Homework 12

CSCI570 Spring 2025

Question 1

1(a) Define your variables. (Describe what they represent in English)

Answer:
x1: Change in the number of type 1 drinks produced (can be positive or negative)
x2: Change in the number of type 2 drinks produced
x3: Change in the number of type 3 drinks produced

Thus, new production quantities are:
Type 1: 200+x1
Type 2: 120+x2
Type 3: 180+x3

1(b) What is the objective function?

Answer Maximize Z = 2*x1 + 2.8*x2 + 1.5*x3

1(c) What are the constraints in your LP?

Answer:

Resource Constraints:

Coffee constraint (due to shortage):
$10x1+16x2+8x3 \leq -600$

Sugar constraint (due to shortage):
$12x1+6x2+15x3 \leq -400$

Milk constraint (due to surplus):
$8x1+3x2+10x3 \leq 256$

Non-negativity of Production:

Cannot produce negative total drinks:
$x1 \geq -200$, $x2 \geq -120$, $x3 \geq -180$

Question 2

2(a) Define your variables. (Describe what they represent in English)

Answer:

Define a binary variable x_u for each vertex $u \in V$:

- $x_u = 1$ if vertex u is included in the clique
- $x_u = 0$ otherwise.

2(b) What is the objective function?

Answer:

Maximize the number of vertices included in the clique:

Maximize $\sum$ below summation $(u \in V)$ (x_u)

2(c) What are the constraints in your LP?

Answer:

To ensure that selected vertices form a valid clique:

For every pair of **non-adjacent** vertices $(u,v) \notin E$, at most one can be included in the clique:

$x_u + x_v \leq 1 \; \forall \; (u,v) \notin E$

Binary constraints on variables:

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

Question 3

3(a) Define your variables. (Describe what they represent in English)
Answer:

Xij - Be the quantity of item *i* ordered from store *j*. These quantities can be real-valued (divisible goods).

3(b) What is the objective function?
Answer:

Minimize the total cost of purchasing all items from all stores:

Minimize $\sum$ i=1 to m $\sum$ j=1 to n (Xij * pij)

Where pij is the price per unit of item i at store j.

3(c) What are the constraints in your LP?
Answer:

Demand Constraints (per item):
For each item i, ensure the total quantity purchased across all stores meets the minimum and does not exceed the maximum required:

$ai \leq \sum j=1 \text{ to } n (Xij) \leq bi \; \forall \; i \in \{1,\ldots,m\}$

Store Order Value Constraints:
Each store j must receive an order worth at least Sj (due to delivery restrictions), and at most W (credit card per-transaction limit):

$Sj \leq \sum i=1 \text{ to } m (Xij * pij) \leq W \; \forall \; j \in \{1,\ldots,n\}$

Non-negativity Constraints:
Quantities ordered must be non-negative:

$Xij \geq 0 \; \forall \; i \in \{1,\ldots,m\}, \; \forall \; j \in \{1,\ldots,n\}$

Question 4

Answer 4

We use a greedy algorithm that builds sets A and B incrementally, maintaining their sizes equal throughout the process.

Algorithm Steps:

1. Initialize: A←∅A, B←∅

2. for i=1 to n:

Select the next two unused vertices, say u = v_2i−1 and v=v_2i

Evaluate two possible assignments:
Option 1: Add u to A, v to B
Option 2: Add v to A, u to B

For each option, compute the increase in the cut size (i.e., the number of edges crossing from A to B)

Choose the assignment that **maximizes** the number of crossing edges at this step

Approximation:

Let's analyse why this greedy strategy guarantees at least **half** the optimal solution:

At each iteration i, two vertices u and v are assigned to opposite sets.

Let:
N Au, N Bu : number of neighbours of u currently in A and B, respectively
N Av, N Bv : same for vertex v

Then:
Placing u ∈ A, v ∈ B contributes N Bu + N Av edges to the cut
Placing u ∈ B, v ∈ A contributes N Au + N Bv edges

The **total** of both options is the total number of edges these two vertices contribute to the current partial cut.

The greedy algorithm chooses the better of the two options, so it adds at least **half** of the total possible contribution from each pair.

Since this reasoning holds at **every step**, and all 2n vertices are assigned through n such iterations, the total number of crossing edges added by the algorithm is at least:

$\frac{1}{2} * |E^*| \leq \frac{1}{2} * OPT$

Where OPT is the size of the maximum equal cut, and $|E^*| \leq |E|$ is the total number of edges added.

Thus, this is a **½ approximation algorithm**.