**SAPTARSHI MANDAL**

**ROLL: 2024PGCSCS09**

**EC ASSIGNMENT 1**

Q1: Solve GAP Problem using Optimal Method  
function analyze\_assignment\_data()

totalFiles = 12;

resultsCollection = cell(totalFiles, 1);

% For saving

instanceIDs = {};

optimalCosts = [];

for fileIdx = 1:totalFiles

dataFile = sprintf('gap%d.txt', fileIdx);

fileHandle = fopen(dataFile, 'r');

if fileHandle == -1

error('Failed to open file %s.', dataFile);

end

% Get number of cases

caseCount = fscanf(fileHandle, '%d', 1);

caseResults = cell(caseCount, 1);

for caseIdx = 1:caseCount

servers = fscanf(fileHandle, '%d', 1);

clients = fscanf(fileHandle, '%d', 1);

costMatrix = fscanf(fileHandle, '%d', [clients, servers])';

resourceMatrix = fscanf(fileHandle, '%d', [clients, servers])';

capacities = fscanf(fileHandle, '%d', [servers, 1]);

allocation = compute\_assignment(servers, clients, costMatrix, resourceMatrix, capacities);

totalCost = sum(sum(costMatrix .\* allocation));

% Save result

problemID = sprintf('c%d-%d', servers\*100 + clients, caseIdx);

caseResults{caseIdx} = sprintf('%s\t%d', problemID, round(totalCost));

% Store for saving

instanceIDs{end+1} = problemID; %#ok<AGROW>

optimalCosts(end+1) = round(totalCost); %#ok<AGROW>

end

fclose(fileHandle);

resultsCollection{fileIdx} = caseResults;

end

% Save results

optimal\_results.instanceIDs = instanceIDs;

optimal\_results.costs = optimalCosts;

save('results\_optimal.mat', 'optimal\_results');

% Save as .txt

T = table(instanceIDs', optimalCosts', 'VariableNames', {'InstanceID', 'OptimalCost'});

writetable(T, 'results\_optimal.txt');

% Display Results

display\_results\_table(resultsCollection, totalFiles);

end

function allocation = compute\_assignment(servers, clients, costMatrix, resourceMatrix, capacities)

objective = -costMatrix(:); % Maximization → convert to minimization

eqConstraints = kron(eye(clients), ones(1, servers));

eqValues = ones(clients, 1);

ineqConstraints = zeros(servers, servers \* clients);

for server = 1:servers

for client = 1:clients

ineqConstraints(server, (client-1)\*servers + server) = resourceMatrix(server, client);

end

end

ineqValues = capacities;

lowerBounds = zeros(servers \* clients, 1);

upperBounds = ones(servers \* clients, 1);

integerVars = 1:(servers \* clients);

options = optimoptions('intlinprog', 'Display', 'off');

solution = intlinprog(objective, integerVars, ineqConstraints, ineqValues, ...

eqConstraints, eqValues, lowerBounds, upperBounds, options);

allocation = reshape(solution, [servers, clients]);

end

function display\_results\_table(resultsCollection, totalFiles)

columnsPerLine = 4;

for startPos = 1:columnsPerLine:totalFiles

endPos = min(startPos + columnsPerLine - 1, totalFiles);

for col = startPos:endPos

fprintf('gap%d\t\t', col);

end

fprintf('\n');

maxCases = max(cellfun(@length, resultsCollection(startPos:endPos)));

for row = 1:maxCases

for col = startPos:endPos

if row <= length(resultsCollection{col})

fprintf('%s\t', resultsCollection{col}{row});

else

fprintf('\t\t');

end

end

fprintf('\n');

end

fprintf('\n');

end

end

% Run the code

analyze\_assignment\_data();

Output:  
