73
26. Mucha, M., Wegrzycki, K., Wlodarczyk, M.: A subquadratic approximation scheme for partition. In: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, 6–9 January 2019, pp. 70–88. SIAM (2019). https://doi.org/10.1137/1.9781611975482.5
27. Nanongkai, D.: Simple FPTAS for the subset-sums ratio problem. Inf. Process. Lett. **113**(19–21), 750–753 (2013). https://doi.org/10.1016/j.ipl.2013.07.009
28. Papadimitriou, C.H.: On the complexity of the parity argument and other inefficient proofs of existence. J. Comput. Syst. Sci. **48**(3), 498–532 (1994). https://doi.org/10.1016/S0022-0000(05)80063-7
29. Pisinger, D.: Linear time algorithms for knapsack problems with bounded weights. J. Algorithms **33**(1), 1–14 (1999). https://doi.org/10.1006/jagm.1999.1034
30. Voloch, N.: Mssp for 2-d sets with unknown parameters and a cryptographic application. Contemp. Eng. Sci. **10**, 921–931 (2017). https://doi.org/10.12988/ces.2017.79101
31. Woeginger, G.J., Yu, Z.: On the equal-subset-sum problem. Inf. Process. Lett. **42**(6), 299–302 (1992). https://doi.org/10.1016/0020-0190(92)90226-L