CSE 200 - Computability and Complexity
The abundance of NP-complete problems

Russell Impagliazzo and Jiawei Gao

Oct., 2016
Goals for today

1. Show that simple $NPC$ problems arise frequently and in varied contexts
2. General format of $NPC$ proof
3. Common mistakes
4. What to do when your problem is $NPC$
A brief history of $P$ vs $NP$

The $P$ vs $NP$ question was implicitly posed in an unpublished letter from Godel to von Neumann:
In the Soviet Union

Yablonsky was investigating perebor, the concept of forced search.

Unfortunately, his students mistook his work on perebor as an algorithmic fact, rather than a conjecture.
Jack Edmonds

Jack Edmonds published the first statement of the $P$ vs $NP$ problem to justify his interest in polynomial time algorithms for matching.
Steven Cook proved that \textit{NPC} problems existed and even gave a combinatorial example, clique. Independently, Leonid Levin did a similar proof (using tiling as the example) in the Soviet Union, but needed to be tactful because Yablonsky’s students were politically powerful. So his work did not receive as much attention at the time.
Richard Karp

Richard Karp’s follow up paper showed six \textit{NPC} problems that had previously been studied by different people.

Since then, he has been one of the deepest researchers both on \textit{NPC} and on methods for coping once your problem is \textit{NPC}
The Colorful Connected Subgraph Problem

A graph is given, together with a color assigned to each vertex. Many vertices may receive the same color. We consider the NP-hard problem of finding a connected subgraph with a minimum number of vertices, such that the subgraph must contain at least one vertex of each color. In particular, we are interested in perfect solutions, in which no color is repeated. Versions of the problem arise in the context of protein-protein interaction networks, social networks and sensor networks. The problem (or even a generalization in which the edges of the graph are weighted) can be solved by a dynamic programming in time polynomial in the size of the graph but exponential in the number of colors. It can also be represented by an integer program with polynomial-bounded numbers of variables and linear constraints. We present a simple fast heuristic algorithm and describe its performance on large 2-dimensional grid graphs, under various specifications of the number of colors and their frequency distribution, using a random model and a semi-random model.

In the random model the color assignment is chosen uniformly at random among assignments with the given frequency distribution. The algorithm reliably gives near-perfect solutions, provided the distribution of color frequencies is not highly skewed.

In the semi-random model a random perfect solution is planted, and the completion of the color assignment is random. Regardless of the frequency distribution the algorithm reliably produces perfect solutions. In this case we extend the algorithm to generate many perfect solutions, and report on its performance.
Say we have problem $A$ that we know is $NPC$, and problem $B$ that we are trying to prove $NPC$. We need to give a mapping reduction $F$ from $A$ to $B$. (Not the other way!)
Note that we are trying to show: if there exist hard instances of $A$, then there exist hard instances of $B$. So we do not need that $F$ includes all instances of $B$ in its range, and we do not need that $F$ can be inverted.
What $A$ and $B$ look like

Since $A, B \in \mathit{NP}$ (need to check for $B$ before starting),

$x_A \in A \iff \exists y_A \in R_A(x_A, y_A)$ and

$x_B \in B \iff \exists y_B \in R_A(x_B, y_B)$

We should have $x_A \in A$ iff $x_B = F(x_A) \in B$. 
Proving reductions correct

So what we need is, for $x_B = F(x_A)$, $\exists y_A R_A(x_A, y_A)$ iff $\exists y_B, R_B(x_B, y_B)$ So what we need to do is:

1. Define carefully the mapping $F$. We need to show how to construct the instance $x_B$ of $B$ from the instance $x_A$ of $A$. Make sure types match, and that $F \in P$.

2. Define a map $G$ from solutions to $x_B$ to solutions to $x_A$. Assume $R_B(x_B, y_B)$. Prove $R_A(x_A, G(y_B))$. This shows that, if $x_B \in B$, then $x_A \in A$.

3. Define a map $H$ from solutions to $x_A$ to solutions to $x_B$. Assume $R_A(x_A, y_A)$. Prove $R_B(x_B, H(y_A))$. This shows that, if $x_A \in A$, then $x_B \in B$. 
Common mistakes

2. We cannot use a solution $y_A$ to define $F$, since we don’t know such a solution yet.
3. $F, G, H$ don’t need to be one-to-one or onto.
4. Remember to give the map $H!!$
Circuit – SAT to 3 – SAT

$F$: Given $C$, we construct $C'(z_1 \ldots z_m)$, where we have a variable $z_i$ for each gate of $C$, and clauses that interpret $g_i = op_i(g_j, g_k)$ for every middle gate, and the clause $z_m$ for the output gate.

$G$: Given a satisfying assignment to $C'$, we use the values for the first $n$ variables to define a satisfying assignment to $C$.

$H$: Given a satisfying assignment to $C$, we use the resulting values given to the gates of $C$ on this assignment to define a satisfying assignment to $C'$. 
Constraint Satisfaction Problem

Like \( NP \) but relation involves just local conditions on variables from a fixed range. \( CSP_{a',r} \) : constraints involve at most \( a \) variables each, variables take on \( r \) values.

\( 3SAT = CSP_{3,2} \).

We’ll reduce from \( 3-SAT \) to \( CSP_{2,3} \), constraints involving just two variables, but where each variable has 3 possible values. From \( CSP_{2,3} \), we’ll be able to further reduce to natural graph problems such as 3-colorability and Big independent set.
The reduction

Say $\Phi = C_1 \land C_2 \land \ldots C_m$ is a $3-CNF$ in variables $x_1 \ldots x_n$, and each $C_i = l_{i,1} \lor l_{i,2} \lor l_{i,3}$.

$F$: We create a CSP with $m$ variables taking on values 1, 2, 3 as follows. The variable $v_i$ represents how we satisfy $C_i$, either through $l_{i,1}, l_{i,2}, l_{i,3}$. Then for each pair $i, j, i', j'$ where $l_{i,j} = \neg l_{i',j'}$, add the constraint that, if $v_i = j$, then $v_{i'} \neq j'$.
Proof of equivalence

\(G\): Assume we have values to the variables \(v_i\) that satisfy all of the constraints of the created CSP. Then we will let \(x\) be assigned \(True\) if there is some \(v_i\) so that \(l_{i,v_i} = x\). \(x\) will be assigned \(False\) otherwise.

Let \(C_i\) be a clause. \(C_i = l_{i,1} \lor l_{i,2} \lor l_{i,3}\). If \(l_{i,v_i}\) is a positive variable \(x\), our rule assigned \(x\) value True, and the clause \(C_i\) is satisfied. If \(l_{i,v_i}\) is \(\neg x\), then we cannot have any \(i'\) where \(l_{i',v_{i'}} = x\), because that would violate the constraint. So our rule assigns \(x\) the value False, and \(C_i\) is satisfied. Since each clause of \(\Phi\) is satisfied, \(\Phi\) is satisfied by this assignment.
Proof of equivalence

\[ H: \] Assume we have a satisfying assignment to the \( \Phi, x_1..x_n \). For each clause \( C_i \), it is satisfied by one of its three literals. Assign value \( v_i \) to be the number of the first satisfied literal. Since we cannot have both \( x \) and \( \neg x \) be satisfying literals for different clauses, this assignment meets all of the constraints of our \( CSP \). Thus, \( CSP_{2,3} \) is \( NP \)-complete.
From $CSP_{2,3}$ to $BigIndSet$

Idea: relationships between two variables = edges in a graph

Say we have a $CSP$ with variables $v_1 .. v_n \in \{1, 2, 3\}$ and constraints $R_{i,j}(v_i, v_j)$ for some pairs $i, j$.

$F$: We create a graph and an integer $k$ from the $CSP$ as follows:
For every $i$, create a triangle $u_{i,1}, u_{i,2}, u_{i,3}$, (where we will put $u_{i,v_i}$ in the independent set). If $\neg R_{i,j}(d, d')$, i.e., we are not allowed to give $v_i = d$ and $v_j = d'$, then we put an additional edge between $u_{i,d}$ and $u_{i',d'}$. We set $k = n$. 
Equivalence

\(G\): Assume \(S\) is an independent set in the constructed graph of size \(k = n\). Then each triangle must have exactly one element in the independent set. We assign \(v_i\) to be the one value \(d\) so that \(u_{i,d} \in S\). This meets every constraint, because if it failed for a constraint between \(v_i\) and \(v_i'\), there’d be an edge between the two vertices \(u_{i,v_i}\) and \(u_{i',v_i'}\), both of which are in the supposed independent set \(S\).
Equivalence

\[ H: \text{ Assume there is an assignment } v_1 ... v_n \text{ that meets all the constraints. Let } S = \{u_i, v_i\}. |S| = n = k. S \text{ has exactly one vertex per triangle, and if there were an edge between the two vertices } u_i, v_i \text{ and } u_{i'}, v_{i'}, \text{ both of which are in the supposed independent set } S, \text{ then } v_i, v_{i'} \text{ would violate the constraint } R_{i,i'}. \text{ So } S \text{ is an independent set in the constructed graph} \]