CAS 781: Data Center Design Assignment 1

Question 1

1. **Finding n (with G/G/n)**

$$\bar{t}_{res} = \frac{1}{\mu_i}\frac{u_i}{1-u_i}\frac{C_{Ai}^2 + C_{Bi}^2}{2},$$

$$u_n' = \frac{u^n + u}{2},$$

$$u = \frac{\lambda}{n\mu}$$

$$n = \frac{\left(\frac{\lambda}{n\mu}\right)^n + \frac{\lambda}{n\mu}}{\mu(1-u)}$$

- For λ=2 → n=2.9774… → n=3
- For λ=8 → n=9.1618… → n=10
- For λ=14 → n=15.2055… → n=16

2. **Finding A matrix**
   - $A = \{a_{ij}\}_{NxN}$, $a_{ij} = d_{ij}\,(W_j + \alpha_j * u)$
   - Power consumption = 100+150*u
   - Utilization = $u = \frac{\lambda}{n\mu} = \frac{2}{3*1}$
   - $a_{ij} = d_{ij}\,(150+100*u) = d_{ij}\,(150 + 100*2/3) = d_{ij} * 216.6666$

3. **Non-linear to linear & MIP**
   - min max A*x
   - subject to $\sum_{si \in |S'|} xi = |S'|$, $x_i \in \{0,1\}$
   - Converting it to linear problem we need to add new constraints:
     - $(a_{1,1} * x_1) + (a_{1,2} * x_2) + \ldots + (a_{1,16} * x_{16}) = z_1$
     - $(a_{2,1} * x_1) + (a_{2,2} * x_2) + \ldots + (a_{2,16} * x_{16}) = z_2$
     - …
     - $(a_{16,1} * x_1) + (a_{16,2} * x_2) + \ldots + (a_{16,16} * x_{16}) = z_{16}$
   - And final problem is minimizing z

4. **TASP-MIP Results**
   - For λ=2 → n=6 → 1st, 3rd,5th, 8th,12th and 15th servers
   - [0, 2, 4, 7, 11, 14]
   - For λ=8 → n=12 → 1st, 2nd, 3rd, 5th,6th, 8th,9th, 12th,13th, 15th and 16th servers
   - [0, 1, 2, 4, 5, 7, 8, 11, 12, 14, 15]
   - For λ=10 → n=16 → All servers

- [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

5. **TASP-LRH Results**
   - For $\lambda=2$ → n=3 → 1st, 2nd, 16th servers
   - [3.575, 4.225, 4.65833333, 5.09166667, 5.09166667, 5.09166667, 5.30833333, 5.30833333, 5.30833333, 5.30833333, 5.09166667, 5.09166667, 5.09166667, 4.875, 4.875, 4.55]
   - For $\lambda=8$ → n=10 → 1st, 2nd, 3rd, 14th, 15th,16th servers
   - We have to choose 4 more servers and 4th, 5th, 6th, 11th, 12th and 13th servers have the same LRH metrics. We can choose 4 of them.
   - [3.795, 4.485, 4.945, 5.405, 5.405, 5.405, 5.635, 5.635, 5.635, 5.635, 5.405, 5.405, 5.405, 5.175, 5.175, 4.83]
   - For $\lambda=14$ → n=16 → All servers
   - [3.91875, 4.63125, 5.10625, 5.58125, 5.58125, 5.58125, 5.81875, 5.81875, 5.81875, 5.81875, 5.58125, 5.58125, 5.58125, 5.34375, 5.34375, 4.9875]

Question 2

Both papers' main purpose is reducing the energy consumption of data centers via reducing the cooling powers.

Zapater et al paper emphasizes close-coupled data centers and free cooling methods. Their aim is to reduce power consumption and increase cooling efficiency with free cooling and outside temperature data. Their algorithm selects various cooling methods (22 or 26-degree C) with similar jobs CPU and memory distances. They also use water/air combination for cooling and carrying the heat instead of just air. Best results are power-balanced budget algorithm. The key points of the paper are outside temperature effects the cooling power consumption and fixed temperatures throughout the year doesn't reduce power consumption. Total 5% energy saving and 24% peak energy saving with their algorithm

TACOMA paper's algorithm designed for systems that energy saving over utilization priority. They developed 2 different algorithms: TASP (Tier 1) and TAWD (Tier 2). TASP is used for selecting servers for activation and TAWD used for workload distribution for servers. Both of them aims to minimize computing and cooling (total) power. However, they don't minimize the number of active serves. One of the disadvantages of these algorithms is they don't have dynamic refinement. In order to calculate results of algorithm, they predict the traffic rate, performance numbers, heat circulation and power consumption. Also switching active serves can cause performance decrease. But its advantages are low process power requirement for algorithms and realistic results on real data centers. 40% energy saving can be achieved with TASP and TAWD usage.

Baran Kaya, 400284996, kayab@mcmaster.ca

Group with: Songyi Wang, 400274292, wangs345@mcmaster.ca

```python
import numpy as np
from itertools import combinations

# Create matrix D (16x16)
D = [[0.015, 0.003, 0.002, 0.001, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0],
      [0.0015, 0.015, 0.003, 0.002, 0.001, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      [0, 0.0015, 0.015, 0.003, 0.002, 0.001,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      [0, 0, 0.0015, 0.016, 0.003, 0.002,
0.001, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      [0, 0, 0, 0.0015, 0.016, 0.003, 0.002,
0.001, 0, 0, 0, 0, 0, 0, 0, 0],
      [0, 0, 0, 0, 0.0015, 0.016, 0.003,
0.002, 0.001, 0, 0, 0, 0, 0, 0, 0],
      [0, 0, 0, 0, 0, 0.0015, 0.017, 0.003,
0.002, 0.001, 0, 0, 0, 0, 0, 0],
      [0, 0, 0, 0, 0, 0, 0.0015, 0.017, 0.003,
0.002, 0.001, 0, 0, 0, 0, 0],
      [0, 0, 0, 0, 0, 0, 0, 0.0015, 0.017,
0.003, 0.002, 0.001, 0, 0, 0, 0],
      [0, 0, 0, 0, 0, 0, 0, 0, 0.0015, 0.017,
0.003, 0.002, 0.001, 0, 0, 0],
      [0, 0, 0, 0, 0, 0, 0, 0, 0, 0.0015,
0.016, 0.003, 0.002, 0.001, 0, 0],
      [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.0015,
0.016, 0.003, 0.002, 0.001, 0],
      [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0.0015, 0.016, 0.003, 0.002, 0.001],
      [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0.0015, 0.015, 0.003, 0.002],
      [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0.0015, 0.015, 0.003],
      [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0.0015, 0.015]]
# Convert D to numpy matrix
```

```
D_mat = np.matrix(D)
# Arrival rate = 2, 8, 14
lambdas = [2, 8, 14]
# Active servers
n_values = [3, 10, 16] #for lambda=2, 8, 14
# Service rate
mu = 1

#Main function
def main():
    print("Please select lambda index: 0 for
2, 1 for 8, 2 for 14")
    index = int(input())

    for num in range(n_values[index], 17):
        # Create A Matrix
        A_mat = create_a_matrix(D_mat,
lambdas[index], n_values[index], mu)
        # Calculate MIP
        val_pair = mip_calculation(A_mat,
num)

    # Call TASP-LRH function
    print("TASP_LRH vector: ",
tasp_lrh(A_mat))

# Create A matrix with lambda, n and mu
def create_a_matrix(d_mat, lambda_value,
n_value, mu_value):
    util = lambda_value / (n_value *
mu_value) #Utilization

    power = 150 + 100*util #Power usage

    #Create matrix A (16x16)
    #A --> a[i, j] = d[i, j] * (W +
alfa*util)
```

```
    A_mat = d_mat * power
    return A_mat

# Find maximum
def
get_constraint_z_from_given_situation(a_mat,
server_set):
    max_res = 0

    for i in range(16):
        res = 0
        for server in server_set:
            res += a_mat.item(i, server)
        if res >= max_res:
            max_res = res

    return max_res

# All combinations
def get_server_combinations(server_list, n):
    return list(combinations(server_list, n))

# MIP
def mip_calculation(a_mat, n):
    server_number_list = np.arange(16)
    all_server_set =
get_server_combinations(server_number_list,
n)

    max_value = 0
    max_server_set = all_server_set[0]

    min_value = 1000000
    min_server_set = all_server_set[0]

    for server_set in all_server_set:
        temp_value =
```

```
get_constraint_z_from_given_situation(a_mat,
server_set)
        if temp_value > max_value:
            max_value = temp_value
            max_server_set = server_set
        if temp_value < min_value:
            min_value = temp_value
            min_server_set = server_set
    print("n: {0}, max value: {1}, max server
set: {2}, min value: {3}, min server set:
{4}"
          .format(n, max_value,
max_server_set, min_value, min_server_set))

    return max_value, min_value

# TASP-LRH calculate
def tasp_lrh(a_mat):
    result = np.zeros(16)
    # Add all rows
    for i in range(16):
        result = np.add(result, a_mat[i,:])

    return result

# Call main function
if __name__ == '__main__':
    main()
```