

Lecture 21: November 26

Lecturer: Micah Adler

Scribes: Xiaotao Liu and Heng Xu

### 21.1 The Subset-Sum Problem

**Definition 21.1** *The subset sum problem.*

**Input:** A set  $S$  of integers,  $\{s_1, s_2, \dots, s_l\}$ , and a target integer  $t$ .

**Question:** Does  $S$  have a subset  $S' \subseteq S$  such that the  $s_i \in S'$  sum exactly to  $t$ ?

**Example 21.2** *Two Subset-Sum problems:*

- $S = \{1, 3, 3, 7\}, t = 13$  - Yes, such a  $S'$  exists ( $S' = \{3, 3, 7\}$ ).
- $S = \{1, 3, 3, 7\}, t = 9$  - No subset  $S'$  exists that sums to 9.

**Theorem 21.3** *The subset sum problem is NP-Complete.*

**Proof:**

1. The Subset Sum problem is in **NP**.

Verifying that a witness  $W' \subseteq S$  sums up to  $t$  can be done in polynomial time.

2.  $3\text{-SAT} \leq_p \text{Subset-Sum}$ .

**Proposed transformation:**

Given  $\Phi$ , construct a set  $S$  of integers and a target integer  $t$  such that there is a valid sum if and only if  $\Phi$  is satisfiable.

The integers  $s_i$  (represented in the decimal system) have two parts:

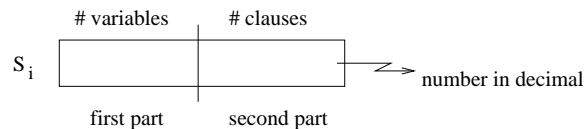


Figure 21.1: The two parts of an integer  $s$ .

For each variable  $X_i$  two integers are constructed,  $s_i$  to represent  $X_i$ , and  $s'_i$  to represent  $\bar{X}_i$ .

The first parts of these integers are  $s_i : 10^{i-1}$  and  $s'_i : 10^{i-1}$ .

The second parts of these integers are constructed as follows:

- If  $X_i$  is in clause  $j$ , the  $j^{th}$  digit from the right in  $S_i$  is a 1 otherwise the  $j^{th}$  digit is 0.
- If  $\bar{X}_i$  is in clause  $j$ , the  $j^{th}$  digit from the right in  $S'_i$  is a 1 otherwise the  $j^{th}$  digit is 0.

Additionally we add helper integers  $h_i$  to  $S$ , for each clause  $j$ , add two copies of  $10^{j-1}$ .

Finally, the target integer is of the form:  $t = \overbrace{11\dots\dots 11}^{\# \text{ of variables}} \overbrace{33\dots\dots 33}^{\# \text{ of clauses}}$

**Example 21.4** Following is the set of integers  $S$  constructed using the previous rules for the example.

$$\phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee x_4) \quad t = 1111333 \text{ (because we have 4 variables and 3 clauses)}$$

Set 'S'	First Part	Second Part
$S_1$	1	001
$S'_1$	1	010
$S_2$	10	110
$S'_2$	10	001
$S_3$	100	001
$S'_3$	100	100
$S_4$	1000	100
$S'_4$	1000	010
$h_1$		1
$h_2$		1
$h_3$		10
$h_4$		10
$h_5$		100
$h_6$		100

Table 21.1: Reduction from CNF to a Subset sum problem.  $S_i$ 's are generated using the rules for construction of the *first* and *second* parts of each integer.  $h_i$ 's are the *helper* integers required in the set  $S$ .

Now we need to show that this construction given  $\Phi$  yields a set  $S$  of integers and a target integer  $t$  such that:

**Claim 21.5** *The following are equivalent:*

1.  $\Phi$  is satisfiable.
2. There is a subset  $S'$  that solves the subset sum problem  $(S, t)$ .

**Proof:** Let  $v$  be the number of variables and  $m$  the number of clauses in  $\Phi$ .

- $\Rightarrow$ :  $\Phi$  is satisfiable. This means that there is a truth assignment  $A$  of the variables in  $\Phi$  such that  $\Phi(A)$  is *true*. We include in the subset  $S'$  for each variable  $X_i$  in  $\Phi$  either  $s_i$ , if  $X_i = \text{true}$  under  $A$ , or  $s'_i$ , if  $X_i = \text{false}$ . This guarantees that the first part of the integers in  $S'$  sum up correctly as no integers have the same digits set to 1 except for  $s_i$  and  $s'_i$  in the first part. Since  $\Phi(A)$  is *true*, each clause of  $\Phi$  must be *true* under  $A$ . This means that for every clause  $c = 1, 2, \dots, m$ , there must be at least one integer  $s_i$  or  $s'_i$  in the so far constructed  $S'$  subset such that the  $c^{\text{th}}$  digit in its second part is 1. Thus for each clause  $c$ , we can add zero, one or both helper variables  $h_c, h'_c$  to  $S'$  to equal the required 3 at the  $c^{\text{th}}$  digit in the second part of  $t$ . Thus adding all selected integers ( $s_i$  or  $s'_i$ ) and if necessary the helper integers for each clause to  $S'$ , we have constructed a valid solution subset  $S'$  for the subset problem  $(S, t)$ .
- $\Leftarrow$ : Say for  $(S, t)$  a subset  $S'$  is given such that its elements sum exactly up to  $t$ . In order to equate the first part of  $t$ ,  $S'$  must contain for all  $i = 1, 2, \dots, v$  either  $s_i$  or  $s'_i$ . Otherwise the  $i^{\text{th}}$  digit of the first

part of  $t$  would be 2, if both  $s_i$  and  $s'_i$  were in  $S'$ , or 0, if none of them were in  $S'$ . So by inspecting if  $s_i$  or  $s'_i$  is in  $S'$ , we can construct an assignment  $A$  for the formula  $\Phi$ : if  $s_i$  is in  $S'$  then  $X_i = \text{true}$ , otherwise ( $s'_i \in S'$ )  $X_i = \text{false}$ .

We now show via proof by contradiction that  $\Phi(A)$  must be *true*:

Assume that  $\Phi(A)$  is *false*. Then one clause (say it is the  $c^{\text{th}}$  clause) in  $\Phi$  must be *false*. This corresponds to the situation that in  $S'$  no integer  $s_i$  or  $s'_i$  is included that has a 1 at its  $c^{\text{th}}$  digit in its second part. (This is true because if such an  $s_i$  existed the assignment  $A$  would dictate that  $X_i = \text{true}$  and the  $c^{\text{th}}$  digit being 1 means after the construction that the literal  $X_i$  is in the  $c^{\text{th}}$  clause. But with  $X_i = \text{true}$  under  $A$ , we have that the literal  $X_i$  in the  $c^{\text{th}}$  clause is *true*. Thus the  $c^{\text{th}}$  clause is *true* under  $A$ . This contradicts the fact that it was just assumed to be *false* under  $A$ . A similar argument holds if there exists an  $s'_i$  such that it has a 1 at its  $c^{\text{th}}$  digit in its second part.) But if there is no integer  $s_i$  or  $s'_i$  having a 1 at the  $c^{\text{th}}$  digit in its second part, the only integers of  $S'$  that can contribute to the  $c^{\text{th}}$  digit of  $t$  in its second part are the two helper variables  $h_c$  and  $h'_c$ . But these two can only contribute in total 2 to the  $c^{\text{th}}$  digit of  $t$ 's second part, and the  $c^{\text{th}}$  digit of  $t$  must be equal to 3. So the subset  $S'$  cannot add up to  $t$  since the construction does not allow any carries. This contradicts the fact that  $S'$  was given to be a valid solution to the subset sum problem.

Thus  $\Phi(A)$  must be *true*. ■

This concludes the proof. ■

## 21.2 Max-Cut Problem

We want to prove that the max-cut problem is **NP-Complete**. To do this, we'll first prove that a problem called not-all-equal 3-SAT (NA= 3-SAT) is **NP-Complete**. We'll then reduce NA= 3-SAT to max-cut.

## 21.3 Not All = 3-SAT (NA = 3-SAT)

**Definition 21.6** *The NA= 3-SAT problem.*

**Input:** A 3-CNF formula  $\Phi$ .

**Question:** *Is there a satisfying assignment to  $\Phi$  such that there is at least one false literal per clause?*

**Theorem 21.7** *NA= 3-SAT is NP-Complete.*

**Proof:** We can verify that an assignment is NA= satisfying in polynomial time. So NA= 3-SAT is in **NP**.

Now we show that  $3\text{-SAT} \leq_p \text{NA} = 3\text{-SAT}$ .

We transform an input  $\Phi$  to 3-SAT problem into an input  $\Phi'$  to NA= 3-SAT. For each clause  $(l_{i1} \vee l_{i2} \vee l_{i3})$ , we add two variables  $x_i$  and  $y_i$  and three clauses as follows. (The variable  $\alpha$  is another one we introduce that we share between clauses.)

$$\begin{aligned}\Phi &= \dots \wedge (l_{i1} \vee l_{i2} \vee l_{i3}) \wedge \dots \\ \Phi' &= \dots \wedge (l_{i1} \vee l_{i2} \vee x_i) \wedge (\neg x_i \vee l_{i3} \vee y_i) \wedge (x_i \vee y_i \vee \alpha) \dots\end{aligned}$$

Note that the transformation from  $\Phi$  to  $\Phi'$  can be done in polynomial time because there is constant amount of work being done per clause.

**Claim 21.8** *If  $\Phi$  is satisfiable, then  $\Phi'$  is NA= satisfiable.*

**Proof:** Given a satisfying assignment to  $\Phi$ , we construct a satisfying assignment to  $\Phi'$ . Set  $\alpha = \text{true}$ . Ignoring the case where  $l_{i1}$ ,  $l_{i2}$ , and  $l_{i3}$  are all false, there are 7 possible settings of  $l_{i1}$ ,  $l_{i2}$ , and  $l_{i3}$ . Examine all possible settings. In each setting,  $\Phi'$  is satisfied.

*ex. if  $l_{i1}$ ,  $l_{i2}$ , and  $l_{i3}$  are all true,*

$l_{i1} = \text{true}$   
 $l_{i2} = \text{true}$   
 $l_{i3} = \text{true}$   
 $x_i = \text{false}$   
 $\neg x_i = \text{true}$   
 $y_i = \text{false}$   
 $\alpha = \text{true}$

■

**Claim 21.9** *If  $\Phi'$  is NA-satisfiable, then  $\Phi$  is satisfiable.*

**Proof:** Assume  $\alpha = \text{true}$  ( $\Phi'$  has at least 1 true and and 1 false per clause. If  $\alpha$  is false in  $\Phi'$ , we can flip it to true, and some other literal in the same clause that is true to false).

At least one of  $l_{i1}$ ,  $l_{i2}$ , or  $l_{i3}$  must be true. ■

This concludes the proof that NA= 3-SAT is **NP-Complete**. ■

## 21.4 Proof that Max-Cut is NP-Complete

Recall that the decision version of max-cut is as follows.

Input: A graph  $G$  and an integer  $k$ .

Question: Does  $G$  have a cut of size at least  $k$ ?

**Theorem 21.10** *Max-cut is NP-Complete.*

**Proof:** Given binary string  $X$ , it is easy to verify in polynomial time that  $X$  describes a valid cut in  $G$  of size  $k$ . So max-cut is in **NP**.

We now show that max-cut is **NP-Hard**. We do this by reducing NA= 3-SAT to max-cut. Let  $\phi$  be an input to NA= 3-SAT. First, to simplify the reduction, we convert  $\phi$  into a restricted but equivalent form. Convert  $\phi$  into a formula  $\phi'$  such that

1. No clause has the form  $(X \vee \neg X \vee \dots)$ , where  $X$  is any literal. This is easy to ensure because the clause will always be satisfied, so we can simply remove it from  $\phi'$ .

- No two clauses share a pair of literals. That is, no two clauses have the form  $(a \vee b \vee c)$  and  $(a \vee b \vee d)$ , for literals  $a, b, c$ , and  $d$ . If  $\phi$  contains any clauses of this form, we can introduce two variables  $x$  and  $y$  and substitute for  $(a \vee b \vee c) \wedge (a \vee b \vee d)$  the equivalent expression  $(a \vee b \vee c) \wedge (a \vee x \vee d) \wedge (b \vee \bar{x} \vee y) \wedge (\bar{b} \vee x \vee y)$ .

So  $\phi'$  is NA= satisfiable if and only if  $\phi$  is NA= satisfiable.

Now, we can use  $\phi'$  to construct an input to the max cut problem. Define a graph  $G_\phi$  as follows:

- Add two vertices  $X_i$  and  $\bar{X}_i$  for every variable in  $\phi'$ .
- Add an edge between two vertices  $X_i$  and  $X_j$  if and only if either (a)  $X_i$  and  $X_j$  occur in the same clause, or (b)  $X_i = \neg X_j$ . Let  $k_\phi = 2m + v$ , where  $m$  is the number of clauses in  $\phi'$  and  $v$  is the number of variables in  $\phi'$ .

Here is an example of this reduction. Suppose  $\phi = (a \vee \bar{b} \vee c) \wedge (\bar{a} \vee \bar{b} \vee \bar{c})$ . Then  $G_\phi$  is given in figure 19.2 and  $k_\phi = v + 2m = 6$ .

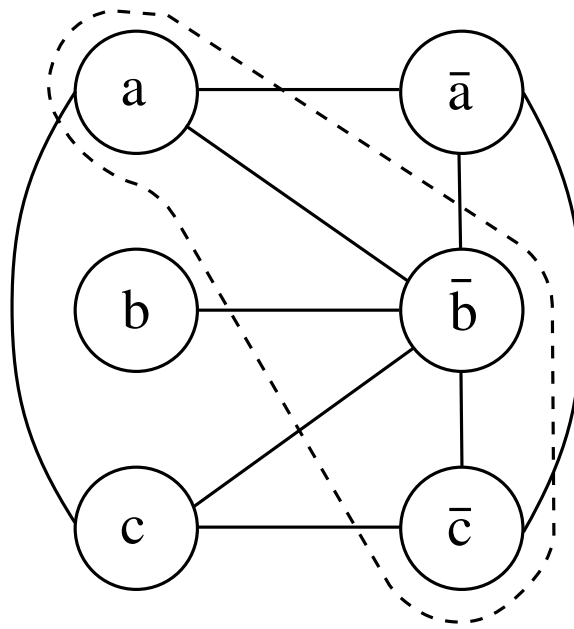


Figure 21.2: Reduction of NA=3-SAT to max-cut. The dashed region is a maximum cut of the graph.

Clearly,  $G_\phi$  and  $k_\phi$  can be calculated in polynomial time.

So we need to prove that this is a valid reduction.

- Let us show that if  $\phi$  is NA= satisfiable, then  $G_\phi$  has a cut of size  $k_\phi$ . Suppose that  $\phi$  has some satisfying assignment. Construct a cut in  $G_\phi$  as follows. Put the vertices that correspond to true literals one side of the cut, and other vertices on the other side.

The cut will cross one edge for each variable in  $\phi$ . Also, for each clause, two literals will be on one side of the cut and one literal will be on the other, because  $\phi$  is NA= satisfied. Therefore, the cut will cross 2 edges for each clause. So the size of the cut is  $v + 2m$ .

2. We show that if  $G_\phi$  has a cut of size  $k_\phi$ , then  $\phi$  is NA= satisfiable. We added one edge in the graph for every variable, and three for every clause. None of these edges are shared between clauses because we specified that no pairs of literals appears in two different clauses. In addition, no clause shares an edge with a variable because we specified that no variable and its negation appears in the same clause. So the number of edges in the graph is  $v + 3m$ .

Each variable can contribute at most one edge to the cut. Each clause can contribute at most two vertices to the cut. So the maximum cut possible has size  $v + 2m$ . If there exists a cut of this size, then it must cut every clause and every variable.

This means that we can construct a NA= satisfying assignment for  $\phi$  by choosing either side of the cut to represent true literals.

■

## 21.5 Other NP–Complete problems

### 21.5.1 Max-bisection

**Definition 21.11** A bisection  $(A, V - A)$  of a graph  $G = (V, E)$  is a partition of  $V$  such that  $|A| = \lfloor |V| / 2 \rfloor$ .

**Definition 21.12** The size of a bisection  $(A, V - A)$  is the number of edges that cross the cut defined by  $(A, V - A)$ .

**Input:** A graph  $G$  and an integer  $k$ .

**Question:** Does  $G$  have a bisection of size at least  $k$ ?

### 21.5.2 Min-Bisection

**Input:** A graph  $G$  and an integer  $k$ .

**Question:** Does  $G$  have a bisection of size at most  $k$ ?

This problem can be proven by reduction from the max-bisection problem using the complement of  $G$ .

### 21.5.3 Hamiltonian Path

**Input:** A graph  $G$ , and two vertices  $a, b \in V_G$ .

**Question:** Is there a path from  $a$  to  $b$  that goes through every node exactly once?

## 21.6 Reference

CLR90 THOMAS H. CORMEN, CHARLES E. LEISERSON, and RONALD L. RIVEST, Introduction to Algorithms, 1990.

FK2000 DIRK FACKEN, PURU KULKARNI, Scribe Notes, Lecture 18, Advanced Algorithms Fall 2000

SK2000 CHARLES SUTTON, YOSHIGA KINUTA, Scribe Notes, Lecture 19, Advanced Algorithms Fall 2000