# CSE 200
# Computability and Complexity
# Homework 4
# NP and beyond
# Polynomial-time Hierarchy
# Consequences of $NP = P$ Due March 1

February 17, 2016

**Low exponential NP-complete problems** We often use the somewhat misleading phrase that $NP$-complete problems are the "hardest in $NP$". Show that this can be misleading, by showing that, for every $\epsilon > 0$ there is an NP-complete problem that can be solved in $2^{n^\epsilon}$ time.

**Canonical form** Two graphs $G$ and $H$ are *isomorphic* if there is a 1-1 onto map $f$ from the vertices of $G$ to the vertices of $H$ so that $u$ and $v$ are adjacent in $G$ if and only if $f(n)$ and $f(v)$ are adjacent in $H$. A *graph canonicalization* is a map $Canon$ from graphs to graphs so that $G$ and $Canon(G)$ are isomorphic, and for every two isomorphic graphs $G$ and $H$, $Canon(G) = Canon(H)$. Prove that, if $P = NP$, there is a graph canonicalization computable in polynomial time.

**Sudoku experiment** We started looking at reducing sudoku problems to $SAT$ last assignment.

The *sudoku* problem of size $n$ is as follows. The input is an $n^2 \times n^2$ matrix $M$ whose entries are either "blank" or an integer between 1 and $n^2$. A solution fills in the blank spaces with integers between 1 and $n^2$. The following constraints must be met: Each integer from 1 to $n^2$ appears exactly once in each row, in each column, and in each $n \times n$ sub-matrix of the form $M[jn+1...(j+1)n][in+1..(i+1)n]$ for

each $0 \leq i, j \leq n-1$. The problem is to find any solution meeting the constraints, or return "no solution possible" if there is no such solution.

Last assignment, you gave at least two different ways to reduce the Sudoku problem to $CNF - SAT$.

Try solving sudoku problems by combining the above reductions with a complete SAT solver, such as MiniSAT.

Use as test inputs puzzles of sizes 9 by 9, 16 by 16, and 25 by 25 (and larger, if possible) generated by picking the entries of the first row, first column and main diagonal as random permutations, condiitioned on all entries being consistent with each other.

Be sure to credit the SAT solver you use.

(MiniSAT has been installed by Glen Little on cseclass02. You can ask Glen for an account or install a SAT solver of your choice.)

The experiment should return the following information: For each size, how much time did the SAT solver take using different reductions? Which reduction is best? Did the difficulty ratings reflect in the time taken by these solvers?

(Note: Be careful not to use up too much computer time. Don't leave programs running unsupervised too long. Depending on your algorithm and reduction, you may find even very small sizes take huge amounts of time. )