CS 3510: Design & Analysis of Algorithms 11/28/2023

Homework 9: Linear Programming & Randomness

Prof. Abrahim Ladha Due: 12/05/2023 11:59pm

Problem 1 Solutions

1. Objective Function: $\max\{f(s,a) + f(s,b)\}$.
   Constraints:
$$f(s,a) \leq 4 \tag{1}$$
$$f(s,b) \leq 5 \tag{2}$$
$$f(a,t) \leq 7 \tag{3}$$
$$f(b,t) \leq 3 \tag{4}$$
$$f(s,a) - f(a,t) \leq 0 \tag{5}$$
$$f(s,b) - f(b,t) \leq 0 \tag{6}$$
$$f(a,t) - f(s,a) \leq 0 \tag{7}$$
$$f(b,t) - f(s,b) \leq 0 \tag{8}$$

2. c: $\begin{bmatrix}1\\1\\0\\0\end{bmatrix}$, b: $\begin{bmatrix}4\\5\\7\\3\\0\\0\\0\\0\end{bmatrix}$, A: $\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\\1 & 0 & -1 & 0\\-1 & 0 & 1 & 0\\0 & 1 & 0 & -1\\0 & -1 & 0 & 1\end{bmatrix}$. Note that this is with $x = \begin{bmatrix}f(s,a) & f(s,b) & f(a,t) & f(b,t)\end{bmatrix}$, there are other solutions.

3. Work should not just be the answer. The answer should be that the objective function maximizes at 7, which is the maximum flow of the graph. Specific answers:
   $f(s,a) = 4, f(s,b) = 3, f(a,b) = 4, f(b,t) = 3$.

4. The dual is computed as follows: the objective function is $\min b^T y$, and the constraints are $A^T y \geq c$ and $y \geq 0$. There are many possibilities - here is one possibility:

   Objective Function:
$$4y_1 + 5y_2 + 7y_3 + 3y_4$$

   Constraints:
$$y_1 + y_5 - y_6 \geq 1 \tag{9}$$
$$y_2 + y_7 - y_8 \geq 1 \tag{10}$$
$$y_3 + y_5 + y_6 \geq 0 \tag{11}$$
$$y_4 - y_7 + y_8 \geq 0 \tag{12}$$

Solving this, the objective function should still output 7, and this corresponds to the min cut of the graph.

Problem 2 Solutions

```
import random

# Determines if a point lies within a circle
def is_inside_circle(x, y, center_x, center_y, radius):
    return (x - center_x) ** 2 + (y - center_y) ** 2 <= radius ** 2

# Counts number of circles point lies within
def count_circles(x, y, circles):
    count = 0
    for (center_x, center_y), radius in circles:
        if is_inside_circle(x, y, center_x, center_y, radius):
            count += 1
    return count

def approximate_intersection_area(num_samples):
    circles = [
        ((100, 100), 31),
        ((-1, 55), 95),
        ((125, 50), 60)
    ]

    min_x = min(center_x - radius for (center_x, center_y), radius in circles)
    max_x = max(center_x + radius for (center_x, center_y), radius in circles)
    min_y = min(center_y - radius for (center_x, center_y), radius in circles)
    max_y = max(center_y + radius for (center_x, center_y), radius in circles)

    inside_region = 0
    for _ in range(num_samples):
        x = random.uniform(min_x, max_x)
        y = random.uniform(min_y, max_y)
        count = count_circles(x, y, circles)
        if count > 1:
            inside_region += 1
    ratio_inside = inside_region / num_samples
    box_area = (max_x - min_x) * (max_y - min_y)
    intersection_area = ratio_inside * box_area
    return intersection_area

# Increase for more accuracy OR run several trials to get an average
# Current num_samples takes ~30-60 seconds to run and outputs 3360-3363
num_samples = 50000000
approx_area = approximate_intersection_area(num_samples)
print(approx_area)
```