Simple FPTAS for the subset-sums ratio problem

Article *in* Information Processing Letters · September 2013

DOI: 10.1016/j.ipl.2013.07.009

Danupon Nanongkai
KTH Royal Institute of Technology

Simple FPTAS for the Subset-Sums Ratio Problem*

Danupon Nanongkai†
Nanyang Technological University (NTU), Singapore

Abstract

In the *subset-sums ratio* problem, we want to find two sets of numbers such that the ratio between the larger and smaller sets (in terms of the summed values) is as small as possible. We show a new Fully Polynomial-Time Approximation Scheme (FPTAS) for this problem which simplifies the algorithm proposed by Bazgan et al. [J. Comput. Syst. Sci. 2002]. The key insight of the new algorithm is to solve the problem with an additional constraint that the output sets must contain some certain numbers. While the new problem is harder (in the sense that the original problem can be reduced to this problem), it still admits an FPTAS, which is surprisingly simpler than the FPTAS of the original problem.

1 Introduction

We consider the following problem.

Subset-sums ratio problem Given n positive integers, $a_1 \leq \cdots \leq a_n$, find two disjoint nonempty subsets $S_1, S_2 \subseteq \{1, \ldots, n\}$ with $\sum_{i \in S_1} a_i \geq \sum_{j \in S_2} a_j$, such that the ratio

$$\frac{\sum_{i \in S_1} a_i}{\sum_{j \in S_2} a_j}$$

is minimized.

Motivated by a decision problem called *subset-sums equality* which is NP-hard, Woeginger and Yu [1] introduced this subset-sums ratio problem

*Preprint (to appear in *Information Processing Letters*).
†Work done while at the University of Vienna, Austria.

and gave a 1.324-approximation algorithm. They left as an open question to decide whether this problem has a polynomial-time approximation scheme (PTAS). The answer to this question was given by Bazgan et al. [2] where it was shown that this problem admits not only a PTAS but also an FPTAS. Many variations of the problem have also been studied (e.g., [3, 4]). Besides being a fundamental research problem, it has also been used as an exercise in textbooks and courses on approximation algorithms (e.g., [5]).

As history reveals, although the problem sounds very simple, its solution is far from trivial. Moreover, its only existing solution is quite complicated as the algorithm and analysis have to deal with different cases and generate a large number of pairs of disjoint sets that satisfy some set of rules.

In this paper, we give a very straightforward algorithm for this problem. The key insight is that, instead of solving this problem directly, we solve a slightly harder problem where we have an additional requirement that some numbers p and q must be in S_1 and S_2.

Restricted Subset-Sums Ratio Problem In this problem, in addition to $a_1, \ldots, a_n$, we are given two integers $1 \leq p < q \leq n$ as part of the input. Our goal is to find S_1 and S_2 as before, but we are restricted to S_1 and S_2 such that $\{\max S_1, \max S_2\} = \{p, q\}$. That is, each of p and q appears as the largest number in S_1 or S_2.

Let $\mathcal{P}(\{a_1, \ldots, a_n\})$ denote the subset-sums ratio problem on an instance $\{a_1, \ldots, a_n\}$. Also let $\mathcal{P}'(\{a_1, \ldots, a_n\}, p, q)$ denote the restricted subset-sums ratio problem on instances $\{a_1, \ldots, a_n\}$, p, and q. Observe a simple fact that $\mathcal{P}(\{a_1, \ldots, a_n\})$ can be solved by solving $\mathcal{P}'(\{a_1, \ldots, a_n\}, p, q)$ for all $1 \leq p < q \leq n$ and output the solution that has smallest ratio. Thus, if the restricted subset-sums ratio problem admits an FPTAS then so does the subset-sums ratio problem. (In particular, if we let (S_1^*, S_2^*) be the optimal solution of $\mathcal{P}(\{a_1, \ldots, a_n\})$ and $p^* = \max S_1^*$ and $q^* = \max S_2^*$, then the $(1+\epsilon)$-approximate solution of $\mathcal{P}'(\{a_1, \ldots, a_n\}, p^*, q^*)$ is $(1+\epsilon)$-approximate in $\mathcal{P}(\{a_1, \ldots, a_n\})$ as well.) The rest of this paper is devoted to showing that the new problem actually admits an FPTAS.

Theorem 1. *For any $0 < \epsilon \leq 1$ there exists a $(1 + \epsilon)$-approximation algorithm for the restricted subset-sums ratio problem that runs in* $\mathrm{poly}(n, 1/\epsilon)$ *time.*

As we will see, showing an FPTAS for the restricted version turns out to be much easier than doing the same thing for the original version. The

Algorithm 1 Compute table T

Initially, let $T_0[x,y] = \emptyset$ for any $na_q \geq x \geq y \geq 0$ except $T_0[a_q, a_p] = (\{q\}, \{p\})$.
for $i = 1, \ldots, q-1$ **do**
  if $i = p$ **then** let $T_i[x,y] = T_{i-1}[x,y]$ for all i, x and y and let $i = i+1$
  for any $na_q \geq x \geq y \geq 0$ **do**
    if $T_{i-1}[x,y] \neq \emptyset$ **then**
      $T_i[x,y] = T_{i-1}[x,y]$.
      Let S_1, S_2 be such that $T_{i-1}[x,y] = (S_1, S_2)$. Let $s_i = \max S_i$.
      if $i < s_1$ **then** $T_i[x + a_i, y] = (S_1 \cup \{i\}, S_2)$
      if $i < s_2$ **then**
        if $y + a_i \leq x$ **then** $T_i[x, y + a_i] = (S_1, S_2 \cup \{i\})$
        else $T_i[y + a_i, x] = (S_2 \cup \{i\}, S_1)$
      end if
    end if
  end for
end for

approach is a standard one which was perhaps first used in [6]: We first show that the problem can be solved in pseudo-polynomial time (cf. Section 2) and then use this pseudo-polynomial time algorithm to get an FPTAS (cf. Section 3). The first part is done using a very basic dynamic programming while the second part only requires a few simple inequalities in the analysis. We note, however, that some of these simple inequalities hold only when we consider the restricted version (e.g. Lemma 3). *This is the main reason that the solution of the new problem is simpler.*

2 Pseudo-polynomial Time Algorithm

In this section we show that $\mathcal{P}'(\{a_1, \ldots, a_n\}, p, q)$ can be solved in $\mathrm{poly}(n, a_q)$ time. One of many ways to solve this easy problem is to define a table $T_i[x,y]$, for any $i < q$ and $x \geq y$. This table keeps one pair of disjoint sets $S_1, S_2 \subseteq \{1, \ldots, i\} \cup \{p, q\}$ such that $\sum_{i \in S_1} a_i = x$, $\sum_{j \in S_2} a_j = y$, and $\{\max S_1, \max S_2\} = \{p, q\}$. This table can be computed easily as in Algorithm 1. (We note that Algorithm 1 may change the value of some entry $T_i[x,y]$ many times. This is not necessary and could be avoided. However, we do not try to avoid this to keep the algorithm description simple.) After this table is computed, we find a pair $x \geq y$ such that $T_{q-1}[x,y] \neq \emptyset$ and x/y is minimum. We output the corresponding sets which yield a ratio of

Algorithm 2 Find x and y that minimizes x/y and $T_{q-1}[x, y] \neq \emptyset$

```
Let MinRatio = a_q/a_p, S*_1 = {q} and S*_2 = {p}.
for any n a_q ≥ x ≥ y ≥ 0 do
    if x/y < MinRatio and T_{q-1}[x, y] ≠ ∅ then
        MinRatio = x/y, S*_1 = S_1 and S*_2 = S_2 where S_1 and S_2 are such
        that T_{q-1}[x, y] = (S_1, S_2)
    end if
end for
return (S*_1, S*_2)
```

x/y. (See Algorithm 2.)

3 FPTAS

To $(1+\epsilon)$-approximate $\mathcal{P}'(\{a_1, \ldots, a_n\}, p, q)$, we use Algorithm 3. In this

Algorithm 3 FPTAS($\{a_1, \ldots, a_n\}$, p, q)

```
if n a_p < a_q then
    return S_1 = {q} and S_2 = {1, ..., p}.
else
    Let δ = (ε/3n) a_p and a'_i = ⌊a_i/δ⌋ for all i.
    Find the optimal solution of P'({a'_1, ..., a'_q}, p, q) in pseudo-
    polynomial time (poly(n, a'_q) time) using the algorithm in Section 2.
    Let (S'_1, S'_2) denote the solution.
    if Σ_{i∈S'_1} a_i ≥ Σ_{j∈S'_2} a_j then
        return S_1 = S'_1 and S_2 = S'_2
    else
        return S_1 = S'_2 and S_2 = S'_1.
    end if
end if
```

algorithm, we first deal with an easy case where $na_p < a_q$. In this case, the optimal solution is $S_1 = \{q\}$ and $S_2 = \{1, \ldots, p\}$. (This is because the restriction that $\sum_{i \in S_1} a_i \geq \sum_{j \in S_2} a_j$ implies that $q \in S_1$ and $p \in S_2$. Moreover, $\sum_{i \in S_1} a_i$ is smallest when $S_1 = \{q\}$ while $\sum_{j \in S_2} a_j$ is largest when $S_2 = \{1, \ldots, p\}$.)

In the other case where $na_p \geq a_q$ we simply scale all numbers down by a factor of $\delta = \frac{\epsilon}{3n} a_p$ and solve the problem $\mathcal{P}'(\{a'_1, \ldots, a'_q\}, p, q)$ in poly(n, a'_q)

time using the pseudo-polynomial time algorithm. This running time is polynomial in n and $1/\epsilon$ since

$$a'_q \le \frac{a_q}{\delta} = \frac{3na_q}{\epsilon a_p} \le \frac{3n^2}{\epsilon}$$

(the last inequality is because $na_p \ge a_q$). We then return either (S'_1, S'_2) or (S'_2, S'_1), depending on which solution is feasible. This gives a solution with the ratio of $\max\left(\frac{\sum_{i\in S'_1} a_i}{\sum_{j\in S'_2} a_j}, \frac{\sum_{j\in S'_2} a_j}{\sum_{i\in S'_1} a_i}\right)$. We now prove that this solution is $(1+\epsilon)$-approximate.

Theorem 2. *For any $0 < \epsilon \le 1$, let (S^*_1, S^*_2) and (S'_1, S'_2) be the optimal solutions of $\mathcal{P}'(\{a_1, \dots, a_n\}, p, q)$ and $\mathcal{P}'(\{a'_1, \dots, a'_n\}, p, q)$, respectively. Then,*

$$\max\left(\frac{\sum_{i\in S'_1} a_i}{\sum_{j\in S'_2} a_j}, \frac{\sum_{j\in S'_2} a_j}{\sum_{i\in S'_1} a_i}\right) \le (1+\epsilon)\frac{\sum_{i\in S^*_1} a_i}{\sum_{j\in S^*_2} a_j}. \tag{1}$$

We prove this theorem by proving three simple lemmas. The first lemma lists all inequalities we need to prove the next two lemmas.

Lemma 3.

$$a_i/\delta - 1 \le a'_i \le a_i/\delta \quad \text{for any } i. \tag{2}$$

$$n\delta \le \frac{\epsilon}{3}\sum_{j\in S} a_j \quad \text{for any } S \in \{S^*_1, S^*_2, S'_1, S'_2\} \tag{3}$$

Proof. Eq. (2) is because $a'_i = \lfloor a_i/\delta \rfloor$. To prove Eq. (3), let $r = \max_{i\in S} i$. Note that either $r = p$ or $r = q$. In either case, $\delta = \frac{\epsilon}{3n}a_p \le \frac{\epsilon}{3n}a_r$. It follows that $n\delta \le \frac{\epsilon}{3}a_r \le \frac{\epsilon}{3}\sum_{j\in S} a_j$. ☐

Remark We note that Eq. (3) holds only for the case of restricted subset-sums ratio problem since it relies on the fact that either p or q is in S. This is the key that leads to a simple solution.

Lemma 4. $\max\left(\frac{\sum_{i\in S'_1} a_i}{\sum_{j\in S'_2} a_j}, \frac{\sum_{j\in S'_2} a_j}{\sum_{i\in S'_1} a_i}\right) \le \frac{\sum_{i\in S'_1} a'_i}{\sum_{j\in S'_2} a'_j} + \frac{\epsilon}{3}$.

Proof. The lemma follows from the following inequalities.

$$\begin{aligned}\frac{\sum_{i\in S'_1} a_i}{\sum_{j\in S'_2} a_j} &\le \frac{\sum_{i\in S'_1} a'_i}{\sum_{j\in S'_2} a'_j} + \frac{n\delta}{\sum_{j\in S'_2} a_j} && \text{by Eq. (2)}\\ &\le \frac{\sum_{i\in S'_1} a'_i}{\sum_{j\in S'_2} a'_j} + \frac{\epsilon}{3} && \text{by Eq. (3) with } S = S'_2\end{aligned}$$

In exactly the same way, we have $\frac{\sum_{j\in S_2'} a_j}{\sum_{i\in S_1'} a_i} \le \frac{\sum_{j\in S_2'} a_j'}{\sum_{i\in S_1'} a_i'} + \frac{\epsilon}{3}$. This is at most $\frac{\sum_{i\in S_1'} a_i'}{\sum_{j\in S_2'} a_j'} + \frac{\epsilon}{3}$ since $\sum_{i\in S_1'} a_i' \ge \sum_{j\in S_2'} a_j'$. This concludes the lemma. □

Define A^* and B^* as follows. If $\sum_{i\in S_1^*} a_i' \ge \sum_{j\in S_2^*} a_j'$ then we let $A^* = S_1^*$ and $B^* = S_2^*$; otherwise, $A^* = S_2^*$ and $B^* = S_1^*$. Note that

$$\frac{\sum_{i\in S_1'} a_i'}{\sum_{j\in S_2'} a_j'} \le \frac{\sum_{i\in A^*} a_i'}{\sum_{j\in B^*} a_j'} \tag{4}$$

since (A^*, B^*) is a feasible solution for the problem $\mathcal{P}'(\{a_1', \ldots, a_n'\}, p, q)$.

Lemma 5. *For any* $0 < \epsilon \le 1$, $\frac{\sum_{i\in A^*} a_i'}{\sum_{j\in B^*} a_j'} \le \left(1 + \frac{\epsilon}{2}\right) \frac{\sum_{i\in A^*} a_i}{\sum_{j\in B^*} a_j}$.

Proof.

$$\begin{aligned}
\frac{\sum_{i\in A^*} a_i'}{\sum_{j\in B^*} a_j'} &\le \frac{\sum_{i\in A^*} a_i}{\sum_{j\in B^*} a_j - n\delta} && \text{by Eq. (2)}\\
&= \left(1 + \frac{n\delta}{\sum_{j\in B^*} a_j - n\delta}\right) \frac{\sum_{i\in A^*} a_i}{\sum_{j\in B^*} a_j}\\
&\le \left(1 + \frac{\epsilon}{3-\epsilon}\right) \frac{\sum_{i\in A^*} a_i}{\sum_{j\in B^*} a_j} && \text{by Eq. (3) with } S \in \{S_1^*, S_2^*\}\\
&\le \left(1 + \frac{\epsilon}{2}\right) \frac{\sum_{i\in A^*} a_i}{\sum_{j\in B^*} a_j} && \text{since } \epsilon \le 1.
\end{aligned}$$

This completes the lemma. □

Now Eq. (1) in the theorem statement is easy to prove:

$$\begin{aligned}
\max\left(\frac{\sum_{i\in S_1'} a_i}{\sum_{j\in S_2'} a_j}, \frac{\sum_{j\in S_2'} a_j}{\sum_{i\in S_1'} a_i}\right) &\le \frac{\sum_{i\in S_1'} a_i'}{\sum_{j\in S_2'} a_j'} + \frac{\epsilon}{3} && \text{by Lemma 4}\\
&\le \frac{\sum_{i\in A^*} a_i'}{\sum_{j\in B^*} a_j'} + \frac{\epsilon}{3} && \text{by Eq. (4)}\\
&\le \left(1 + \frac{\epsilon}{2}\right) \frac{\sum_{i\in A^*} a_i}{\sum_{j\in B^*} a_j} + \frac{\epsilon}{3} && \text{by Lemma 5}\\
&\le (1+\epsilon) \frac{\sum_{i\in S_1^*} a_i}{\sum_{j\in S_2^*} a_j}
\end{aligned}$$

where the last inequality is because $1 \le \frac{\sum_{i\in A^*} a_i}{\sum_{j\in B^*} a_j} \le \frac{\sum_{i\in S_1^*} a_i}{\sum_{j\in S_2^*} a_j}$.

4 Conclusion

This paper presents a simpler FPTAS for the subset-sums ratio problem. The key idea is to solve this problem through a slightly harder problem called restricted subset-sums ratio problem. We note that the simplicity of our solution comes with a slightly higher running time, and it might be possible to improve this running time without sacrificing the simplicity too much. For example, one possibility is to solve the problem through a less restricted version of the restricted subset-sums ratio problem where we only require that $\max(S_1 \cup S_2) = q$. This will reduce the search space from $O(n^2)$ (for trying all possible values of p and q) to $O(n)$. It is also interesting to see whether the technique in this paper can be extended to other variations of the subset-sums ratio problem (e.g. [3, 4]). Since the time complexities of algorithms in [2] and this paper are quite high ($O(n^5)$ or more), it might be useful to improve this running time.

Acknowledgement The author realized the complication of the existing solution of this problem when he taught *Advanced Algorithms* course in Winter 2011/12 term at the University of Vienna, Austria. He would like to thank the students in this course and especially his teaching assistant, Veronika Loitzenbauer, for pointing this out and providing comments. He would also like to thank the anonymous reviewers for many helpful comments.

References

[1] G. J. Woeginger, Z. Yu, On the Equal-Subset-Sum Problem, Inf. Process. Lett. 42 (6) (1992) 299–302.

[2] C. Bazgan, M. Santha, Z. Tuza, Efficient Approximation Algorithms for the SUBSET-SUMS EQUALITY Problem, J. Comput. Syst. Sci. 64 (2) (2002) 160–170, announced in ICALP 1998.

[3] M. Cieliebak, S. Eidenbenz, A. Pagourtzis, Composing Equipotent Teams, in: FCT, 2003, pp. 98–108.

[4] M. Cieliebak, S. Eidenbenz, A. Pagourtzis, K. Schlude, On the Complexity of Variations of Equal Sum Subsets, Nord. J. Comput. 14 (3) (2008) 151–172.

[5] V. V. Vazirani, Approximation algorithms, Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[6] O. H. Ibarra, C. E. Kim, Fast approximation algorithms for the knapsack and sum of subset problems, J. ACM 22 (4) (1975) 463–468.