COMS4236, Spring 2014 Final Exam Solutions

Problem 1.

1. a. The decision problem corresponding to this optimization problem is: "Given an instance I of W-MAX-3SAT (i.e. a set of clauses and associated weights) and an integer k, does there exist a truth assignment with value at least k?". The decision problem is in NP because we can guess a truth assignment, and then in polynomial time we can determine all the clauses that it satisfies, add up the weights of these clauses to compute the value of the assignment, and verify that the value is at least k. From the assumption that P=NP, it follows that the decision problem is in P.

We use a polynomial-time procedure for the decision problem to compute the optimal value k^*. The optimal value is the maximum integer k for which the answer to the decision problem is Yes. The optimal value is between 0 and the sum M of the weights of the clauses. We can find the optimal value k^* by performing a binary search in the range $[0, M]$ using the polynomial-time algorithm for the decision problem to answer questions of the form "Does there exist a truth assignment with value at least k?" The number of questions needed to determine the optimal value k^* is $\log M$, which is polynomial in the number of bits of the given clause weights. Hence the running time is polynomial in the size of the input.

b. After determining the optimal value k^*, we compute an assignment that achieves this value by setting the truth value of the variables one by one. Consider the following problem: "Given an instance I of W-MAX-3SAT, a partial assignment α to a subset of the variables, and an integer k, does there exist an assignment to all the variables that extends α and which satisfies a set of clauses with weight at least k?" This problem is in NP, hence also in P according to the hypothesis. Let $DEC(I, \alpha, k)$ be a polynomial-time procedure that decides this problem. We can compute an optimal assignment as follows.

α = empty assignment(all variables unassigned)
for $i = 1$ to n **do**
  if $DEC(I, \alpha \cup \{x_i = 1\}, k^*)$ =true **then**
    $\alpha = \alpha \cup \{x_i = 1\}$
  else
    $\alpha = \alpha \cup \{x_i = 0\}$

The assignment α at the end of the algorithm has value k^*, and therefore

it is an optimal assignment.

c. The problem "Given an instance I of the W-MAC-3SAT problem and integer k, are there at least two assignments that have value at least k" is in NP, because we can guess two assignments and verify that their value is at least k. From the hypothesis that P=NP, the problem is also in P. Thus, once we have computed the optimal value k^*, we can determine in polynomial time whether there are at least two assignments with value k^* or whether there is a unique optimal assignment.

2. We reduce 3SAT to the problem of determining whether there are two or more optimal assignments for a given instance of W-MAX-3SAT. This implies that, if the problem of part 1c is in P, then so is 3SAT, and hence P=NP.

Let F be a given set of 3-clauses $C_1, \ldots, C_m$ in variables $x_1, \ldots, x_n$. Introduce a new variable z and variables $y_1, \ldots y_m$ (corresponding to the clauses of F), and let F' be the following set of clauses: For each clause $C_i = a_i \vee b_i \vee c_i$ of F (where a_i, b_i, c_i are literals), F' contains 2 clauses $a_i \vee b_i \vee y_i$ and $\bar{y}_i \vee c_i \vee z$. In addition F' contains the clauses $\bar{z} \vee x_j$ for all $j = 1, \ldots, n$, and $\bar{z} \vee y_i$ for all $i = 1, \ldots, m$. All clauses of F' have weight 1. (F' contains some clauses with 2 literals. If we want all clauses to have exactly 3 literals, then we can modify the reduction as we did in class to show the NP-completeness of 3SAT with exactly 3 literals per clause: Replace every 2-literal clause $\bar{z} \vee x_j$ by two clauses $\bar{z} \vee x_j \vee x_l$ and $\bar{z} \vee x_j \vee \bar{x}_l$, where x_l is any variable other than x_j, and similarly replace each clause $\bar{z} \vee y_i$ by two clauses $\bar{z} \vee y_i \vee x_l$ and $\bar{z} \vee y_i \vee \bar{x}_l$.)

The claim is that (i) if F is satisfiable then F' has at least 2 satisfying assignments, whereas (ii) if F is not satisfiable then F' has a unique satisfying assignment. Since all the clauses of F' have weight 1, this means that F is unsatisfiable iff the constructed instance F' of W-MAX-3SAT has a unique optimal assignment.

Proof of (i): Note that every clause of F' contains a positive literal. One satisfying assignment of F' consists of setting all variables to 1 (true). For every satisfying assignment τ of F we can form another satisfying assignment for F' by setting variables x_j according to τ, setting $z = 0$, and giving y_i the truth value of the literal c_i for each $i = 1, \ldots, m$.

Proof of (ii): As above, the all-1 assignment satisfies all the clauses of F'. Suppose that there is another satisfying assignment. If $z = 1$, then all the other variables must also be 1 because of the clauses $\bar{z} \vee x_j$ and $\bar{z} \vee y_i$. Therefore, the other satisfying assignment must have $z = 0$. Consequently, for every clause $C_i = a_i \vee b_i \vee c_i$ of F, the other assignment must satisfy both

clauses $a_i \vee b_i \vee y_i$ and $\bar{y}_i \vee c_i$. If $y_i = 1$ then c_i must be 1, and if $y_i = 0$ then at least one of a_i, b_i must be 1. Hence in either case the clause C_i is satisfied by the assignment. Thus, the assignment to the variables x_j satisfies all the clauses of F, contradicting the assumption that F is not satisfiable.

Problem 2

1. By the Space Hierarchy Theorem, L =SPACE($\log n$) is a proper subset of SPACE(n), hence L $\neq$ PSPACE. If P = L, it follows that P $\neq$ PSPACE.

2. We know that P $\subseteq$ PSPACE. We use the hypothesis to show the other direction, PSPACE $\subseteq$ P. Let L be any language in PSPACE, i.e. $L \in$ SPACE(n^c) for some constant c. Define the language $L' = pad(L, n^{2c})$. Then $L' \in$ SPACE($\sqrt{n}$). Thus, by the hypothesis it follows that $L' \in P$, i.e. $L' \in$ TIME(n^d) for some constant d. Consequently we can decide L in time $O(n^{2cd})$ as follows: Given an input x of length n, pad out x to length n^{2c}, and apply the polynomial algorithm for L' which will run in time $O((n^{2c})^d) = O(n^{2cd})$. Therefore, $L \in P$.

3. Let's call this the "2-prime problem". We show that this problem is in coNP, i.e. that the complementary problem is in NP. A suitable certificate that a given number N is not the product of two primes is the prime factorization of N. Nondeterministically guess the prime factorization of N, i.e. guess a list of (at most $\log N$) numbers, all $\leq N$, verify that the numbers are all primes and that their product is equal to N. If these verifications hold and if the factorization does not consist of exactly two primes, then accept; otherwise reject. The nondeterministic algorithm runs in polynomial time and accepts iff N is not the product of two primes. (Alternatively, we can show that the complementary problem is in NP by noting that a number N is not the product of two primes iff either N is a prime or N is the product of 3 integers > 1.)

We know that if an NP-complete problem is in coNP then NP=coNP. Thus, if the 2-prime problem is NP-complete then NP=coNP.

Problem 3.

Let $D = A \cdot B$. We want to test if $C = D$, i.e. whether $C[i,j] = D[i,j]$ for all $1 \leq i, j \leq n$. If x, y are respectively a row n-vector and column n-vector of variables, the expression $x \cdot C \cdot y$ is the polynomial $\sum_{i=1}^{n} \sum_{j=1}^{n} C[i,j] x_i y_j$ and similarly $x \cdot D \cdot y = \sum_{i=1}^{n} \sum_{j=1}^{n} D[i,j] x_i y_j$. Thus, $C = D$ iff the two

polynomials are identical, i.e. if their difference is the 0 polynomial. The degree of the polynomials is 2. If $C = D$ then any choice of values for the variables yields the same value for the polynomials. If $C \neq D$ then by the Schwartz-Zippel Lemma, if we pick random values for the variables from a set of size M (for example from the set $\{1, \ldots, M\}$) then the probability that they yield the same value for the polynomials is at most $2/M$, and thus it is at most 0.01 if $M \geq 200$.

Thus, we use the following randomized algorithm. Pick random integer values in $[1, 200]$ for the entries of the row vector x and column vector y, and compute $x \cdot (C \cdot y)$ and $(x \cdot A) \cdot (B \cdot y)$; if they are equal then accept else reject. If $A \cdot B = C$ then the algorithm accepts with probability 1. If $A \cdot B \neq C$ then it accepts with probability at most 0.01, i.e., it rejects with probability at least 0.99. The algorithm involves three matrix-vector products ($C \cdot y$, $x \cdot A$ and $B \cdot y$) which take time $O(n^2)$ and two vector-vector products that take time $O(n)$. Thus, the total running time is $O(n^2)$.

Problem 4

The SCG problem is NL-complete.

1. SCG is in NL. For each pair of nodes u, v, guess a path from u to v, one node at a time. The algorithm has to keep track only of u, v and the current node on the path, i.e. $O(\log n)$ bits.

2. SCG is NL-hard. We will reduce from the Graph Reachability problem. Given a directed graph $G = (N, E)$ and two nodes s, t, we will construct in log space another graph G' such that G has path from s to t iff G' is strongly connected. The graph $G' = (N, E')$ has the same set N of nodes as G. Its set E' of edges includes all edges of G, and in addition it includes an edge to s from every other node of the graph, and an edge from t to every other node of the graph. That is, $E' = E \cup \{(u, s) | u \in N - \{s\}\} \cup \{(t, v) | v \in N - \{t\}\}$. Clearly, G can be constructed from G by a log space TM.

Claim: G has a path from s to $t \iff G'$ is strongly connected.

Proof:

($\Rightarrow$) For every pair of nodes u, v, we can form a path from u to v in G' by taking first the edge (u, s) if $u \neq s$, then following the path of G from s to t, and finally taking the edge (t, v) if $v \neq t$.

($\Leftarrow$) The graph G' contains a path from s to t, and hence it contains in particular a simple path. The path does not contain any edge coming into s or going out of t, hence it consists entirely of edges that are in G. Therefore, G contains a path from s to t.

Problem 5.

1. *Problem is in PSPACE.* We define a recursive function $\text{Win}(H, col)$ which takes as input a hypergraph H and an assignment col of colors (red or blue) to an initial subset of nodes $\{1, \ldots, i-1\}$ and returns true iff Player 1 can win the game on H with the colors of nodes $\{1, \ldots, i-1\}$ fixed by col. The main call is $\text{Win}(H, \emptyset)$ where $\emptyset$ is the empty assignment.

The recursive algorithm is as follows

```
Win(H, col) {
if all nodes are colored in col then
  if there is no monochromatic hyperedge then
    return true
  else
    return false
else
  let i be the next uncolored node
  if i is odd then
    /* i is colored by Player 1 */
    return Win(H, col ∪ [col(i) = red]) ∨ Win(H, col ∪ [col(i) = blue])
  else
    /* i is colored by Player 2 */
    return Win(H, col ∪ [col(i) = red]) ∧ Win(H, col ∪ [col(i) = blue])
}
```


The depth of the recursion is n, the number of nodes. The nonrecursive implementation of the algorithm has to keep track of the recursion stack, which is essentially the partial coloring α and constant additional information, thus it needs space $O(n)$.

2. *Problem is PSPACE-hard.* We reduce from QSAT. We may assume without loss of generality that in the given QSAT formula the quantifiers alternate starting and ending with a universal quantifier (we can always add extra dummy variables that don't appear in the formula to ensure this). That is, the given QSAT formula F is of the form

$$F = \forall x_1 \exists x_2 \forall x_3 \cdots \forall x_n . C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

Note that the question of whether a given hypergraph has a legal 2-coloring (called the "Hypergraph 2-coloring problem") is very similar to the NAE-SAT problem. In fact the Hypergraph 2-coloring problem can be viewed as the special case of the NAE-SAT problem where the NAE-SAT instance has no negative literals. Our reduction from QSAT to the Hy-

pergraph 2-Coloring Game problem will be similar to the NP-completeness reduction for NAE-SAT.

Our hypergraph $H = (N, E)$ contains a node z, $2n$ nodes $\{x_j, \bar{x}_j | j = 1, \ldots, n\}$ corresponding to the literals, and n additional dummy nodes $\{y_j | j = 1, \ldots, n\}$. The ordering of the nodes is as follows: $z, x_1, x_2, \ldots, x_n, \bar{x}_1, y_1, \bar{x}_2, y_2, \ldots \bar{x}_n, y_n$.

The hypergraph H contains the following set E of hyperedges:

- a hyperedge $\{x_j, \bar{x}_j\}$ for every $j = 1, \ldots, n$.
- a hyperedge $C'_i = C_i \cup \{z\}$ for every $i = 1, \ldots, m$ that contains the literals of clause C_i and node z.

Clearly the reduction can be carried out in polynomial time (and in log space).

Note that the dummy nodes $\{y_j | j = 1, \ldots, n\}$ are not involved in any hyperedge. Their purpose is only to conform to the requirement that the players alternate choosing colors for the nodes in the prescribed order. According to this order, Player 1 chooses the color for node z, all nodes x_j with even index j (the ones corresponding to the existentially quantified variables in the QSAT formula F) and for all nodes $\bar{x}_j$ with both odd and even index. Player 2 chooses the color for the odd-indexed nodes x_j (the ones corresponding to the universally quantified variables in the QSAT formula F) and for the dummy variables y_i.

Claim: The QSAT formula F is true $\Longleftrightarrow$ Player 1 has a winning strategy in the 2-coloring game for the hypergraph H.

Proof:

($\Rightarrow$) Suppose that F is true. Thus, the Existential player has a winning strategy in the QSAT game for formula F. Player 1 simulates the winning strategy of the existential player where we let red color correspond to truth value 1 and blue to 0. Player 1 chooses blue for node z, then chooses the values of the even-indexed nodes x_j following the winning strategy of the Existential player, and chooses for each node $\bar{x}_j$ the opposite color to that of x_j. Clearly this coloring is legal for all the hyperedges $\{x_j, \bar{x}_j\}$. Furthermore, since every clause C_i contains a true literal, every hyperedge C'_i contains a literal node colored red and node z that is colored blue, hence it is legally colored.

($\Leftarrow$) Suppose that Player 1 has a winning strategy for H. We show that the Existential Player has a winning strategy in the QSAT game for F. Player 1 starts the game for H by assigning a color to node z. After that, players 1 and 2 alternate selecting colors for the nodes $x_1, \ldots x_n$. The Existential player follows the strategy of Player 1 where the color of node z

corresponds to the truth value 0 (false) and the opposite color to the value 1. Because of the hyperedges $\{x_j, \bar{x}_j\}$, each node $\bar{x}_j$ gets opposite color to that of x_j. Each hyperedge C'_i contains a literal node with color opposite to z, and thus the corresponding clause C_i contains a true literal. That is, the truth assignment at the end of the QSAT play satisfies all the clauses and hence the Existential player wins. Therefore, F is true.