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

Russell Impagliazzo and Jiawei Gao

Oct., 2016
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 $k = n$. 
Example

Say we have four variables $A, B, C, D$, and constraints:

1. $A \neq C$
2. If $A = 1$ then $B = 1$
3. If $B = 1$ then $C \neq 2$
4. $C \neq D$

What does the corresponding instance of BIS look like?
G: Assume $S$ is an independent set in the constructed graph of size $k = n$. Then each triangle $v_{i,1}, v_{i,2}, v_{i,3}$ 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

\( H: \) Assume there is an assignment \( v_1...v_n \) that meets all the constraints. Let \( S = \{u_{i,v_i}\}. \) \( |S| = n = k. \) \( S \) has exactly one vertex per triangle, and if there were an edge between the two vertices \( u_{i,v_i} \) and \( u_{i',v_i'}. \), both of which are in the supposed independent set \( S, \) then \( v_i, v_i' \) would violate the constraint \( R_{i,i'}. \) So \( S \) is an independent set in the constructed graph.
A 3-coloring of a graph is a labeling of its vertices with 3 colors, $R$, $G$, $B$, so that adjacent vertices have different labels. The 3-coloring problem is, given a graph, determine whether such a coloring exists.
**NP-completeness of 3-coloring**

We already saw 3-coloring $\in NP$. We will reduce $CSP_{2,3}$ to 3-coloring. One issue is that there is complete symmetry between the three colors, but not in the values 1,2,3 assigned the variables in a $CSP$. So we’ll use the following “gadget”. Three nodes 1, 2, 3 connected in a triangle, with whatever color we color node $i$ being identified with a variable having value $i$. Saying a vertex can only have a subset of values, say 1, 2, is equivalent to putting an edge to node 3. We’ll use this as a short-hand in our construction.
The mapping

\( F \): We are given a CSP in variables \( v_1 \ldots v_n \). Create the triangle mentioned above. Add one vertex \( u_1 \ldots u_n \) for each variable. For each pair of values \( d, d' \) with \( d \neq d' \), where \( v_j = d, v_k = d' \) is inconsistent with a constraint, let \( d'' \) be the third value. We add two new vertices, \( a_{j,k,d,d'} b_{j,k,d,d'} \) with \( u_j \) connected to \( a_{j,k,d,d'} \) connected to \( b_{j,k,d,d'} \) connected to \( u_k \). We allow colors \( d \) and \( d'' \) for \( a \), and \( d' \) and \( d'' \) for \( b \). If it is inconsistent to give \( v_j \) and \( v_k \) the same value \( d \), we add three new vertices, \( a_{j,k,d}, b_{j,k,d}, c_{j,k,d} \) connecting \( u_j \) to \( u_k \) in a line, and allow \( a \) colors \( d, d', d'' \), \( b \) colors \( d', d'' \) and \( c \) colors \( d'', d \).
Example

What does the sub-graph expressing $A \neq C$ look like? What about $A = 1 \implies B = 1$?
**Equivalence**

**G:** Say that the graph we constructed is 3-colorable. To give values to the variables, we: identify the color given vertex $d \in \{1, 2, 3\}$ with value $d$, and give $v_i$ the value corresponding to the color of $u_i$.

This obeys the constraints, because if $v_i$ and $v_j$ had values $d \neq d'$ which violated a constraint, we'd color $u_j d$, $u_k d'$, and then $a_{j,k,d,d'}$ and $b_{j,k,d,d'}$ would be neighboring vertices both colored $d''$, contradicting the properties of a valid 3-coloring.

Similarly, if both are given value $d$, which violates the constraint, $a_{j,k,d}$ would be colored $d'$, $c_{j,k,d}$ would be colored $d''$, and $b_{j,k,d}$ has no possible color.
Equivalence

\( H: \) Conversely, say we have values \( d_i \) for each variable that meets all constraints. Then we color the triangle with say 1 colored \( R \), 2 colored \( B \) and 3 colored \( G \), and color each \( u_i \) with the color corresponding to \( d_i \).

For the gadget vertices, if \( d \neq d' \) violates the constraint for \( v_j, v_k \), we know either \( d_j \neq d \) or \( d_k \neq d' \). If the former, we color \( a_{j,k,d,d'} \) \( d \), and \( b_{j,k,d,d'} d'' \), and color a \( d'' \) and b \( d' \) if the latter. For \( v_j = v_k = d \) violating the constraints, we know either \( d_j \neq d \) or \( d_k \neq d \), so in the first case, we color a \( d \), and then color c to be different from \( d_k \), and b to be different from the color of c, and symmetrically if \( d_k \neq d \).
The subset sum problem has as input a list of $n$ integers $a_1..a_n$ and a target integer $T$. The question is: Does there exist a subset $S$ of $1, ... n$ so that $\sum_{i \in S} a_i = T$?

This problem arises in scheduling, in load-balancing, and in lattice based cryptography.
To specify a subset takes \( n \) bits, and verifying it just involves adding the given integers. So \( \text{SubsetSum} \in NP \).

We’ll reduce from Big Independent Set. Say we are given a graph \( G \), and a target size \( k \). We’ll construct integers as follows. First, let’s look at our integers base 4, and think of their having digits that correspond to edge positions, so for each edge \( e_1 \ldots e_m \), they will have digits \( d_1, \ldots d_m \), and also a high order value \( H \), beyond the \( m \)'th digit. The high order value will count the size of the set.
The integers

For each vertex $v$, we will create an integer $a_v$ whose first $m$ base 4 digits are 1 in position $j$ iff $v \in e_j$ and 0 otherwise. The high order value will always be 1.

We also have a “slack” integer $b_e$ for each edge $e$, that is 1 in the position corresponding to $e$, 0 everywhere else, and 0 as its high order part.

The target $T$ will have all 1’s as its digits, and $k$ as its high order part.
Say we have a solution $\sum_{v \in S_V} a_v + \sum_{e \in S_E} b_e = T$. We let $I = S_V$. Note that for each digit, corresponding to $e = \{u, v\}$, there are 3 integers in our input with a 1 in that digit, $a_u$, $a_v$ and $b_e$. So we will never have any carries when adding a subset base 4. To get a 1 in that digit, as in the target, at most one of $a_u$ and $a_v$ must be in $S_V$, so $I = S_V$ is an independent set. Since there are no carries, the high order part of the sum is the sum of the high order bits, and must be $k$. Only $a_v$ has a non-zero high order bit, so we must have $k = |S_V| = |I|$. as desired
Say we have an independent set $I$ in $G$ of size $k$. Then we pick $S$ as follows: Include $a_v$ if $v \in I$, include $b_e$ if neither endpoint of $e$ is in $I$. Then for every edge $e = \{u, v\}$, we’ve included exactly one of $a_u, a_v, \text{and } b_e$. So the $e$’th digit of the sum will be 1 for every $e$. The high order part of the sum will be $|S_V| = |I| = k$. Thus, the sum will match the target.
Meaning of $NP$-completeness

If an optimization problem is shown to be $NP$-complete, and $P$ is very different from $NP$, that means: We will not have a fast algorithm that can find the exact optimal solution on every instance of the problem. This still leaves open many possibilities.