# Lab11-NP Reduction

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2020.

∗ If there is any problem, please contact TA Shuodian Yu.
∗ Name:Hongjie Fang Student ID:518030910150 Email: galaxies@sjtu.edu.cn

1. What is the "certificate" and "certifier" for the following problems?

   (a) *ZERO-ONE INTEGER PROGRAMMING*: Given an integer $m \times n$ matrix $A$ and an integer $m$-vector $b$, is there an integer $n$-vector $x$ with elements in the set $\{0, 1\}$ such that $Ax \leq b$.

   (b) *SET PACKING*: Given a finite set $U$, a positive integer $k$ and several subsets $U_1, U_2, \ldots, U_m$ of $U$, is there $k$ or more subsets which are disjoint with each other?

   (c) *STEINER TREE IN GRAPHS*: Given a graph $G = (V, E)$, a weight $w(e) \in Z_0^+$ for each $e \in E$, a subset $R \subset V$, and a positive integer bound $B$, is there a subtree of $G$ that includes all the vertices of $R$ and such that the sum of the weights of the edges in the subtree is no more than $B$.

**Solution.** Here are the certificates and certifiers for the problems above.

(a) *ZERO-ONE INTEGER PROGRAMMING*:

- **Certificate**. An integer $n$-vector $x$ with elements in the set $\{0, 1\}$.
- **Certifier**. Check that the inequality $Ax \leq b$ holds for the given matrix $A$, vector $b$ and certificate vector $x$.

(b) *SET PACKING*:

- **Certificate**. A set $\mathcal{U}$ which contains $k$ or more subsets $U_{i_1}, U_{i_2}, \cdots, U_{i_k}, \cdots$.
- **Certifier**. Check that the subsets in set $\mathcal{U}$ are disjoint with each other.

(c) *STEINER TREE IN GRAPHS*:

- **Certificate**: A subtree $T$ of $G$ includes all the vertices of the given subset $R$.
- **Certifier**: Check that the sum of weights of the edges in the subtree $T$ is no more than a given number $B$.

□

2. Algorithm class is a democratic class. Denote class as a finite set $S$ containing every students. Now students decided to raise a student union $S' \subseteq S$ with $|S'| \leq K$ .

   As for the members of the union, there are many different opinions. An opinion is a set $S_o \subseteq S$. Note that number of opinions has nothing to do with number of students.

   The question is whether there exists such student union $S' \subseteq S$ with $|S'| \leq K$, that $S'$ contains at least one element from each opinion. We call this problem *ELECTION* problem, prove that it is NP-complete.

**Theorem 1.** *ELECTION problem is an NP problem.*

**Proof.** Here are the certificate and certifier of the *ELECTION* problem.

- **Certificate**. A student union $S' \subseteq S$ with $|S'| \leq K$.
- **Certifier**. Check that $S'$ contains at least one student from each given opinion.

Suppose the total number of students in all given opinions, which is also the length of the input, is $n$. Given a certificate $t$, we can implement an $O(n)$ time certifier by checking if each student in each opinion is in the certificate student union $S'$. Therefore, *ELECTION* problem is an NP problem. □

**Lemma 2.** 3-SAT $\leq_p$ *ELECTION*.

**Proof.** Given an instance $\Phi$ of 3-SAT which contains $n$ elements, we construct an instance of *ELECTION* that has a student union $S' \subseteq S$ with $|S'| \leq K$ which contains at least one student from each given opinion if and only if $\Phi$ is satisfiable.

**Construction.** First we can construct an algorithm class with $2n$ students, and let us call them $S_1, S_2, \cdots, S_n, \tilde{S_1}, \tilde{S_2}, \cdots, \tilde{S_n}$, which correspond to $x_1, x_2, \cdots, x_n, \overline{x_1}, \overline{x_2}, \cdots, \overline{x_n}$ in $\Phi$ respectively. Then we construct the opinions as follows.

- For each clause in $\Phi$, we construct an opinion only containing the three corresponding students. For example, a clause $(x_1 \vee \overline{x_2} \vee x_4)$ in $\Phi$ corresponds to an opinion $\{S_1, \tilde{S_2}, S_4\}$ in our instance of *ELECTION* problem.
- We add $n$ extra opinions $\{S_1, \tilde{S_1}\}, \{S_2, \tilde{S_2}\}, \cdots \{S_n, \tilde{S_n}\}$ in our instance of *ELECTION* problem.

Finally we set $K = n$.

**Claim and Proof.** The corresponding *ELECTION* instance constructed by $\Phi$ is satisfiable if and only if $\Phi$ is satisfiable. Here is the specific proof.

- ($\Longleftarrow$) Given a satisfying assignment of $\Phi$, we can construct the student union based on the satisfying assignment as follows.
  - If $x_i$ is in the assignment, then we choose $S_i$;
  - If $x_i$ is not in the assignment, that is, $\overline{x_i}$ is in the assignment, then we choose $\tilde{S_i}$.

  Therefore, the number of students in the student union is exactly $n$. Since every clause in 3-SAT is satisfied, there must exist a student in the student union in every opinion constructed by clause according to the corresponding relations above. Since the satisfying assignment must choose either $x_i$ or $\overline{x_i}$ for all $i$, the extra opinions can also be satisfied according to the corresponding relations above. Hence, the corresponding *ELECTION* instance constructed by $\Phi$ is satisfiable if $\Phi$ is satisfiable.

- ($\Longrightarrow$) Given a satisfying solution to *ELECTION* instance constructed by $\Phi$. The number of students in the student union in solution should be at least $n$ because of the extra opinions. Combining with the given restriction $|S'| \leq K = n$, we know that the number of students in the student union in solution should be exactly $n$. Moreover, the student union should not include both student $S_i$ and student $\tilde{S}_i$, or the number of students in the student union will be more than $n$, which contradicts the given restriction. Therefore, we can construct the assignment of $\Phi$ as follows.
  - If $S_i$ is in the student union, then we choose $x_i$;
  - If $\tilde{S}_i$ is in the student union, then we do not choose $x_i$, that is, we choose $\overline{x_i}$.

  Therefore, since we choose at least one student from the opinion constructed by clause, every clause in $\Phi$ must be satisfied in the assignment above. Therefore, the assignment of the instance $\Phi$ of 3-SAT is a satisfying assignment. Hence, $\Phi$ is satisfiable if the corresponding *ELECTION* instance constructed by $\Phi$ is satisfiable.

□

**Theorem 3.** *ELECTION problem is an NP-complete problem.*

**Proof.** According to Theorem 1, we know that *ELECTION* problem is an NP problem. Then according to Lemma 2, we know that the 3-SAT can be reduced to *ELECTION* problem. Since 3-SAT is an NP-complete problem, *ELECTION* problem should be an NP-complete problem. □

3. Not-All-Equal Satisfiability (NAE-SAT) is an extension of SAT where every clause has at least one true literal and at least one false one. NAE-3-SAT is the special case where each clause has exactly 3 literals. Prove that NAE-3-SAT is NP-complete. (Hint : reduce 3-SAT to NAE-$k$-SAT for some $k > 3$ at first)

**Theorem 4.** NAE-3-SAT *problem is an NP problem.*

**Proof.** Here are the certificate and certifier of the NAE-3-SAT problem.

- **Certificate**. An assignment of truth values to the $n$ boolean variables $x_1, x_2, \cdots, x_n$.
- **Certifier**. Check that each clause has at least one true literal and one false literal.

Given a certificate assignment, we can easily implement a polynomial time certifier by checking if each clause has at least one true literal and one false literal. Therefore, NAE-3-SAT problem is an NP problem. □

**Lemma 5.** 3-SAT $\leq_p$ NAE-4-SAT

**Proof.** Given an instance $\Phi$ of 3-SAT which contains $n$ elements, we construct an instance of NAE-4-SAT that is satisfiable if and only if $\Phi$ is satisfiable.

**Construction.** The instance of NAE-4-SAT has $(n + 1)$ elements, and the construction process is as follows.

- Use $n$ variables $y_1, y_2, \cdots, y_n$ to represent $x_1, x_2, \cdots, x_n$ respectively;
- Add an additional variable $z$;
- For every clause $(x_i \vee x_j \vee x_k)$ in $\Phi$, we construct a clause $(y_i, y_j, y_k, z)$ in our instance. For example, a clause $(x_1 \vee \overline{x_2} \vee x_3)$ in $\Phi$ corresponds to a clause $(y_1, \overline{y_2}, y_3, z)$ in our instance.

**Claim and Proof.** The corresponding NAE-4-SAT instance constructed by $\Phi$ is satisfiable if and only if $\Phi$ is satisfiable. Here is the specific proof.

- ($\Longleftarrow$) Given a satisfying assignment of $\Phi$, we can construct an assignment of the corresponding instance of NAE-4-SAT as follows.
  - For all $i$, Set $y_i$ to $x_i$;
  - Set $z$ to `False`.

  According to the satisfying assignment of the 3-SAT instance $\Phi$, at least one of $x_i$, $x_j$ and $x_k$ for each clause $(x_i \vee x_j \vee x_k)$ is `True`. Since we set $y_i$ to $x_i$, at least one of $y_i$, $y_j$ and $y_k$ in the corresponding clause $(y_i, y_j, y_k, z)$ is `True`. Moreover, $z$ is `False` for the clause $(y_i, y_j, y_k, z)$. Therefore, the assignment of the corresponding instance of NAE-4-SAT is the satisfying assignment. Hence, the corresponding NAE-4-SAT instance constructed by $\Phi$ is satisfiable if $\Phi$ is satisfiable.
- ($\Longrightarrow$) Given a satisfying assignment of the corresponding NAE-4-SAT instance constructed by $\Phi$, we can construct an assignment of $\Phi$ as follows.
  - For all $i$, Set $x_i$ to $y_i \oplus z$, where operator $\oplus$ means "xor".

  Let us make some discussions.
  - If $z$ is `False`, then we have $x_i = y_i$ for all $i$. For each clause $(y_i, y_j, y_k, z)$, since there must be a true literal and $z$ is already `False`, at least one of $y_i$, $y_j$ and $y_k$ should be `True`. As a result, at least one of $x_i$, $x_j$ and $x_k$ should be `True`. Hence, this assignment is a satisfying assignment.

– If $z$ is `True`, then we have $x_i = \overline{y_i}$ for all $i$. For each clause $(y_i, y_j, y_k, z)$, since there must be a false literal and $z$ is already `True`, at least one of $y_i$, $y_j$ and $y_k$ should be `False`, which means at least one of $\overline{y_i}$, $\overline{y_j}$ and $\overline{y_k}$ should be `True`. As a result, at least one of $x_i$, $x_j$ and $x_k$ should be `True`. Hence, this assignment is a satisfying assignment.

In summary, $\Phi$ is satisfiable if the corresponding NAE-4-SAT instance constructed by $\Phi$ is satisfiable.

□

**Lemma 6.** NAE-4-SAT $\leq_p$ NAE-3-SAT

**Proof.** Given an instance $\Phi$ of NAE-4-SAT which contains $n$ elements, we construct an instance of NAE-3-SAT that is satisfiable if and only if $\Phi$ is satisfiable.

**Construction.** The instance of NAE-3-SAT has $(n + m)$ elements, where $m$ is the number of clauses in $\Phi$, and the construction process is as follows.

- Use $n$ variables $y_1, y_2, \cdots, y_n$ to represent $x_1, x_2, \cdots, x_n$ respectively;
- Add additional variables $z_1, z_2, \cdots, z_m$;
- For $p$-th clause $(x_i, x_j, x_k, x_l)$ in $\Phi$, we construct two clauses $(y_i, y_j, z_p)$ and $(y_k, y_l, \overline{z_p})$ in our instance. For example, the 2nd clause $(x_1, \overline{x_2}, x_3, x_4)$ in $\Phi$ corresponds to two clauses $(y_1, \overline{y_2}, z_2)$ and $(y_3, y_4, \overline{z_2})$ in our instance.

**Claim and Proof.** The corresponding NAE-3-SAT instance constructed by $\Phi$ is satisfiable if and only if $\Phi$ is satisfiable. Here is the specific proof.

- ($\Longleftarrow$) Given a satisfying assignment of $\Phi$, we can construct an assignment of the corresponding instance of NAE-3-SAT as follows.
  – For all $i$, set $y_i$ to $x_i$;
  – For $p$-th clause $(x_i, x_j, x_k, x_l)$, set the value of $z_p$ according to the following table.

| $(x_i, x_j, x_k, x_l)$ | $z_p$ |
|---|---|
| (`True`, `True`, $\cdot$, $\cdot$) | `False` |
| (`False`, `False`, $\cdot$, $\cdot$) | `True` |
| ($\cdot$, $\cdot$, `True`, `True`) | `True` |
| ($\cdot$, $\cdot$, `False`, `False`) | `False` |
| otherwise | Arbitrary (assume `True`) |

  According to the satisfying assignment of $\Phi$, at least one literal is `True` and at least one literal is `False` in clause $(x_i, x_j, x_k, x_l)$. Since we set $y_i$ to $x_i$, then in the table above, we use $z_p$ to balance the `True` and `False` in two corresponding clauses $(y_i, y_j, w_p)$ and $(y_k, y_l, \overline{w_p})$, so at least one literal is `True` and at least one literal is `False` in the two corresponding clauses of the NAE-3-SAT instance. Therefore, the assignment of the corresponding instance of NAE-3-SAT is the satisfying assignment. Hence, the corresponding NAE-3-SAT instance constructed by $\Phi$ is satisfiable if $\Phi$ is satisfiable.
- ($\Longrightarrow$) Given a satisfying assignment of the corresponding NAE-3-SAT instance constructed by $\Phi$, we can construct an assignment of $\Phi$ as follows.
  – For all $i$, Set $x_i$ to $y_i$.

We know that one clause $(x_i, x_j, x_k, x_l)$ in $\Phi$ corresponds to two clauses $(y_i, y_j, w_p)$ and $(y_k, y_l, \overline{w_p})$ in NAE-4-SAT instance constructed by $\Phi$. Then we can make the following discussions.

- If $w_p$ is `True`, then at least one of $y_i$ and $y_j$ is `False` and at least one of $y_k$ and $y_l$ is `True`;
- If $w_p$ is `False`, then at least one of $y_i$ and $y_j$ is `True` and at least one of $y_k$ and $y_l$ is `False`.

Since we set $x_i$ to $y_i$, at least one of $x_i$, $x_j$, $x_k$ and $x_l$ is `True` and at least one of them is `False`. Therefore, the assignment of $\Phi$ is the satisfying assignment. Hence, $\Phi$ is satisfiable if the corresponding NAE-3-SAT instance constructed by $\Phi$ is satisfiable.

□

**Theorem 7.** NAE-3-SAT *is an NP complete problem.*

**Proof.** According to Theorem 4, we know that NAE-3-SAT problem is an NP problem. Then according to Lemma 5 and Lemma 6, we know that 3-SAT can be reduced to NAE-3-SAT. Since 3-SAT is an NP-complete problem, NAE-3-SAT problem should be an NP-complete problem. □

4. In the Lab10, we have introduced Minimum Constraint Data Retrieval Problem (MCDR). Prove that MCDR (Version 1 or 2) is NP-complete. (Hint : reduce from VERTEX-COVER or 3-SAT)

**Definition 1** (MCDR, Decision Problem)**.** *Given $k$ data items $D = \{d_1, d_2, \cdots, d_k\}$ located in $n$ different broadcasting channels $\mathbf{C} = \{C_1, C_2, \cdots, C_n\}$, and the target data items set $D_q \subseteq D$. Each data item $d_i$ has its length $l_i$, and it is located in some positions in some channels. Each channel has a different broadcast cycle length. Given a switch parameter $h$ and an access latency parameter $t$, then the Minimum Constraint Data Retrieval Problem is whether there exists a data retrieval schedule, under the constraint of at most $h$ switches and at most $t$ access latency.*

**Theorem 8.** MCDR *problem is an NP problem.*

**Proof.** Here are the certificate and certifier of the MCDR problem.

- **Certificate**. A data retrieval schedule $S$ that can get all the target data items.
- **Certifier**. Check that $S$ is under the constraint of at most $h$ switches and at most $t$ access latency.

Given a certificate $S$, we can easily implement an polynomial time certifier by checking whether the schedule $S$ satisfies the requirements (this only need us to simulate the schedule). Therefore, MCDR problem is an NP problem. □

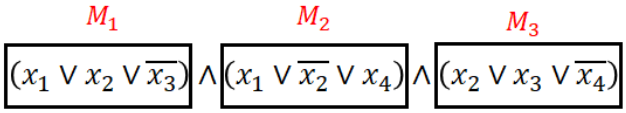
**Lemma 9.** 3-SAT $\leq_p$ MCDR

**Proof.** Given an instance $\Phi$ of 3-SAT which contains $n$ elements and $m$ clauses, we construct an instance of MCDR that is satisfiable if and only if $\Phi$ is satisfiable.

**Construction.** Let us call our MCDR instance $\Sigma$. Here are the specific construction steps:

- **Step 1**: $\Sigma$ has $2n$ channels, say channel $x_1$, channel $\overline{x_1}$, channel $x_2$, channel $\overline{x_2}$, $\cdots$, channel $x_n$ and channel $\overline{x_n}$;
- **Step 2**: $\Sigma$ has $m$ clause data, say data $M_1$, data $M_2$, $\cdots$, and data $M_m$;
- **Step 3**: If the $i$-th clause includes three literals $x_i, x_j, x_k$, then we add data $M_i$ to channel $x_i$, channel $x_j$ and channel $x_k$. For example, if the 2nd clause is $(x_1 \vee \overline{x_2} \vee x_4)$, then we will add data $M_2$ to channel $x_1$, channel $\overline{x_2}$ and channel $x_4$;
- **Step 4**: Now we can count the number of data items in each channel. Suppose the maximum number of data items in one channel is $L$;
- **Step 5**: $\Sigma$ has $n(L+1)$ element data, say data $C_{i,j}$ $(i = 1, 2, \cdots, n; j = 1, 2, \cdots, L+1)$;
- **Step 6**: For all $i$, add data $C_{i,1}$, data $C_{i,2}$, $\cdots$, data $C_{i,L+1}$ to channel $x_i$ and channel $\overline{x_i}$;
- **Step 7**: Re-arrange the data in each channel. Set the cycle of each channel to $2(L+1)$. Define the ascending order of the element data as $C_{i,1}, C_{i,2}, \cdots, C_{i,L+1}$, and define the descending order of the element data as $C_{i,L+1}, C_{i,L}, \cdots, C_{i,1}$. Then use the element data in ascending order to separate the clause data in channel $x_i$ for all $i$; and use the element data in descending order to separate the clause data in channel $\overline{x_i}$ for all $i$. The order of clause data can be arbitrary. If the number of element data is greater than the number of clause data, we need to add a blank slot between two continuous element data.
- **Step 8**: The target data is the clause data and the element data.
- **Step 9**: Set $h$ to $(n-1)$, and set $t$ to $2n(L+1)$.

Here is a specific example of the construction process. Assume that the instance $\Phi$ contains four elements $x_1$, $x_2$, $x_3$ and $x_4$ and three clauses $(x_1 \vee x_2 \vee \overline{x_3})$, $(x_1 \vee \overline{x_2} \vee x_4)$ and $(x_2 \vee x_3 \vee \overline{x_4})$. Therefore, $n = 4$ and $m = 3$ and we can calculate that $L = 2$. As a result, we have 8 channels and $n(L+1)+m = 15$ data. Fig. 1 shows our corresponding MCDR instance $\Sigma$.




8 Channels: $\{x_1, \overline{x_1}, x_2, \overline{x_2}, x_3, \overline{x_3}, x_4, \overline{x_4}\}$
15 Data: $\{M_1, M_2, M_3\} \cup \{C_{i,j} \mid i = 1,2,3,4,\ j = 1,2,3\}$
No more than 3 switches
No more than 24 access latency

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Channel $x_1$ | $C_{1,1}$ | $M_1$ | $C_{1,2}$ | $M_2$ | $C_{1,3}$ | | $C_{1,1}$ | $M_1$ | $C_{1,2}$ | $M_2$ | $C_{1,3}$ | | ... ... | | | |
| Channel $\overline{x_1}$ | $C_{1,3}$ | | $C_{1,2}$ | | $C_{1,1}$ | | $C_{1,3}$ | | $C_{1,2}$ | | $C_{1,1}$ | | ... ... | | | |
| Channel $x_2$ | $C_{2,1}$ | $M_1$ | $C_{2,2}$ | $M_3$ | $C_{2,3}$ | | $C_{2,1}$ | $M_1$ | $C_{2,2}$ | $M_3$ | $C_{2,3}$ | | ... ... | | | |
| Channel $\overline{x_2}$ | $C_{2,3}$ | $M_2$ | $C_{2,2}$ | | $C_{2,1}$ | | $C_{2,3}$ | $M_2$ | $C_{2,2}$ | | $C_{2,1}$ | | ... ... | | | |
| Channel $x_3$ | $C_{3,1}$ | $M_3$ | $C_{3,2}$ | | $C_{3,3}$ | | $C_{3,1}$ | $M_3$ | $C_{3,2}$ | | $C_{3,3}$ | | ... ... | | | |
| Channel $\overline{x_3}$ | $C_{3,3}$ | $M_1$ | $C_{3,2}$ | | $C_{3,1}$ | | $C_{3,3}$ | $M_1$ | $C_{3,2}$ | | $C_{3,1}$ | | ... ... | | | |
| Channel $x_4$ | $C_{4,1}$ | $M_2$ | $C_{4,2}$ | | $C_{4,3}$ | | $C_{4,1}$ | $M_2$ | $C_{4,2}$ | | $C_{4,3}$ | | ... ... | | | |
| Channel $\overline{x_4}$ | $C_{4,3}$ | $M_3$ | $C_{4,2}$ | | $C_{4,1}$ | | $C_{4,3}$ | $M_3$ | $C_{4,2}$ | | $C_{4,1}$ | | ... ... | | | |

Figure 1: The corresponding MCDR instance $\Sigma$

**Claim and Proof.** $\Sigma$ is satisfiable if and only if $\Phi$ is satisfiable. Here is the specific proof.

- ($\Longleftarrow$) If $\Phi$ is satisfiable, then suppose the satisfying assignment of $\Phi$ is $y_1, y_2, \cdots, y_n$, where $y_i = x_i$ or $y_i = \overline{x_i}$ for all $i$. We can start at channel $y_1$, and move to channel $y_2$ at the end of the cycle in channel $y_1$, and then move to channel $y_3$ at the end of the cycle in channel $y_2$. Repeat this process $(n-1)$ times, so we can collect all the data items since $y_1, y_2, \cdots, y_n$ is the satisfying assignment of $\Phi$. We need to switch $(n-1)$ times, and actually we collect all the data items at the end of time $2n(L+1)-1$. Both of them is within the constraints of switch number $h$ and access time $t$. Therefore, this is a satisfying solution to $\Sigma$. Hence, $\Sigma$ is satisfiable if and only if $\Phi$ is satisfiable.
- ($\Longrightarrow$) If $\Sigma$ is satisfiable, then we can find a schedule which collects all the target data within the constraints of switch number $h$ and access time $t$. According to our design, we have to switch $(n-1)$ times to collect all the element data because each element data is only in two channel. Moreover, we cannot access both channel $x_i$ and channel $\overline{x_i}$ because it will cost one switch and we have no switch opportunity left. The access latency also restricts us to collect all the clause data in $n$ cycles. Suppose we visit channel $y_1$, channel $y_2$, $\cdots$, channel $y_n$, where $y_i = x_i$ or $y_i = \overline{x_i}$ for all $i$, then $y_1, y_2, \cdots, y_n$ should be a satisfying assignment of $\Phi$ since we have covered all the clauses. Hence, $\Phi$ is satisfiable if $\Sigma$ is satisfiable.

**Note.** The original method of designing ascending order and descending order of the element data is to prevent switching in the middle of the cycle. In fact, we even do not need to constraint of switch number $h$ using this design (you can figure out the reason yourself). We do not use this property in our proof because we have the constraint of switch number $h$, but it is a quite tricky design so I want to keep the design in the final solution. □

**Theorem 10.** MCDR *problem is an NP-complete problem.*

**Proof.** According to Theorem 8, we know that MCDR problem is an NP problem. Then according to Lemma 9, we know that 3-SAT can be reduced to MCDR problem. Since 3-SAT is an NP-complete problem, MCDR problem should be an NP-complete problem. □

**Remark:** Please include your .pdf, .tex files for uploading with standard file names.