

CSE 200 HW 3 Solutions

Chris Calabro (mainly)

May 31, 2010

1 NP and $co - NP$

Consider the problem Greater Maximum Independent Set: Given G_1 and G_2 , is the maximum independent set for G_1 strictly larger than that for G_2 ? Show that this problem is in NP if and only if it is in $Co-NP$ if and only if $NP = co-NP$.

We first show that the GMI problem above is hard for both NP and $co-NP$ under polynomial time mapping reductions.

First, we can reduce the NP -complete Big Independent Set problem to GMI as follows: Given a graph G and an integer k , we let G_1 be G and G_2 have $k - 1$ nodes and no edges. Since the maximum independent set for G_2 is of size $k - 1$, this will be a positive example of GMI if and only if $G = G_1$ has an independent set of size $\geq k$.

Similarly, we can reduce the $co-NP$ complete problem, the complement of Big Independent Set, to GMI by setting G_1 to be an independent set of size k and $G_2 = G$. Then G_1 has a strictly larger maximum independent set than G_2 if and only if G_2 has no independent set of size $\geq k$.

Then, if GMI is in $co-NP$, since $co-NP$ is closed under mapping reductions, and Big Independent Set is complete for NP under mapping reductions, all of NP is in $co-NP$ and $NP = co - NP$. Similarly, if GMI is in NP , since it is $co-NP$ hard under mapping reductions, $co - NP \subset NP$ and so $co - NP = NP$.

Conversely, if $NP = co - NP$, there is a relation $R \in P$ and a polynomial p so that $(G, k) \notin BIS$ if and only if $\exists y, |y| = p(n), R((G, k), y)$. Then $(G_1, G_2) \in GIS$ if and only if $\exists k, S, y$ so that S is an independent set of size k in G_1 , and $R((G_2, k), y)$, i.e., maximum independent set for $G_1 \geq k$, and y is a proof that G_2 has no independent set of size k . Since the size of the witness is polynomial ($p(n) + n + O(\log n)$), and all constraints can be verified in polynomial time, this shows that $BIS \in NP$ (if $NP = co - NP$).

2 NP -completeness

For a language L and function $g : \mathbb{N} \rightarrow \mathbb{N}$ with $g(n) \geq n$, define

$$\text{pad}(L, g) = \{x\#^{g(|x|)-|x|} \mid x \in L\},$$

where $\#$ is some symbol not in any string in L . The following lemma will help us translate results up and down the various time hierarchies, and will be used in the next 2 problems.

Lemma 1. *Let $f, g : \mathbb{N} \rightarrow \mathbb{N}$ with $f(n) \in \Omega(n)$, $g(n) \geq n$, g is time constructible. Then*

$$L \in \text{NTIME}(f \circ g) \Leftrightarrow \text{pad}(L, g) \in \text{NTIME}(f).$$

The same result holds for randomized and deterministic time complexity, by the same arguments.

Proof. (\Rightarrow) $\exists N \in \text{NTM } \mathcal{L}(N) = L$ and $\text{time}(N) \in O(f \circ g)$. Define

$N'(y)$
 if y has the form $x\#^{g(|x|)-|x|}$, return $N(x)$
 reject

Then $\mathcal{L}(N') = \text{pad}(L, g)$ and $\text{time}(N') \in O(n + f(n)) = O(f)$. (the n comes from checking the form of y ¹)

(\Leftarrow) $\exists N' \in \text{NTM } \mathcal{L}(N') = \text{pad}(L, g)$ and $\text{time}(N') \in O(f)$. Define

$N(x)$
 return $N'(x\#^{g(|x|)-|x|})$

Then $\mathcal{L}(N') = L$ and $\text{time}(N') \in O(g(n) + f \circ g(n)) = O(f \circ g)$. (the $g(n)$ comes from computing g)

The remaining 2 claims use the same argument. □

Claim 2. $\forall k > 0 \exists L \in \text{NP-complete } L \in \text{TIME}(2^{n^{1/k}})$.

Proof. Consider the reduction f that takes x and appends $n^k - n$ $\#$ symbols to the end of x . f reduces SAT to $L =_{\text{def}} \text{pad}(SAT, n^k)$. L is still in NP , since we can certify that the prefix is in SAT . So L is NP-complete. $SAT \in \text{TIME}(2^n)$, and so, by lemma 1, $\text{pad}(L, n^k) \in \text{TIME}(2^{n^{1/k}})$. □

3 Polynomial-time hierarchy

Claim 3. $\text{NP} \subseteq \text{TIME}(n^{O(\lg n)}) \Rightarrow \forall i \Sigma_i \subseteq \text{TIME}(n^{O(\lg n)^{2^i-1}})$.

Proof. We use induction on i . The base case of $i = 0$ holds since $\Sigma_0 = P = \text{TIME}(n^{O(1)})$. So suppose $i \geq 1$. Then from the inductive hypothesis

$$\Sigma_i = \text{NP}^{\Sigma_{i-1}} \subseteq \text{NP}^{\text{TIME}(n^{O(\lg n)^{2^{i-1}-1}})}.$$

¹A technicality: $\exists c > 0$ s.t. if $g(|x|)$ takes longer to compute than cn , then y definitely does not have the right form, so we really can check the form of y in time $O(n)$.

Let $L \in \Sigma_i$ be solvable by a $p \in \text{poly}$ time NTM with oracle access to a language in $\text{TIME}(n^{O(\lg n)^{2^{i-1}-1}})$. Then by simulating the oracle with a deterministic machine, we see that $L \in \text{NTIME}(p \cdot p^{O(\lg p)^{2^{i-1}-1}}) \subseteq \text{NTIME}(n^{O(\lg n)^{2^{i-1}-1}})$. By lemma 1 we have

$$\text{pad}(L, n^{O(\lg n)^{2^{i-1}-1}}) \in \text{NTIME}(n) \subseteq \text{NP} \subseteq \text{TIME}(n^{O(\lg n)}).$$

By lemma 1 again, we have the very painful expression

$$\begin{aligned} L &\in \text{TIME}(n^{O(\lg n)^{2^{i-1}-1} \cdot O(\lg n^{O(\lg n)^{2^{i-1}-1}})}) \\ &= \text{TIME}(n^{O(\lg n)^{2^i-1}}), \end{aligned}$$

which completes the inductive step. \square

4 Space complexity

Let F be a polynomial-time function where $|F(x)| = |x|$. If F has a fixed-point k on x , then it has one with $k < 2^{|x|}$. Otherwise, since F preserves input size, by the pigeonhole principle, there are at most $2^{|x|}$ strings of length $|x|$, there are $0 \leq k_1 < k_2 < 2^n$ with $F^{k_1}(x) = F^{k_2}(x)$, and since k_1 is not a fixed point, $k_1 < k_2 - 1$. Then the pattern of values between $F^{k_1}(x)$ and $F^{k_2}(x)$ will repeat indefinitely, and there will never be a fixed point.

Then we can compute whether F has a fixed point on x in $PSPACE$ as follows: In the k 'th iteration, we maintain k and $F^k(x)$. In each iteration, we compute $F(F^k(x))$ and then increment k . If $k > 2^n$, we reject, if $F^{k+1}(x) = F^k(x)$ we accept. The total space required is that to store $k, F^k(x), F^{k+1}(x)$ and that to compute F . $k < 2^n$, and F^k and F^{k+1} are also of length n , so the first three require only linear space. Since F is computable in polynomial time, it is also computable in polynomial space, so the total space is polynomial.

Thus, if $P = PSPACE$, we can compute $FIX - F$ in P .

5 Sudoku

Let us generically call each row, column, or $n \times n$ n -aligned submatrix of the board a *region*. We use the following 2 reductions from Sudoku to SAT:

A We associate with each cell (i, j) and color k a variable $x_{i,j,k}$ representing whether that cell gets that color, for a total of n^6 variables. Then we add clauses expressing

- each cell gets a color
(n^4 clauses of width n^2)
- in each region, no 2 cells get the same color
($3n^2 \binom{n^2}{2} n^2 \in O(n^8)$ clauses of width 2)

- the given entries in the diagonal blocks
(n^3 clauses of width 1)

B Same as *A*, but add clauses expressing that each cell gets at most one color ($n^4 \binom{n^2}{2} \in O(n^8)$ clauses of width 2).

We used Zchaff 2007.3.12 on a Macbook Pro 2.33 GHz Intel Core 2 Duo with 2 GB RAM and running Mac OS X 10.4.9. For each value of n and each reduction, we collected 100 samples (except for $n = 5$, where we collected only 10 samples) of the running times of Zchaff, as reported by itself, on the formulas output by the corresponding reduction on inputs drawn from the distribution on boards specified by the problem.

For the reduction *A*, the running times in seconds were

n	mean	std dev
2	9.22e-05	1.95e-11
3	3.97e-04	9.19e-10
4	3.83e-02	5.55e-03
5	no solution in 2 minutes	

For the reduction *B*,

n	mean	std dev
2	9.45e-05	1.97e-11
3	5.34e-04	1.40e-09
4	1.04e-02	7.07e-05
5	6.25e+01	6.46e+02

Of course the running times increased as a function of n , but the times jumped from negligible to more than I am willing to wait too quickly to draw much of a conclusion other than that for Zchaff and these reductions 256 variables is easy but 625 is hard. Using the reduction *B* made the $n = 5$ case just barely feasible.

It is interesting to note that the probability of being satisfiable is $\frac{1}{2}$ for $n = 2$ and seems to be 1 for $n = 3, 4$.