CSE200 Lecture Notes – NP-complete problems (I)

Lecture by Russell Impagliazzo
Notes by Jiawei Gao

February 9, 2016

Intuitively, any Turing machine can be simulated by Boolean circuits (thus we can build computers using logic gates). The next theorem claims for any problem in P there is a circuit family of polynomial size that computes it.

Theorem 1. Every language in P has a P-uniform circuit family computing each $L_n = L \cap \{0,1\}^n$.

Using this theorem, we prove the NP-completeness of CircuitSAT.

Claim 2. CircuitSAT is NP-complete.

Proof. Recall that a language $L \in \mathsf{NP}$ means $x \in L$ iff $\exists y, |y| = \ell(n), R(x,y), R \in \mathsf{P}$.

Given x, we perform the following steps:

1. Set $n' = n + \ell(n)$. (For simplicity we fix n and use ℓ to represent $\ell(n)$.)
2. Construct a circuit $C_{n'}(x_1, \ldots, x_n, y_1, \ldots, y_\ell) = R(x,y)$.
3. Fix the gates in $C_{n'}$ corresponding to $x_1, \ldots, x_n$ to the actual input bits, and get circuit $C'(y_1, \ldots, y_\ell)$.

We prove that $C'(y_1, \ldots, y_\ell)$ is satisfiable iff $x \in L$.

- If $x \in L$, and $\exists y, R(x,y)$. Define $y_1, \ldots, y_\ell$ to be the bits of y.
  $C'(y_1, \ldots, y_\ell) = C_{n'}(x_1, \ldots, x_n, y_1, \ldots, y_\ell) = R(x,y) = 1$. Thus C' is satisfiable.
- If C' is satisfiable, namely $\exists y_1, \ldots, y_\ell$, $C'(y_1, \ldots, y_\ell) = 1$. Let $y = y_1 \ldots y_\ell$,
  $C'(y_1, \ldots, y_\ell) = C_{n'}(x_1, \ldots, x_n, y_1, \ldots, y_\ell) = R(x,y)$. Thus if C' is satisfiable, then $x \in L$.

□

Next we reduce from CircuitSAT to 3SAT.

Claim 3. 3SAT is NP-complete.

Proof. From circuit C on n variables and m gates (defined in the previous lecture), we create a 3-CNF $\varphi(x_1, \ldots, x_n, z_{n+1}, \ldots, z_m)$.

The main technique is to encode the gates using clauses: for the i^{th}gate $g_i = op_i(g_j, g_k)$, write $z_i = op_i(z_j, z_k)$ as a CNF.

We give the example of construction of clauses for gate of form $z_i = z_j \wedge z_k$. (Other operations are constructed in similar ways.) This can be done by hard-coding the truth table of the gate.

- On input 00, output must be 0. So we create clause $(z_j \vee z_k \vee \neg z_i)$.
- On input 01, output must be 0. So we create clause $(z_j \vee \neg z_k \vee \neg z_i)$.
- On input 10, output must be 0. So we create clause $(\neg z_j \vee z_k \vee \neg z_i)$.
- On input 11, output must be 1. So we create clause $(\neg z_j \vee \neg z_k \vee z_i)$.

(Explanation of the above: For input 00, we have $(\neg z_j \wedge \neg z_k) \rightarrow \neg z_i$. It is equivalent to $\neg(\neg z_j \wedge \neg z_k) \vee \neg z_i$ and thus equivalent to $z_j \vee z_k \vee \neg z_i$)

Finally, because we expect the circuit output "true", we add a clause (z_m) to φ.

For every gate, we create at most four clauses. The size of φ we constructed is linear to the size of C. Also, the reduction runs in linear time.

Proof of correctness:

- Assume $\varphi(x_1, \ldots, x_n, z_{n+1}, \ldots, z_m) = 1$. Then define a satisfying assignment to C as $x_1, \ldots, x_n$. Because each $z_i = op_i(z_j, z_k), z_i = g_i(x_1, \ldots, x_n)$, we get $1 = z_m = g_m(x_1, \ldots, x_n) = C(x_1, \ldots, x_n)$.
- Assume $C(x_1, \ldots, x_n) = 1$. Let $z_i = g_i(x_1, \ldots, x_n), i = n+1, \ldots, m$.
  $g_i = op_i(g_j, g_k)$, so $z_i = op_i(g_j, g_k)$, so all but the last clause of φ, is satisfied.
  $z_m = g_m(x_1, \ldots, x_n) = 1$, so last clause (z_m) is also satisfied.

□

Next we reduce from 3SAT to Independent Set.

Independent Set problem (IndSet)

- Instance: $G = (V, E)$, $1 \leq k \leq |V|$.
- Solution: $S \subseteq V$, $\{u, v\} \in E \rightarrow u \notin S \vee v \notin S$, $|S| = k$.
- Question: Is there an independent set in G of size k?

Claim 4. IndSet is NP-complete.

Example: Clauses $(x \vee y \vee z), (\bar{x} \vee y \vee \bar{z}), (x \vee x), (\bar{w} \vee y \vee \bar{z})$.

- For each occurrence in the CNF, we create a vertex.
- For each variable and its negation, link an edge between them.
- To avoid double-counting the same clause, we add edges between all pairs of nodes in the same clause.
- Finally, set k to be the number of clauses.

Proof. Given a 3CNF φ, we construct instance (G, k) of IndSet.

Let $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where the clauses are $C_i = \ell_{i1} \vee \ell_{i2} \vee \ell_{i3}$.

$(G(V,E), k)$ is defined as follows:

- $V = \{v_{i1}, v_{i2}, v_{i3} \mid i = 1, \ldots, m\}$.
- $E = \{\{v_{i1}, v_{i2}\}, \{v_{i2}, v_{i3}\}, \{v_{i2}, v_{i1}\} \mid i \in 1, \ldots, m\} \cup \{\{v_{ij}, v_{i'j'}\} \mid \ell_{ij} = \neg\ell_{i'j'} \mid i = 1, \ldots, m, j = 1, 2, 3\}$.
- $k = m$.

(Note: This is not a one-to-one mapping from independent sets to satisfying assignments. An independent set may correspond to multiple assignments, where some variables can be either true or false.)

Proof of correctness:

- Assume $\varphi(x_1, \ldots, x_n) = 1$. Clause C_i is true for $i = 1, \ldots, m$.
  $C_i = \ell_{i1} \vee \ell_{i2} \vee \ell_{i3} = 1$. Let ℓ_{ij} be the first literal set to true in C_i, we add v_{ij} in S. i.e. set $S = \{v_{ij} \mid \ell_{ij}$ is the first literal satisfying $C_i\}$.
  $|S| = m = k$, because we only put one vertex per clause in S, we can't have added both endpoints of edges in the first category (edges between literals in the same clause).
  Since we only added vertices corresponding to the true literals to S, and edges go between $\ell, \neg\ell$, we can't have added both endpoints of any edge of the second type (edges between literals and their negations).
  So S is an independent set of size k.
- Say S is an independent set of size k.
  Set $x_h = 1$ if $x_h = \ell_{ij}, v_{ij} \in S$.
  $x_h = 0$ if $\neg x_h = \ell_{ij}, v_{ij} \in S$.
  $x_h = 0$ if neither.
  Because of edges between $\ell_{ij} = \neg\ell_{i'j'}$, we have assigned a variable 0 or 1.
  $|S| = k = m$, because all clauses are fully connected, S must have exactly one v_{ij} from each clause C_i, $\ell_{ij} = 1$ under our assignment. So each clause C_i is satisfied. So φ is satisfied.

□

Conclusions: (1) CircuitSAT is NP-complete. (2) 3-SAT is NP-complete. (3) IndSet is NP-complete. We have showed a combinatorial problem is NP-complete. (4) The big abstract problem "Is P = NP" is equivalent with the concrete problem "Is IndSet in P"?