On Strong NP-Completeness of Rational Problems

Dominik Wojtczak

University of Liverpool, UK

Abstract. The computational complexity of the partition, 0-1 subset sum, unbounded subset sum, 0-1 knapsack and unbounded knapsack problems and their multiple variants were studied in numerous papers in the past where all the weights and profits were assumed to be integers. We re-examine here the computational complexity of all these problems in the setting where the weights and profits are allowed to be any rational numbers. We show that all of these problems in this setting become strongly NP-complete and, as a result, no pseudo-polynomial algorithm can exist for solving them unless P=NP. Despite this result we show that they all still admit a fully polynomial-time approximation scheme.

1 Introduction

The problem of partitioning a given set of items into two parts with equal total weights (that we will refer to as PARTITION) goes back at least to 1897 [14]. A well-known generalisation is the problem of finding a subset with a given total weight (0-1 SUBSET SUM) and the same problem where each item can be picked more than once (UNBOUNDED SUBSET SUM). Finally, these are commonly generalised to the setting where each item also has a profit and the aim is to pick a subset of items with the total profit higher than a given threshold, but at the same time their total weight smaller than a given capacity (0-1 KNAPSACK), and again its variant where each item can be picked more than once (UNBOUNDED KNAPSACK).

The SUBSET SUM problem has numerous applications: its solutions can be used for designing better lower bounds for scheduling problems (see, e.g., [8] and [7]) and it appears as a subproblem in numerous combinatorial problems (see, e.g., [17]). At the same time, many industrial problems can be formulated as knapsack problems: cargo loading, cutting stock, capital budgeting, portfolio selection, interbank clearing systems, knapsack cryptosystems, and combinatorial auctions to name a couple of examples (see Chapter 15 in [12] for more details regarding these problems and their solutions).

The decision problems studied in this paper were among the first ones to be shown to be NP-complete [11]. At the same they are considered to be the easiest problems in this class, because they are polynomial time solvable if items' weights and profits are represented using the unary notation (in other words, they are only *weakly* NP-complete). In particular, they can be solved in polynomial time when these numbers are bounded by a fixed constant (and the number of items is unbounded). Furthermore, the optimisation version of all these decision problems admit *fully polynomial-time approximation schemes (FPTAS)*, i.e., we can find a solution with a value at least equal to $(1 - \epsilon)$ times the optimal in time polynomial in the size of the input and $1/\epsilon$ for any $\epsilon > 0$.

To the best of our knowledge, the computational complexity analysis of all these problems was only studied so far under the simplifying assumption that all the input

values are integers. However, in most settings where these problems are used, these numbers are quite likely to be in fact some rational numbers instead. We were surprised to discover that the computational complexity in such a rational setting was not properly studied before. Indeed, as pointed out in [12]:

> A rather subtle point is the question of rational coefficients. Indeed, most textbooks get rid of this case, where some or all input values are non-integer, by the trivial statement that multiplying with a suitable factor, e.g. with the smallest common multiple of the denominators, if the values are given as fractions or by a suitable power of 10, transforms the data into integers. Clearly, this may transform even a problem of moderate size into a rather unpleasant problem with huge coefficients.

This clearly looks like a fundamental gap in the understanding of the complexity of these computational problems. Allowing the input values to be rational makes a lot of sense in many settings. For example, we encountered this problem when studying an optimal control in multi-mode systems with discrete costs [15] and looking at the weighted voting games (see, e.g., [4] where the weights are defined to be rational). Another interesting problem is checking whether we have the exact amount when paying for something. While we take decimal monetary systems for granted these days, there were plenty of non-decimal monetary systems in use not so long ago. For example, in the UK between 1717 and 1816 one pound sterling was worth twenty shillings, one shilling was worth twelve pence, and one guinea was worth twenty one shillings.

We show here that this makes a significant difference and in fact all these decision problems in this setting become strongly NP-complete [6], i.e., they are NP-complete even when all their numerical values are bounded by a polynomial in the size of the rest of the input or, equivalently, if all the numerical values are represented in unary. To prove this we will show an NP-completeness of a new variant of the SATISFIABLITY problem and use results regarding distribution of prime numbers. As a direct consequence of our result, there does not exist any pseudo-polynomial algorithms for solving these decision problems unless P=NP. At the same time, we will show that they still all admit a fully polynomial-time approximation scheme (see, e.g., [12]). This may sound incorrect, because the paper that introduced strong NP-completeness [6] also showed that no strongly NP-hard problem can admit an FPTAS unless P=NP. However, the crucial assumption made there is that the objective function is integer valued, which does not hold in our case.

Related work. The decision problems studied in this paper are so commonly used that they have already been thousands of papers published about them and and many of their variants, including multiple algorithms and heuristic for solving them precisely and approximately. There are also two full-length books, [20] and [12], solely dedicated to these problems.

Several extensions of the classic knapsack problem were shown to be strongly NP-complete. These include partially ordered knapsack[10] (where we need to pick a set of items closed under predecessor), graph partitioning[10] (where we need to partition a graph into m disjoint subgraphs under cost constraints), multiple knapsack problem [2], knapsack problem with conflict graphs[16] (where we restrict which pairs of items can

be picked together), and quadratic knapsack problem[5,12] (where the profit of packing an item depends on how well it fits together with the other selected items).

The first FPTAS for the optimisation version of the KNAPSACK problem was established in 1975 by Ibarra and Kim[9] and independently by Babat[1]. Multiple other, more efficient, FPTAS for these problems followed (see, e.g., [12]).

Plan of the paper. In the next section, we introduce all the used notation as well as formally define all the decision problems that we study in this paper. In Section 3, we analyse the amount of space one needs to write down the first n primes in unary as well as a unique representation theorem concerning sums of rational numbers. In Section 4, we define a couple of new variants of the well-known satisfiability problem for Boolean formulae in 3-CNF form and show them to be NP-complete. Our main result, concerning the strong NP-hardness of all the decision problems studied in this paper with rational inputs, can be found in Section 5 and it builds on the results from Sections 3 and 4. We briefly discuss the existence of FPTAS for the optimisation version of our decision problems in Section 6. Finally, we conclude in Section 7.

2 Background

Let $\mathbb{Q}_{\geq 0}$ be the set of non-negative rational numbers. We assume that a non-negative rational number is represented as usual as a pair of its numerator and denumerator, both of which are natural numbers that do not have a common divisor > 1. A rational number is written in the unary notation, if both of these two numbers are represented in unary. For two natural numbers a and n, let $a \mod n \in \{0, \ldots, n-1\}$ denote the remainder of dividing a by n. For any two numbers $a, b \in \{1, \ldots, n\}$, we define their addition modulo n, written as $\oplus_n$, as follows: $a \oplus_n b = ((a + b - 1) \mod n) + 1$. Note that we subtract and add 1 in this expression so that the result of this operation belongs to $\{1, \ldots, n\}$. Similarly we define the subtraction modulo n, written as $\ominus_n$, as $a \ominus_n b = ((n + a - b - 1) \mod n) + 1$. We assume that $\oplus_n$ and $\ominus_n$ operators have higher precedence than the usual $+$ and $-$ operators.

We now formally define all the decision problems that we study in this paper.

Definition 1 (SUBSET SUM **problems**). *Assume we are given a list of n items with rational non-negative weights $A = \{a_1, \ldots, a_n\}$ and a target total weight $W \in \mathbb{Q}_{\geq 0}$.*

0-1 SUBSET SUM: Does there exists a subset B of A such that the total weight of B is equal to W?

UNBOUNDED SUBSET SUM: Does there exist a list of non-negative integer quantities $(q_1, \ldots, q_n)$ such that

$$\sum_{i=1}^{n} q_i \cdot a_i = W?$$

(Intuitively, q_i denotes the number of times the i-th item in A is chosen.)

A natural generalisation of this problem where each item gives us a profit when picked is the well-known knapsack problem.

Definition 2 (KNAPSACK **problems**). *Assume there are n items whose non-negative rational weights and profits are given as as a list $L = \{(w_1, v_1), \ldots, (w_n, v_n)\}$. Let the capacity be $W \in \mathbb{Q}_{\geq 0}$ and the profit threshold be $V \in \mathbb{Q}_{\geq 0}$.*

0-1 KNAPSACK*: Is there a subset of L whose total weight does not exceed W and total profit is at least V?*

UNBOUNDED KNAPSACK*: Is there a list of non-negative integers $(q_1, \ldots, q_n)$ such that*

$$\sum_{i=1}^{n} q_i \cdot w_i \leq W \qquad and \qquad \sum_{i=1}^{n} q_i \cdot v_i \geq V?$$

(Intuitively, q_i denotes the number of times the i-th item in A is chosen.)

Finally, a special case of the SUBSET SUM problem is the PARTITION problem.

Definition 3 (PARTITION **problem**). *Assume we are given a list of n items with non-negative rational weights $A = \{a_1, \ldots, a_n\}$.*

Can the set A be partitioned into two sets with equal total weights?

3 Prime Suspects

In this section we first show that writing down all the first n prime numbers in the unary notation can be done using space polynomial in n. Let π_i denote the i-th prime number. The following upper bound is known for π_i.

Theorem 1 (inequality (3.13) in [18]).

$$\pi_i < i(\log i + \log \log i) \quad for \quad i \geq 6$$

This estimate gives us the following corollary that will be used in the main result of this paper.

Corollary 1. *The total size of the first n prime numbers, when written down in unary, is $\mathcal{O}(n^2 \log n)$. Furthermore, they can be computed in polynomial time.*

Proof. Let $n \geq 6$, because otherwise the problem is trivial. Thanks to Theorem 1, it suffices to list all natural numbers smaller than $2n \log n$ (because $2n \log n \leq n(\log n + \log \log n)$) and use the sieve of Eratosthenes to remove all nonprime numbers from this list. It follows that writing down the first n prime numbers requires $\mathcal{O}(n^2 \log n)$ space. The sieve can easily be implemented in polynomial time and, to be precise, in this case $\mathcal{O}(n^2 \log^2 n)$ additions and $\mathcal{O}(n \log n)$ bits of memory would suffice. $\qquad\square$

Now we prove a result regarding a unique representation of rational numbers expressed as sums of fractions with prime denominators, which in a way is quite similar to the Chinese remainder theorem.

Lemma 1. *Let $(p_1, \ldots, p_n)$ be a list of n different prime numbers. Let $(a_0, a_1, \ldots, a_n)$ and $(a_0, b_1, \ldots, b_n)$ be two lists of integers such that $|a_i| < p_i$ and $|b_i| < p_i$ hold for all $i = 1, \ldots, n$. We then have*

$$a_0 + \frac{a_1}{p_1} + \ldots + \frac{a_n}{p_n} = b_0 + \frac{b_1}{p_1} + \ldots + \frac{b_n}{p_n}$$

if and only if

$$a_i = b_i \ for \ all \ i = 0, \ldots, n.$$

Proof. ($\Leftarrow$) If $a_i = b_i$ for all $i = 0, \ldots, n$ holds then obviously

$$a_0 + \frac{a_1}{p_1} + \ldots + \frac{a_n}{p_n} = b_0 + \frac{b_1}{p_1} + \ldots + \frac{b_n}{p_n}.$$

($\Rightarrow$) We need to consider two cases: $a_0 = b_0$ and $a_0 \neq b_0$. In the first case, suppose that $a_j \neq b_j$ for some $j \in \{1, \ldots, n\}$. If we multiply

$$\frac{a_1 - b_1}{p_1} + \ldots + \frac{a_n - b_n}{p_n} \quad \text{by} \quad \prod_{i=1}^{n} p_i$$

then we would get an integer, which is not divisible by p_j (because $0 < |a_j - b_j| < p_j$) and so this expression cannot be equal to 0. Therefore, in this case,

$$a_0 + \frac{a_1}{p_1} + \ldots + \frac{a_n}{p_n} \neq b_0 + \frac{b_1}{p_1} + \ldots + \frac{b_n}{p_n}.$$

In the second case, if $a_i = b_i$ for all $i = 1, \ldots, n$ holds then clearly

$$a_0 + \frac{a_1}{p_1} + \ldots + \frac{a_n}{p_n} \neq b_0 + \frac{b_1}{p_1} + \ldots + \frac{b_n}{p_n}.$$

Otherwise, again suppose that $a_j \neq b_j$ for some $j \in \{1, \ldots, n\}$. If we multiply

$$a_0 - b_0 + \frac{a_1 - b_1}{p_1} + \ldots + \frac{a_n - b_n}{p_n} \quad \text{by} \quad \prod_{i=1}^{n} p_i$$

then we would get an integer, which is not divisible by p_j (because $0 < |a_j - b_j| < p_j$) and so this expression cannot be equal to 0. Therefore, again, in this case,

$$a_0 + \frac{a_1}{p_1} + \ldots + \frac{a_n}{p_n} \neq b_0 + \frac{b_1}{p_1} + \ldots + \frac{b_n}{p_n}.$$

$\square$

4   In the Pursuit of Satisfaction

The Boolean satisfiability (SATISFIABILITY) problem for formulae was the first problem to be shown NP-complete by Cook [3] and Levin [13]. Tovey showed in [21] that SATISFIABILITY is also NP-complete for 3-CNF formulae in which each variable occurs at most 4 times. We will denote the set of such formulae by 3-CNF$_{\leq 4}$. Schaefer defined in [19] the ONE-IN-THREE-SAT problem for 3-CNF formulae in which one asks for an truth assignment that makes exactly one literal in each clause true, and showed it to be NP-complete. We define here a new ALL-THE-SAME-SAT problem for 3-CNF formulae, which asks for a valuation that makes exactly the same number of literals true in every clause (this may be zero, i.e., such a valuation may not make the formula true). This problem will be a crucial ingredient in the proof of the main result of this paper.

The first step is to show that ONE-IN-THREE-SAT problem is NP-complete even when restricted to 3-CNF$_{\leq 4}$ formulae.

5

Theorem 2. *The* ONE-IN-THREE-SAT *problem for 3-CNF$_{\leq 4}$ is* NP-*complete.*

Proof. Obviously the problem is in NP, because we can simply guess a valuation and check how many literals are true in each clause in linear time.

To prove NP-hardness, we are going to reduce from the SATISFIABILITY problem for 3-CNF$_{\leq 4}$, which is NP-complete [21]. Assume we are given a 3-CNF$_{\leq 4}$ formula

$$\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$$

with m clauses $C_1, \ldots, C_m$ and n propositional variables $v_1, \ldots, v_n$, where $C_j = x_j \vee y_j \vee z_j$ for $j = 1, \ldots, m$ and each x_j, y_j, z_j is a literal equal to v_i or $\neg v_i$ for some i. We will construct a 3-CNF$_{\leq 4}$ formula ϕ' with $3m$ clauses and $n + 4m$ propositional variables such that ϕ is satisfiable iff ϕ' is an instance of the ONE-IN-THREE-SAT problem. This will be based on the construction already given in [19].

The formula ϕ' is constructed by replacing each clause in ϕ with three new clauses. Specifically, the j-th clause $C_j = x_j \vee y_j \vee z_j$ is replaced by $C'_j := (\neg x_j \vee a_j \vee b_j) \wedge (b_j \vee y_j \vee c_j) \wedge (c_j \vee d_j \vee \neg z_j)$ where a_j, b_j, c_j, d_j are four fresh propositional variables. It is quite straightforward to check that only a valuation that makes C_j true can be extended to a valuation that makes exactly one literal true in each of the clauses in C'_j. Notice that such a constructed ϕ' is a 3-CNF$_{\leq 4}$ formula, because this transformation does not increase the number of occurrences of any of the original variables in ϕ and each of the new variables is used at most twice.

Now, if there exists a valuation that makes every clause in ϕ true, then as argued above it can be extended to a valuation that makes exactly one literal true in every clause in ϕ'.

To show the other direction, let ν be a valuation that makes exactly one literal true in every clause in ϕ'. Consider for every $j = 1, \ldots, m$ the projection of ν on the set of variables occurring in the clause C_j. Suppose that such a valuation makes C_j false. It follows that it would not be possible to extend this valuation to a valuation that makes exactly one literal true in every clause in C'_j. However, we already know that ν is such a valuation, so this leads to a contradiction. □

Although Theorem 2 is of independent interest, all that we need it for is to prove our next theorem.

Theorem 3. *The* ALL-THE-SAME-SAT *problem for 3-CNF$_{\leq 4}$ formulae is* NP-*complete.*

Proof. The ALL-THE-SAME-SAT problem is clearly in NP, because we can simply guess a valuation and check whether it makes exactly the same number of literals in every clause true.

To proof NP-hardness, we reduce from the ONE-IN-THREE-SAT problem for 3-CNF$_{\leq 4}$ formulae (Theorem 2). Let ϕ be any 3-CNF$_{\leq 4}$ formula and let us consider a new formula $\phi' = \phi \vee (x \wedge x \wedge \neg x)$, where x is a fresh variable that does not occur in ϕ. Notice that ϕ' is also a 3-CNF$_{\leq 4}$ formula. We claim that ϕ is an instance of ONE-IN-THREE-SAT iff ϕ' is an instance of ALL-THE-SAME-SAT.

($\Rightarrow$) If ν is a valuation that makes exactly one literal in every clause in ϕ true, then extending it by setting $\nu'(x) = \perp$ would make exactly one literal in every clause in ϕ' true.

($\Longleftarrow$) Let ν be a valuation that makes the same number of literals in every clause in ϕ' true. It cannot be that this number is 0 or 3, because there is at least one true literal and one false literal in the clause $(x \wedge x \wedge \neg x)$.

If ν makes exactly one literal in every clause in ϕ' true, then the same holds for ϕ.

If ν makes exactly two literals in every clause true, then consider the valuation ν' such that $\nu'(y) = \neg \nu(y)$ for every propositional variable y in ϕ'. Notice that ν' makes exactly one literal in every clause in ϕ' true, so the same holds for ϕ. $\qquad\square$

5   Being Rational Makes You Stronger

In this section, we build on the results obtained in the previous two section and show strong NP-hardness of all the decision problems defined in Section 2. As a direct consequence, no pseudo-polynomial algorithm can exist for solving any of these problems unless P=NP. Instead of showing strong NP-hardness for each of these problems separably, we will show one "master" reduction for the UNBOUNDED SUBSET SUM problem instead. This reduction will then be reused to show strong NP-hardness of the PARTITION problem, and from these two results the strong NP-hardness of all the other problems will follow.

Theorem 4. *The* UNBOUNDED SUBSET SUM *problem with rational weights is strongly NP-complete.*

Proof. For a given instance $A = \{w_1, \ldots, w_n\}$ and target weight W, we know that the quantities q_i, for all $i = 1, \ldots, n$, have to satisfy $q_i \leq W/w_i$. This in fact shows that the problem is in NP, because all the quantities q_i when represented in binary can be written down in polynomial space and can be guessed at the beginning. We can then simply verify whether $\sum_{i=1}^{n} q_i \cdot a_i = W$ holds in polynomial time by adding the rational numbers inside this sum one by one (while representing all the numerators and denominators in binary).

To prove strong NP-hardness, we provide a reduction from the ALL-THE-SAME-SAT problem for 3-CNF$_{\leq 4}$ formulae (which is NP-complete due to Theorem 3). Assume we are given a 3-CNF$_{\leq 4}$ formula

$$\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$$

with m clauses $C_1, \ldots, C_m$ and n propositional variables $x_1, \ldots, x_n$, where $C_j = a_j \vee b_j \vee c_j$ for $j = 1, \ldots, m$ and each a_j, b_j, c_j is a literal equal to x_i or $\neg x_i$ for some i. For a literal l, we write that $l \in C_j$ iff l is equal to a_j, b_j or c_j. We will now construct a set of items A of size polynomial in $n + m$ and a polynomial weight W such that A with the total weight W is a positive instance of UNBOUNDED SUBSET SUM iff ϕ is satisfiable.

We first need to construct a list of $n + m$ different prime numbers $(p_1, \ldots, p_{n+m})$ that are all larger than $n + 5$. It suffices to pick $p_i = \pi_{i+n+5}$ for all i, because clearly $\pi_j > j$ for all j. Thanks to Corollary 1, we can list all these primes numbers in the unary notation in time and space polynomial in $n + m$.

The set A will contain one item per each literal. We set the weight of the item corresponding to the literal x_i to

$$1 + \frac{1}{p_i} - \frac{1}{p_{i\oplus_n 1}} + \sum_{\{j|x_i\in C_j\}} \left(\frac{1}{p_{n+j}} - \frac{1}{p_{n+j\oplus_m 1}} \right)$$

and corresponding to the literal $\neg x_i$ to

$$1 + \frac{1}{p_i} - \frac{1}{p_{i\oplus_n 1}} + \sum_{\{j|\neg x_i\in C_j\}} \left(\frac{1}{p_{n+j}} - \frac{1}{p_{n+j\oplus_m 1}} \right).$$

Notice that each of these weights is $\geq 1 - \frac{5}{p_1} > 0$, because each literal occurs at most four times in ϕ, and $p_1 > 5$ is the smallest prime number among p_i-s. At the same time, all of them are also $\leq 1 + \frac{5}{p_1} < 2$. Moreover, they can all be written in unary using polynomial space, because each literal occurs in at most four clauses and so this sum will have at most 11 terms in total. We can then combine all these terms into a single rational number. Its denominator will be at most equal to p_{n+m}^{10}, because p_{n+m} is the largest prime number among p_i-s and one of these terms is equal to 1. Its numerator has to be smaller than $2p_{n+m}^{10}$, because this rational number is < 2. So both of them will have size $\mathcal{O}((2n+m)^{10}\log^{10}(2n+m))$ when written down in unary, because $p_{n+m} = \pi_{2n+m+5} < 2(2n+m+5)\log(2n+m+5)$ due to Theorem 1. Set A has $2n$ such items and so all its elements' weights can be written down in unary using $\mathcal{O}(n(2n+m)^{10}\log^{10}(2n+m))$ space.

Notice that the total weight of A is equal to

$$2n + \sum_{i=1}^n \left(\frac{2}{p_i} - \frac{2}{p_{i\oplus_n 1}} \right) + \sum_{j=1}^m \left(\frac{3}{p_{n+j}} - \frac{3}{p_{n+j\oplus_m 1}} \right)$$

because there are $2n$ literals, each variable corresponds to two literals, and each clause contains exactly three literals. As both of the two sums in this expression are telescoping, we get that the total weight is in fact equal to $2n$. We claim that the target weight $W = n$ is achievable by picking items from A (each item possibly multiple times) iff ϕ is a positive instance of ALL-THE-SAME-SAT.

($\Rightarrow$) Let q_i and q_i' be the number of times an item corresponding to, respectively, literal x_i and $\neg x_i$ is chosen so that the total weight of all these items is n.

For $i = 1, \ldots, n$, we define $t_i := q_i + q_i'$. For $j = 1, \ldots, m$, we define t_{n+j} to be the number of times an item corresponding to a literal in C_j is chosen. For example, if $C_j = x_1 \vee \neg x_2 \vee x_5$ then $t_{n+j} = q_1 + q_2' + q_5$. Finally, let $T := \sum_{i=1}^n q_i + q_i'$ be the total number of items chosen. Notice that $T \leq W/(1 - \frac{5}{p_1}) < W/(1 - \frac{5}{n+5}) = n+5$.

Now the total weight of the selected items can be expressed using t_i-s as follows:

$$\sum_{i=1}^n t_i + \sum_{i=1}^n \frac{t_i - t_{i\ominus_n 1}}{p_i} + \sum_{j=1}^m \frac{t_{n+j} - t_{n+j\ominus_m 1}}{p_{n+j}} \tag{$\star$}$$

Notice that $|t_i - t_{i\ominus_n 1}| < n+5$ and $p_i > n+5$ for all $i = 1, \ldots, n$, and $|t_{n+j} - t_{n+j\ominus_m 1}| < n+5$ and $p_{n+j} > n+5$ for all $j = 1, \ldots, m$. It now follows from Lemma

8

1 that $(\star)$ can be equal to n if and only if $\sum_{i=1}^{n} t_i = n$, and $t_1 = t_2 = \ldots = t_n$, and $t_{n+1} = t_{n+2} = \ldots = t_{n+m}$. The first two facts imply that for all $i = 1, \ldots, n$, we have $t_i = 1$ and so exactly one item corresponding to either x_i or $\neg x_i$ is chosen. The last fact states that in each clause exactly the same number of items corresponding to its literals is chosen. It is now easy to see that the ALL-THE-SAME-SAT condition is satisfied by ϕ for the valuation ν such that, for all $i \in \{1, \ldots, n\}$, we set $\nu(x_i) = \top$ iff $q_i = 1$.

$(\Leftarrow)$ Let ν be a valuation for which ϕ satisfies the ALL-THE-SAME-SAT condition. We set the quantities q_i and q'_i, the number of times an item corresponding to the literal x_i and $\neg x_i$ is picked, as follows. If $\nu(x_i) = \top$ then we set $q_i = 1$ and $q'_i = 0$. If $\nu(x_i) = \bot$ then we set $q_i = 0$ and $q'_i = 1$.

Let us define t_i-s as before. Note that we now have $t_i = 1$ for all $i = 1, \ldots, n$ and $t_{n+1} = t_{n+2} = \ldots = t_{n+m}$, because the ALL-THE-SAME-SAT condition is satisfied by ν. We can now easily see from the expression $(\star)$ that the total weight of the just picked items is equal to n. $\qquad\square$

Although the strong NP-hardness complexity of the PARTITION problem does not follow from the statement of Theorem 4, it follows from its proof as follows.

Theorem 5. *The* PARTITION *problem with rational weights is strongly NP-complete.*

Proof. Just repeat the proof of Theorem 4 without any change. In this case we know that $q_i \in \{0, 1\}$, which does not make any difference to the used reasoning. Notice that the target weight W chosen in the reduction is exactly equal to half of the total weights of all the items in A, so the UNBOUNDED SUBSET SUM problem instance constructed can also be considered to be a PARTITION problem instance. $\qquad\square$

Now, as the SUBSET SUM problem is a generalisation of the PARTITION problem, we instantly get the following result.

Corollary 2. *The* SUBSET SUM *problem with rational weights is strongly NP-complete.*

Finally, we observe that the 0-1 KNAPSACK and UNBOUNDED KNAPSACK problems are generalisations of the SUBSET SUM and UNBOUNDED SUBSET SUM problems, respectively. To see this just restrict the weight and profit of each item to be equal to each other as well as require $V = W$. Any such an instance is a positive instance of 0-1 KNAPSACK (UNBOUNDED KNAPSACK) if and only if it is a positive instance of SUBSET SUM (respectively, UNBOUNDED SUBSET SUM).

Corollary 3. *The 0-1* KNAPSACK *and* UNBOUNDED KNAPSACK *problems with rational weights are strongly NP-complete.*

6 Approximability

In this section, we briefly discuss the counter-intuitive fact that the optimisation version of all the decision problems defined in Section 2 admit a fully polynomial-time approximation scheme (FPTAS) even though we just showed them to be strongly NP-complete. First, let us restate a well-known result concerning this.

Corollary 4 (Corollary 8.6 in [22]). *Let Π be an NP-hard optimisation problem satisfying the restrictions of Theorem 8.5 in [22] (first shown in [6]). If Π is strongly NP-hard, then Π does not admit an FPTAS, assuming $P \neq NP$.*

The crucial assumption made in Theorem 8.5 of [22] is that the objective function is integer valued, which does not hold in our case, so there is no contradiction.

First, let us formally define the optimisation version of some of the decision problems studied. The optimisation version of the 0-1 KNAPSACK problem with capacity W asks for a subset of items with the maximum possible total profit and whose weight does not exceed W. As for the SUBSET SUM problem, its optimisation version asks for a subset of items whose total weight is maximal, but $\leq W$. The optimisation version of the other decision problems from Section 2 can also be defined (see, e.g., [12]).

Now, let us formally define what we mean by an approximation algorithm for these problems. We say that an algorithm is a *constant factor approximation algorithm* with a *relative performance* ρ iff, for any problem instance, I, the cost of the solution that it computes, $f(I)$, satisfies:

- for a maximisation problem: $(1 - \rho) \cdot \mathrm{OPT}(I) \leq f(I) \leq \mathrm{OPT}(I)$
- for a minimisation problem: $\mathrm{OPT}(I) \leq f(I) \leq (1 + \rho)\mathrm{OPT}(I)$

where $\mathrm{OPT}(I)$ is the optimal cost for the problem instance I. We are particularly interested in polynomial-time approximation algorithms. A polynomial-time approximation scheme (PTAS) is an algorithm that, for every $\rho > 0$, runs in polynomial-time and has relative performance ρ. Note that the running time of a PTAS may depend in an arbitrary way on ρ. Therefore, one typically strives to find a fully polynomial-time approximation scheme (FPTAS), which is an algorithm that runs in polynomial-time in the size of the input and $1/\rho$.

We will focus here on defining an FPTAS for 0-1 KNAPSACK problem with rational coefficients. An FPTAS for the other optimisation problems considered in this paper can be defined in an essentially the same way and thus their details are omitted.

Theorem 6. *The 0-1 KNAPSACK problem with rational coefficients admits an FPTAS.*

Proof. We claim that we can simply reuse any FPTAS for the 0-1 KNAPSACK problem with integer coefficients for this problem. Let I be a 0-1 KNAPSACK instance with rational coefficients. We turn I into an instance with integer coefficients only, I', by the usual trick of multiplying all the rational coefficients by the least common multiple of the denominators of all the rational coefficients in I. Let us denote this least common multiple by α. Assuming that all the coefficients in I are represented in binary, then when multiplying them by α (again in binary representation), their size will only increase polynomially. Therefore, the size of I' is just polynomially larger than I.

Notice that $\alpha \cdot \mathrm{OPT}(I) = \mathrm{OPT}(I')$. In fact, the profit of any subset of items A in I', denoted by $\mathsf{profit}'(A)$, is α times bigger than the profit of this set of items in I, denoted by $\mathsf{profit}(A)$. Let us now run on I' any FPTAS for 0-1 KNAPSACK problem with integer coefficients with relative performance ρ (e.g., [9]). This will return as a solution a subset of items, B, such that $\mathsf{profit}'(B) \geq (1 - \rho)\mathrm{OPT}(I')$. This implies that $\mathsf{profit}(B) \geq (1 - \rho)\mathrm{OPT}(I)$ so the same subset of items B has also the same relative performance ρ on the original instance I. □

7 Conclusions

In this paper we studied how the computational complexity of the PARTITION, 0-1 SUBSET SUM, UNBOUNDED SUBSET SUM, 0-1 KNAPSACK, UNBOUNDED KNAPSACK problems changes when items' weights and profits can be any rational numbers. We showed here, as opposed to the setting where all these values are integers, that all these problems are strongly NP-hard, which means that there does not exists a pseudo-polynomial algorithm for solving them unless P=NP. Nevertheless, we also showed that all these problem admit an FPTAS just like in the original setting. Finally, we just want to point out that if we restrict ourselves to only rational weights and profits with a finite representation as decimal numerals, then these problems are no longer strongly NP-complete. This is because we can just multiply all the input numbers by sufficiently high power of 10 and get an instance, with all integer coefficients, whose size is polynomial in the size of the original instance.

References

1. LG Babat. Linear functions on n-dimensional unit cube. *Doklady Akademii Nauk SSSR*, 221(4):761–762, 1975.
2. Chandra Chekuri and Sanjeev Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing*, 35(3):713–728, 2005.
3. Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.
4. Edith Elkind, Leslie Ann Goldberg, Paul W Goldberg, and Michael Wooldridge. On the computational complexity of weighted voting games. *Annals of Mathematics and Artificial Intelligence*, 56(2):109–131, 2009.
5. Giorgio Gallo, Peter L Hammer, and Bruno Simeone. Quadratic knapsack problems. In *Combinatorial optimization*, pages 132–149. Springer, 1980.
6. Michael R Garey and David S Johnson. "Strong" NP-Completeness Results: Motivation, Examples, and Implications. *Journal of the ACM (JACM)*, 25(3):499–508, 1978.
7. Christelle Guéret and Christian Prins. A new lower bound for the open-shop problem. *Annals of Operations Research*, 92:165–183, 1999.
8. JA Hoogeveen, Henk Oosterhout, and Steef L van de Velde. New lower and upper bounds for scheduling around a small common due date. *Operations research*, 42(1):102–110, 1994.
9. Oscar H Ibarra and Chul E Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM (JACM)*, 22(4):463–468, 1975.
10. David S Johnson and KA Niemi. On knapsacks, partitions, and a new dynamic programming technique for trees. *Mathematics of Operations Research*, 8(1):1–14, 1983.
11. Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.
12. Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.
13. Leonid Anatolevich Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3):115–116, 1973.
14. George B Mathews. On the partition of numbers. *Proceedings of the London Mathematical Society*, 1(1):486–490, 1897.
15. Mahmoud A. A. Mousa, Sven Schewe, and Dominik Wojtczak. Optimal Control for Multi-Mode Systems with Discrete Costs. In *Proc. of FORMATS*, pages 77–96. Springer, September 2017.
16. Ulrich Pferschy and Joachim Schauer. The knapsack problem with conflict graphs. *J. Graph Algorithms Appl.*, 13(2):233–249, 2009.

17. David Pisinger. An exact algorithm for large multiple knapsack problems. *European Journal of Operational Research*, 114(3):528–541, 1999.

18. J Barkley Rosser and Lowell Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois Journal of Mathematics*, 6(1):64–94, 1962.

19. Thomas J Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 216–226. ACM, 1978.

20. Martello Silvano and Toth Paolo. Knapsack problems: algorithms and computer implementations, 1990.

21. Craig A Tovey. A simplified np-complete satisfiability problem. *Discrete applied mathematics*, 8(1):85–89, 1984.

22. Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.