

# CSE 200 HW 2 Solutions

Russell Impagliazzo with Chris Calabro

May 12, 2010

## 1 Restricted 3-SAT

Show that 3-SAT remains *NP*-complete when restricted to formulas where every variable appears in at most three clauses.

Since we can verify that a formula has the above property in polynomial time, the restricted 3-SAT is still in *NP*.

We reduce from arbitrary 3-SAT to the restricted 3-SAT as follows. Let  $\Phi$  be a 3-CNF in variables  $x_1, \dots, x_n$  and with clauses  $C_1, \dots, C_m$ . We construct  $\Phi'$  as follows: Our new set of variables contains a variable  $x_{i,j}$  for each  $1 \leq i \leq n$  and  $1 \leq j \leq m$ . For each clause  $C_j$ , we replace each variable  $x_i$  in the clause by  $x_{i,j}$  and each negated variable  $\neg x_i$  with  $\neg x_{i,j}$ . Call the result  $C'_j$ , and add it to  $\Phi'$ . Note that  $x \leftarrow y$  is equivalent to  $y \vee \neg x$ . Thus, we can express  $x_{i,1} \leftarrow x_{i,2} \leftarrow \dots \leftarrow x_{i,n} \leftarrow x_{i,1}$  as a set of  $m$  clauses of size 2. For each  $i$ , we add these  $m$  clauses to  $\Phi'$ . These two types of clauses are all the clauses in  $\Phi'$ . Note that each  $x_{i,j}$  can only appear up to three times in  $\Phi'$ : once in  $C'_j$ , and once each in the implications between  $x_{i,j-1}$  and  $x_{i,j}$  and between  $x_{i,j}$  and  $x_{i,j+1}$  (operations in indices being done modulo  $m$ ).

This gives the reduction. We now need to show that  $\Phi'$  is satisfiable if and only if  $\Phi$  is. First, assume  $\Phi$  is satisfiable, by assigning each  $x_i$  value  $a_i$ . Then consider the assignment to the variables  $x_{i,j}$  that assigns each  $x_{i,j}$  value  $a_i$ . Since each clause  $C_j$  is satisfied by  $\vec{a}$ , and we assign each  $x_{i,j}$  the value  $x_i$  was assigned, each  $C'_j$  is satisfied. Since we give all  $x_{i,j}$  the same value, each implication is satisfied. Thus,  $\Phi'$  is satisfiable.

If  $\Phi'$  is satisfiable by  $x_{i,j} = a_{i,j}$ , we must have all  $a_{i,j}$  with the same  $i$  equal to satisfy the implication clauses. Define  $a_i$  to be this common value of  $a_{i,j}$ . Then the assignment giving  $x_i$  value  $a_i$  satisfies  $\Phi$ , since  $a_{i,j}$  satisfies each clause  $C'_j$  and we give the  $x_i$  the same value as  $x_{i,j}$ .

Thus, the reduction exactly preserves satisfiability, and hence the restricted version is still *NP*-complete.

## 2 Larger Independent Set

The larger independent set problem is, given a graph  $G$  and an independent set  $I$  in  $G$ , is there an independent set of  $G$  larger than  $I$ ? This problem is in *NP*,

since an independent set larger than  $I$  would have size at most  $n$ , and could be verified in polynomial-time.

To show that it is  $NP$ -complete, we reduce from the Big Independent Set problem: Given  $G, k$ , is there an independent set  $I$  of size  $k$  in  $G$ ?

Given  $G = (V, E)$  and  $k$ , we let  $V'$  be  $V$  together with  $k - 1$  new nodes,  $w_1, \dots, w_{k-1}$ . We let  $E'$  be  $E$  together with  $v, w_j$  for each  $v \in V$  and  $1 \leq j \leq k - 1$ . We let  $I$  be the set  $\{w_1, \dots, w_{k-1}\}$ . The output of the reduction is  $G' = (V', E'), I$ .

We need to show that there is an independent set larger than  $I$  in  $G'$  if and only if there is an independent set of size  $k$  in  $G$ . Assume  $I'$  is an independent set larger than  $I$  in  $G'$ . Since  $|I| = k - 1$ ,  $|I'| \geq k$ . Thus, at least one  $v \in V$  is in  $I'$ , since there are only  $k - 1$  nodes not in  $V$  in  $V'$ . Since there is an edge from  $v$  to each  $w_j$  in  $E'$  and  $I'$  is an independent set in  $G'$ , no  $w_j$  is in  $I'$ . Thus,  $I' \subset V$ . Since all edges from  $G$  are also in  $G'$ ,  $I'$  is an independent set in  $G$  of size at least  $k$ . Taking any  $k$  nodes of  $I'$  gives an independent set of size exactly  $k$ .

Conversely, assume  $G$  has an independent set of size  $k$ ,  $I'$ . Then  $I'$  is also an independent set in  $G'$ , since we added no edges between nodes in  $V$ . Since it is of size  $k$ , and  $I$  is of size  $k - 1$ ,  $|I'| > |I|$ , and there is a bigger independent set in  $G'$  than  $I$ .

Thus, the reduced instance is a member of BiggerIndependentSet if and only if the original instance is a member of BigIndependentSet. Thus, the reduction is correct, and BiggerIndependentSet is  $NP$ -complete.

### 3 Abstract Sudoku

We can represent a solution by giving a color for every variable, and since colors are between 1 and  $k \leq n$ , this will take at most  $n \log k \leq n \log m$  bits. Then to verify a solution, we simply run through all sets  $S_i$  and verify that each color occurs once. Thus, abstract sudoku is in  $NP$ .

To show that it is  $NP$ -complete, we reduce from the known  $NP$ -complete problem, 3-colorability. Given a graph  $G = (V, E)$ , we create an abstract sudoku instance with variables  $x_v$  for  $v \in V$  and  $x_e$  for  $e \in E$ . We set  $k = 3$ . For each edge  $e = \{u, v\}$ , we add the set  $S_e = \{x_u, x_v, x_e\}$ .

We need to show that the constructed abstract sudoku problem is satisfiable if and only if  $G$  is 3-colorable. Assume the sudoku problem is satisfiable by colors  $a_v$   $v \in V$  and  $a_e$ ,  $e \in E$ . Then let  $\chi$  color each node  $u$  the value  $a_u$  assigned to  $x_u$ . For any edge,  $e = \{u, v\}$ , since all values assigned to  $x_u, x_v, x_e$  must be distinct, in particular,  $\chi(u) = a_u \neq a_v = \chi(v)$ , so this is valid coloring. Since  $k = 3$ , we only use three values to color all nodes. Thus,  $\chi$  is a 3-coloring of  $G$ .

In the reverse direction, if  $\chi$  is a 3-coloring of  $G$ , say with colors 1, 2, 3, let  $\bar{a}$  assign each  $x_u$  color  $\chi(u)$ . Then assign  $x_e$  the remaining color not used on either of its two endpoints. Since each endpoint has a distinct color, and we assign  $x_e$  the third color, each of three colors appears once in the set of assigned

colors to  $x_u, x_v, x_e$  for each  $e = \{u, v\}$ . Thus,  $\vec{a}$  is a valid solution to the abstract sudoku puzzle.

Since the reduced instance is equivalent to the original instance, the reduction is valid and we have shown Abstract Sudoku is  $NP$ -complete.

## 4 Bounded Abstract Tiling

This problem is very similar to the proof that abstract tiling is non-recursive, so I will only give a sketch of what additional steps are needed.

First, note that since  $T$  is given in unary, and  $\Sigma$  is part of the input, the description size of the  $T \times T$  matrix of elements of  $\Sigma$  is polynomial, and given such a matrix, we only need to check that each of  $O(T^2)$  consecutive  $2 \times 3$  submatrices is in the allowed set of tiles. Thus, it is polynomial time to verify a solution, and the problem is in  $NP$ .

For the reduction, let  $L$  be any problem in  $NP$ , and let  $M$  be a non-deterministic 1-tape machine accepting  $L$  in time  $T(n) = poly(n)$ . Given  $x$ , as before, we first construct  $M'$  that on empty input, writes  $x$  and simulates  $M$ . Such a machine has only  $O(n)$  more states than  $M$ , which has some fixed number of states, so  $O(n)$  states total. It also takes  $T'(n) = T(n) + O(n)$  time, to write  $x$  and then simulate  $M$ . Then we let  $\Sigma'$  be  $\Sigma$  together with  $\Sigma \times Q$ , where  $Q$  is the set of states for  $M'$ , together with edge symbols. Note that since  $\Sigma$  is constant, and  $|Q|$  is  $O(n)$ ,  $|\Sigma'| = O(n)$ .

We construct the set of tiles as before. Note that each tile is a six-tuple of elements of  $\Sigma'$  and so there are  $O(|\Sigma'|^6) = O(n^6)$  such tiles. (Actually, a sharper analysis gives  $O(n^2)$ , since at most 2 of the six elements involve states.) We let  $T = T'(n)$ , so  $1^T$  is of polynomial length. From the above, the reduction is polynomial time. The proof of equivalence is the same as in class.

## 5 Reduction from sudoku to SAT

Chris's solution:

Here are 2 reductions from  $n^2 \times n^2$  sudoku to SAT. Each relies on representing a solution as a function  $f$  from the set of cells to the set  $[n^2]$ . We will use the generic term *region* to refer to a row, column, or  $n$ -aligned  $n \times n$  submatrix of the sudoku board. Note that there are  $3n^2$  such regions.

1. Each variable represents whether a particular ordered pair is in  $f$ ; i.e. variable  $x_{i,j,k}$  represents whether cell  $(i, j)$  has the value  $k$ . We need  $n^6$  such variables. The constraint that each cell gets at least one number can be represented by  $n^4$   $n^2$ -clauses. The constraint that no 2 cells in the same region get the same number can be represented by  $3n^2 \binom{n^2}{2} n^2 = \Theta(n^8)$  2-clauses. (That each cell gets at most one number is forced by the above.)
2. We represent the number in each cell in binary. This requires  $2(\lg n)n^4 = \Theta(n^4 \lg n)$  variables. That each cell gets a number is now represented

implicitly. That no 2 cells in the same region get the same number can be represented with  $3n^2 \binom{n^2}{2} n^2 = \Theta(n^8)$  4 lg  $n$ -clauses.

Despite the fact that the 2nd method generates fewer variables and a smaller maximum clause width, SAT solvers would probably perform worse on such instances than on the instances generated by the 1st method since the 1st method generates so many 2-clauses, which promote unit propagation.

Russell's solution: (a bit more detailed and has a different approach)

There are some obvious ways of performing the reduction. In these ways, the variables are  $x_{i,j,k}$  representing " $M[i][j] = k$ ". Let  $B_{(a,b)}$  be the  $(a, b)$ 'th block, the set of positions  $(i, j)$  with  $(a-1)n+1 \leq i \leq an$  and  $(b-1)n+1 \leq j \leq bn$ .

We can define the following sets of clauses:

$V_{\leq 1}$ : (at most one value)

For each  $(i, j)$ , there is at most one  $k$  with  $M[i][j] = k$ : for each  $1 \leq i \leq n^2, 1 \leq j \leq n^2, 1 \leq k < k' \leq n^2, \neg x_{i,j,k} \vee \neg x_{i,j,k'}$ .

( $O(n^6)$  clauses of size 2.)

$V_{\geq 1}$ : (at least one value)

For each  $(i, j)$ , there is at least one such  $k$  with  $M[i][j] = k$ : for each  $1 \leq i \leq n^2, 1 \leq j \leq n^2, \vee_{1 \leq k \leq n^2} x_{i,j,k}$ . ( $O(n^4)$  clauses of size  $n^2$ )

$R_{\leq 1}$ : (at most one  $k$  in each row)

For each row,  $k$  appears at most once: for each  $1 \leq i \leq n^2, 1 \leq k \leq n^2, 1 \leq j < j' \leq n^2, \neg x_{i,j,k} \vee \neg x_{i,j',k}$ . ( $O(n^8)$  clauses of size 2.)

$R_{\geq 1}$ : (at least one  $k$  in each row)

For each row there is at least one  $k$ : with  $M[i][j] = k$ : for each  $1 \leq i \leq n^2, 1 \leq k \leq n^2, \vee_{1 \leq j \leq n^2} x_{i,j,k}$ .

( $O(n^4)$  clauses of size  $n^2$ )

$C_{\leq 1}$  For each column,  $k$  appears at most once: for each  $1 \leq j \leq n^2, 1 \leq k \leq n^2, 1 \leq i < i' \leq n^2, \neg x_{i,j,k} \vee \neg x_{i',j,k}$ . ( $O(n^8)$  clauses of size 2.)

$C_{\geq 1}$ : (at least one  $k$  in each column)

For each  $1 \leq j \leq n^2, 1 \leq k \leq n^2, \vee_{1 \leq i \leq n^2} x_{i,j,k}$ .

( $O(n^4)$  clauses of size  $n^2$ )

$B_{\leq 1}$  For each block,  $k$  appears at most once: for each  $1 \leq a \leq n, 1 \leq b \leq n, 1 \leq k \leq n^2, (i, j) \in B_{a,b}, (i', j') \in B_{a,b}, (i, j) \neq (i', j'),$

$\neg x_{i,j,k} \vee \neg x_{i',j',k}$ . ( $O(n^8)$  clauses of size 2.)

$B_{\geq 1}$ : (at least one  $k$  in each block)

For each  $1 \leq a \leq n, 1 \leq b \leq n, 1 \leq k \leq n^2, \vee_{(a-1)n+1 \leq i \leq an, (b-1)n+1 \leq j \leq bn} x_{i,j,k}$ .

( $O(n^4)$  clauses of size  $n^2$ )

Many combinations of these constraints are valid reductions. (Together with  $x_{i,j,k}$  whenever we are given that  $M[i, j] = k$ ).

For example,  $V_{\geq 1}, R_{\leq 1}, B_{\leq 1}, C_{\leq 1}$ . People were conjecturing different kinds of tradeoffs between more clauses versus smaller clause size. But actually, for complete methods (algorithms that explore all possibilities), it is usually better to have as many constraints as possible. My guess is it would be better to just include all constraints above than to select subsets for these methods. Other heuristic search methods such as WalkSAT might get confused by a large number

of “less important” constraints. However, all of these methods have  $O(n^6)$  variables, which may make them prohibitively large.

But here’s another method that uses  $O(n^5)$  variables instead. Consider variables  $Y_{i,b,k}$  and  $Z_{a,j,k}$  for  $1 \leq i \leq n^2, 1 \leq a, b \leq n, 1 \leq k \leq n^2$  that express: there is a  $k$  in row  $i$  in a column between  $n(b-1)+1$  and  $nb$  (i.e., in the intersection of row  $i$  and block  $B_{i \text{ div } n, b}$ ) and likewise for columns. These variables also determine the solution by  $M[i][j] = k$  if and only if  $Y_{i, j \text{ div } n, k}$  and  $Z_{i \text{ div } n, j, k}$ . Plus, the above constraints can be “factored” into these new variables:

For example:  $R_{\leq 1}$ : (at most one  $k$  in each row)

For each  $1 \leq i \leq n^2, 1 \leq k \leq n^2, 1 \leq b < b' \leq n, \neg Y_{i,b,k} \vee \neg Y_{i,b',k}$ . ( $O(n^6)$  clauses of size 2.)

$R_{\geq 1}$ : (at least one  $k$  in each row)

For each row there is at least one  $k$ : with  $M[i][j] = k$ : for each  $1 \leq i \leq n^2, 1 \leq k \leq n^2, \vee_{1 \leq b \leq n} Y_{i,b,k}$ .  
( $O(n^4)$  clauses of size  $n$ )

This both reduces the number of variables from  $n^6$  to  $2n^5$  and the maximum clause width from  $n^2$  to  $n$ . (We’ll also have some clauses of size 4 saying that there can’t be two values in the same square,  $\neg Y_{i,b,k} \vee \neg Y_{i,b,k'} \vee \neg Z_{a,j,k} \vee \neg Z_{a,j,k'}$  for  $1 \leq i \leq n^2, 1 \leq j \leq n^2, a = i \text{ div } n, b = j \text{ div } n, 1 \leq k < k' \leq n^2$ .) My feeling is that this would be a dramatic improvement over the reductions above, but no experiments have been made.