ORIE 6300 Mathematical Programming I November 18, 2014

Lecture 24

Lecturer: David P. Williamson *Scribe: Dave Lingenbrink*

1 Decision Problem as a Subset

Definition 1 *We denote the encoding of an input to a problem by* $\langle \cdot \rangle$.

For example, the input to the LP

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

can be denoted $\langle A, b, c \rangle$.

Definition 2 *The set of all binary strings, is defined as* $\{0,1\}^* = \{0, 1, 00, 01, 10, 11, 000, \ldots\}$

Definition 3 *A decision problem is one such that the expected output is either YES or NO. It is represented by a set* $A \in \{0,1\}^*$ *of exactly those inputs whose outputs are YES.*

An LP can be seen as a decision problem. Consider the LP we defined above. We can define the decision problem

$$LP = \{\langle A, b, c, t \rangle. \text{ There is a solution } x \text{ s.t. } Ax \leq b,\ x \geq 0, \text{ and } c^T x \leq t\}.$$

If we want to find an optimal solution, we can begin at $t = -\infty$ and decide whether $\langle A, b, c, t \rangle \in LP$, then we can find optimal solution t^* as the point where the answer "switches" from YES to NO.

Another example is the Traveling Salesman Problem. This has decision problem

$$TSP = \{\langle n, c, B \rangle : \text{ There is a tour of length} \leq B \text{ (i.e. } \textstyle\sum_{j=1}^{n-1} c(\pi(j), \pi(j+1)) + c(\pi(n), \pi(1)) \leq B)\}.$$

Definition 4 x *is a yes instance of a decision problem* A *if* $x \in A$. x *is a no instance of a decision problem* A *if* $x \notin A$. *An algorithm* $\mathcal{A}$ *decides* A *if* $\mathcal{A}(x)$ *outputs YES iff* $x \in A$.

Definition 5 *We define* $|x|$ *to be the length of the string* x *(e.g. the number of bits it takes to represent* x*).*

2 Definition of Polynomial Time

Definition 6 *We say* $\mathcal{A}$ *runs in polynomial time if there exists a polynomial* p *such that the number of steps of* $\mathcal{A}$ *on input* x *is no more than* $p(|x|)$.

Definition 7 *If we denote a computational problem as* π*, then the set of polynomial-time decision problems, denoted by* $\mathcal{P}$*, is defined as:*

$$\mathcal{P} = \{\pi : \textit{There is an algorithm to decide } \pi \textit{ in polynomial time}\}.$$

For example, $LP \in \mathcal{P}$.

Definition 8 *A decision problem is in* $\mathcal{NP}$ *if there exists a verifier* $\mathcal{A}(\cdot,\cdot)$*, polynomials* p_1, p_2 *such that*

- *for all $x \in A$ there exists a $y \in \{0,1\}^*$ where $|y| \leq p_1(|x|)$ such that $\mathcal{A}(x,y)$ outputs YES.*
- *for all $x \notin A$, there eixsts a $y \in \{0,1\}^*$ where $|y| \leq p_1(|x|)$ such that $\mathcal{A}(x,y)$ outputs NO.*
- *the number of steps of $\mathcal{A}(x,y)$ is no more than $p_2(|x|+|y|)$.*

$\mathcal{NP}$ means non-deterministic polynomial time.

For example, $TSP \in \mathcal{NP}$. The "proof" is a tour of cost $\leq B$

$$y = \langle \pi(1), \pi(2), \ldots, \pi(n) \rangle.$$

The verifier $\mathcal{A}$ checks that

$$\sum_{j=1}^{n-1} c(\pi(j), \pi(j+1)) + c(\pi(n), \pi(1)) \leq B$$

and that π is a permutation.

Next, $LP \in \mathcal{NP}$, since we our verifier $\mathcal{A}$ can just ignore the proof y and compute in polytime whether $x \in A$. Thus, $\mathcal{P} \subseteq \mathcal{NP}$. The answer to whether $\mathcal{P} = \mathcal{NP}$ is not known, but if someone solves it, they will have solved one of the seven Millennium Prize Problems and will win a \$1,000,000 prize!

Definition 9 *For decision problems A, B, a polyonimal-time reduction from A to B is a polynomial time algorithm $\mathcal{A}$ such that $\mathcal{A}(x) \in B$ iff $x \in A$. In other words, $\mathcal{A}(x)$ is a yes instance of B iff x is a yes instance of A. We write $A \leq_{\mathcal{P}} B$.*

Lemma 1 *If $A \leq_{\mathcal{P}} B$ and $B \leq_{\mathcal{P}} C$, then $A \leq_{\mathcal{P}} C$.*

Proof: Let $\mathcal{A}$ be the polynomial time reduction from A to B and $\mathcal{A}'$ be the polynomial time reduction from B to C. So, $\mathcal{A}(x) \in B$ iff $x \in A$, and $A'(y) \in C$ iff $y \in B$. Thus, $\mathcal{A}'(\mathcal{A}(x)) \in C$ iff $x \in A$. All we need to show is that $\mathcal{A}' \circ A$ runs in time polynomial in $|x|$. This is true since polynomials are closed under composition. Thus, the lemma is proved. $\square$

Definition 10 *B is a $\mathcal{NP}$-complete problem if $B \in \mathcal{NP}$ and for all $A \in \mathcal{NP}$, $A \leq_{\mathcal{P}} B$.*

Theorem 2 *If B is $\mathcal{NP}$-complete and $B \in P$, then $\mathcal{P} = \mathcal{NP}$.*

Proof: We know that $\mathcal{P} \subseteq \mathcal{NP}$. Pick a $A \in \mathcal{NP}$. By definition of $\mathcal{NP}$-competeness, $A \leq_{\mathcal{P}} B$. Let $\mathcal{A}$ be a polynomial time algorithm for deciding B and let $\mathcal{A}'$ be the polynomial time algorithm for reducing A to B. Then, $\mathcal{A}'(x) \in B$ iff $x \in A$.

So, given input x, we run $\mathcal{A}(\mathcal{A}'(x))$. This will output YES iff $x \in A$ and runs in polynomial time. So, $A \in \mathcal{P}$, and $\mathcal{NP} \subseteq \mathcal{P}$. Thus, $\mathcal{NP} = \mathcal{P}$. $\square$

3 Outline of Strategy for Proving $\mathcal{NP}$-Completeness

Consider some problem B that we want to show is $\mathcal{NP}$-complete. First, we show that $B \in \mathcal{NP}$. Next, show that for some $\mathcal{NP}$-complete A, $A \leq_{\mathcal{P}} B$.

Lemma 3 *Given the above, B is $\mathcal{NP}$-complete.*

Proof: $B \in \mathcal{NP}$, so all we need to show is that for any $C \in \mathcal{NP}$, $C \leq_{\mathcal{P}} B$. Since A is $\mathcal{NP}$-complete, $C \leq_{\mathcal{P}} A$. We know $A \leq_{\mathcal{P}} B$, so by transitivity, $C \leq_{\mathcal{P}} B$, and B is $\mathcal{NP}$-complete, as desired. $\square$

4 Some $\mathcal{NP}$-Complete Problems

- **Satisfiability**: Given boolean variables $x_1, \ldots, x_n$ and clauses of disjunctions of variables or negations (for example, one clause could be $x_1 \vee \overline{x_5} \vee x_{17}$). Is there an assignment of true and false to the $\{x_i\}$ that satisfies all clauses?

- **Partition**: Given n integers $a_1, \ldots, a_n$ such that $\sum_{i=1}^{n} a_i$ is even. Does there exist a partition of $\{1, \ldots, n\}$ into S and T such that
$$\sum_{i \in S} a_i = \sum_{j \in T} a_j?$$

- **3-partition**: Given $3n$ integers $a_1, \ldots, a_{3n}$ and b such that $b/4 < a_i < a/2$ for all i and $\sum_{i=1}^{3n} a_i = nb$, does there exist a partition of $\{1, \ldots, 3n\}$ into n sets $T_1, \ldots, T_n$ such that
$$\sum_{i \in T_j} a_i = b \quad \forall j = 1, \ldots n?$$

Finally, consider the Knapsack problem. Recall this problem has n items of sizes $s_1, \ldots, s_n$, values $v_1, \ldots, v_n$ and a total knapsack size of B. The decision problem is:

Given input V, does there exist a $S \subset \{1, \ldots, n\}$ such that $\sum_{i \in S} s_i \leq B$ and $\sum_{i \in S} v_i \geq V$?

First, we see that the knapsack problem is in $\mathcal{NP}$. Just let our verifier check the list of items in S and check whether the items' sizes are no more than B and the values are at least V. We will come back to showing the knapsack problem is $\mathcal{NP}$-complete.