CSOR4231 Algorithm
HW6

Jialin Zhao — jz2862

April 30, 2018

1 Problem I

A large store has m customers and n products and maintains an m * n matrix A such that A_{ij} = 1 if customer i has purchased product j; otherwise, A_{ij} = 0.
Two customers are called orthogonal if they did not purchase any products in common. Your task is to help the store determine a maximum subset of orthogonal customers.
Give an efficient algorithm for this problem or state its decision version and prove it is NP complete.

Decision Version SOC(D):

Given an m*n matrix A represent the purchasing relationship between m customers and n products using 1 else 0, whether there is a subset of orthogonal customers of size at least k.

Proof of NPC:

Proof of NP:
Given a instance subset of size k, the certifier just needs to traverse all customers in this instance and check if there is a pair of two customers connected to a same product. This will take $O(kn)$ time which is within polynomial time.

$$SOC(D) \in NP$$

Proof of reduction from an NPC:
For an arbitrary problem of independent set decision version IS(D):

- Input: a graph G(V,E), k
- Output: if there is a independent set of size at least k

Transformation:
Add all nodes to a set as different customers of size m;
Add all edges to a set as different products of size n;
Create a matrix in size m*n initialized with 0 for each element, for each edge i(x,y) set the element in matrix A[x][i] and matrix A[y][i] as 1 to represent customer x,y bought a same product i.
So we can see this reduction takes polymonial time O(mn).

- Input: the matrix A,k
- Output: if there is a subset of orthogonal customers of size at least k

A short proof for equivalence:

- Forward: If there is a independent subset of size k which means that there is no edge between any two elements x,y, then for any column i in matrix A there is no A[x][i]=A[y][i]=1. Construct a SOC instance with the same set of size k then it satisfies that for every customer there is no same products.
- Backward: If there is a SOC of size k which means for any pair of customers in this subset any column i in matrix A there is no A[x][i]=A[y][i]=1. Construct a IS instance with the same set of size k then it satisfies that for every pair of nodes in it there is no edges between them.

$$\text{which means:} IS(D) \leq_p SOC(D)$$

so that the SOC(D) is NP complete.

2 Problem II

Suppose you had a polynomial-time algorithm that, on input a graph, answers yes if and only if the graph has a Hamiltonian cycle.
Show how, on input a graph G = (V, E), you can return in polynomial time

- **a Hamiltonian cycle in G, if one exists,**
- **no, if G does not have a Hamiltonian cycle.**

Description:

Suppose G is Hamiltonian. Let C be any Hamiltonian cycle in G. Then, if we remove any edge not in C, the resulting graph will still be Hamiltonian. Denote the given algorithm as H(G) then my algorithm will run H(G') when checking on every edges and choose delete or keep the edge and finally return the reduced graph as the Hamiltonian cycle we want.

Pseudocode:

Algorithm 1 Function HCycle($G(V, E)$)

```
if H(G)=FALSE then
  return NO
end if
G' = G
for every edge e in E do
  if H(G'-e) then
    delete e from G'
  end if
end for
return G'
```

Proof of Correctness:

When there is no Hamiltonian cycle, the algorithm will return false correctly.
When there is a Hamiltonian cycle, we know it consists of exactly $|V|$ edges. We only need to prove that:

- In every step we deleting/keeping an edge, there still is a Hamiltonian cycle in the remaining graph;
- After we checked all edges, the remaining graph contains $|V|$ edges which forms a Hamiltonian cycle.

The constraint 1 is satisfied during every loop of our algorithm.
The constraint 2 is satisfied after all loops because every edges remained cannot be deleted which means that all of them is part of the Hamiltonian cycle.
So, by simple induction we can conclude that the correctness of this algorithm that it can return a Hamilton cycle.

Time Complexity:

Suppose that the polymonial time cost of using the algoithim H(G) to check if there is a Hamilton cycle in G is $O(P(|E|))$;
Then the time cost by checking every edge in our algorithm cost time P(G');
There is totally $|E|$ loops, so the total runnning time is $O(|E|P(|E|))$

3 Problem III

There is a set of ground elements $E = e_1, e_2, ..., e_n$ and a collection of m subsets $S_1, S_2, ..., S_m$of the ground elements (that is, $Si \subseteq E \; for \; 1 \leq i \leq m$). The goal is to select a minimum cardinality set A of ground elements such that A contains at least one element from each subset S_i. State the decision version of this problem and prove that it is NP-complete.

Decision Version CS(D):

Given $E = e_1, e_2, ..., e_n$ and a collection of m subsets $S_1, S_2, ..., S_m$, the output is whether there is a cardinality set of size at most k.

Proof of NPC:

Proof of NP:

Given a instance set of size k, the certifier just needs to traverse all subsets and check if there is an element in common. For every subset this will take at most $O(n^2)$ time so that this will take $O(m * n^2)$ time which is within polynomial time.

$$CS(D) \in NP$$

Proof of reduction from an NPC:

For an arbitrary problem of vertex cover decision version VC(D):

- Input: a graph G(V,E), k
- Output: if there is a vertex cover of size at most k

Transformation:
Create a set A consisting of all nodes in V;
Create collection S of subsets {u,v} for every edges in E;
So we can see this reduction takes polymonial time $O(|V|| + |E|)$.

- Input: the set A and subsets S, k
- Output: if there is a cardinality set of size at most k

A short proof for equivalence: an arbitrary vertex cover decision problem VC(D) can be reduced to a 2-size CS(D) problem.

- Forward: If there is a vertex cover C of size k in G which means that for every edge e in G the nodes in C contain at least one side of node in e. Then in the related CS(D) problem C of size k contains at least 1 element in every 2-size subsets in G.
- Backward: If there is a cardinality set C of size k for set A and 2-size subsets S which means that C contains at least 1 element in every 2-size subsets. Then in the related VC(D) problem the vertex cover C of size k contains at least one side for every edge in G.

$$\text{which means:} VC(D) \leq_p CS(D)$$

so that the CS(D) is NP complete.

4 Problem IV

A paper mill manufactures rolls of paper of standard width 3 meters. Customers want to buy paper rolls of shorter width, and the mill has to cut such rolls from the 3m rolls.
For example, one 3 m roll can be cut into 2 rolls of width 93cm and one roll of width 108cm; the remaining 6cm goes to waste. The mill receives an order of

- **97 rolls of width 135cm**
- **610 rolls of width 108cm**
- **395 rolls of width 93cm**
- **211 rolls of width 42cm**

Form a linear program to compute the smallest number of 3m rolls that have to be cut to satisfy this order, and explain how they should be cut.

Formulation of Linear Programming:

Varibles:Cutting Patterns and numbers of each
v_1 2 * 135.
v_2 1 * 135, 1 * 108, 1 * 42.
v_3 1 * 135, 1 * 93, 1 * 42.
v_4 1 * 135, 3 * 42.
v_5 2 * 108, 2 * 42.
v_6 1 * 108, 2 * 93.
v_7 1 * 108, 1 * 93, 2 * 42.
v_8 1 * 108, 4 * 42.
v_9 3 * 93.
v_{10} 2 * 93, 2 * 42.
v_{11} 1 * 93, 4 * 42.
v_{12} 7 * 42.
Objective:

$$min(\sum_{i=1}^{12} v_i)$$

Constraints:

$$2v_1 + v_2 + v_3 + v_4 \geq 97$$

$$v_2 + 2v_5 + v_6 + v_7 + v_8 \geq 610$$

$$v_3 + 2v_6 + v_7 + 3v_9 + 2v_{10} + v_{11} \geq 395$$

$$v_2 + v_3 + 3v_4 + 2v_5 + 2v_7 + 4v_8 + 2v_{10} + 4v_{11} + 7v_{12} \geq 211$$

$$\forall v_i \geq 0$$

5 Problem V

Formulate linear or integer programs for the following optimization problems.

5.1 Min-cost flow: Given a flow network with capacities c_e and costs a_e on every edge e, and supplies s_i on every vertex i, find a feasible flow $f : E \rightarrow R_+$ that is, a flow satisfying edge capacity constraints and node supplies that minimizes the total cost of the flow.

Objective:

$$min(\sum_{e \in E}(f_e * c_e))$$

Constraints:

$$for \ \forall \ i, \ \sum_{e \ in \ i}(f_e) - \sum_{e \ out \ of \ i}(f_e) = s_i$$

$$for \ \forall \ e, \ 0 \leq f_e \leq c_e$$

5.2 The assignment problem: There are n persons and n objects that have to be matched on a one-to-one basis. There's a given set A of ordered pairs (i, j), where a pair (i, j) indicates that person i can be matched with object j. For every pair $(i, j) \in A$, theres a value a_{ij} for matching person i with object j. Our goal is to assign persons to objects so as to maximize the total value of the assignment.

Objective:

Associate every assignment with x_{ij} which means $x_{ij} = 1$ represents that the person i is matched with object j and otherwise $x_{ij} = 0$ represents the person i is not matched with object j.

$$max(\sum_{(i,j) \in A}(x_{ij} * a_{ij}))$$

Constraints:

$$for \ \forall \ i \in \{1, 2...n\}, \ \sum_{j|(i,j) \in A} x_{ij} = 1$$

$$for \ \forall \ j \in \{1, 2...n\}, \ \sum_{i|(i,j) \in A} x_{ij} = 1$$

$$for \ \forall \ (i, j) \in A, 0 \leq x_{ij} \leq 1$$

5.3 Uncapacitated facility location: There is a set F of m facilities and a set D of n clients. For each facility $i \in F$ and each client $j \in D$, there is a cost c_{ij} of assigning client j to facility i. Further, there is a one-time cost f_i associated with opening and operating facility i. Find a subset F' of facilities to open that minimizes the total cost of (i) operating the facilities in F' and (ii) assigning every client j to one of the facilities in F'.

Objective:

Associate every facility $i \in F$ with y_i which is either 1 representing the facility needs to be activated or 0 representing the facility is not in use.

Associate every pair $(i, j) \in D$ with $x_{i,j}$ which is either 1 representing the client j is assigned to facility i or 0 representing the client j is not assigned to facility i.

$$min(\sum_{i \in F}(y_i * f_i) + \sum_{i \in F, j \in D}(x_{ij} * a_{ij}))$$

Constraints:

$$\forall j \in D, \sum_{i \in F} x_{ij} = 1$$

$$\forall i \in F, \forall j \in D, y_i \geq x_{ij}$$

$$\forall i \in F,\ 0 \leq y_i \leq 1$$

$$\forall i \in F, \forall j \in D, 0 \leq x_{ij} \leq 1$$