Subset Sum Is NP-Complete

In this lecture, we prove NP-hardness of the subset sum (= exact change) problem.

1 What Is Subset Sum Problem

Motivation. We have coins valued at s_1, ..., and s_n cents. We need to pay the amount S. Can we pay it without change? In other words, can we select a subset of the set of the coins so that thee sum of their values is exactly S?

Example. Suppose that we have three coins of 1 cent ($s_1 = s_2 = s_3 = 1$), one 5-cent coin $s_4 = 5$, two dimes $s_5 = s_6 = 10$, and one quarter $s_7 = 25$. Then:

- if we need to pay $S = 42$ cents, we can do it, since

$$s_1 + s_2 + s_4 + s_5 + s_7 = 1 + 1 + 5 + 10 + 25 = 42;$$

- on the other hand, if we need to pay $S = 19$ cents, then, as one can check, there is no way to give exact change.

Comment. For the US coins, the problem is simple, since the coins are simple: 1, 5, 10, 25, 50, and 100 cents. However, historically, there were coins of not very usual value. For example:

- In Russia until the 1990s, we had 3-kopeck and 15-kopeck coins. This has historical reasons: Russian conquered another kingdom that used altyns, 1 altyn was 3 kopecks, so a natural 5-altyn coin became a 15-kopeck coin.
- In the past, in Russia, there were also efimkas – 64-kopeck coins. This coin corresponded to Joachimsthalers, perfect quality silver coins produced in the German town of Joachimsthal. In other countries, the name of the same coin was abbreviated to its second part – thalers, and this name got pronounced as dollars.
- Russia was not the only country with a weird coin system. For example, until the 1970s, England had shilling coins (12 pence), pound coins (20 shillings = 240 pence), and guinea coins (21 shillings = 252 pence).

How can we describe this problem in precise terms? For each coin i, either we select it or we don't. We can describe it by introducing, for each coin i, the value t_i:

- t_i be true (i.e., 1) if we select the i-th coin, and
- t_i is false (i.e., 0) if we don't select this coin.

For example, the above arrangement for paying 42 cents can be described as

$$t_1 = 1, \quad t_2 = 1, \quad t_3 = 0, \quad t_4 = 1, \quad t_5 = 1, \quad t_6 = 0, \quad t_7 = 1.$$

In general, the overall value of all selected coins is equal to

$$t_1 \cdot s_1 + \ldots + t_n \cdot s_n.$$

Indeed, in this sum:

- the value of each coin which is not selected is multiplied by 0 – thus ignored,
- so we only count coins that are selected.

In these terms, the problem of finding the exact change can be formulated as follows.

Subset sum problem.

- *given:* positive integers $s_1, \ldots, s_n$ and S;
- *find:* the values t_i (equal to 0 or 1) for which $t_1 \cdot s_1 + \ldots + t_n \cdot s_n = S$ (or produce a message that this equality is not possible).

How we prove that this problem is NP-complete. We prove it by showing that 3-SAT can be reduced to this problem.

2 The Algorithm U_1 That Transforms a 3-CNF Formula x into an Instance $x' = U_1(x)$ of the Subset Sum Problem

Input to the algorithm. We start with a 3-CNF formula $C_1 \,\&\, \ldots \,\&\, C_k$, in which each clause C_j has:

- either a form $a \vee b$
- or a form $a \vee b \vee c$,

where a, b, and c are literals, i.e.,

- either the Boolean variables $v_1, \ldots, v_\ell$
- or their negations $\neg v_1, \ldots, \neg v_\ell$.

Construction. Let us build a table with:

- $k + \ell$ columns $v_1, \ldots, v_\ell$, $C_1, \ldots, C_k$, and
- $2k + 2\ell + 1$ rows $v_1, \neg v_1, \ldots, v_\ell, \neg v_\ell$, $C'_1, C''_1, \ldots, C'_k, C''_k$, and S.

This table will be filled with 0s and 1s – with the exception of the last row. The table is filled as follows:

- for each column v_i, we place 1 in rows v_i and $\neg v_i$ and in row S, and we place 0 in all other rows;
- for each column C_j, we place 1 in the rows corresponding to the literals that form the clause C_j and to the rows C'_j and C''_j, we place 3 in row S, and we place 0 in all other rows.

In the last row, we thus get ℓ 1s followed by k 3s.

Then:

- the first $n = 2k + 2\ell$ rows will describe decimal numbers $s_1, \ldots, s_n$ (i.e., the coin values), and
- the last row will describe the sum S that we want to pay.

Example. Let us start with a 3-CNF formula $(\neg v_1 \vee v_2) \,\&\, (v_1 \vee \neg v_2 \vee v_3)$. Here, we have:

- three Boolean variables v_1, v_2, and v_3 (so that $\ell = 3$), and
- $k = 2$ clauses $C_1 = \neg v_1 \vee v_2$ and $C_2 = v_1 \vee \neg v_2 \vee v_3$.

By applying the above algorithm, we get the following table:

	v_1	v_2	v_3	C_1	C_2
v_1	1	0	0	0	1
$\neg v_1$	1	0	0	1	0
v_2	0	1	0	1	0
$\neg v_2$	0	1	0	0	1
v_3	0	0	1	0	1
$\neg v_3$	0	0	1	0	0
C'_1	0	0	0	1	0
C''_1	0	0	0	1	0
C'_2	0	0	0	0	1
C''_2	0	0	0	0	1
S	1	1	1	3	3

This means that we want to obtain the value $S = 11133$ by using the following $2k + 2\ell = 10$ coins:

- the coins $v_1 = 10001$, $\neg v_1 = 10010$, $v_2 = 01010$, $\neg v_2 = 01001$, $v_3 = 00101$, $\neg v_3 = 00100$, and
- the coins $C'_1 = C''_1 = 00010$ and $C'_2 = C''_2 = 00001$.

3 Algorithm U_2 That Transforms a Solution y' to the Exact Change Problem into a Satisfying Vector $y = U_2(y')$ for the Original Formula x

Algorithm. Once you have the selection of coins whose values add up to S, then, for each Boolean variable v_i:

- make this variable true if the coin v_i is in this selection, and
- make the variable v_i false if the coin $\vec{v}_i$ is in this selection.

Why this words. First, let us show that for each variable v_i, either the coin v_i or the coin $\neg v_i$ will be in the selection – but not both. Indeed:

- If none of these two coins will be in the selection, then in the column v_i, the sum will be 0 – and the sum S has 1 in this column.
- If both coins were selected, then in the column v_i, the sum will be $1+1 = 2$, and the sum S has 1 in this column.

Let us now show that the resulting collection of literals satisfies the original formula, i.e., makes each of its clauses C_j true. Indeed, in the C_j column, the sum is 3, and the C_i' and C_i'' rows can add to at most 2. Thus, there is at least one 1 in this column – which means exactly that the corresponding literal is in the set of selected coins, hence true – and thus, the whole clause is true.

Example. In the above example, we can select coins $\neg v_1$, $\neg v_2$, v_3, C_1', C_1'', and C_2'. One can check that the sum of their values is indeed 11133:

	v_1	v_2	v_3	C_1	C_2
$\neg v_1$	1	0	0	1	0
$\neg v_2$	0	1	0	0	1
v_3	0	0	1	0	1
C_1'	0	0	0	1	0
C_1''	0	0	0	1	0
C_2'	0	0	0	0	1
S	1	1	1	3	3

Then, according to the algorithm, we select v_1 =“false”, v_2 =“false”, and v_3 =“true”. One can see that for this selection, the original formula is true.

4 Algorithm U_3 That Transforms a Satisfying Vector y for the Original 3-CNF Formula x into a Solution $y' = U_2(y)$ to the Corresponding Subset Sum Problem $x' = U_1(x)$

Algorithm. Suppose that have a satisfying vector for the original 3-CNF formula. Then, we select the coins corresponding to the literals which are true. This guarantees that the sums in the v_i columns are 1s.

For each C_j-column, add one of two coins C_j' and/or C_j'' if in this column, with coins corresponding to literals, we get less than 3.

Example. In the above formula, one of the satisfying vectors is $v_1 =$"false", $v_2 =$"false", and $v_3 =$"true". For this vector, the following literals are true: $\neg v_1$, $\neg v_2$, and v_3. Let us select the corresponding coins; these selections are marked by placing the corresponding row in bold:

	v_1	v_2	v_3	C_1	C_2
v_1	1	0	0	0	1
$\neg\mathbf{v_1}$	**1**	**0**	**0**	**1**	**0**
v_2	0	1	0	1	0
$\neg\mathbf{v_2}$	**0**	**1**	**0**	**0**	**1**
$\mathbf{v_3}$	**0**	**0**	**1**	**0**	**1**
$\neg v_3$	0	0	1	0	0
C_1'	0	0	0	1	0
C_1''	0	0	0	1	0
C_2'	0	0	0	0	1
C_2''	0	0	0	0	1
S	1	1	1	3	3

The sum of the values of the selected coins is 11112. So, to get 11133, we add:

- two coins C_1' and C_1'' corresponding to the clause C_1, and
- a coin C_2'' corresponding to the clause C_2.

Now, the sum of the values of all selected coins is exactly $S = 11133$:

	v_1	v_2	v_3	C_1	C_2
v_1	1	0	0	0	1
$\mathbf{\neg v_1}$	**1**	**0**	**0**	**1**	**0**
v_2	0	1	0	1	0
$\mathbf{\neg v_2}$	**0**	**1**	**0**	**0**	**1**
$\mathbf{v_3}$	**0**	**0**	**1**	**0**	**1**
$\neg v_3$	0	0	1	0	0
$\mathbf{C_1'}$	**0**	**0**	**0**	**1**	**0**
$\mathbf{C_1''}$	**0**	**0**	**0**	**1**	**0**
$\mathbf{C_2'}$	**0**	**0**	**0**	**0**	**1**
C_2''	0	0	0	0	1
S	1	1	1	3	3

Try it on the example of a formula $(v_1 \vee \neg v_2 \vee v_3) \,\&\, (\neg v_1 \vee v_2 \vee \neg v_3)$.

Homework. Now, you should be ready to do the corresponding homework.