Composing Equipotent Teams

Mark Cieliebak[1], Stephan Eidenbenz[2], and Aris Pagourtzis[3]

[1] Institute of Theoretical Computer Science, ETH Zürich, `cieliebak@inf.ethz.ch`
[2] Basic and Applied Simulation Science (CCS-5), Los Alamos National Laboratory[†], `eidenben@lanl.gov`
[3] Department of Computer Science, School of ECE, National Technical University of Athens, Greece[‡], `pagour@cs.ntua.gr`

Abstract. We study the computational complexity of k Equal Sum Subsets, in which we need to find k disjoint subsets of a given set of numbers such that the elements in each subset add up to the same sum. This problem is known to be NP-complete. We obtain several variations by considering different requirements as to how to compose teams of equal strength to play a tournament. We present:

- A pseudo-polynomial time algorithm for k Equal Sum Subsets with $k = O(1)$ and a proof of strong NP-completeness for $k = \Omega(n)$.
- A polynomial-time algorithm under the additional requirement that the subsets should be of equal cardinality $c = O(1)$, and a pseudo-polynomial time algorithm for the variation where the common cardinality is part of the input or not specified at all, which we proof NP-complete.
- A pseudo-polynomial time algorithm for the variation where we look for two equal sum subsets such that certain pairs of numbers are not allowed to appear in the same subset.

Our results are a first step towards determining the dividing lines between polynomial time solvability, pseudo-polynomial time solvability, and strong NP-completeness of subset-sum related problems; we leave an interesting set of questions that need to be answered in order to obtain the complete picture.

1 Introduction

The problem of identifying subsets of equal value among the elements of a given set is constantly attracting the interest of various research communities due to its numerous applications, such as production planning and scheduling, parallel processing, load balancing, cryptography, and multi-way partitioning in VLSI design, to name only a few. Most research has so far focused on the version where the subsets must form a partition of the given set; however, the variant where we skip this restriction is interesting as well. For example, the Two Equal

[†] LA–UR–03:1158; work done while at ETH Zürich.

[‡] Work partially done while at ETH Zürich, supported by the Human Potential Programme of EU, contract no HPRN-CT-1999-00104 (AMORE).

SUM SUBSETS problem can be used to show NP-hardness for a minimization version of PARTIAL DIGEST (one of the central problems in computational biology whose exact complexity is unknown) [4]. Further applications may include: forming similar groups of people for medical experiments or market analysis, web clustering (finding groups of pages of similar content), or fair allocation of resources.

Here, we look at the problem from the point of view of a tournament organizer: Suppose that you and your friends would like to organize a soccer tournament (you may replace soccer with the game of your choice) with a certain number of teams that will play against each other. Each team should be composed of some of your friends and – in order to make the tournament more interesting – you would like all teams to be of equal strength. Since you know your friends quite well, you also know how well each of them plays. More formally, you are given a set of n numbers $A = \{a_1, \ldots, a_n\}$, where the value a_i represents the excellence of your i-th friend in the chosen game, and you need to find k teams (disjoint subsets[1] of A) such that the values of the players of each team add up to the same number.

This problem can be seen as a variation of BIN PACKING with fixed number of bins. In this new variation we require that all bins should be filled to the same level while it is not necessary to use all the elements. For any set A of numbers, let $\mathrm{sum}(A) := \sum_{a \in A} a$ denote the sum of its elements. We call our problem k EQUAL SUM SUBSETS, where k is a fixed constant:

Definition 1 **(k EQUAL SUM SUBSETS).** *Given is a set of n numbers $A = \{a_1, \ldots, a_n\}$. Are there k disjoint subsets $S_1, \ldots, S_k \subseteq A$ such that $sum(S_1) = \ldots = sum(S_k)$?*

The problem k EQUAL SUM SUBSETS has been recently shown to be NP-complete for any constant $k \geq 3$ [3]. The NP-completeness of the particular case where $k = 2$ has been shown earlier by Woeginger and Yu [8]. To the best of our knowledge, the variations of k EQUAL SUM SUBSETS that we study in this paper have not been investigated before in the literature.

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 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 $k = O(1)$.

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 want to have the same number of elements in each subset (this would be especially useful in composing

[1] Under a strict formalism we should define A as a set of elements which have *values* $\{a_1, \ldots, a_n\}$. For convenience, we prefer to identify elements with their values. Moreover, the term "disjoint subsets" refers to subsets that contain elements of A with different indices.

teams for a tournament). We thus define k Equal Sum Subsets of Specified Cardinality as follows:

Definition 2 (k Equal Sum Subsets of Specified Cardinality). *Given are a set of n numbers $A = \{a_1, \ldots, a_n\}$ and a cardinality c. Are there k 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?*

There are two nice variations of this problem, depending on the parameter c. The first is to require c to be a fixed constant; this corresponds to always playing a specific game (e.g. if you always play soccer then it is $c = 11$). We call this problem k Equal Sum Subsets of Cardinality c. The second variation is to require only that all teams should have an equal number of players, without specifying this number; this indeed happens in several "unofficial" tournaments, e.g. when composing groups of people for medical experiments, or in online computer games. We call the second problem k Equal Sum Subsets of Equal Cardinality.

Let us now consider another aspect of the problem. Your teams would be more efficient and happy if they consisted of players that like each other or, at least, that do not hate each other. Each of your friends has a list of people that she/he prefers as team-mates or, equivalently, a list of people that she/he would not like to have as team-mates. In order to compose k equipotent teams respecting such preferences/exclusions, you should be able to solve the following problem:

Definition 3 (k Equal Sum Subsets with Exclusions). *Given are a set of n numbers $A = \{a_1, \ldots, a_n\}$, and an* exclusion graph *$G_{ex} = (A, E_{ex})$ with vertex set A and edge set $E_{ex} \subseteq A \times A$. Are there k disjoint subsets $S_1, \ldots, S_k \subseteq A$ with $sum(S_1) = \ldots = sum(S_k)$ such that each set S_i is an independent set in G_{ex}, i.e., there is no edge between any two vertices in S_i?*

An overview of the results presented in this paper is given below. In Section 2, we propose 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[2] for $k = O(1)$. For k Equal Sum Subsets with $k = \Omega(n)$, we show strong NP-completeness[3] in Section 3 by proposing a reduction from 3-Partition.

[2] That is, the running time of the algorithm is polynomial in (n, m), where n denotes the cardinality of the input set and m denotes the largest number of the input, but it is not necessarily polynomial in the length of the representation of the input (which is $O(n \log m)$).

[3] This means that the problem remains NP-hard even when restricted to instances where all input numbers are polynomially bounded in the cardinality of the input set. In this case, no pseudo-polynomial time algorithm can exist for the problem (unless P = NP). For formal definitions and a detailed introduction to the theory of NP-completeness the reader is referred to [5].

In Section 4, we propose a polynomial-time algorithm for k Equal Sum Subsets of Cardinality c. The algorithm uses exhaustive search and runs in time $O(n^{kc})$, which is polynomial in n as the two parameters k and c are fixed constants. For k Equal Sum Subsets of Specified Cardinality, we show NP-completeness; the result holds also for k Equal Sum Subsets of Equal Cardinality. However, we show that none of these problems is strongly NP-complete, by presenting an algorithm that can solve them in pseudo-polynomial time.

In Section 5, we study k Equal Sum Subsets with Exclusions, which is NP-complete since it is a generalization of k Equal Sum Subsets. We present a pseudo-polynomial time algorithm for the case where $k = 2$. We also give a modification of this algorithm that additionally guarantees that the two sets will have an equal (specified or not) cardinality.

We conclude in Section 6 presenting a set of open questions and problems.

1.1 Number Representation

In many of our proofs, we use numbers that are expressed in the number system of some base B. We denote by $\langle a_1, \dots, a_n \rangle$ the number $\sum_{1 \le i \le n} a_i B^{n-i}$; we say that a_i is the i-th digit of this number. In our proofs, we will choose base B large enough such that even adding up all numbers occurring in the reduction will not lead to carry-digits from one digit to the next. Therefore, we can add numbers digit by digit. The same holds for scalar products. For example, having base $B = 27$ 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 will generally make liberal use of the notation and allow different bases for each digit. We define the concatenation of two numbers by $\langle a_1, \dots, a_n \rangle \parallel \langle b_1, \dots, b_m \rangle := \langle a_1, \dots, a_n, b_1, \dots, b_m \rangle$, i.e., $\alpha \parallel \beta = \alpha B^m + \beta$, where m is the number of digits in β. We will use $\Delta_n(i) := \langle 0, \dots, 0, 1, 0, \dots, 0 \rangle$ for the number that has n digits, all 0's except for the i-th position where the digit is 1. Furthermore, $\mathbb{1}_n := \langle 1, \dots, 1 \rangle$ is the number that has n digits, all 1's, and $\mathbb{0}_n := \langle 0, \dots, 0 \rangle$ has n zeros. Notice that $\mathbb{1}_n = B^n - 1$.

2 A Pseudo-Polynomial Time Algorithm for k Equal Sum Subsets with $k = O(1)$

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 [5]. For constant k, this algorithm runs in pseudo-polynomial time.

For an instance $A = \{a_1, \dots, a_n\}$ of k Equal Sum Subsets, let $S = \text{sum}(A)$. We define boolean variables $F(i, s_1, \dots, s_k)$, where $i \in \{1, \dots, n\}$ and $s_j \in \{0, \dots, \lfloor \frac{S}{k} \rfloor\}$ for $1 \le j \le k$. Variable $F(i, s_1, \dots, s_k)$ will be TRUE if there are k disjoint subsets $X_1, \dots, X_k \subseteq \{a_1, \dots, a_i\}$ with $\text{sum}(X_j) = s_j$, for $1 \le j \le k$.

There are k sets of equal sum 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 follows:

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

The 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 following

Theorem 1. *There is a dynamic programming algorithm that solves k* Equal Sum Subsets *for input $A = \{a_1, \ldots, a_n\}$ in time $O(\frac{n \cdot S^k}{k^{k-1}})$, where $S = sum(A)$. For $k = O(1)$ this algorithm runs in pseudo-polynomial time.*

3 Strong NP-Completeness of k Equal Sum Subsets with $k = \Omega(n)$

In Section 2 we gave a pseudo-polynomial time algorithm for k Equal Sum Subsets assuming that k is a fixed constant. We will now show that this is unlikely if k is a fixed *function* of the cardinality n of the input set. In particular, we will prove that k Equal Sum Subsets is strongly NP-complete if $k = \Omega(n)$.

Let $k = \frac{n}{q}$ for some fixed integer $q \geq 2$. We provide a polynomial reduction from 3-Partition, which is defined as follows: Given a multiset of $n = 3m$ numbers $P = \{p_1, \ldots, p_n\}$ and a number h with $\frac{h}{4} < p_i < \frac{h}{2}$, for $1 \leq i \leq n$, are there m pairwise disjoint sets $T_1, \ldots, T_m$ such that $\text{sum}(T_j) = h$, for $1 \leq j \leq m$? Observe that in a solution for 3-Partition, there are exactly three elements in each set T_j.

Lemma 1. *If $k = \frac{n}{q}$ for some fixed integer $q \geq 2$, then* 3-Partition *can be reduced to k* Equal Sum Subsets.

Proof. Let $P = \{p_1, \ldots, p_n\}$ and h be an instance of 3-Partition. If all elements in P are equal, then there is a trivial solution. Otherwise, let $r = 3 \cdot (q-2) + 1$ and

$$
\begin{aligned}
a_i &= \langle p_i \rangle \mathbin{||} \mathbb{0}_r, \text{ for } 1 \leq i \leq n \\
b_j &= \langle h \rangle \mathbin{||} \mathbb{0}_r, \text{ for } 1 \leq j \leq \frac{2n}{3} \\
d_{k,\ell} &= \langle 0 \rangle \mathbin{||} \Delta_r(k), \text{ for } 1 \leq k \leq r, 1 \leq \ell \leq \frac{n}{3}
\end{aligned}
$$

Here, we use base $B = 2nh$ for all numbers. Let A be the set containing all numbers a_i, b_j and $d_{k,\ell}$. We will use A as an instance of k Equal Sum Subsets. The size of A is $n' = n + \frac{2n}{3} + r \cdot \frac{n}{3} = n + \frac{2n}{3} + (3 \cdot (q-2) + 1) \cdot \frac{n}{3} = q \cdot n$. We prove that there is a solution for the 3-Partition instance P and h if and only if there are $\frac{n'}{q}$ disjoint subsets of A with equal sum.

"only if": Let $T_1, \ldots, T_m$ be a solution for the 3-Partition instance. This induces m subsets of A with sum $\langle h \rangle \parallel \mathbb{0}_r$, namely $S_i = \{a_i \mid p_i \in T_i\}$. Together with the $\frac{2n}{3}$ subsets that contain exactly one of the b_j's each, we have $n = \frac{n'}{q}$ subsets of equal sum $\langle h \rangle \parallel \mathbb{0}_r$.

"if": Assume there is a solution $S_1, \ldots, S_n$ for the k Equal Sum Subsets instance. Let S_j be any set in this solution. Then $\mathrm{sum}(S_j)$ will have a zero in the r rightmost digits, since for each of these digits, there are only $\frac{n}{3}$ numbers in A for which this digit is non-zero (which are not enough to have one of them in each of the n 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 $\frac{n'}{q} = n$ disjoint sets, there must be at least one b_j in one of the subsets in the solution. Thus, for all j 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 $n \cdot h$, therefore $\mathrm{sum}(S_j) = h$, which means that all a_i's and all b_j's appear in the solution. More specifically, there are $\frac{2n}{3}$ sets in the solution such that each of them contains exactly one of the b_j's, and each of the remaining $\frac{n}{3}$ sets in the solution consists only of a_i's, such that the corresponding a_i's add up to h. Therefore, the latter sets immediately yield a solution for the 3-Partition instance. □

In the previous proof, r is a constant, therefore numbers a_i and b_j are polynomial in h and numbers $d_{k,\ell}$ are bounded by a constant. Since 3-Partition is strongly NP-complete [5], k Equal Sum Subsets is strongly NP-hard for $k = \frac{n}{q}$ as well. Obviously, k Equal Sum Subsets is in NP even if $k = \frac{n}{q}$ for some fixed integer $q \geq 2$, thus we have the following

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

4 Restriction to Equal Cardinalities

In this section we study the setting where we do not only require the teams to be of equal strength, but to be of equal cardinality as well. If we are interested in a specific type of game, e.g. soccer, then the size of the teams is also fixed, say $c = 11$, and we have k Equal Sum Subsets of Cardinality c. This problem is solvable in time polynomial in n by exhaustive search as follows: compute all $N = \binom{n}{c}$ subsets of the input set A that have cardinality c; consider all $\binom{N}{k}$ possible sets of k subsets, and for each one check if it consists of disjoint subsets of equal sum. This algorithm needs time $O(n^{ck})$, which is polynomial in n, since c and k are constants.

On the other hand, if the size of the teams is not fixed, but given as part of the input, then we have k Equal Sum Subsets of Specified Cardinality. We show that this problem is NP-hard by modifying a reduction used in [3] to show NP-completeness of k Equal Sum Subsets. The reduction is from Alternating Partition, which is the following NP-complete [5] variation of Partition: Given n pairs of numbers $(u_1, v_1), \ldots, (u_n, v_n)$, are there two disjoint sets of indices I and J with $I \cup J = \{1, \ldots, n\}$ such that $\sum_{i \in I} u_i + \sum_{j \in J} v_j = \sum_{i \in I} v_i + \sum_{j \in J} u_j$ (equivalently, $\sum_{i \in I} u_i + \sum_{j \in J} v_j = \sum_{i \notin I} u_i + \sum_{j \notin J} v_j$)?

Lemma 2. Alternating Partition *can be reduced to* k Equal Sum Subsets of Specified Cardinality *for any* $k \geq 2$.

Proof. We transform a given Alternating Partition instance with pairs $(u_1, v_1), \ldots, (u_n, v_n)$ into a k Equal Sum Subsets of Specified Cardinality instance as follows: Let $S = \sum_{i=1}^{n} (u_i + v_i)$. For each pair (u_i, v_i) we create two numbers $u'_i = \langle u_i \rangle \parallel \Delta_n(i)$ and $v'_i = \langle v_i \rangle \parallel \Delta_n(i)$. In addition, we create $k-2$ (equal) numbers $b_1, \ldots, b_{k-2}$ with $b_i = \langle \frac{S}{2} \rangle \parallel \Delta_n(n)$. Finally, for each b_i we create $n-1$ numbers $d_{i,j} = \langle 0 \rangle \parallel \Delta_n(j)$, for $1 \leq j \leq n-1$. While we set the base of the first digit to $k \cdot S$, for all other digits it suffices to use base $n+1$, in order to ensure that no carry-digits can occur in any addition 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 Equal Sum Subsets of Specified Cardinality.

Assume first that we are given a solution for the Alternating Partition instance, i.e., two indices sets I and J. We create k equal sum subsets $S_1, \ldots, S_k$ as follows: for $i = 1, \ldots, k-2$, we have $S_i = \{b_i, d_{i,1}, \ldots, d_{i,n-1}\}$; 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 $\langle \frac{S}{2} \rangle \parallel \mathbb{1}_n$.

Now assume we are given a solution for the k Equal Sum Subsets 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, and the elements in each S_i sum up to $\langle \frac{S}{2} \rangle \parallel \mathbb{1}_n$. Since the first digit of each b_i equals $\frac{S}{2}$, we may assume w.l.o.g. that for each $1 \leq i \leq k-2$, 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). Therefore, all u'_i's and v'_i's (and only these numbers) are in the remaining two subsets; this yields an alternating partition for the original instance, as u'_i and v'_i can never be in the same subset since both have the $(i+1)$-th digit non-zero. $\square$

Since the problem k Equal Sum Subsets of Specified Cardinality is obviously in NP, we get the following:

Theorem 3. *For any* $k \geq 2$, k Equal Sum Subsets of Specified Cardinality *is* NP*-complete.*

Remark: Note that the above reduction, hence also the theorem, holds also for the variation k Equal Sum Subsets 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 in [8] to establish the NP-completeness of Two Equal Sum Subsets (called Equal-Subset-Sum there).

However, these problems are not strongly NP-complete for fixed constant k. We will now describe how to convert the dynamic programming algorithm of Section 2 to a dynamic programming algorithm for k Equal Sum Subsets of Specified Cardinality and for k Equal Sum Subsets of Equal Cardinality.

It suffices to add to our variables k more dimensions corresponding to cardinalities of the subsets. 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 $1 \le j \le k$, and $c_j \in \{0, \dots, \lfloor \frac{n}{k} \rfloor\}$ for $1 \le j \le k$. Variable $F(i, s_1, \dots, s_k, c_1, \dots, c_k)$ will be TRUE if there are k disjoint subsets $X_1, \dots, X_k \subseteq \{a_1, \dots, a_i\}$ with $\text{sum}(X_j) = s_j$ and the cardinality of X_j is c_j, for $1 \le j \le k$.

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 $F(n, s, \dots, s, c, \dots, c) = \text{TRUE}$. Also, 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 $F(n, s, \dots, s, d, \dots, d) = \text{TRUE}$.

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

Each variable $F(i, s_1, \dots, s_k, c_1, \dots, c_k)$, for $i \in \{2, \dots, n\}$, $s_j \in \{0, \dots, \lfloor \frac{S}{k} \rfloor\}$, and $c_j \in \{0, \dots, \lfloor \frac{n}{k} \rfloor\}$, can be expressed recursively as follows:

$$F(i, s_1, \dots, s_k, c_1, \dots, c_k) = F(i-1, s_1, \dots, s_k, c_1, \dots, c_k) \vee \bigvee_{\substack{1 \le j \le k \\ s_j - a_i \ge 0 \\ c_j > 0}} F(i-1, s_1, \dots, s_j - a_i, \dots, s_k, c_1, \dots, c_j - 1, \dots, c_k).$$

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 $n \lfloor \frac{S}{k} \rfloor^k \lfloor \frac{n}{k} \rfloor^k$ variables, and computing each variable takes at most time $O(k)$. This yields the following:

Theorem 4. *There is a dynamic programming algorithm that solves k* Equal Sum Subsets of Specified Cardinality *and k* Equal Sum Subsets of Equal Cardinality *for input $A = \{a_1, \dots, a_n\}$ in running time $O(\frac{S^k \cdot n^{k+1}}{k^{2k-1}})$, where $S = sum(A)$. For $k = O(1)$ this algorithm runs in pseudo-polynomial time.*

5 Adding Exclusion Constraints

In this section we study the problem k Equal Sum Subsets with Exclusions where we are additionally given an *exclusion graph* (or its complement: a *preference graph*) and ask for teams that take this graph into account.

Obviously, k Equal Sum Subsets with Exclusions is NP-complete, since k Equal Sum Subsets (shown NP-complete in [3]) is the special case where the exclusion graph is empty ($E_{ex} = \emptyset$). Here, we present a pseudo-polynomial algorithm for the case $k = 2$, using a dynamic programming approach similar-in-spirit to the one used for finding two equal sum subsets (without exclusions) [1].

Let $A = \{a_1, \ldots, a_n\}$ and $G_{ex} = (A, E_{ex})$ be an instance of k Equal Sum Subsets with Exclusions. We assume w.l.o.g. that the input values are ordered, i.e., $a_1 \leq \ldots \leq a_n$. Let $S = \sum_{i=1}^{n} a_i$.

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 $X \subseteq \{a_1, \ldots, a_k\}$, $a_k \in X$, $\mathrm{sum}(X) = t$, and X is independent in G_{ex}. For a TRUE entry $F(k, t)$ we store the corresponding set 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 indices $k, \ell \in \{1, \ldots, n\}$ such that $F(k, t) = F(\ell, t) =$ TRUE; in this case, sets $X(k, t)$ and $X(\ell, t)$ constitute a solution: $\mathrm{sum}(X(k, t)) = \mathrm{sum}(X(\ell, t)) = t$, 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 $F(k, t) =$ FALSE for $1 \leq t < a_k$ and for $\sum_{i=1}^{k} a_i < t \leq S$; moreover, we set $F(k, a_k) =$ TRUE and $X(k, a_k) = \{a_k\}$. Observe that these equations already 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 $a_k \leq t \leq \sum_{i=1}^{k} a_i$ 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 the subset $X(\ell, t - a_k)$ remains independent in G_{ex} when adding a_k. The recursive computation is as follows.

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

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 to the problem 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, computation of $F(k, t)$ takes time $O(n)$ and the total time complexity of the algorithm is $O(n^2 \cdot S)$. Therefore, we have the following

Theorem 5. Two Equal Sum Subsets with Exclusions *can be solved for input* $A = \{a_1, \ldots, a_n\}$ *and* $G_{ex} = (A, E_{ex})$ *in pseudo-polynomial time* $O(n^2 \cdot S)$, *where* $S = sum(A)$.

Remarks: Observe that the problem k Equal Sum Subsets of Cardinality c with Exclusions, where cardinality c is constant, and an exclusion graph is given, can be solved by exhaustive search in time $O(n^{kc})$ in the same way as the problem k Equal Sum Subsets of Cardinality c is solved (see Section 4).

Moreover, we can have a pseudo-polynomial time algorithm for k Equal Sum Subsets of Equal Cardinality with Exclusions, where the cardinality is part of the input, if $k = 2$, by modifying the dynamic programming algorithm for Two Equal Sum Subsets with Exclusions as follows. We introduce a further dimension in our table F, the cardinality, and set $F(k, t, c)$ to TRUE if there is a set X with $\mathrm{sum}(X) = t$ (and all other conditions as before), and such that the cardinality of X equals c. Again, we can fill the table recursively, and we stop as soon as we find values $k, \ell \in \{1, \ldots, n\}, t \in \{1, \ldots, S\}$ and $c \in \{1, \ldots, n\}$ such that $F(k, t, c) = F(\ell, t, c) =$ TRUE, which yields a solution. Notice that the corresponding two sets must be disjoint, since otherwise removing their intersection would yield two subsets of smaller equal cardinality that are independent in G_{ex}; thus, the algorithm - which constructs two sets of minimal cardinality - would have stopped earlier. Table F now has $n^2 \cdot S$ entries, thus we can solve Two Equal Sum Subsets with Exclusions in time $O(n^3 \cdot S)$.

Note that the above sketched algorithm does not work for specified cardinalities, because there may be exponentially many ways to construct a subset of the correct cardinality.

6 Conclusion – Open Problems

In this work we studied the problem k Equal Sum Subsets and some of its variations. We presented a pseudo-polynomial time algorithm for constant k, and proved strong NP-completeness for non-constant k, namely for the case in which we want to find $\frac{n}{q}$ subsets of equal sum, where n is the cardinality of the input set and q a constant. We also gave pseudo-polynomial time algorithms for the k Equal Sum Subsets of Specified Cardinality problem and for the Two Equal Sum Subsets with Exclusions problem, as well as for variations of them.

Several questions remain open. Some of them are: determine the exact borderline between pseudo-polynomial time solvability and strong NP-completeness for k Equal Sum Subsets, for k being a function different than $\frac{n}{q}$, for example $k = \frac{\log n}{q}$; find faster dynamic programming algorithms for k Equal Sum Subsets of Specified Cardinality; and, finally, determine the complexity of k Equal Sum Subsets with Exclusions, i.e. is it solvable in pseudo-polynomial time or strongly NP-complete?

Another promising direction is to investigate approximation versions related to the above problems, for example "given a set of numbers A, find k subsets of A with sums that are as similar as possible". For $k = 2$, the problem has been studied by Bazgan et al. [1] and Woeginger [8]; an FPTAS was presented in [1]. We would like to find out whether there is an FPTAS for any constant k. Finally, it would be interesting to study phase transitions of these problems with respect to their parameters, in a spirit similar to the work of Borgs, Chayes and Pittel [2], where they analyzed the phase transition of Two Equal Sum Subsets.

Acknowledgments

We would like to thank Peter Widmayer for several fruitful discussions and ideas in the context of this work.

References

1. C. Bazgan, M. Santha, and Zs. Tuza; *Efficient approximation algorithms for the Subset-Sum Equality problem*; Proc. ICALP'98, pp. 387–396.
2. C. Borgs, J.T. Chayes, and B. Pittel; *Sharp Threshold and Scaling Window for the Integer Partitioning Problem*; Proc. STOC'01, pp. 330–336.
3. M. Cieliebak, S. Eidenbenz, A. Pagourtzis, and K. Schlude; *Equal Sum Subsets: Complexity of Variations*; Technical Report 370, ETH Zürich, Department of Computer Science, 2003.
4. M. Cieliebak, S. Eidenbenz, and P. Penna; *Noisy Data Make the Partial Digest Problem NP-hard*; Technical Report 381, ETH Zürich, Department of Computer Science, 2002.
5. M.R. Garey and D.S. Johnson; *Computers and Intractability: A Guide to the Theory of NP-completeness*; Freeman, San Francisco, 1979.
6. R.M. Karp; *Reducibility among combinatorial problems*; in R.E. Miller and J.W. Thatcher (eds.), Complexity of Computer Computations, Plenum Press, New York, pp. 85 – 103, 1972.
7. S. Martello and P. Toth; *Knapsack Problems*; John Wiley & Sons, Chichester, 1990.
8. G.J. Woeginger and Z.L. Yu; *On the equal-subset-sum problem*; Information Processing Letters, 42(6), pp. 299–302, 1992.