Nordic Journal of Computing 14(2008), 151–172.

ON THE COMPLEXITY OF VARIATIONS OF EQUAL SUM SUBSETS

MARK CIELIEBAK
Institute of Theoretical Computer Science, ETH Zürich
8092 Zürich, Switzerland
mark@dreamboxx.com

STEPHAN EIDENBENZ
Los Alamos National Laboratory
CCS-5, MS M997, PO Box 1663, Los Alamos, NM87545
Los Alamos National Laboratory Publication No. LA-UR:04-4791
eidenben@lanl.gov

ARIS T. PAGOURTZIS
School of Electrical and Computer Engineering
National Technical University of Athens
Polytechnioupoli, 15780 Zografou, Athens, Greece
pagour@cs.ntua.gr

KONRAD SCHLUDE
Institute of Theoretical Computer Science, ETH Zürich
8092 Zürich, Switzerland
schlude@inf.ethz.ch

Abstract. The Equal Sum Subsets problem, where we are given a set of positive integers and we ask for two nonempty disjoint subsets such that their elements add up to the same total, is known to be NP-hard. In this paper we give (pseudo-)polynomial algorithms and/or (strong) NP-hardness proofs for several natural variations of Equal Sum Subsets. Among others we present (1) a framework for obtaining NP-hardness proofs and pseudo-polynomial time algorithms for Equal Sum Subsets variations, which we apply to variants of the problem with additional selection restrictions, (2) a proof of NP-hardness and a pseudo-polynomial time algorithm for the case where we ask for two subsets such that the ratio of their sums is some fixed rational $r > 0$, (3) a pseudo-polynomial time algorithm for finding k subsets of equal sum, with $k = O(1)$, and a proof of strong NP-hardness for the same problem with $k = \Omega(n)$, (4) algorithms and hardness results for finding k equal sum subsets under the additional requirement that the subsets should be of equal cardinality.

Our results are a step towards determining the dividing lines between polynomial time solvability, pseudo-polynomial time solvability, and strong NP-completeness of subset-sum related problems.

ACM CCS Categories and Subject Descriptors: F.2 [Analysis of Algorithms and Problem Complexity]

Key words: equal sum subsets, partition, knapsack problems, strong NP-completeness, pseudo-polynomial algorithms

Received August 2004; revised July 2007; accepted October 2008. Communicated by Andrzej Lingas.

1. Introduction

In this paper, we study the complexity of variations of Equal Sum Subsets (ESS), which is defined as follows: Given a set A of n positive integers, are there two nonempty disjoint subsets $X, Y \subseteq A$ such that their elements sum up to the same total?

The problem Equal Sum Subsets is a relaxation of Partition in the sense that we do not require the two subsets to cover all input numbers. Partition is a special case of another well-known problem Subset Sum, where the goal is to find one subset whose elements add up to a particular value; Subset Sum, in turn, is a special case of Knapsack. Equal Sum Subsets can be also seen as a variation of Bin Packing with exactly two bins, if we require that both bins should be filled to the same level and not insist on necessarily using all the elements. An even more related problem is a generalization of Knapsack, called Multiple Knapsack, in which there are several knapsacks instead of only one. Partition, Subset Sum, Knapsack, Multiple Knapsack, and Bin Packing, as well as many variations of them, are all NP-hard problems with numerous applications in production planning and scheduling (see for instance [Martello and Toth 1990] for a survey). They all admit pseudo-polynomial time algorithms that lead to polynomial time approximation schemes for the corresponding optimization versions [Ibarra and Kim 1975],[Lawler 1979], [Kellerer 1999],[Chekuri and Khanna 2005],[de la Vega and Lueker 1981],[Karmarkar and Karp 1982].

Our particular interest in Equal Sum Subsets comes from computational biology, namely from its connection to the Partial Digest problem. We briefly illustrate this connection; more details can be found in [Cieliebak 2003]. In the Partial Digest problem we are given a multiset D of distances and are asked to find coordinates of points on a line such that D is exactly the multiset of all pair-wise distances of these points. Partial Digest is a basic problem from DNA sequencing [Setubal and Meidanis 1997],[Pevzner 2000]. Neither a polynomial-time algorithm nor a proof of NP-completeness is known for this problem. In [Cieliebak *et al.* 2003] it is shown that Partial Digest is NP-hard if some of the distances are missing in the input, which is usually the case for real-life data; the proof of this result is done by reduction from Equal Sum Subsets.

Regarding the complexity of Equal Sum Subsets, it is known that it is NP-complete [Woeginger and Yu 1992] and that it admits a pseudo-polynomial algorithm [Bazgan *et al.* 2002] as explained below. Bazgan, Santha and Tuza [Bazgan *et al.* 2002] study an optimization version of Equal Sum Subsets, which they call Subset-Sums Ratio, in which the ratio of the sums of the two subsets should be as close to 1 as possible. They give a pseudo-polynomial time algorithm for Subset-Sums Ratio which leads to an FPTAS for the problem; the first part of that algorithm is in fact a pseudo-polynomial algorithm for Equal Sum Subsets with time complexity $O(n^2S)$, where S is the sum of all numbers in the input set A. It should be also mentioned that Subset-Sums Ratio was first defined in [Woeginger and Yu 1992] (stated as 'computing the similarity number of a set') and a simple 1.324-approximation algorithm was given there. To the best of our knowledge, no other results concerning Equal Sum Subsets have appeared in the literature.

As observed in [Bazgan *et al.* 2002], an interesting special case of Equal Sum Subsets is defined if we restrict the sum of the n given numbers to be smaller than $2^n - 1$; then at least two of the $2^n - 1$ non-empty subsets of the numbers must have equal sum, hence, the decision version of Equal Sum Subsets becomes trivial. In other words, the search version of Equal Sum Subsets for instances with total sum $< 2^n - 1$ is guaranteed to always have a solution and therefore belongs to the class of function problems TFNP [Meggido and Papadimitriou 1991]; what is quite intriguing is that, while it is unlikely that this problem is FNP-complete (FNP is the function analog of NP), no polynomial-time algorithm for finding the (undoubtedly existing) two equal sum subsets is known [Papadimitriou 1994].

In order to better understand Equal Sum Subsets as a combinatorial problem, we study different natural variations of the problem, which we introduce in the following. A summary of our results is shown in Table I.

1.1 Variations of two equal sum subsets

In some settings, it is required that the two subsets fulfil additional requirements. One such requirement is that the subsets have to respect a given set of exclusions, for instance if we want to find groups of people for medical experiments that fulfil some interaction restrictions. This yields the following problem definition, where $\text{sum}(S) = \sum_{x \in S} x$ denotes the sum of all elements in set S.

Definition 1. (ESS with Exclusions) *Given a set A of n positive integers and an* exclusion graph $G_{ex} = (A, E_{ex})$*, are there two nonempty disjoint subsets $X, Y \subseteq A$ with sum $(X) =$ sum (Y) such that each of the two sets is an independent set in G_{ex}?*

The problem ESS with Exclusions is obviously NP-complete, since Equal Sum Subsets is its special case where the exclusion graph is empty. We give a pseudo-polynomial time algorithm for this problem in Section 3.1, using a dynamic programming approach similar-in-spirit to the one used for finding two equal sum subsets (without exclusions) [Bazgan *et al.* 2002].

If we do not want to exclude elements, but on the contrary we want to ensure that some numbers of the input *occur* in the subsets, then this yields the following two problems: In ESS with Enforced Element we enforce one element of the input numbers, say the last, to be in one of the subsets; in Alternating Equal Sum Subsets we have for each input number a “partner”, and if a number occurs in one set, then its partner has to be in the other set. This problem is the “partial” equivalent of a variation of Partition [Garey and Johnson 1979, p. 223] which we call Alternating Partition.

We show in Sections 3.2 and 3.3 that both problems above are NP-complete, by reducing Alternating Partition to the former and Equal Sum Subsets to the latter, respectively; we present pseudo-polynomial time algorithms for both problems. We would like to point out at this time that the reductions for showing NP-completeness are not too involved; however, we present them in this work for two reasons: first, to make the reader become familiar with our vector representation for large numbers (see below); and second, because the technique used can serve as a framework for obtaining NP-hardness proofs for several other variations of Equal Sum Subsets. In fact, Cieliebak [Cieliebak 2003] has used this framework to prove NP-completeness for several variants of Equal Sum Subsets where the solution elements can be selected from two different sets, instead of one.

We then study the relaxation of Equal Sum Subsets that asks for two subsets such that the *ratio* of their sums is exactly r, for some fixed rational $r > 0$. This problem, that we refer to as Factor–r Sum Subsets, is very closely related to the minimization version of Equal Sum Subsets studied in [Bazgan *et al.* 2002]. In Section 4, we show that Factor–r Sum Subsets is NP-complete for any rational factor $r > 0$, by giving two reductions from Exact 3–SAT: one that works for all $r > 0$ with $r \notin \{1, 2, \frac{1}{2}\}$, and one that works for the cases $r = 2$ and $r = \frac{1}{2}$. The case $r = 1$ is just Equal Sum Subsets. We also present a pseudo-polynomial time algorithm for this problem too.

1.2 Variations of k equal sum subsets for $k > 2$

In the second part of the paper, we turn to the generalization of Equal Sum Subsets where we do not ask for only two, but for k disjoint equal sum subsets from a given set of numbers, for a given $k > 2$:

Definition 2. (k Equal Sum Subsets) *Given a multiset of n positive integers $A = \{a_1, \ldots, a_n\}$, are there k nonempty disjoint subsets $S_1, \ldots, S_k \subseteq \{a_1, \ldots, a_n\}$ such that sum$(S_1) = \ldots =$ sum(S_k)?*

Observe that we allow multisets here, in contrast to Equal Sum Subsets, which becomes trivial if any number occurs more than once in the input. Note also that if we require that our subsets yield a full partition of the given numbers, our problem would turn into a variation of Partition with k sets instead of 2.

We first show in Section 5.1 that k Equal Sum Subsets is NP-complete for any integer $k > 2$, by giving a reduction from Alternating Partition. Then we study the influence of parameter k on the complexity of k Equal Sum Subsets in more depth. We have introduced parameter k for the number of equal size subsets as a fixed constant that is part of the problem definition. An interesting variation is to allow k to be a (fixed) function of the number of input elements n, e.g. $k = \frac{n}{q}$ for some constant q. In the sequel, we will always consider k as a function of n; whenever k is a constant we simply write $k = O(1)$. In Section 5.2, we present a dynamic programming algorithm for k Equal Sum Subsets with running time $O(\frac{nS^k}{k^{k-1}})$, where n is the cardinality of the input set and S is the sum of all numbers in the input set; the algorithm runs in pseudo-polynomial time for $k = O(1)$. On the other hand, we show that k Equal Sum Subsets is strongly NP-complete for $k = \Omega(n)$. We obtain this result by giving a reduction from 3–Partition.

We then return to the case where k is a fixed constant. The definition of k Equal Sum Subsets corresponds to the situation in which it is allowed to form subsets that do not have the same number of elements. In some cases, this makes sense; however, we may also wish to have the same number of elements in each subset. Such problems occur for instance when we are given a set of, say, soccer players, together with values that represent their strength, and we want to compose teams of equal strength and size to play a tournament. We study three variations of k Equal Sum Subsets with *equal* cardinalities, where either the cardinality is a fixed constant, i.e., part of the problem definition (k ESS of Cardinality c), or we specify the cardinality of the subsets in the input (k ESS of Specified Cardinality), or we only ask for subsets of equal cardinality, but do not specify their cardinality at all (k ESS of Equal Cardinality).

In Section 5.3, we first observe that there is a simple polynomial time algorithm for k ESS of Cardinality c that uses exhaustive search and runs in time $O(n^{kc})$, which is polynomial in n, since the two parameters k and c are fixed constants. We then show that, on the other hand, k ESS of Specified Cardinality and k ESS of Equal Cardinality are NP-complete. To establish this result, we give a reduction from Alternating Partition to the first problem; a similar reduction can be used to prove NP-completeness for the second problem. In addition, we show that none of these two problems can be strongly NP-complete, by presenting a dynamic programming algorithm that solves them in pseudo-polynomial time.

2. Notation

We do not distinguish between sets and multisets in our notation, and denote a multiset with elements $1, 1, 3, 5, 5$, and 8 by $\{1, 1, 3, 5, 5, 8\}$. Subtracting an element from a multiset will remove it only once (if it is there), thus $\{1, 1, 3, 5, 5, 8\} - \{1, 4, 5, 5\} = \{1, 3, 8\}$. For a set or multiset S we denote by $|S|$ the cardinality of S, e.g. $|\{1, 1, 3, 5, 5, 8\}| = 6$. We denote by $\text{sum}(S)$ the sum of all elements in a set or multiset S, i.e., $\text{sum}(S) = \sum_{x\in S} x$. E.g. $\text{sum}(\{1, 1, 3, 5, 5, 8\}) = 23$. For simplicity, we slightly abuse standard terminology and call two subsets X, Y of a given multiset $A = \{a_1, \dots, a_n\}$ *disjoint* if there are disjoint (in the usual sense) subsets of indices $I_X, I_Y \subseteq \{1, \dots, n\}$ such that $X = \cup_{i\in I_X}\{a_i\}$ and $Y = \cup_{i\in I_Y}\{a_i\}$; note that X and Y may be multisets as well.

We introduce a vector representation for large numbers that will allow us to add up numbers digit by digit, like polyadic numbers. The numbers are expressed in the number system of some base Z. We denote by $\langle a_1, \dots, a_n\rangle$ the number $\sum_{1\le i\le n} a_i Z^{n-i}$; we say that a_i is the i-th digit of this number. In our proofs, we will choose base Z large enough such that the additions that we will perform do not lead to carry-overs from one digit to the next. Hence, we can add numbers digit by digit. The same holds for scalar multiplications. For example, having base $Z = 27$ and numbers $\alpha = \langle 3, 5, 1\rangle, \beta = \langle 2, 1, 0\rangle$, then $\alpha + \beta = \langle 5, 6, 1\rangle$ and $3 \cdot \alpha = \langle 9, 15, 3\rangle$. We define the concatenation of two numbers by $\langle a_1, \dots, a_n\rangle \circ \langle b_1, \dots, b_m\rangle := \langle a_1, \dots, a_n, b_1, \dots, b_m\rangle$, i.e., $\alpha \circ \beta = \alpha Z^m + \beta$, where m is the number of digits in β. Let $\Delta_n(i) := \langle 0, \dots, 0, 1, 0, \dots, 0\rangle$ be the number that has n digits, all 0's except for the i-th position, where the digit is 1. Moreover, $\mathbb{1}_n := \langle 1, \dots, 1\rangle$ has n digits, all 1's, and $\mathbb{0}_n := \langle 0, \dots, 0\rangle$ has n zeros. Notice that $\mathbb{1}_n = \frac{Z^n-1}{Z-1}$.

TABLE I: A summary of our results. All results are new, except those of the first line which concern our archetypical problem EQUAL SUM SUBSETS.

Equal Sum Subsets (ESS) Variations		
Problem	Hardness Result	Complexity Upper Bound
EQUAL SUM SUBSETS (ESS)	NP-hard [Woeginger and Yu 1992], pseudo-poly algorithm [Bazgan *et al.* 2002]	$O(n^2 \cdot S)$
ESS WITH EXCLUSIONS	NP-hard, pseudo-poly algorithm	$O(n^2 \cdot S)$
ESS WITH ENFORCED ELEMENT	NP-hard, pseudo-poly algorithm	$O(n^2 \cdot S)$
ALTERNATING EQUAL SUM SUBSETS	NP-hard, pseudo-poly algorithm	$O(n^2 \cdot S^2)$
FACTOR–r SUM SUBSETS	NP-hard, pseudo-poly algorithm for any rational $r > 0$	$O(n^2 \cdot S^2)$
k EQUAL SUM SUBSETS (k ESS)	NP-hard, pseudo-poly algorithm for any fixed constant integer $k > 2$	$O(\frac{nS^k}{k^{k-1}})$
	strongly NP-hard for $k = \Omega(n)$	
k ESS OF CARDINALITY c	in P for any fixed constant integers $k \geq 2$, $c \geq 1$	$O(n^{kc})$
k ESS OF SPECIFIED CARDINALITY	NP-hard, pseudo-poly algorithm for any fixed constant integer $k \geq 2$	$O(\frac{S^k \cdot n^{k+1}}{k^{2k-1}})$
k ESS OF EQUAL CARDINALITY	NP-hard, pseudo-poly algorithm for any fixed constant integer $k \geq 2$	$O(\frac{S^k \cdot n^{k+1}}{k^{2k-1}})$

3. Complexity of Equal Sum Subsets with Selection Constraints

3.1 Element exclusion

In this section, we study variations of EQUAL SUM SUBSETS where we add specific requirements that a solution must fulfil. In particular, we study variations where the two subsets take into account some exclusions or enforcement of specific elements of the input.

We first study the problem ESS WITH EXCLUSIONS, where we are additionally given an *exclusion graph* (or its complement: a *preference graph*) and ask for two subsets of equal sum that take this graph into account (cf. Definition 1). Obviously, ESS WITH EXCLUSIONS is NP-complete, since EQUAL SUM SUBSETS is the special case where the exclusion graph is empty ($E_{ex} = \emptyset$) and EQUAL SUM SUBSETS is NP-complete [Woeginger and Yu 1992]. Here, we present a pseudo-polynomial algorithm for the problem, using a dynamic programming approach similar-in-spirit to the one used for finding two equal sum subsets (without exclusions) [Bazgan *et al.* 2002].

THEOREM 1. ESS WITH EXCLUSIONS *can be solved in pseudo-polynomial time* $O(n^2 \cdot S)$, *where* $S = \text{sum}(A)$.

PROOF. Let $A = \{a_1, \ldots, a_n\}$ and $G_{ex} = (A, E_{ex})$ be an instance of ESS WITH EXCLUSIONS. We assume w.l.o.g. that the input values are in increasing order, i.e., $a_1 \leq \ldots \leq a_n$.

We define Boolean variables $F(k,t)$ for $k \in \{1, \ldots, n\}$ and $t \in \{1, \ldots, S\}$. Variable $F(k,t)$ will be TRUE if there exists a set $X \subseteq A$ such that $X \subseteq \{a_1, \ldots, a_k\}$, $a_k \in X$, $\text{sum}(X) = t$, and X is independent in G_{ex}. For a TRUE entry $F(k,t)$, we store a corresponding set X in a second variable $X(k,t)$.

We compute the value of all variables $F(k,t)$ by iterating over t and k. The algorithm runs until it finds the smallest $t \in \{1, \ldots, S\}$ for which there are two different indices $k, \ell \in \{1, \ldots, n\}$ such that $F(k,t) = F(\ell,t) = \text{TRUE}$; in this case, sets $X(k,t)$ and $X(\ell,t)$ constitute a solution because $\text{sum}(X(k,t)) = \text{sum}(X(\ell,t)) = t$, both sets are disjoint due to minimality of t, and both sets are independent in G_{ex}.

We initialize the variables as follows. For all $1 \leq k \leq n$, we set $F(k,t) = \text{FALSE}$, for $1 \leq t < a_k$ and for $\sum_{i=1}^{k} a_i < t \leq S$; moreover, we set $F(k, a_k) = \text{TRUE}$ and $X(k, a_k) = \{a_k\}$. Observe that these equations define $F(1,t)$, for $1 \leq t \leq S$, and $F(k,1)$, for $1 \leq k \leq n$.

After initialization, the table entries for $k > 1$ and $a_k < t \leq \sum_{i=1}^{k} a_i$ can be computed recursively: $F(k,t)$ is TRUE if there exists an index $\ell \in \{1, \ldots, k-1\}$ such that $F(\ell, t - a_k)$ is TRUE, and such that the subset $X(\ell, t - a_k)$ remains independent in G_{ex} when adding a_k. The recursive computation is

$$F(k,t) \quad = \quad \bigvee_{\ell=1}^{k-1} [\, F(\ell, t - a_k) \ \wedge\ \forall a \in X(\ell, t - a_k),\ (a, a_k) \notin E_{ex}\,].$$

If $F(k,t)$ is set to TRUE due to $F(\ell, t - a_k)$, then we set $X(k,t) = X(\ell, t - a_k) \cup \{a_k\}$. The key observation for showing correctness is that for each $F(k,t)$ considered by the algorithm there is at most one $F(\ell, t - a_k)$ that is TRUE, for $1 \leq \ell \leq k - 1$; if there were two, say ℓ_1, ℓ_2, then $X(\ell_1, t - a_k)$ and $X(\ell_2, t - a_k)$ would be a solution for the problem instance, and the algorithm would have stopped earlier—a contradiction. This means that all subsets considered are constructed in a unique way, and therefore, no information can be lost.

In order to determine the value $F(k,t)$, the algorithm considers $k-1$ table entries. As shown above, only one of them may be TRUE; for such an entry, say $F(\ell, t - a_k)$, the (at most ℓ) elements of $X(\ell, t - a_k)$ are checked to see if they exclude a_k. Hence, the computation of $F(k,t)$ takes time $O(n)$, and the total time complexity of the algorithm is $O(n^2 \cdot S)$. □

3.2 Element enforcement

If we do not want to exclude elements, but on the contrary, we want to ensure that a specific element of the input occurs in one of the two equal sum subsets, then this is the ESS WITH ENFORCED ELEMENT problem:

DEFINITION 3. (ESS WITH ENFORCED ELEMENT) *Given a set $A = \{a_1, \ldots, a_n\}$ of n positive integers, are there two disjoint subsets $X, Y \subseteq A$ with sum $(X) =$ sum (Y) such that $a_n \in X$?*

We show that this problem is NP-complete by reduction from ALTERNATING PARTITION.

THEOREM 2. ESS WITH ENFORCED ELEMENT *is* NP-*complete.*

PROOF. The problem is obviously in NP. For the proof of NP-hardness, we give a reduction from ALTERNATING PARTITION, which is the following NP-complete variation of PARTITION [Garey and Johnson 1979, p. 223]: Given n pairs of positive integers $(u_1, v_1), \ldots, (u_n, v_n)$, are there two nonempty disjoint sets of indices I and J with $I \cup J = \{1, \ldots, n\}$ such that $\sum_{i\in I} u_i + \sum_{j\in J} v_j = \sum_{i\in I} v_i + \sum_{j\in J} u_j$?

Consider an instance of ALTERNATING PARTITION $(u_1, v_1), \ldots, (u_n, v_n)$. Let $S = \sum_{i=1}^{n}(u_i + v_i)$, $a_i = \langle u_i \rangle \circ \Delta_n(i)$ and $b_i = \langle v_i \rangle \circ \Delta_n(i)$, $1 \le i \le n$, and let $c = \langle \frac{S}{2} \rangle \circ 1_n$. For these numbers, we use base $Z = 2 \cdot S \cdot n$, which is large enough such that no carry-overs from one digit to the next occur in the following additions.

The a_i's, b_i's, and c are an instance of ESS WITH ENFORCED ELEMENT such that c, which is the last element in the input, is the enforced element. We show that there exists a solution for the ALTERNATING PARTITION instance if and only if there exists a solution for the ESS WITH ENFORCED ELEMENT instance.

Assume that index sets I and J are a solution for the ALTERNATING PARTITION instance. Then $\sum_{i\in I} u_i + \sum_{j\in J} v_j = \sum_{i\in I} v_i + \sum_{j\in J} u_j = \frac{S}{2}$. Let $X = \{c\}$ and $Y = \{a_i \mid i \in I\} \cup \{b_j \mid j \in J\}$. Then

$$\begin{aligned}
\text{sum}\,(Y) &= \sum_{i\in I} a_i + \sum_{j\in J} b_j \\
&= \sum_{i\in I}(\langle u_i \rangle \circ \Delta_n(i)) + \sum_{j\in J}(\langle v_j \rangle \circ \Delta_n(j)) \\
&= \langle \sum_{i\in I} u_i + \sum_{j\in J} v_j \rangle \circ (\sum_{i\in I} \Delta_n(i) + \sum_{j\in J} \Delta_n(j)) \\
&= \langle \frac{S}{2} \rangle \circ \sum_{i=1}^{n} \Delta_n(i) \\
&= \langle \frac{S}{2} \rangle \circ 1_n \\
&= \text{sum}\,(X),
\end{aligned}$$

thus, X and Y are a solution for the ESS WITH ENFORCED ELEMENT instance.

For the opposite direction, let X and Y be a solution for the ESS WITH ENFORCED ELEMENT instance with $c \in X$. All numbers in the input have $n + 1$ digits. For each index $i \in \{2, \dots, n + 1\}$, only three numbers, namely c, a_i and b_i, have a 1 in the i'th digit, all other numbers in the input have a 0 in the i'th digit. For each digit the sum over all elements in X and in Y yields the same result. Therefore, since $c \in X$, exactly one of a_i or b_i can be in Y for each $1 \le i \le n$. Moreover, $X = \{c\}$, since any other element would add a second 1 in some digit i, which then could not be equalized by elements in Y. Summing up the first digit of all elements in Y yields exactly the first digit of c, which is $\frac{S}{2}$. Thus, $I := \{i \in \{1, \dots, n\} \mid a_i \in Y\}$ and $J := \{j \in \{1, \dots, n\} \mid b_j \in Y\}$ yield a solution for the ALTERNATING PARTITION instance. □

We now show that ESS WITH ENFORCED ELEMENT is not NP-complete in the strong sense.

THEOREM 3. ESS WITH ENFORCED ELEMENT *can be solved in pseudo-polynomial time* $O(n^2 \cdot S)$, *where* $S = sum(A)$.

PROOF. We present a dynamic programming algorithm similar to the algorithm for ESS WITH EXCLUSIONS. We use the same Boolean variables without the requirement of excluding elements. More specifically, $F(k, t)$ will be TRUE if there exists a set $X \subseteq A$ such that $X \subseteq \{a_1, \dots, a_k\}$, $a_k \in X$, and $\text{sum}(X) = t$.

We compute the value of all variables $F(k, t)$ by iterating first over $t \in \{1, \dots, S\}$, where $S = \text{sum}(A)$ and then over $k \in \{1, \dots, n\}$. The algorithm runs until it finds the smallest t for which there is an index $k \neq n$ such that $F(k, t) = F(n, t) = \text{TRUE}$; in this case, the corresponding sets constitute a solution because they have the same sum, they are disjoint due to minimality of t, and a_n is present in one of them. Note that it is not hard to reconstruct these sets.

Initialization is done as before and the recursive computation is now much simpler:

$$F(k, t) \quad = \quad \bigvee_{\ell=1}^{k-1} F(\ell, t - a_k)\,.$$

As before, the computation of $F(k, t)$ takes time $O(n)$, and the total time complexity of the algorithm is $O(n^2 \cdot S)$. □

3.3 Alternating equal sum subsets

We now turn to the problem ALTERNATING EQUAL SUM SUBSETS, which is the "partial" equivalent of ALTERNATING PARTITION, which we used in the previous proof. In ALTERNATING EQUAL SUM SUBSETS, we are given pairs of numbers and we require for each element that we use in one set that its partner will be in the other set. Formally, the problem is defined as follows:

DEFINITION 4. (ALTERNATING EQUAL SUM SUBSETS) *Given n pairs of positive integers* $(u_1, v_1), \ldots, (u_n, v_n)$*, are there two nonempty disjoint sets of indices I and J such that* $\sum_{i \in I} u_i + \sum_{j \in J} v_j = \sum_{i \in I} v_i + \sum_{j \in J} u_j$*?*

We show that this problem is NP-complete by reduction from EQUAL SUM SUBSETS.

THEOREM 4. ALTERNATING EQUAL SUM SUBSETS *is* NP*-complete.*

PROOF. The problem is obviously in NP. For the proof of NP-hardness, we give a reduction from EQUAL SUM SUBSETS. Given an instance of EQUAL SUM SUBSETS, i.e., a set of numbers $A = \{a_1, \ldots, a_n\}$, we reduce it to an instance of ALTERNATING EQUAL SUM SUBSETS by setting $B = 2 \cdot \sum_{i=1}^{n} a_i$ and mapping each number a_i to a pair (u_i, v_i), with $u_i = B + a_i$ and $v_i = B$. Note that we use offset B only for technical reasons, since all input numbers for ALTERNATING EQUAL SUM SUBSETS are required to be positive. Clearly, if there are nonempty disjoint sets $X, Y \subseteq A$ such that $\mathrm{sum}(X) = \mathrm{sum}(Y)$, then $I := \{i \mid a_i \in X\}$ and $J := \{j \mid a_j \in Y\}$ are disjoint index sets such that $\sum_{i \in I} u_i + \sum_{j \in J} v_j = \sum_{i \in I} v_i + \sum_{j \in J} u_j$. Conversely, if there is a solution for the ALTERNATING EQUAL SUM SUBSETS instance, i.e., appropriate sets of indices I and J, then the sets $X = \{a_i \mid i \in I\}$ and $Y = \{a_j \mid j \in J\}$ form obviously a solution for the EQUAL SUM SUBSETS instance. □

We now show that ALTERNATING EQUAL SUM SUBSETS admits a pseudo-polynomial time algorithm.

THEOREM 5. ALTERNATING EQUAL SUM SUBSETS *can be solved in pseudo-polynomial time* $O(n^2 \cdot S^2)$*, where* $S = \sum_{1 \le i \le n} \max\{u_i, v_i\}$*.*

PROOF. We present a dynamic programming algorithm similar to the algorithms for ESS WITH ENFORCED ELEMENT (Thm. 3) and ESS WITH EXCLUSIONS (Thm. 1).

Let $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$. We use Boolean variables $F(k, t_1, t_2)$ for $k \in \{1, \ldots, n\}$, $t_1, t_2 \in \{1, \ldots, S\}$ which will be TRUE if there exist sets $X, Y \subseteq \{u_1, \ldots, u_k\} \cup \{v_1, \ldots, v_k\}$ such that:

- for all $i \in \{1, \ldots, k-1\}$ it holds $u_i \in X \Leftrightarrow v_i \in Y$ and $v_i \in X \Leftrightarrow u_i \in Y$,
- either $u_k \in X \wedge v_k \in Y$ or $v_k \in X \wedge u_k \in Y$,
- $\mathrm{sum}(X) = t_1$ and $\mathrm{sum}(Y) = t_2$.

The algorithm computes the value of all variables $F(k, t_1, t_2)$ by iterating first over $t_1 \in \{1, \ldots, S\}$, then over $t_2 \in \{1, \ldots, S\}$, and finally over $k \in \{1, \ldots, n\}$. The algorithm runs until it finds a value t for which there is an index k such that $F(k, t, t) = \mathrm{TRUE}$; in this case, the corresponding sets constitute a solution and it is not hard to reconstruct them from the table entries. The variables are initialized in a way similar to the one in the proof of Thm. 1.

The recursive computation is as follows:

$$F(k, t_1, t_2) \quad = \quad \bigvee_{\ell=1}^{k-1} [\, F(\ell, t_1 - u_k, t_2 - v_k) \vee F(\ell, t_1 - v_k, t_2 - u_k) \,].$$

The computation of $F(k, t_1, t_2)$ takes time $O(n)$, and the total time complexity of the algorithm is $O(n^2 \cdot S^2)$. □

4. Complexity of Factor-*r* Sum Subsets

In this section, we study the Factor–*r* Sum Subsets problem, where we ask for two subsets such that the ratio of their sums is r.

Definition 5. (Factor–*r* Sum Subsets) *Given a multiset A of n positive integers, are there two nonempty disjoint subsets $X, Y \subseteq A$ such that sum $(X) = r \cdot$ sum (Y)?*

For $r = 1$, the problem is exactly Equal Sum Subsets and therefore is NP-complete [Woeginger and Yu 1992]. Here we show that Factor–*r* Sum Subsets is NP-complete for *any* fixed rational $r > 0$. The proof of NP-hardness consists of two different reductions from Exact 3–SAT, where the second reduction is just for the cases $r = 2$ and $r = \frac{1}{2}$.

Lemma 1. Factor–*r* Sum Subsets *is* NP-*hard for any rational $r > 0$ with $r \notin \{1, 2, \frac{1}{2}\}$.*

Proof. We present a reduction from Exact 3–SAT, which is an NP-complete restriction of One–in–Three 3–SAT [Garey and Johnson 1979, p. 259] defined as follows: Given a set of m clauses $c_1, \ldots, c_m$ over n Boolean variables $x_1, \ldots, x_n$ such that each clause contains three positive literals, is there a (satisfying) assignment for the variables that satisfies exactly one literal per clause?

Let $r = p/q$, where p and q are positive integers with no common divisor except 1 (coprimes) and $p < q$. (The case $p > q$ is equivalent by interchanging sets X and Y in the problem definition.) We consider several cases, depending on the values of p and q. We only give a detailed proof for the first case; for the other cases the proof is quite similar, so we just mention the construction of the necessary numbers.

Case 1: $p > 3$. Consider an instance of Exact 3–SAT with a set of n variables $V = \{v_1, \ldots, v_n\}$ and a set of m clauses $C = \{c_1, \ldots, c_m\}$. An instance of Factor–*r* Sum Subsets is constructed as follows. For each variable v_i a number $a_i = \sum_{v_i \in c_j} \Delta_m(j)$ is defined. Value a_i has m digits, and its non-zero digits correspond to clauses where v_i appears. Two additional numbers a_{n+1} and a_{n+2} are constructed which are multiples of 1_m: $a_{n+1} = (p-1) \cdot 1_m$ and $a_{n+2} = q \cdot 1_m$. For all numbers we use base $Z = q(p+q+2)+1$. This way we will avoid carry-overs from one digit to the next when adding a_i's. Let $A = \{a_1, \ldots, a_{n+2}\}$. In the following, we show that there is a solution for the Exact 3–SAT instance if and only if there are two nonempty disjoint subsets $X, Y \subseteq A$ such that sum $(X) = r \cdot$ sum (Y).

"only if": Assume that there exists an exact satisfying assignment for the clauses in C. This implies that there exists a subset $R \subseteq \{a_1, \ldots, a_n\}$ such that sum $(R) = 1_m$, since for each clause c_j there is exactly one of the three variables in c_j set to TRUE, say v_k, and the corresponding a_k has a 1 in the j-th digit. We define a set R to contain exactly these a_i's; then sum $(R) = 1_m$. Hence, by setting $X = R \cup \{a_{n+1}\}$

and $Y = \{a_{n+2}\}$, we have $\text{sum}(X) = p \cdot 1_m = r \cdot q \cdot 1_m = r \cdot \text{sum}(Y)$, thus X and Y yield a solution for the Factor–r Sum Subsets instance.

"if": For the opposite direction, assume that non-empty sets X, Y exist such that $\text{sum}(X) = r \cdot \text{sum}(Y)$; equivalently, $q \cdot \text{sum}(X) = p \cdot \text{sum}(Y)$. Observe that summing the i'th digit of all numbers in the input set A yields $p + q + 2$. Moreover, even when multiplying each number in A by q we get only total $q(p + q + 2)$ in the i'th digit, and no carry-overs occur, since we choose base Z sufficiently large. Since $q \cdot \text{sum}(X) = p \cdot \text{sum}(Y)$, we have $qx_i = py_i$, where x_i and y_i are the i'th digit of $\text{sum}(X)$ resp. $\text{sum}(Y)$, for $1 \leq i \leq n$. This implies that for each digit i either $x_i = y_i = 0$, or q divides y_i and p divides x_i (since p and q are coprimes). Observe that not all digits can be 0, since we have assumed that X and Y are non-empty.

We now show that $x_j = p$ and $y_j = q$ for every non-zero digit j: Since p divides x_j and q divides y_j, there exist two positive integers k and ℓ such that $x_j = k \cdot p$ and $y_j = \ell \cdot q$. Then $qx_j = py_j$ implies that $k = \ell$. Moreover, we have $p + q + 2 \geq x_j + y_j = k(p + q)$, hence $2 \geq (k - 1)(p + q)$, and this inequality can only hold for k = 1, since $q > p > 3$ and k is positive. Thus, $x_j = p$ and $y_j = q$.

Since only five numbers in A have non-zero value in the j'th digit, and the corresponding values are $1, 1, 1, p-1$ and q, we can only achieve $x_j = p$ if $X = \{a_{n+1}\} \cup R$, where R is a subset of A such that $\text{sum}(R)$ has a 1 in the j'th digit. Thus, the only way to get $y_j = q$ is to have $Y = \{a_{n+2}\}$. Since a_{n+1} has value $p-1$ in every digit, no digits in $\text{sum}(X)$ can be 0, hence also in $\text{sum}(Y)$. Thus, the variables corresponding to numbers in R form an exact satisfying assignment for the given clauses.

We now sketch the proof for the remaining combinations of values of p and q:

Case 2: $p = 3, q > 4$. Numbers $a_1, \ldots, a_n$ are constructed as in Case 1, $a_{n+1} = 3 \cdot 1_m$, and $a_{n+2} = (q - 1) \cdot 1_m$.

Case 3: $p = 3, q = 4$. Numbers $a_1, \ldots, a_n$ are constructed as in Case 1, $a_{n+1} = 3 \cdot 1_m$, and $a_{n+2} = 2 \cdot 1_m$.

Case 4: $p = 2, q > 3$. Numbers $a_1, \ldots, a_n$ are constructed as in Case 1, and only one additional number $a_{n+1} = (q - 1) \cdot 1_m$ is used.

Case 5: $p = 2, q = 3$. For each variable v_i let $a_i = \sum_{v_i \in c_j} 3 \cdot \Delta_m(j)$, i.e., a_i has a digit 3 in each position that corresponds to a clause that contains v_i. We also set $a_{n+1} = 1_m$. Note that $\text{sum}(A) = 10 \cdot 1_m$. As in Case 1, the direction "only if" is easy: any exact satisfying assignment for the clauses in C corresponds to numbers a_i that add up to $3 \cdot 1_m$, which together with a_{n+1} constitute X. For the "if" direction, we observe that the only way to have the required ratio is by having two sets X and Y such that $\text{sum}(X) = 4 \cdot 1_m$ and $\text{sum}(Y) = 6 \cdot 1_m$; this implies $a_{n+1} \in X$, and for each $j \in \{1, \ldots, m\}$ there is exactly one further number $a_i \in X$ that has non-zero digit j. Hence, the variables corresponding to $X - \{a_{n+1}\}$ constitute an exact satisfying assignment.

Case 6: $p = 1, q > 2$. Numbers $a_1, \ldots, a_n$ are constructed as in Case 1, and there is only one additional number $a_{n+1} = q \cdot 1_m$. □

Lemma 2. Factor–r Sum Subsets *is* NP-*hard for* $r = 2$ *and* $r = \frac{1}{2}$.

Proof. We use a restricted, but still NP-hard version of Exact 3–SAT for a reduction to Factor–r Sum Subsets for the case $r = 2$ (of course, the case $r = \frac{1}{2}$ is identical). In the following, let always $r = 2$. Given an Exact 3–SAT instance with variables $v_1, \dots, v_n$ and clauses $c_1, \dots, c_m$ with only positive literals, let $G = (V, E)$ be the graph with vertices $V = \{v_1, \dots, v_n\}$ (i.e., each variable corresponds to a vertex) and, for $i, j \in \{1, \dots, n\}$, edges $(v_i, v_j) \in E$ if and only if v_i and v_j both occur in a clause c_k, for some $k \in \{1, \dots, m\}$. The Exact 3–SAT variation in which the corresponding graph G is connected is still NP-hard, because we could use a polynomial algorithm for this variation to solve the unrestricted Exact 3–SAT problem by applying the algorithm to each component of the corresponding graph.

We reduce Exact 3–SAT with a connected graph to Factor–r Sum Subsets as follows. We construct an instance A of Factor–r Sum Subsets by defining one number a_i for each variable v_i by $a_i := \sum_{v_i \in c_j} \Delta_n(j)$, where we set the j-th digit to 1 if v_i appears as a literal in clause c_j. We let the base Z of these numbers be 7. Observe that among all a_i's there are exactly three ones in each digit.

Assume that we are given an exact satisfying assignment for the variables of the Exact 3–SAT instance. We then construct sets $X, Y \subseteq A$, where Y contains all numbers a_i for which the corresponding variable v_i has been set to TRUE, and X contains all remaining numbers. Thus, $\text{sum}\,(Y) = \langle 1, 1, \dots, 1 \rangle$ and $\text{sum}\,(X) = \langle 2, 2, \dots, 2 \rangle$, and therefore, X and Y yield a solution for the Factor–r Sum Subsets instance.

For the opposite direction, assume that we are given a solution X and Y for the Factor–r Sum Subsets instance with $\text{sum}\,(X) = 2 \cdot \text{sum}\,(Y)$. Since each digit is set to 1 in exactly three of the numbers a_i, and since no carry-overs can occur when summing up the a_i's because base Z is sufficiently large, $\text{sum}\,(Y)$ must contain only ones and zeros in its digits, and $\text{sum}\,(X)$ contains only twos and zeros. Since the sets are not empty, at least one digit must be set to 1. We assign the value TRUE to a variable v_i with corresponding number a_i if $a_i \in Y$, and we assign the value FALSE, if $a_i \in X$. Thus, if a clause $c_j = (v_f, v_g, v_h)$ exists, then either one of the three numbers a_f, a_g, or a_h is in Y and the other two numbers are in X, or neither X nor Y contain a_f, a_g, or a_h. In the latter case, we know that $\text{sum}\,(X)$ and $\text{sum}\,(Y)$ would contain a 0 at position j.

However, the numbers $\text{sum}\,(X)$ and $\text{sum}\,(Y)$ cannot contain any zero digits because of the connectedness of graph G. In order to see this, assume for the sake of contradiction that $\text{sum}\,(Y)$ contains some digits that are 0. Then $\text{sum}\,(X)$ must have digits with value 0 at the same positions. Consider the set S of all variables that occur in clauses which correspond to zero digits in $\text{sum}\,(X)$ and $\text{sum}\,(Y)$. Then the subgraph of G with only the vertices corresponding to variables from set S must be a component in the graph G without any edges to other vertices: If such an edge existed, it would imply that the corresponding digit is not set to 0 in either $\text{sum}\,(X)$ or $\text{sum}\,(Y)$. To see this, consider an edge $e = (v_f, v_g)$ arising from clause $c_j = (v_f, v_g, v_h)$ with $v_f \in S$ and $v_g \notin S$. Then $a_g \in X \cup Y$, but a_f (and a_h) must be in $X \cup Y$ as well, in order to achieve the factor 2 in the j-th digit.

Thus, there can be no zeroes in any digit in $\text{sum}\,(X)$ or $\text{sum}\,(Y)$, and our assignment is a solution for the Exact 3–SAT instance. □

Since FACTOR–r SUM SUBSETS is obviously in NP, Lemmata 1 and 2 and the NP-completeness of EQUAL SUM SUBSETS yield the following theorem.

THEOREM 6. FACTOR–r SUM SUBSETS *is* NP-*complete for any rational* $r > 0$.

However, FACTOR–r SUM SUBSETS is not NP-complete in the strong sense:

THEOREM 7. FACTOR–r SUM SUBSETS *can be solved in pseudo-polynomial time* $O(n^2 \cdot S^2)$, *where* $S = sum(A)$.

PROOF. We present a dynamic programming algorithm that combines techniques used in the pseudo-polynomial algorithm for EQUAL SUM SUBSETS by Bazgan, Santha and Tuza [Bazgan *et al.* 2002] and the algorithm for ALTERNATING EQUAL SUM SUBSETS (Thm. 5).

We use Boolean variables $F(k, t_1, t_2)$ for $k \in \{1, \ldots, n\}$, $t_1, t_2 \in \{1, \ldots, S\}$ which will be TRUE if there exist disjoint subsets $X, Y \subseteq A$ such that $\text{sum}(X) = t_1$, $\text{sum}(Y) = t_2$, and $a_k \in X \cup Y$.

The algorithm fills the table in the same order as in the proof of Thm. 5 using the following recursion:

$$F(k, t_1, t_2) \quad = \quad \bigvee_{\ell=1}^{k-1} [\, F(\ell, t_1 - a_k, t_2) \vee F(\ell, t_1, t_2 - a_k)\,].$$

The algorithm stops once it finds a value t for which there is an index k such that $F(k, t, rt)$ is TRUE; the corresponding sets constitute a solution which is easy to find.

The computation of each entry $F(k, t_1, t_2)$ takes time $O(n)$, hence the total time complexity of the algorithm is $O(n^2 \cdot S^2)$. □

REMARK 1. The above complexity bound does not depend on r, therefore the result holds even if r is given as part of the input.

5. Complexity of *k* Equal Sum Subsets

5.1 NP-completeness of k equal sum subsets

We now turn to the generalization of EQUAL SUM SUBSETS where we ask for k subsets of equal sum, instead of two. This is the problem k EQUAL SUM SUBSETS (Definition 2). We first show its NP-hardness by reduction from ALTERNATING PARTITION.

THEOREM 8. k EQUAL SUM SUBSETS *is* NP-*complete for any* $k > 2$.

PROOF. The problem is obviously in NP. To show NP-hardness we reduce ALTERNATING PARTITION to it. We transform a given ALTERNATING PARTITION instance with pairs $(u_1, v_1), \ldots, (u_n, v_n)$ into a k EQUAL SUM SUBSETS instance as follows. For each pair (u_i, v_i) we construct two numbers $u'_i = \langle u_i \rangle \circ \Delta_n(i)$ and $v'_i = \langle v_i \rangle \circ \Delta_n(i)$. In addition, we construct $k-2$ (equal) numbers $c_1, \ldots, c_{k-2}$ with $c_i = \langle \frac{1}{2} \sum_i (u_i + v_i) \rangle \circ 1_n$. We set base $Z = (n+1) \cdot k \cdot \sum_i (u_i + v_i)$, which is chosen sufficiently large to ensure that no carry-overs from one digit to the next occur in any of the following additions.

Assume that we are given a solution for the Alternating Partition instance, i.e., two index sets I and J such that $\sum_{i \in I} u_i + \sum_{j \in J} v_j = \sum_{i \in I} v_i + \sum_{j \in J} u_j$. We construct k equal sum subsets $S_1, \ldots, S_k$ as follows. For $i = 1, \ldots, k-2$, we let $S_i = \{c_i\}$; for the remaining two subsets, we let $u'_i \in S_{k-1}$, if $i \in I$, and $v'_i \in S_{k-1}$, if $i \in J$, and we let $u'_i \in S_k$, if $i \in J$, and $v'_i \in S_k$, if $v_i \in I$. Obviously, all S_i sum up to the same sum $\langle \frac{1}{2} \sum_i (u_i + v_i) \rangle \circ 1_n$, thus we have a solution for the k Equal Sum Subsets instance.

For the opposite direction, assume that we are given a solution for the k Equal Sum Subsets instance, i.e., k equal sum subsets $S_1, \ldots, S_k$. Since each of the n rightmost digits is set to 1 in exactly k numbers, we can assume w.l.o.g. that $S_i = \{c_i\}$ for $i = 1, \ldots, k-2$. The remaining two subsets naturally form an alternating partition, as u'_i and v'_i can never be in the same subset for any $i = 1, \ldots, n$, and all numbers u'_i and v'_i must occur in one of the remaining two subsets in order to match the ones in the n rightmost digits of the other subsets. □

5.2 *k equal sum subsets for k = O(1) and k = Ω(n)*

We now study the impact of the size of parameter k on the complexity of k Equal Sum Subsets. In particular, the following two theorems show that the problem can be solved in pseudo-polynomial time if k is a constant, while it becomes strongly NP-hard if k is linear in n.

Theorem 9. *The problem k* Equal Sum Subsets *with input $A = \{a_1, \ldots, a_n\}$ can be solved in time $O(\frac{n \cdot S^k}{k^{k-1}})$, where $S = sum(A)$. For $k = O(1)$, this time complexity is pseudo-polynomial.*

Proof. We present a dynamic programming algorithm for k Equal Sum Subsets that uses basic ideas of well-known dynamic programming algorithms for Bin Packing with fixed number of bins (for the definition see [Garey and Johnson 1979, p. 226]).

For an instance $A = \{a_1, \ldots, a_n\}$ of k Equal Sum Subsets, let $S = \mathrm{sum}(A)$. We define Boolean variables $F(i, s_1, \ldots, s_k)$, where $i \in \{1, \ldots, n\}$ and $s_j \in \{0, \ldots, \lfloor \frac{S}{k} \rfloor\}$, for $1 \leq j \leq k$. Variable $F(i, s_1, \ldots, s_k)$ will be TRUE if there are k disjoint subsets $X_1, \ldots, X_k \subseteq \{a_1, \ldots, a_i\}$ with $\mathrm{sum}(X_j) = s_j$, for $1 \leq j \leq k$. Given this, there is a solution for the k Equal Sum Subsets instance if and only if there exists a value $s \in \{1, \ldots, \lfloor \frac{S}{k} \rfloor\}$ such that $F(n, s, \ldots, s) = \text{TRUE}$.

Clearly, $F(1, s_1, \ldots, s_k)$ is TRUE if and only if either $s_i = 0$, for $1 \leq i \leq k$, or there exists index j such that $s_j = a_1$ and $s_i = 0$, for all $1 \leq i \leq k, i \neq j$. For $i \in \{2, \ldots, n\}$ and $s_j \in \{0, \ldots, \lfloor \frac{S}{k} \rfloor\}$, variable $F(i, s_1, \ldots, s_k)$ can be expressed recursively as

$$\begin{aligned} F(i, s_1, \ldots, s_k) &= F(i-1, s_1, \ldots, s_k) \vee \\ & \bigvee_{1 \leq j \leq k,\ s_j - a_i \geq 0} F(i-1, s_1, \ldots, s_{j-1}, s_j - a_i, s_{j+1}, \ldots, s_k). \end{aligned}$$

The Boolean value of all variables can be determined in time $O(\frac{nS^k}{k^{k-1}})$, since there are $n\lfloor \frac{S}{k} \rfloor^k$ variables, and computing each variable takes at most time $O(k)$. This yields the claim. □

The previous theorem shows that there is a pseudo-polynomial time algorithm for k Equal Sum Subsets if k is a fixed constant. We now show that this it is unlikely that the problem admits a pseudo-polynomial algorithm if k is a fixed *function* of the cardinality n of the input set. In fact, we prove that k Equal Sum Subsets is strongly NP-complete if $k = \Omega(n)$.

Theorem 10. *k Equal Sum Subsets is NP-complete in the strong sense for $k = \frac{n}{p}$, for any fixed integer $p \geq 2$.*

Proof. The problem is obviously in NP. To prove strong NP-hardness, we give a reduction from 3–Partition, which is NP-complete in the strong sense [Garey and Johnson 1979, pp. 99–100] and defined as follows: Given $3n$ positive integers $q_1, \ldots, q_{3n}$ and an integer h such that $\sum_{i=1}^{3n} q_i = nh$ and $\frac{h}{4} < q_i < \frac{h}{2}$, for $i \in \{1, \ldots, 3n\}$, are there n disjoint triples of q_i's such that each triple adds up to h?

Let $Q = \{q_1, \ldots, q_{3n}\}$ and h be an instance of 3–Partition. If all elements in Q are equal, then there is a trivial solution. Otherwise, let $r = 3 \cdot (p-2) + 1$ and

$$\begin{aligned} a_i &= \langle q_i \rangle \circ 0_r, \text{ for } 1 \leq i \leq 3n, \\ b_j &= \langle h \rangle \circ 0_r, \text{ for } 1 \leq j \leq 2n, \text{ and} \\ d_{\ell,m} &= \langle 0 \rangle \circ \Delta_r(\ell), \text{ for } 1 \leq \ell \leq r, 1 \leq m \leq n. \end{aligned}$$

Here, we use base $Z = 6nh$ for all numbers. Let A be the multiset that contains all numbers a_i, b_j and $d_{\ell,m}$. Multiset A is an instance of k Equal Sum Subsets. The cardinality of A is $n' = 3n + 2n + r \cdot n = 5n + (3 \cdot (p-2) + 1) \cdot n = 3pn$. Since r is a constant, the numbers a_i and b_j are polynomial in h, and numbers $d_{\ell,m}$ are bounded by a constant. We now prove that there is a solution for the 3–Partition instance if and only if there are $k = \frac{n'}{p} = 3n$ disjoint subsets of A with equal sum.

"only if": Assume that there is a solution for the 3–Partition instance, i.e., n triples $T_1, \ldots, T_n$ that each sum up to h. This induces n subsets of A with sum $\langle h \rangle \circ 0_r$, namely $S_k = \{a_i \mid q_i \in T_k\}$. Together with the $2n$ subsets that contain exactly one of the b_j's each, we have $3n$ subsets of equal sum $\langle h \rangle \circ 0_r$.

"if": Assume that there is a solution $S_1, \ldots, S_{3n}$ for the k Equal Sum Subsets instance (recall that for our instance $k = 3n$.) Let S_j be any set in this solution.

Then $\mathrm{sum}(S_j)$ has a zero in the r rightmost digits, since for each of these digits there are only n numbers in A for which this digit is non-zero, which are not enough to have one of them in each of the $3n$ sets S_j. Thus, only numbers a_i and b_j can occur in the solution; moreover, we only need to consider the first digit of these numbers, as the other are zeros.

Since not all numbers a_i are equal, and the solution consists of $\frac{n'}{q} = 3n$ disjoint sets, there must be at least one b_j in one of the subsets in the solution. Thus, for $1 \leq j \leq 3n$, we have $\mathrm{sum}(S_j) \geq h$. On the other hand, the sum of all a_i's and of all b_j's is exactly $3n \cdot h$, therefore $\mathrm{sum}(S_j) = h$, for all $1 \leq j \leq 3n$, which means that all a_i's and all b_j's must appear in the solution. More specifically, there must be $2n$ sets in the solution such that each of them contains exactly one of the b_j's, and each of the remaining n sets in the solution consists only of q_i's, such that the corresponding q_i's add up to h. Thus, the latter sets immediately yield a solution for the 3–Partition instance. □

5.3 k equal sum subsets with equal cardinalities

In this section, we study k Equal Sum Subsets in the setting where we do not only require the subsets to be of equal sum, but to be of equal cardinality as well. This yields the following three problem definitions, depending on whether the cardinality is part of the problem definition (k ESS of Cardinality c), part of the input (k ESS of Specified Cardinality), or not specified at all (k ESS of Equal Cardinality).

Definition 6. (k ESS of Cardinality c) *Given a multiset A of n positive integers, are there k nonempty disjoint subsets $S_1, \ldots, S_k \subseteq A$ with sum$(S_1) = \ldots =$ sum(S_k) such that each S_i has cardinality c?*

Definition 7. (k ESS of Specified Cardinality) *Given a multiset A of n positive integers and a positive integer c, are there k nonempty disjoint subsets $S_1, \ldots, S_k \subseteq A$ with sum$(S_1) = \ldots =$ sum(S_k) such that each S_i has cardinality c?*

Definition 8. (k ESS of Equal Cardinality) *Given a multiset A of n positive integers, are there k nonempty disjoint subsets $S_1, \ldots, S_k \subseteq A$ with sum$(S_1) = \ldots =$ sum(S_k) such that all S_i's have the same cardinality?*

We show that the the first problem can be solved in polynomial time (note that in this case, the cardinality c is a fixed constant), while the other two problems are NP-complete.

Theorem 11. *The problem k* ESS of Cardinality *c can be solved in time $O(n^{kc})$.*

Proof. We use exhaustive search: We simply compute all $N = \binom{n}{c}$ subsets of the input set A that have cardinality c; then we consider all $\binom{N}{k}$ possible combinations of k subsets, and for each one we check if it consists of disjoint subsets of equal sum. This algorithm needs time $O(n^{kc})$, which is polynomial in n because c and k are constants. □

Next, we show that k ESS OF SPECIFIED CARDINALITY, where the size of the subsets is given as part of the input, is NP-complete, by modifying the reduction from ALTERNATING PARTITION used in the proof of Thm. 8 to show NP-completeness of k EQUAL SUM SUBSETS.

THEOREM 12. *k ESS OF SPECIFIED CARDINALITY is NP-complete for any $k \geq 2$.*

PROOF. The problem is obviously in NP. To show NP-hardness, we transform a given ALTERNATING PARTITION instance $(u_1, v_1), \ldots, (u_n, v_n)$ into a k ESS OF SPECIFIED CARDINALITY instance as follows. Let $S = \sum_{i=1}^{n}(u_i + v_i)$. For each pair (u_i, v_i) we construct two numbers $u_i' = \langle u_i \rangle \circ \Delta_n(i)$ and $v_i' = \langle v_i \rangle \circ \Delta_n(i)$. In addition, we construct $k-2$ (equal) numbers $b_1, \ldots, b_{k-2}$ with $b_i = \langle \frac{S}{2} \rangle \circ \Delta_n(n)$. Finally, for each b_i we construct $n-1$ numbers $d_{i,j} = \langle 0 \rangle \circ \Delta_n(j)$, for $1 \leq j \leq n-1$. We set the base of the numbers to $(n+1) \cdot k \cdot S$ in order to ensure that no carry-overs from one digit to the next occur in any additions in the following proof. The set A that contains all u_i''s, v_i''s, b_i's, and d_{ij}'s, together with chosen cardinality $c := n$, is our instance of k ESS OF SPECIFIED CARDINALITY.

Assume first that we are given a solution for the ALTERNATING PARTITION instance, i.e., two index sets I and J. We construct k equal sum subsets $S_1, \ldots, S_k$ as follows. For $i = 1, \ldots, k-2$, we set $S_i = \{b_i, d_{i,1}, \ldots, d_{i,n-1}\}$; for the remaining two subsets, we let $u_i' \in S_{k-1}$, if $i \in I$, and $v_j' \in S_{k-1}$, if $j \in J$, and we let $u_j' \in S_k$, if $j \in J$, and $v_i' \in S_k$, if $i \in I$. Clearly, all these sets have n elements, and their sum is $\langle \frac{S}{2} \rangle \circ 1_n$. Hence, the sets S_i yield a solution for the k ESS OF SPECIFIED CARDINALITY instance.

For the opposite direction, assume that we are given a solution for the k ESS OF SPECIFIED CARDINALITY instance, i.e., k equal sum subsets $S_1, \ldots, S_k$ of cardinality n. In this case, all numbers participate in the sets S_i, since there are exactly $k \cdot n$ numbers in the input A. The elements in each set S_i sum up to $\langle \frac{S}{2} \rangle \circ 1_n$ by definition. Since the first digit of each b_i equals $\frac{S}{2}$, we may assume w.l.o.g. that for each $i \in \{1, \ldots, k-2\}$, set S_i contains b_i and does not contain any number with non-zero first digit, i.e., it does not contain any u_j' or any v_j'. Then all u_i''s and v_i''s, and only these numbers, are in the remaining two subsets. This yields immediately a solution for the ALTERNATING PARTITION instance, as the two subsets yield the same sum $\langle \frac{S}{2} \rangle \circ 1_n$, and since u_i' and v_i' can never be in the same subset, as both have the $(i+1)$-th digit non-zero. □

Note that the above reduction works in a similar fashion for the problem k ESS OF EQUAL CARDINALITY. This requires to employ a method where additional extra digits are used in order to force the equal sum subsets to include *all* augmented numbers that correspond to numbers in the ALTERNATING PARTITION instance; a similar method has been used by Woeginger and Yu [Woeginger and Yu 1992] to establish the NP-completeness of EQUAL SUM SUBSETS (called EQUAL-SUBSET-SUM there).

We finally show that the problems k ESS OF SPECIFIED CARDINALITY and k ESS OF EQUAL CARDINALITY are not strongly NP-complete for fixed constant k, by describing a dynamic programming algorithm for the two problems that needs pseudo-polynomial time.

THEOREM 13. *The problems k* ESS OF SPECIFIED CARDINALITY *and k* ESS OF EQUAL CARDINALITY *with input $A = \{a_1, \dots, a_n\}$ can be solved in time $O(\frac{S^k \cdot n^{k+1}}{k^{2k-1}})$, where $S = sum(A)$. For $k = O(1)$, this time is pseudo-polynomial.*

PROOF. The algorithm is very similar-in-spirit to the dynamic programming algorithm from Thm. 9. In fact, it suffices to add to our variables k more dimensions corresponding to cardinalities of the subsets. More precisely, we define Boolean variables $F(i, s_1, \dots, s_k, c_1, \dots, c_k)$, where $i \in \{1, \dots, n\}$, $s_j \in \{0, \dots, \lfloor \frac{S}{k} \rfloor\}$, for $1 \leq j \leq k$, and $c_j \in \{0, \dots, \lfloor \frac{n}{k} \rfloor\}$, for $1 \leq j \leq k$. Variable $F(i, s_1, \dots, s_k, c_1, \dots, c_k)$ will be TRUE if there are k disjoint subsets $X_1, \dots, X_k \subseteq \{a_1, \dots, a_i\}$ such that $\text{sum}(X_j) = s_j$ and $|X_j| = c_j$, for $1 \leq j \leq k$. There are k subsets of equal sum and equal cardinality c if and only if there exists a value $s \in \{1, \dots, \lfloor \frac{S}{k} \rfloor\}$ such that $F(n, s, \dots, s, c, \dots, c) =$ TRUE. Moreover, there are k subsets of equal sum and equal (non-specified) cardinality if and only if there exists a value $s \in \{1, \dots, \lfloor \frac{S}{k} \rfloor\}$ and a value $d \in \{1, \dots, \lfloor \frac{n}{k} \rfloor\}$ such that $F(n, s, \dots, s, d, \dots, d) =$ TRUE.

Clearly, $F(1, s_1, \dots, s_k, c_1, \dots, c_k) =$ TRUE if and only if either $s_i = 0$ and $c_i = 0$, for $1 \leq i \leq k$, or there exists an index j such that $s_j = a_1, c_j = 1$, and $s_i = 0$ and $c_i = 0$ for all $1 \leq i \leq k, i \neq j$.

For $i \in \{2, \dots, n\}$, $s_j \in \{0, \dots, \lfloor \frac{S}{k} \rfloor\}$, and $c_j \in \{0, \dots, \lfloor \frac{n}{k} \rfloor\}$, the truth value of variable $F(i, s_1, \dots, s_k, c_1, \dots, c_k)$ can be expressed recursively as

$$F(i, s_1, \dots, s_k, c_1, \dots, c_k) = F(i-1, s_1, \dots, s_k, c_1, \dots, c_k) \lor \bigvee_{1 \leq j \leq k,\ s_j - a_i \geq 0,\ c_j > 0} F(i-1, s_1, \dots, s_j - a_i, \dots, s_k, c_1, \dots, c_j - 1, \dots, c_k).$$

The Boolean value of all variables can be determined in time $O(\frac{S^k \cdot n^{k+1}}{k^{2k-1}})$, since there are $n \cdot \lfloor \frac{S}{k} \rfloor^k \cdot \lfloor \frac{n}{k} \rfloor^k$ variables, and computing each variable takes at most $O(k)$ time. This yields the claim. □

REMARK 2. A theorem similar to Thm. 10 can be shown for both k ESS OF SPECIFIED CARDINALITY and k ESS OF EQUAL CARDINALITY, namely that they are NP-complete in the strong sense for $k = \Omega(n)$.

6. Conclusions and Open Questions

We studied several variations of the EQUAL SUM SUBSETS problem: We gave (pseudo-)polynomial time algorithms and proved NP-completeness for several variations of EQUAL SUM SUBSETS where the choice of elements is restricted. Furthermore, we proved analogous results for the variation where we specify a rational factor between the sum of the two subsets. Our techniques can be used to obtain similar results for even more variations of EQUAL SUM SUBSETS, for example for ESS OF

DIFFERENT CARDINALITY, where we ask for two equal sum subsets of *different* cardinality, and for ESS FROM TWO SETS, where we ask for two subsets of equal sum that are drawn from *two* different sets; for detailed proofs of NP-completeness of those problems and of further variants the reader is referred to [Cieliebak *et al.* 2002] and [Cieliebak 2003]. Let us also note that it is not hard to devise pseudo-polynomial time algorithms for all those problems by appropriately adapting algorithms presented in this work.

We also studied the case where we ask for $k > 2$ equal sum subsets, and showed that the problem becomes strongly NP-hard if the number of subsets is linear in n (the size of the input), while it can be solved in pseudo-polynomial time if we ask for only a constant number of subsets. In the latter case (constant k), if we require in addition that all subsets be of equal cardinality, then the problem is polynomial-time solvable if the cardinality is also constant, while it is NP-hard otherwise (but not in the strong sense).

Our studies call forth several questions in the realm of EQUAL SUM SUBSETS that are still open:

- The problem k EQUAL SUM SUBSETS is solvable in pseudo-polynomial time for constant k, while it is strongly NP-complete for k linear in n. What is the exact borderline between pseudo-polynomial time solvability and strong NP-completeness?
- The dynamic programming algorithms for k EQUAL SUM SUBSETS and its variations run in pseudo-polynomial time if $k = O(1)$. However, their running times are highly exponential in k. Moreover, there exists a gap between the time complexity of the pseudo-polynomial time algorithm for EQUAL SUM SUBSETS [Bazgan *et al.* 2002], which is $O(n^2S)$, and the complexity of the algorithm described in Theorem 9 for k EQUAL SUM SUBSETS which is $O(nS^2)$ for $k = 2$ (note that the algorithm works for any $k \geq 2$). Thus, the existence of faster algorithms for k EQUAL SUM SUBSETS in the general case, or at least for small constant values of k, is a clear possibility. Finding such algorithms is an interesting open problem.
- We have only studied problems where the subsets need to have sums of ratio exactly 1 (or, more generally, $r > 0$). There are natural optimization versions related to the studied problems; for instance such a version of k EQUAL SUM SUBSETS might ask to find k subsets of a given set A such that the ratio between the greatest and the smallest sum, among the k subsets, is minimized. This problem has been studied only for $k = 2$ by Woeginger and Yu [Woeginger and Yu 1992], who gave a 1.324-approximation algorithm, and by Bazgan, Santha and Tuza [Bazgan *et al.* 2002], who presented an FPTAS. As far as we know the approximability of the problem for $k > 2$ is open.

  A closely related problem is MULTIPLE KNAPSACK, which is the generalization of KNAPSACK in which multiple knapsacks are given instead of only one. Chekuri and Khanna [Chekuri and Khanna 2005] have recently shown the existence of an FPTAS for MULTIPLE KNAPSACK, while a PTAS for the case of identical bins (UNIFORM MULTIPLE KNAPSACK) had been earlier given by

Kellerer [Kellerer 1999]. One might be tempted to use one of those approximation schemes as a 'black box' in order to obtain an FPTAS for the optimization version of k EQUAL SUM SUBSETS; however it is not obvious how to do this in polynomial time, since one might need to check all possible sum values in order to find a solution close enough to the optimal (it is not hard to see that a mere binary search would not work).

- Going back to the case of two subsets, one might wonder whether the techniques used in [Bazgan *et al.* 2002] (see above) can be adapted in a straightforward manner in order to derive an FPTAS for the problem that asks for two sets with ratio as close to r as possible, for any fixed ratio $r > 0$ (this can be seen as the optimization version of FACTOR–r SUM SUBSETS). Unfortunately, there seems to be no easy way to do this for the following reasons. A fundamental ingredient of the FPTAS proposed in [Bazgan *et al.* 2002] is a pseudo-polynomial dynamic programming algorithm that solves the optimization problem exactly. That algorithm, similar to our algorithm for ESS WITH EXCLUSIONS, fills two tables, one with Boolean entries and one with entries that are sets of elements. The correctness of the algorithm depends crucially on the fact that the first time two subsets of equal sum are found, the algorithm can immediately stop returning an optimal solution. However, this would not work for FACTOR–r SUM SUBSETS, since in this case there can be two subsets of equal sum which certainly (for $r \neq 1$) do not constitute a feasible solution but may participate in a feasible solution; hence, both subsets should be stored, possibly resulting to an exponential blow-up of the space needed. It therefore remains open whether one can obtain an FPTAS for this problem by devising a more involved adaptation of the above mentioned FPTAS, or by using different techniques.
- In the context of optimization variations it is worth investigating whether there is any generic algorithmic technique that can be used to obtain approximation schemes for such problems. Pruhs and Woeginger [Pruhs and Woeginger 2004] have developed such a tool for a wide class of subset selection problems; unfortunately, their technique applies only to problems where *one* subset is sought. The question is whether a similar technique can be developed, appropriate for problems that ask for two or more subsets with specific properties.

Acknowledgements

We would like to thank Peter Widmayer for his encouragement and fruitful discussions. We would also like to thank the anonymous referees for suggestions and comments that substantially improved the quality of the presentation.

Preliminary versions of several results in this article have appeared in "Composing Equipotent Teams" (by M. Cieliebak, S. Eidenbenz, and A. Pagourtzis), Proceedings of the 14th International Symposium on Fundamentals of Computation Theory (FCT 2003), Malmö, Sweden, in [Cieliebak *et al.* 2002], and in M. Cieliebak's Ph.D. Thesis [Cieliebak 2003].

Aris Pagourtzis acknowledges support from National Technical University of Athens through the Basic Research Support program PEVE 2007.

Finally, special thanks go to our children Kira Magdalena, Jano Leonardo, Tayra, Yaris, Lyas, Malin, Vassiliki, Aryiri Maria, Theofanis, Simon, and Roger, for the joy they have brought to our lives and for several sleepless nights that provided us with additional time to complete this work.

References

Bazgan, C., Santha, M., and Tuza, Zs. 2002. Efficient approximation algorithms for the Subset–Sum Equality problem. *Journal of Computer and System Sciences 64*, 2, 160–170.

Chekuri, C. and Khanna, S. 2005. A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem. *SIAM Journal on Computing 35*, 3, 713–728.

Cieliebak, M. 2003. *Algorithms and Hardness Results for DNA Physical Mapping, Protein Identification, and Related Problems*. PhD thesis, ETH Zürich, Department of Computer Science.

Cieliebak, M., Eidenbenz, S., Pagourtzis, A., and Schlude, K. 2002. Equal Sum Subsets: Complexity of Variations. Tech. Report 370, ETH Zurich, Department of Computer Science.

Cieliebak, M., Eidenbenz, S., and Penna, P. 2003. Noisy Data Make the Partial Digest Problem NP-hard. In *Proc. of the* 3rd *Workshop on Algorithms in Bioinformatics (WABI 2003)*, 111–123.

de la Vega, W. F. and Lueker, G. S. 1981. Bin packing can be solved within $1+\varepsilon$ in linear time. *Combinatorica 1*, 4, 349–355.

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

Ibarra, O. H. and Kim, C. E. 1975. Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems. *Journal of the ACM 22*, 4, 463–468.

Karmarkar, N. and Karp, R. M. 1982. An Efficient Approximation Scheme for the One-Dimensional Bin-Packing Problem. In *Proc. 23rd Annual Symposium on Foundations of Computer Science, (FOCS 1982)*. IEEE, 312–320.

Kellerer, H. 1999. A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem. In *Proc. RANDOM-APPROX 1999*, Volume 1671 of *Lecture Notes in Computer Science*. Springer, 51–62.

Lawler, E. L. 1979. Fast Approximation Algorithms for Knapsack Problems. *Mathematics of Operation Research 4*, 4, 339–356.

Martello, S. and Toth, P. 1990. *Knapsack Problems*. John Wiley & Sons.

Meggido, N. and Papadimitriou, C. H. 1991. On total functions, existence theorems, and computational complexity. *Theoretical Computer Science 81*, 317–324.

Papadimitriou, C. H. 1994. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences 48*, 498–532.

Pevzner, P. A. 2000. *Computational Molecular Biology: An Algorithmic Approach*. MIT Press.

Pruhs, K. and Woeginger, G. J. 2004. Approximation Schemes for a Class of Subset Selection Problems. In *Proc. of LATIN 2004*, Volume 2976 of *LNCS*, 203–211.

Setubal, J. and Meidanis, J. 1997. *Introduction to Computational Molecular Biology*. PWS Boston.

Woeginger, G. J. and Yu, Z. L. 1992. On the equal–subset–sum problem. *Information Processing Letters 42*, 6, 299–302.