Journal of Computer and System Sciences **64**, 160–170 (2002)
doi:10.1006/jcss.2001.1784, available online at http://www.idealibrary.com on IDEAL®

Efficient Approximation Algorithms for the SUBSET-SUMS EQUALITY Problem[1]

Cristina Bazgan

Université Paris-Sud, LRI, bât. 490, F-91405 Orsay, France
E-mail: bazgan@lri.fr

Miklos Santha

CNRS, URA 410, Université Paris-Sud, LRI, bât. 490, F-91405 Orsay, France
E-mail: santha@lri.fr

and

Zsolt Tuza

Computer and Automation Institute, Hungarian Academy of Sciences,
H-1111 Budapest, Kende u. 13-17, Hungary
E-mail: tuza@sztaki.hu

Received June 3, 1999; revised February 26, 2001

We investigate the problem of finding two nonempty disjoint subsets of a set of n positive integers, with the objective that the sums of the numbers in the two subsets be as close as possible. In two versions of this problem, the quality of a solution is measured by the ratio and the difference of the two partial sums, respectively. Answering a problem of G. J. Woeginger and Z. Yu (1992, *Inform. Process. Lett.* **42**, 299–302) in the affirmative, we give a fully polynomial-time approximation scheme for the case where the value to be optimized is the ratio between the sums of the numbers in the two sets. On the other hand, we show that in the case where the value of a solution is the positive difference between the two partial sums, the problem is not 2^{n^k}-approximable in polynomial time unless $P = NP$, for any constant k. In the positive direction, we give a polynomial time algorithm that finds two subsets for which the difference of the two sums does not exceed $K/n^{\Omega(\log n)}$, where K is the greatest number in the instance. © 2002 Elsevier Science (USA)

[1] This research was supported by the ESPRIT Working Group Rand2 21726 and by the bilateral project Balaton, Grants 97140 (APAPE, France) and F-36/96 (TéT Alapítvány, Hungary).

1. INTRODUCTION

KNAPSACK is a well-known problem which was shown to be NP-complete in 1972 by Karp [3]. It remains NP-complete even if the size of each object is equal to its value. This particular case is called SUBSET-SUM problem. Ibarra and Kim [2] gave a fully polynomial-time approximation scheme for the optimization problem associated with KNAPSACK which, therefore, applies to SUBSET-SUM as well. The most efficient fully polynomial-time approximation scheme known for the SUBSET-SUM problem is due to Kellerer *et al.* [4]. The running time of their algorithm is $O(\min\{n/\varepsilon, n+(1/\varepsilon)^2\log(1/\varepsilon)\})$, and the space required is $O(n+1/\varepsilon)$, where n is the number of the integers and ε the accuracy.

The input to an instance of SUBSET-SUM is a set of n positive integers $a_1, ..., a_n$ and another positive integer b. The question is to decide if there exists a subset of $\{a_1, ..., a_n\}$ whose sum is equal to b. In the optimization version the goal is to find a set of numbers whose sum is as large as possible under the constraint that it does not exceed b.

Woeginger and Yu [7] introduced a related problem called SUBSET-SUMS EQUALITY. Given n positive integers, the question is to decide if there exist two disjoint nonempty subsets whose sums are equal. They also defined a related optimization problem that we call SUBSET-SUMS RATIO; it requires one to find two disjoint subsets with the ratio of their sums being as close to 1 as possible. In the same paper they proved the NP-completeness of SUBSET-SUMS EQUALITY and gave a polynomial-time 1.324-approximation algorithm for SUBSET-SUMS RATIO. They left it as an open question to decide whether this problem has a polynomial-time approximation scheme.

In this paper we answer their question in the affirmative by showing the stronger assertion that actually SUBSET-SUMS RATIO has a *fully polynomial-time* approximation scheme.

The problems defined by Woeginger and Yu have some interesting special instances. Consider the case where the sum of the n numbers is less than $2^n - 1$. It is immediately seen by the pigeonhole principle that there always exist two disjoint nonempty subsets whose sums are equal. Nonetheless, no polynomial-time algorithm is known so far to find two such subsets effectively. We call this latter problem PIGEONHOLE SUBSET-SUMS. This problem is a well-known member of what Meggido and Papadimitriou [5, 6] call the class TFNP of total functions. This class contains function problems associated with languages in NP where, for every instance of the problem, a solution is guaranteed to exist. Other examples in the class are FACTORING, SECOND HAMILTONIAN CYCLE, and HAPPYNET.

Many functions in TFNP (like the examples quoted above) have a challenging intermediate status between FP and FNP, the function classes associated with P and NP. Although these problems are not NP-hard unless $NP = co\text{-}NP$, no polynomial-time algorithm is known for them.

Although the polynomial-time solvability of PIGEONHOLE SUBSET-SUMS still remains open, we will show that in a sense this problem is more approximable in polynomial time than SUBSET-SUMS EQUALITY. For this purpose, we define a further related optimization problem that we call SUBSET-SUMS DIFFERENCE. Here the value of a solution is the positive difference between the sums of the two sets plus 1. The same problem, with the additional constraint that the sum of the numbers is less than $2^n - 1$, is called PIGEONHOLE SUBSET-SUMS DIFFERENCE.

The existence of a fully polynomial-time approximation scheme for SUBSET-SUMS RATIO implies that, for any constant k, there is a polynomialtime $2^n/n^k$-approximation algorithm for PIGEONHOLE SUBSET-SUMS DIFFERENCE. We will show an even stronger result, giving a polynomial-time $2^n/n^{\Omega(\log n)}$-approximation for this problem. This will follow from a more general theorem: we will show that SUBSET-SUMS DIFFERENCE has a polynomial-time $K/n^{\Omega(\log n)}$-approximation algorithm where K is the largest number in the input. On the other hand, we also present a negative result for SUBSET-SUMS DIFFERENCE, proving that it is not 2^{n^k}-approximable in polynomial time unless $P = NP$, for any constant k.

Showing that PIGEONHOLE SUBSET-SUMS (a total function) is better approximable than the corresponding NP search problem is somewhat analogous to the result we have obtained in [1]. There we have shown that there is a polynomial-time approximation scheme for finding another Hamiltonian cycle in cubic Hamiltonian graphs if a Hamiltonian cycle is given in the input (again a total function). On the other hand, finding the longest cycle is not even constant approximable in cubic Hamiltonian graphs, unless $P = NP$.

The paper is organized as follows. In Section 2 we give the necessary definitions. In Section 3 we describe a fully polynomial-time approximation scheme for SUBSET-SUMS RATIO and in Section 4 we prove our results on SUBSET-SUMS DIFFERENCE.

2. PRELIMINARIES

Let us recall a few notions concerning approximability. Given an instance I of an optimization problem A, and a feasible solution y of I, we denote by $m(I, y)$ the value of the solution y and by $opt_A(I)$ the value of an optimum solution of I. The *performance ratio* of y is

$$R(I, y) = \max\left\{\frac{m(I, y)}{opt_A(I)}, \frac{opt_A(I)}{m(I, y)}\right\}.$$

For a constant $c > 1$, an algorithm is a *c-approximation* if, for any instance I of the problem, it returns a solution y such that $R(I, y) \leqslant c$. We say that an optimization problem is *constant approximable* if it admits a polynomial-time c-approximation for some $c < 1$. An optimization problem has a *polynomial-time approximation scheme* (a *ptas*, for short) if, for every constant $\varepsilon > 0$, there exists a polynomial-time $(1+\varepsilon)$-approximation for it. An optimization problem has a *fully polynomial-time approximation scheme* (an *fptas*, for short) if, for every constant $\varepsilon > 0$, there exists an $(1+\varepsilon)$-approximation algorithm for it which is polynomial both in the size of the input and in $1/\varepsilon$. The set of problems having an fptas is denoted by FPTAS.

An algorithm for a problem is called *pseudo-polynomial* if its running time is polynomial in the size of the input and in the unary representation of the largest number occurring in the input.

Let us now give the formal definitions of the problems to be investigated.

Subset-Sums Equality

Input: A set $\{a_1, ..., a_n\}$ of positive integers.

Question: Are there two disjoint nonempty subsets $S_1, S_2 \subseteq \{1, ..., n\}$ such that

$$\sum_{i \in S_1} a_i = \sum_{i \in S_2} a_i?$$

Pigeonhole Subset-Sums

Input: A set $\{a_1, ..., a_n\}$ of positive integers such that $\sum_{i=1}^{n} a_i < 2^n - 1$.

Output: Two disjoint nonempty subsets $S_1, S_2 \subseteq \{1, ..., n\}$ such that

$$\sum_{i \in S^1} a_i = \sum_{i \in S_2} a_i.$$

Subset-Sums Ratio

Input: A set $\{a_1, ..., a_n\}$ of positive integers.

Output: Two disjoint nonempty subsets $S_1, S_2 \subseteq \{1, ..., n\}$ with

$$\sum_{i \in S_1} a_i \geqslant \sum_{i \in S_2} a_i$$

such that the ratio

$$\frac{\sum_{i \in S_1} a_i}{\sum_{i \in S_2} a_i},$$

termed the value of the output, is minimized.

Subset-Sums Difference

Input: A set $\{a_1, ..., a_n\}$ of positive integers.

Output: Two disjoint nonempty subsets $S_1, S_2 \subseteq \{1, ..., n\}$, with

$$\sum_{i \in S_1} a_i \geqslant \sum_{i \in S_2} a_i$$

such that the difference

$$\sum_{i \in S_1} a_i - \sum_{i \in S_2} a_i + 1,$$

the value of the output, is minimized.

Remark. The reason to add 1 in the value of the solution in the above problem is that otherwise the optimum value might be 0, and the performance ratio could not be defined in that case.

PIGEONHOLE SUBSET-SUMS DIFFERENCE

Input: A set $\{a_1, \ldots, a_n\}$ of positive integers such that $\sum_{i=1}^{n} a_i < 2^n - 1$.

Output: The same as for SUBSET-SUMS DIFFERENCE.

3. SUBSET-SUMS RATIO HAS AN FPTAS

In the first part of this section we give a pseudo-polynomial algorithm for the SUBSET-SUMS RATIO problem that we use afterward to construct an fptas.

3.1. *A pseudo-polynomial algorithm*

We assume that the n numbers are in increasing order, $a_1 < \cdots < a_n$, and we set $B = \sum_{i=1}^{n} a_i$. We are going to give an algorithm that finds an optimum solution in time $O(nB^2)$.

The main part of the algorithm will be dynamic programming procedure. We will fill out (maybe partially) two tables, $t[0..n, 0..B]$ with values in $\{0, 1\}$ and $c[0..n, 0..B]$ whose entries are *subsets* of $\{1, \ldots, n\}$.

The completed parts of the tables will satisfy the following properties for $(i, j) \neq (0, 0)$:

1. $t[i, j] = 1$ if and only if there exists a set $S \subseteq \{1, \ldots, n\}$ with $\sum_{k \in S} a_k = j$, $i \in S$, and $h \notin S$ for all $i < h \leqslant n$.

2. $c[i, j] = S$, where $S \subseteq \{1, \ldots, n\}$ is the subset satisfying the above conditions, if such an S exists; and $S = \varnothing$ otherwise.

We stop this procedure if, for some j, two integers $i_1 \neq i_2$ are found such that $t[i_1, j] = t[i_2, j] = 1$. Actually, the procedure will be stopped when the first (smallest) such j is found. Otherwise the tables will be filled out completely. The procedure is as follows:

```
$t[0, 0] := 1, c[0, 0] := \varnothing$;
for $i = 1$ to $n$ do $t[i, 0] := 0, c[i, 0] := \varnothing$;
for $j = 1$ to $B$ do $t[0, j] := 0, c[0, j] := \varnothing$;
for $j = 1$ to $B$ do
  for $i = 1$ to $n$ do
    if ($j \geqslant a_i$ and $\exists\, k \in \{0, \ldots, i-1\}$ with $t[k, j-a_i] = 1$) then
      $t[i, j] := 1, c[i, j] := c[k, j-a_i] \cup \{i\}$;
    else $t[i, j] := 0, c[i, j] := \varnothing$;
  if ($\exists\, i_1 \neq i_2$ with $t[i_1, j] = t[i_2, j] = 1$) then STOP.
```

If the optimum of the instance is 1, then the procedure is stopped when the smallest integer is found which is the sum of two different subsets. The minimality of the sum ensures that these subsets are in fact disjoint.

Otherwise the tables t and c will be completed and we continue the algorithm. We call an integer $j > 0$ *candidate* if it is the sum of some input elements; that is, if we have $t[i, j] = 1$ for some i. For each candidate j, let $i(j)$ be the (unique) integer with this property. Moreover, for every candidate j, let k_j be the greatest candidate less than j such that $c[i(j), j] \cap c[i(k_j), k_j] = \emptyset$, if there is any. Then the optimum solution is the couple $(c[i(j), j], c[i(k_j), k_j])$ for which j/k_j is minimized.

One can see that the above algorithm is pseudo-polynomial.

3.2. *The fptas*

Similar to the previous algorithm, we start with sorting the input in increasing order; that is, after this preprocessing we have $a_1 < a_2 < \cdots < a_n$.

For $m = 2, \ldots, n$, let us denote by I_m the instance of Subset-Sums Ratio which consists of the m smallest numbers $a_1, \ldots, a_m$. At the top level, the algorithm executes its main procedure on the inputs I_m, for $m = 2, \ldots, n$ and takes as solution the best among the solutions obtained for these instances.

Given any ε in the range $0 < \varepsilon < 1$, we set

$$k(m) = \varepsilon^2 \cdot a_m / (2m).$$

Let $n_0 \leqslant n$ be the greatest integer such that $k(n_0) < 1$. We now describe the algorithm on the instance I_m.

If $m \leqslant n_0$, then we apply the pseudo-polynomial algorithm of the previous subsection to I_m. Since $a_{n_0} \leqslant 2n/\varepsilon^2$, this will take polynomial time.

If $n_0 < m \leqslant n$, then we transform the instance I_m into another one that contains only polynomial-size numbers. Set $a'_i = \lfloor a_i / k(m) \rfloor$ for $i = 1, \ldots, m$. Observe that $a'_m = \lfloor 2m/\varepsilon^2 \rfloor$ is indeed of polynomial size. Let us denote by I'_m the instance of Subset-Sums Ratio that contains the numbers a'_i such that $a'_i \geqslant m/\varepsilon$. Suppose that I'_m contains t numbers, $a'_{m-t+1}, \ldots, a'_m$. Since $\varepsilon < 1$, we have $a'_m \geqslant m/\varepsilon$, and therefore $t > 0$. We will distinguish between two cases according to the value of t.

Case 1: $t = 1$. Let j be the smallest nonnegative integer such that $a_{j+1} + \cdots + a_{m-1} < a_m$. If $j = 0$, then the solution will be $S_1 = \{m\}$ and $S_2 = \{1, \ldots, m-1\}$. Otherwise the solution will be $S_1 = \{j, j+1, \ldots, m-1\}$ and $S_2 = \{m\}$.

Case 2: $t > 1$. We solve (exactly) I'_m, using the pseudo-polynomial algorithm which will take only polynomial time on this instance. Then we distinguish between two cases, depending on the value of the optimum of I'_m.

Case 2a: $opt(I'_m) = 1$. The algorithm returns the solution which realizes this optimum for I'_m.

Case 2b: $opt(I'_m) > 1$. In this case we generate a sufficiently rich collection of pairs of subsets in the following way. We consider 3^{t-1} pairs $P(\bar{v}, m)$, $Q(\bar{v}, m)$ of disjoint sets,

$$P(\bar{v}, m), Q(\bar{v}, m) \subseteq \{m-t+1, ..., m\},$$

parameterized by the vectors

$$\bar{v} = (v_1, ..., v_{t-1}) \in \{0, 1, 2\}^{t-1}.$$

The sets are defined according to the rule

$$\begin{array}{llll} m-t+i \in P(\bar{v}, m) & \text{and} \quad m-t+i \notin Q(\bar{v}, m) & \text{if} & v_i = 1, \\ m-t+i \notin P(\bar{v}, m) & \text{and} \quad m-t+i \in Q(\bar{v}, m) & \text{if} & v_i = 2, \\ m-t+i \notin P(\bar{v}, m) & \text{and} \quad m-t+i \notin Q(\bar{v}, m) & \text{if} & v_i = 0, \end{array}$$

for $1 \leqslant i \leqslant t-1$, and we put m into $P(\bar{v}, m)$. Define $R_1(\bar{v}, m) = P(\bar{v}, m)$ if $\sum_{i \in P(\bar{v}, m)} a_i > \sum_{i \in Q(\bar{v}, m)} a_i$ and $R_1(\bar{v}, m) = Q(\bar{v}, m)$ otherwise. Let $R_2(\bar{v}, m)$ be the other set.

The pair $S_1(\bar{v}, m)$, $S_2(\bar{v}, m)$ is defined as follows. Let j be the smallest nonnegative integer such that

$$\sum_{i \in R_2(\bar{v}, m)} a_i + \sum_{i=j+1}^{m-t} a_i < \sum_{i \in R_1(\bar{v}, m)} a_i.$$

If $j = 0$, then $S_1(\bar{v}, m) = R_1(\bar{v}, m)$ and $S_2(\bar{v}, m) = R_2(\bar{v}, m) \cup \{1, ..., m-t\}$. Otherwise, if $m \in R_1(\bar{v}, m)$, then $S_1(\bar{v}, m) = R_2(\bar{v}, m) \cup \{j, ..., m-t\}$ and $S_2(\bar{v}, m) = R_1(\bar{v}, m)$. In the opposite case, where $m \in R_2(\bar{v}, m)$, we define $S_1(\bar{v}, m) = R_1(\bar{v}, m)$ and $S_2(\bar{v}, m) = R_2(\bar{v}, m) \cup \{j+1, ..., m-t\}$. Finally, we choose a vector $\bar{v} \in \{0, 1, 2\}^{t-1}$ for which the ratio

$$\sum_{i \in S_1(\bar{v}, m)} a_i \Big/ \sum_{i \in S_2(\bar{v}, m)} a_i$$

is minimized. The solution given by the algorithm is then $S_1 = S_1(\bar{v}, m)$ and $S_2 = S_2(\bar{v}, m)$.

THEOREM 1. *The above algorithm yields an* $(1+\varepsilon)$*-approximation, in time polynomial in n and* $1/\varepsilon$.

Proof. The algorithm clearly works in polynomial time whenever the number 3^{t-1} of vectors is polynomial in Case 2b. Since $opt(I'_m) > 1$ in that case, all the 2^t subsets of the set $\{a'_{m-t+1}, ..., a'_m\}$ make up mutually distinct sums. Since

$$a'_m \leqslant 2m/\varepsilon^2,$$

we have

$$\sum_{i=m-t+1}^{m} a'_i < 2m^2/\varepsilon^2.$$

Therefore

$$2^t \leqslant 2m^2/\varepsilon^2,$$

and thus $t \leqslant 2\log(m/\varepsilon)+1$.

We will prove now that the algorithm indeed yields an $(1+\varepsilon)$-approximation. Let m be an integer such that a_m is the greatest element occurring in an optimum solution. Then, clearly, this optimum solution for I_n is optimum for I_m as well. We prove that the algorithm yields an $(1+\varepsilon)$-approximation on the instance I_m.

If $m \leqslant n_0$, then the pseudo-polynomial algorithm yields an optimum solution. Hence, let us suppose that $m > n_0$.

In Case 1, if $j=0$, then the given solution is optimum. If $j>0$, then $a'_j \geqslant m/\varepsilon$ and

$$a_j < (a'_j+1)\,k(m) < \frac{2m}{\varepsilon}\frac{\varepsilon^2 a_m}{2m} = \varepsilon a_m.$$

So in this case,

$$\sum_{i\in S_1} a_i \Big/ \sum_{i\in S_2} a_i \leqslant 1 + a_j/a_m < 1+\varepsilon.$$

In Case 2a, we have

$$\frac{\sum_{i\in S_1} a_i}{\sum_{i\in S_2} a_i} \leqslant \frac{\sum_{i\in S_1} k(m)\cdot(1+a'_i)}{\sum_{i\in S_2} k(m)\cdot a'_i} = 1+\frac{|S_1|}{\sum_{i\in S_2} a'_i} \leqslant 1+\frac{t}{m/\varepsilon} < 1+\varepsilon.$$

In Case 2b, let $S_1 = S_1(\bar{v}, m)$ and $S_2 = S_2(\bar{v}, m)$ for some $\bar{v} \in \{0, 1, 2\}^{t-1}$. If $j=0$, since we add all the other integers $a_1, \ldots, a_{m-t}$ to the smallest set, which remains the smallest one, then S_1, S_2 is an optimum solution among the solutions parameterized by the vector $\bar{v}$. Otherwise, we have

$$\sum_{i\in R_2(\bar{v},m)} a_i + \sum_{i=j+1}^{m-t} a_i < \sum_{i\in R_1(\bar{v},m)} a_i \leqslant \sum_{i\in R_2(\bar{v},m)} a_i + \sum_{i=j}^{m-t} a_i.$$

Therefore

$$\sum_{i\in S_1} a_i \Big/ \sum_{i\in S_2} a_i \leqslant 1 + a_j \Big/ \sum_{i\in S_2} a_i \leqslant 1 + a_j/a_m < 1+\varepsilon. \quad \blacksquare$$

4. SUBSET-SUMS DIFFERENCE

Since Subset-Sums Ratio has a fptas, from the approximation point of view, we cannot distinguish Subset-Sums Equality from Pigeonhole Subset-Sums when the value of a solution is the ratio between the sums of the two sets. The situation changes drastically when a harder problem is considered, where the value of a solution is the difference between the two sums. In this section we show that Pigeonhole Subset-Sums Difference has a polynomial-time $2^n/n^{\Omega(\log n)}$-approximation, and on the other hand Subset-Sums Difference is not 2^{n^k}-approximable in polynomial time unless $P = NP$, for any constant k.

The fptas for Subset-Sums Ratio gives a polynomial-time $2^n/n^k$-approximation for Pigeonhole Subset-Sums Difference when we take $\varepsilon = 1/n^k$. But, in fact, one can do better than that.

Theorem 2. *Subset-Sums Difference has a polynomial-time $K/n^{\Omega(\log n)}$-approximation, where K is the greatest number in the instance.*

Proof. We will describe a polynomial-time algorithm that finds a solution of value at most $K/n^{\Omega(\log n)}$. Since the optimum value of each instance is at least 1 by definition, the assertion will follow.

Let $a_1 < a_2 < \cdots < a_n$ be an instance of Subset-Sums Difference, and let us define $a_0 = 0$. Consider the sequence

$$0 = a_0 < a_1 < a_2 \cdots < a_n = K.$$

Notice that at most $n/3$ of the consecutive differences $a_i - a_{i-1}$ can be as large $3K/n$; that is, at least $2n/3$ differences are similar than $3K/n$. From these differences smaller than $3K/n$, we choose every second one (in the order of their occurrence) and create the sequence

$$a_1^{(1)} < a_2^{(1)} < \cdots < a_{n^{(1)}}^{(1)},$$

to which we adjoin $a_0^{(1)} = 0$. We also set $K^{(1)} = a_{n^{(1)}}^{(1)}$, where $K^{(1)} < 3K/n$ and $n^{(1)} \geqslant n/3$.

We repeat this type of difference selection $t = \lfloor \log_3 n \rfloor$ times, creating the sequences

$$0 = a_0^{(i)} < a_1^{(i)} < a_2^{(i)} < \cdots < a_{n^{(1)}}^{(i)} = K^{(i)}$$

for $i = 2, ..., t$, with $K^{(i)} < 3K^{(i-1)}/n^{(i-1)}$ and $n^{(i)} \geqslant n^{(i-1)}/3$. After that, we still have $n^{(t)} \geqslant n/3^t \geqslant 1$ numbers, from which we select the smallest one, namely $a_1^{(t)}$.

Observe that each number in the sequence of step i represents a signed subset-sum, some of the input elements occurring with "+" and some with "−" (and some missing). The numbers with the same sign specify a subset, and the difference between the sum of the numbers of the "+" subset and the sum of the numbers of the "−" subset is at most $K^{(i)}$.

We are going to show that $K^{(t)} = K/n^{\Omega(\log n)}$. We have

$$K^{(1)} < \frac{3K}{n},$$

and

$$K^{(i)} < \frac{3K^{(i-1)}}{n^{(i-1)}}$$

for $i = 2, ..., t$. Taking the product of these inequalities, we obtain

$$K^{(t)} < \frac{3^{t(t+1)/2} \cdot K}{n^t} = K/n^{\Omega(\log n)}.$$

Since the value of the solution is at most $K^{(t)}$, the statement follows. ∎

COROLLARY 1. *PIGEONHOLE SUBSET-SUMS DIFFERENCE has a polynomial-time* $2^n/n^{\Omega(\log n)}$*-approximation.*

Finally, we show a simple nonapproximability result for SUBSET-SUMS DIFFERENCE which is in strong opposition with the approximability of PIGEONHOLE SUBSET-SUMS DIFFERENCE.

THEOREM 3. *If* $P \neq NP$*, then, for any constant k, SUBSET-SUMS DIFFERENCE is not* 2^{n^k}*-approximable in polynomial time.*

Proof. We prove that if SUBSET-SUMS DIFFERENCE were 2^{n^k}-approximable in polynomial time, then SUBSET-SUMS EQUALITY would admit a polynomial-time algorithm. Given an instance $I = \{a_1, a_2, ..., a_n\}$ of SUBSET-SUMS EQUALITY, we create (in polynomial time) an instance $I' = \{b_1, b_2, ..., b_n\}$ of SUBSET-SUMS DIFFERENCE where $b_i = 2^{n^k} \cdot a_i$. The size of I' is polynomial in the size of I, and clearly I is a positive instance if and only if the value of an optimum solution for I' is 1. Let q denote this optimum value, and let s be the value of the solution for I' given by the 2^{n^k}-approximation algorithm.

We claim that $q = 1$ if and only if $s = 1$. The "if" part is trivial. For the "only if" part, let us suppose that $s > 1$. We have

$$s \leqslant 2^{n^k} \cdot q,$$

because the solution was given by a 2^{n^k}-approximation algorithm. Since every element in I' is a multiple of 2^{n^k}, the value of a solution for I' is either 1 or greater than 2^{n^k}. Therefore, we also have

$$s > 2^{n^k},$$

and thus $q > 1$. ∎

REFERENCES

1. C. Bagzan, M. Santha, and Zs. Tuza, On the approximation of finding a(nother) Hamiltonian cycle in cuboc Hamiltonian graphs, *in* "15th Annual Symposium on Theoretical Aspects of Computer Science," Lecture Notes in Computer Science, Vol. 1373, pp. 276–286, Springer-Verlag, Berlin/New York, 1998.
2. O. H. Ibarra and C. E. Kim, Fast approximation algorithms for the Knapsack and sum of subset problems, *J. Assoc. Comput. Mach.* **22** (1975), 463–468.
3. R. M. Karp, Reducibility among combinatorial problems, *in* "Complexity of Computer Computations" (R. E. Miller and J. W. Thatcher, Eds.), pp. 85–103, 1972.
4. H. Kellerer, R. Mansini, U. Pferschy, and M. G. Speranza, "An Efficient Fully Polynomial Approximation Scheme for the subset-sum Problem," Technical Report 14, Faculty of Economics, University of Graz, 1997; see also "Proceedings of the 8th ISAAC Symposium," Lecture Notes in Computer Science, Vol. 1350, pp. 394–403, Springer-Verlag, Berlin/New York, 1997.
5. N. Megiddo and C. Papadimitriou, On total functions, existence theorems and computational complexity, *Theoret. Comput. Sci.* **81** (1991), 317–324.
6. C. Papadimitriou, On the complexity of the parity argument and other inefficient proofs of existence, *J. Comput. System Sci.* **48** (1994), 498–532.
7. G. J. Woeginger and Z. Yu, On the equal-subset-sum problem, *Inform. Process. Lett.* **42** (1992), 299–302.