CSC373H1 Summer 2014 Assignment 4

Names: John Armstrong, Henry Ku
SNs\CDF username: 993114492\g2jarmst, 998551348\g2kuhenr

Question #	Score
1	
2	
3	
4	
Total	

Acknowledgements:

"We declare that we have not used any outside help in completing this assignment."

Name: John Armstrong, Henry Ku

Date: August 8, 2014

Q1. The Mute Prison

Claim: The mute prison problem is NP-complete.

Proof:

1. Show the mute prison problem is NP.

2. Show the mute prison problem is NP-hard.

1. Suppose we are given a certificate S and have access to value k and matrix T. We can verify that the certificate is satisfiable in the following way. Suppose each element in S represents an inmate. Verification would involve iterating on each inmate in the following way:

```
for inmate in S do
    j = 1;
    while j ⩽ m do
        if T[inmate, j] then
            for (otherinmate ≠ inmate) in S do
                if T[otherinmate, j] then
                    S is not a subset of inmates who do don't speak the same language;
                    return 0;
                end
            end
        end
        j++;
    end
end
return 1;
```

Clearly, the verification that S is a subset where no two inmates speak the same language can run in polynomial time O(mn^2). Once this verification is complete all that is left to do is to verify that |S| $\geqslant$ k, which is O(1). Therefore the mute prison problem is NP. ■

2. To show that the mute prison problem is NP-hard we must perform a reduction using an NP-complete problem. We will use a reduction on NP-complete 3-SAT in CNF, in order to show 3-SAT $\leqslant_p$ Mute Prison Problem.

Properties of Reduction

Suppose that ϕ is an instance of 3-SAT and C_1, C_2, ..., C_m are the clauses of ϕ. By construction of 3-SAT in CNF we have $C_i = (z_{i1} \vee z_{i2} \vee z_{i3})$. In the reduction each C_i's boolean value will represent a boolean value for each language, L_i, spoken by some inmate(s), precisely, $L_i = C_i = (z_{i1} \vee z_{i2} \vee z_{i3})$. Each boolean value for L_i has a specific mean:

$$L_i = \begin{cases} 1 & \text{if } L_i \text{ is spoken by at most 1 inmate} \\ 0 & \text{if } L_i \text{ is spoken by at least 1 inmate} \end{cases}$$

Producing L_1, L_2, ..., L_m will take polynomial time since we iterate through each C_i and perform a boolean or operation on each z_i in C_i which takes O(m).

Finally, the mute prison problem requires a matrix T to produce the subset of inmates S. Let T be an m x m matrix, so that no inmates are left without a language. The rows in T will represent inmates and the columns will represent languages such that column i represents L_i. The algortihm that peforms the reduction will iterate through each L_i. If $L_i = 1$ then set T[i, i] = 1, else if $L_i = 0$ then set T[1, i] = T[2, i] = ... = T[m, i] = 1. Assigning all inmates to speak L_i, when $L_i = 0$, will guarantee that $|S| = \emptyset$. Alternatively, $\forall$ i, if $L_i = 1$ then $|S| = m$. So that if ϕ in 3-SAT is satisfiable, then T will satisfy the mute prison problem if we set k = m. Again this process is polynomial as it iterates through m L_i's and assigns at most m inmates the language L_i, so it will run $O(m^2)$.

ϕ of 3-SAT is satisfiable $\rightarrow$ L and k of mute prison problem is satisfiable

Suppose ϕ of 3-SAT is satisfiable, then each clause $C_1, C_2, ..., C_m$ is satisfied. A set of $L_1, ..., L_m$ is produced such that $\forall$ L_i, $L_i = 1$. Then we form matrix T of size m x m, such that T resembles the identity matrix as each T[i,i] = 1. Also, k =m, so that when S is assembled all m inmates speak a different language, then $|S| \geqslant k$ is satisfied.

L and k of mute prison problem is satisfiable $\rightarrow$ ϕ of 3-SAT is satisfiable

Suppose that T and k of the mute prison problem are satisfiable. Also, suppose $|S| = m = k$. Suppose T is an m x m matrix that resembles an identity matrix. We will attribute the m columns in T to variables $L_1, ..., L_m$, such that, $1 \leqslant i \leqslant m$, and set $L_i = 1$ if the column has at most one entry equal to 1, and set $L_i = 0$ otherwise. Since T and k satisfy the problem then all $L_i = 1$. We then form m clauses of a 3-SAT CNF, call them C_i, ..., C_m. Each C_i relates to L_i, so that the boolean value of $C_i = (z_{i1} \vee z_{i2} \vee z_{i3}) = 1$. Thus set any one of the z_{i1}, z_{i2}, or z_{i3} to 1 (or true) to set C_i to 1. If a column in T has more than one entry with 1 then clearly the mute prison problem would not be satistified and some $C_i = 0$ (or false) and ϕ would not be satisfied. It follows that all C_i equal 1 since all L_i equal 1, thus $\phi = (C_1 \wedge C_2 \wedge ... \wedge C_m)$ is satisfiable.

So, ϕ of 3-SAT is satisfiable $\Leftrightarrow$ L and k of mute prison problem is satisfiable . Also, because the reduction was shown to be polynomial it is proven that the mute prison problem is NP-hard. ■

By the proofs <u>1.</u> and <u>2.</u> it follows that the mute prison problem is NP-complete. ■

Q2. The Nonsense Prerequisites

Claim: The nonsense prerequisites problem is NP-complete.

Proof:

1. Show the nonsense prerequisites problem is NP.

2. Show the nonsense prerequisites problem is NP-hard.

1. Suppose we know G(V, E) and k and we are given E' as a certificate. We verify the certificate with the following algorithm:

```
E" = E - E';
Produce function w, such that ∀ (u, v) ∈ E", w(u, v) = -1;
Produce new G'(V, E", w);
for v in V do
    Perform Bellman-Ford(G', w, v);
    for each edge (u, v) ∈ G'.E" do
        if v.d > u.d + w(u, v) then
            There is a cycle and the certificate is not satisfiable;
            return 0;
        end
    end
end
return 1;
```

If there is a cycle in G'(V, E") then setting each edge in G' to a weight -1 will produce a negative edge cycle which, after relaxations, we can identify easily. Given that G(V, E") may or may not be connected, to locate a cycle in the graph we must perform the relaxation with Bellman-Ford |V| times. Bellman-Ford runs at O(VE), it is executed |V| times in the verifier, thus we have O(V^2E) for our algorithm. Since |V| = n, and |E| = O(n^2), the verifier runs O(n^4). So the verifier is polynomial and then the nonsense prerequisites problem is NP. ■

2. To show the nonsense prerequisites problem is NP-hard, as directed by the problem set, we will perform a reduction using NP-complete problem VECTOR COVER. So, we will show VECTOR COVER $\leqslant_p$ The Nonsense Prerequisites Problem.

Properties of Reduction

Take the G(V, E) and k given to the VECTOR COVER problem. k represents $|S| \leqslant k$, such that $S \subseteq V$ such that if $(u, v) \in E$, then $u \in S$ or $v \in S$. However, in the nonsense prerequisites, the k corresponds to edges that when removed from the graph will make it acyclic. It follows that the reduction must somehow convert the vertices in G to represent edges. This is done by splitting each vertex in two, so given V = {v_1, v_2, ..., v_n}, produce V' = {v_{pre-1}, v_{post-1}, v_{pre-2}, v_{post-2}, ..., v_{pre-n}, v_{post-n}}, and ∀ i, $1 \leqslant i \leqslant n$, ($v_{pre-i}$, v_{post-i}) is a directed edge such that (v_{pre-i}, v_{post-i}) ∈ E'. Also, we must create a circumstance in the new graph where each undirected edge (v_i, v_j) ∈ E, becomes directed edges (v_{post-i}, v_{pre-j}) ∈ E' and (v_{post-j}, v_{pre-i}) ∈ E'. This construction guarantees in G'(V', E') that when we enter any v_{pre-i} we can walk a path $v_{pre-i} \rightarrow v_{post-i} \rightarrow v_{pre-j} \rightarrow v_{post-j} \rightarrow v_{pre-j}$, and indeed this is a cycle. Thus, we have a cycles, such that if (v_i, v_j) ∈ E, then the cycle is limited to the new vertices {v_{pre-i}, v_{post-i}, v_{pre-j}, v_{post-j}}. So the reduction is complete and can easily be performed in polynomial time. O(nα(m + n)) to produce new directed edges from m existing edges, splitting vertices in V and creating new edges, and adding them to the new graph G' using make-set, union, and link.

G(V, E), k of VECTOR COVER is satisfiable $\rightarrow$
G*(V*, E*), k of the nonsense prerequisite is satisfiable
Suppose using undirected G(V, E) and k, VECTOR COVER is satisfied. Suppose also that we have access to S = $\{s_1, ..., s_q\}$, which is the vertex cover of G and $|S| \leqslant k$. We perform the reduction and have G*(V*, E*). It follows in G* any cycles is limited to $\{v_{pre-i}, v_{post-i}, v_{pre-j}, v_{post-j}\}$. To break a cycle in G* we could remove any edge from the cycle, but to do this efficiently we need to remove edges that break many cycles at once. This is precisely E' = $\{(s_{pre-1}, s_{post-1}), ..., (s_{pre-q}, s_{post-q})\}$, because in G* the edges in E' that correspond to vertices in S, are precisely the set of edges that appear in all cycles. Thus $|E'| = |S| \leqslant k$, and so G*(V*, E*) and k of the nonsense prerequisite is satisfiable.
G(V, E), k of the nonsense prerequisite is satisfiable $\rightarrow$
G*(V*, E*), k of VECTOR COVER is satisfiable
Suppose G(V, E), k when used in the nonsense prerequisite problem is satisfiable. Now, to establish a contradiction, suppose the original graph, G*(V*, E*) and k in VERTEX COVER were not satisfiable. This would mean that the set of vertex cover $S \subseteq V$, $|S| >k$. But since G(V, E) and k were satisfiable then $|E'| \leqslant k$. But by the construction of the reduction this in impossible. Since every $(v_{pre-i}, v_{post-i}) \in E'$ corresponds to a vertex $v_i \in S$, this will mean that there is some $v_i \in S$ that is not represented in E', since $|E'| < |S|$. This means that there is some cycle left over in G when E - E' is performed. So then a contradiction is reached based on our original assumption, and so G*(V*, E*), k of VECTOR COVER must be satisfiable.

By proving both directions, it follows that G(V, E), k of VECTOR COVER is satisfiable $\leftrightarrow$
G*(V*, E*), k of the nonsense prerequisite is satisfiable. ■

Additionally, since the reduction can be performed in polynomial time, then the nonsense prerequisite problem is NP-hard. ■

Q3. T-rex Christmas

1. {0, 1}-integer programming formulation:

Minimize:

P

Subject To:

a)

$x_q^1 \in \{0, 1\}$ $\forall$ q s.t. q represents a path of transferring a present, i $\rightarrow$ j, such that L[i, j] = 1

$x_q^0 \in \{0, 1\}$

b)

$x_q^1 + x_q^0 \geqslant 1$

c)

$\sum_{\forall q} x_q^{f(q,\ m)} \leqslant P$ $\forall$ m $\in$ {0, 1, ..., n-1}

Explanation of Constraints:

a) x_q^1 and x_q^0 represent the direction (x_q^1 clockwise and x_q^0 counter-clockwise) a gift q will be passed from i to j, such that i, j $\in$ {0, 1, ..., n-1}. If $x_q^1 = 1$ then the gift will be sent clockwise, if $x_q^1 = 0$ it will not be sent clockwise (same for x_q^0 but counter-clockwise).

b) This constraint limits x_q^1 and x_q^0 so both cannot be 0, and it is tightly bound when only one of the variables equals 1.

c) Assume some linear function f, and also assume m represents a pass (m, m+1[n]). f(q, m) = 1 if on the clockwise direction of passing a gift from i to j a pass in (m, m+1[n]) occurs. Conversely, f(q, m) = 0 if on the counter-clockwise direction of passing a gift from i to j a pass in (m, m+1[n]) occurs. We can assume linearity of f because the computation of the intersection of gift direction and passes will have been computed previously and values will be contained in some 2-D matrix which f has access to at will. Finally, each summation for each m will represent, when the IP completes, each P_m. Clearly if the pass (m, m+1[n]) is used during the passing of gift q, then $x_q^{f(q,\ m)} = 1$, the it is added to P_m. Additionally, the constraint is limited to P, as a tight bound, P is some value that will equal the maximum of all P_m's.

2. LP relaxation:

Minimize:

P

Subject To:

a)

$0 \leqslant x_q^1 \leqslant 1$ $\forall$ q s.t. q represents a path of transferring a present, i $\rightarrow$ j, such that L[i, j] = 1

$0 \leqslant x_q^0 \leqslant 1$

b)

$x_q^1 + x_q^0 \geqslant 1$

c)

$\sum_{\forall q} x_q^{f(q,\ m)} \leqslant P$ $\forall$ m $\in$ {0, 1, ..., n-1}

Rounding Scheme:

Let all x_1^1*, x_1^0*, x_2^1*, x_2^0*, ..., x_s^1*, x_s^0* (s being the number of gifts to be exchanged) and P be the solution returned by the relaxed LP. Allow us to round all x_q^1*'s and x_q^0*'s to produce y_1^1, y_1^0, y_2^1, y_2^0, ..., y_s^1, y_s^0, and P'. So, let this represent the solution to the integer programming such that:

$$y_q^1 \ or \ y_q^0 = \begin{cases} 1 & \text{if corresponding } \mathrm{x}_q^{1*} \text{ or } \mathrm{x}_q^{0*} \geqslant \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{P'} = \max_{0 \leqslant r \leqslant n-1} (\text{P'}_r = \textstyle\sum_{\forall q} \mathrm{y}_q^{f(q,\ r)})$$

Proof that Rounding Scheme works:

All constraints in the interger program will be satisfied. For example a) is satisfied because y_q^1 or $\mathrm{y}_q^0 \in \{0, 1\}$. Constraint b) is satisfied even in the unfortunate case where x_q^{1*} and x_q^{0*} equals 1/2 or greater, then y_q^1 and y_q^0 will equal 1. Additionally both y_q^1 and y_q^0 cannot both equal 0 by the constraint b). Finally, in constraint c), P is a bound specified by the algorithm, and indeed with the specified values for y_q^1 and y_q^0 the constraint will be satisfied by replacing P with P'. Therefore we conclude that the result of our rounding scheme will be feasible in the integer program.

Proof of 2-approximation:

In both the relaxed linear program and rounding scheme function f will return the same results, such that the result of the relaxed linear program, since P is minimized, P will equal some P_m, for passes at (m, m+1[n]). Indeed suppose $\mathrm{P} = \mathrm{P}_m = \sum_{\forall q} \mathrm{x}_q^{f(q,\ m)*}$. It follows, from our rounding scheme, that the rounding solution must be $\mathrm{P'} = \mathrm{P'}_r = \sum_{\forall q} \mathrm{y}_q^{f(q,\ r)}$ representing the number of passes at some (r, r+1[n]). Then the proof is such that:

$\mathrm{x}_q^{1*} \geqslant \frac{1}{2} \rightarrow \mathrm{y}_q^1 = 1$

$\mathrm{x}_q^{0*} \geqslant \frac{1}{2} \rightarrow \mathrm{y}_q^0 = 1$

then,

$2\mathrm{x}_q^{1*} \geqslant 1 \rightarrow 2\mathrm{x}_q^{1*} \geqslant \mathrm{y}_q^1$

$2\mathrm{x}_q^{0*} \geqslant 1 \rightarrow 2\mathrm{x}_q^{0*} \geqslant \mathrm{y}_q^0$

finally,

$\mathrm{P'}_r = \sum_{\forall q} \mathrm{y}_q^{f(q,\ r)} \leqslant 2^*\mathrm{P}_r = 2^*\sum_{\forall q} \mathrm{x}_q^{f(q,\ r)}$

$\mathrm{P'}_r = \sum_{\forall q} \mathrm{y}_q^{f(q,\ r)} \leqslant 2^*\mathrm{P}_m = 2^*\sum_{\forall q} \mathrm{x}_q^{f(q,\ m)}$ # Since $\mathrm{P}_r \leqslant \mathrm{P}_m$

$\mathrm{P'}_r = \sum_{\forall q} \mathrm{y}_q^{f(q,\ r)} \leqslant 2^*\mathrm{P} \leqslant 2^*\mathrm{LPOPT} \leqslant 2^*\mathrm{IPOPT}$ # Since we established that $\mathrm{P} = \mathrm{P}_m$

Therefore, the rounding scheme attains a 2-approximation. ■

Q4. Vertex Cover

1. $\{0, 1\}$-integer programming formulation:

Minimize:

$\sum_{u\in V} C_v(u)x_u + \sum_{(u,v)\in E} C_e(u,v)x_{(u,v)}$

Subject To:

a)

$x_u \in \{0, 1\} \quad \forall\, u \in V$

$x_{(u,v)} \in \{0, 1\} \quad \forall\, (u, v) \in E$

b)

$x_u + x_v + x_{(u,v)} \geqslant 1$

Explanation of Constraints:

a)

$x_u \in \{0, 1\}$ represents whether or not vertex u is in our vertex cover S.
If $x_u = 0$ then it is not in our vertex cover, and if $x_u = 1$, then it is in our vertex cover.

The second constraint $x_{(u,v)} \in \{0, 1\}$ represents whether or not the edge (u, v) is covered by the vertices in our vertex cover S. If $x_{(u,v)} = 0$ then either vertex u or v are in the cover S, if $x_{(u,v)} = 1$ then neither u nor v are in S.

b)

$x_u + x_v + x_{(u,v)} \geqslant 1$ shows that if either u, or v (or both) are in the cover, then we have at least one, x_u or x_v, equal to one. If u and v are not in the cover then x_u and x_v equals zero. So, in this case, $x_{(u,v)}$ must equal 1, and this indicates that the vertices in the edge (u, v) are not in the cover.

2. LP relaxation:

Minimize:

$\sum_{u\in V} C_v(u)x_u + \sum_{(u,v)\in E} C_e(u,v)x_{(u,v)}$

Subject To:

a)

$0 \leqslant x_u \leqslant 1 \quad \forall\, u \in V$

$0 \leqslant x_{(u,v)} \leqslant 1 \quad \forall\, (u, v) \in E$

b)

$x_u + x_v + x_{(u,v)} \geqslant 1$

Rounding Scheme:

$\forall$ u $\in$ V, and $\forall$ (u, v) $\in$ E let all x_u* and $x_{(u,v)}*$ be the solution to the relaxed LP. Allow us to round all x_u*'s and $x_{(u,v)}*$'s to the interger programming solution, such that all y_u's and $y_{(u,v)}$'s will be a feasible IP solution:

$$y_u = \begin{cases} 1 & \text{if corresponding } x_u{}^* \geqslant \frac{1}{3} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{(u,v)} = \begin{cases} 1 & \text{if corresponding } x_{(u,v)}{}^* \geqslant \frac{1}{3} \\ 0 & \text{otherwise} \end{cases}$$

Proof that Rounding Scheme works:

HENRY TO DO

Proof of 3-approximation:

HENRY TO DO (add intro text if needed)

${x_u}^* \geqslant \frac{1}{3} \rightarrow y_u = 1$

${x_{(u,v)}}^* \geqslant \frac{1}{3} \rightarrow y_{(u,v)} = 1$

then,

$3{x_u}^* \geqslant 1 \rightarrow 3{x_u}^* \geqslant y_u$

$3{x_{(u,v)}}^* \geqslant 1 \rightarrow 3{x_{(u,v)}}^* \geqslant y_{(u,v)}$

finally,

$\sum_{u \in V} C_v(u)y_u + \sum_{(u,v) \in E} C_e(u,v)y_{(u,v)} \leqslant \sum_{u \in V} C_v(u)(3x_u) + \sum_{(u,v) \in E} C_e(u,v)(3x_{(u,v)})$
$\sum_{u \in V} C_v(u)y_u + \sum_{(u,v) \in E} C_e(u,v)y_{(u,v)} \leqslant 3(\sum_{u \in V} C_v(u)x_u) + 3(\sum_{(u,v) \in E} C_e(u,v)x_{(u,v)})$
$\sum_{u \in V} C_v(u)y_u + \sum_{(u,v) \in E} C_e(u,v)y_{(u,v)} \leqslant 3(\sum_{u \in V} C_v(u)x_u + \sum_{(u,v) \in E} C_e(u,v)x_{(u,v)})$
$\sum_{u \in V} C_v(u)y_u + \sum_{(u,v) \in E} C_e(u,v)y_{(u,v)} \leqslant 3\text{*LPOPT} \leqslant 3\text{*IPOPT}$

Therefore, the rounding scheme attains a 3-approximation. ■