**SAPTARSHI MANDAL**

**ROLL: 2024PGCSCS09**

**EC ASSIGNMENT 2**  
Q1:

Code:  
function finalAllocations = executeGAPCases()

totalFiles = 12;

finalAllocations = cell(1, totalFiles);

formattedOutput = cell(totalFiles, 1);

% Collect results to save

allInstanceIDs = {};

allProfits = [];

for fileIndex = 1:totalFiles

fileName = sprintf('gap%d.txt', fileIndex);

fileID = fopen(fileName, 'r');

if fileID == -1

error('Cannot open file %s.', fileName);

end

numInstances = fscanf(fileID, '%d', 1);

instanceResults = cell(numInstances, 1);

for caseIndex = 1:numInstances

numMachines = fscanf(fileID, '%d', 1);

numTasks = fscanf(fileID, '%d', 1);

profitMatrix = fscanf(fileID, '%d', [numTasks, numMachines])';

demandMatrix = fscanf(fileID, '%d', [numTasks, numMachines])';

capacityVector = fscanf(fileID, '%d', [numMachines, 1]);

allocationMatrix = optimizeGAP(numMachines, numTasks, profitMatrix, demandMatrix, capacityVector);

totalProfit = sum(sum(profitMatrix .\* allocationMatrix));

finalAllocations{fileIndex} = allocationMatrix;

problemID = sprintf('c%d%d-%d', numMachines, numTasks, caseIndex);

instanceResults{caseIndex} = sprintf('%s %d', problemID, totalProfit);

% Store results for later saving

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

allProfits(end+1) = totalProfit; %#ok<AGROW>

end

formattedOutput{fileIndex} = instanceResults;

fclose(fileID);

end

displayFormattedResults(formattedOutput, totalFiles);

% Save results to .mat

approx\_results.instanceIDs = allInstanceIDs;

approx\_results.profits = allProfits;

save('results\_approx.mat', 'approx\_results');

% Optionally save as .txt

T = table(allInstanceIDs', allProfits', 'VariableNames', {'InstanceID', 'Profit'});

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

end

function allocationMatrix = optimizeGAP(numMachines, numTasks, profitMatrix, demandMatrix, capacityVector)

allocationMatrix = zeros(numMachines, numTasks);

efficiencyScore = profitMatrix ./ (demandMatrix + 1e-6);

[~, sortedIndices] = sort(efficiencyScore(:), 'ascend');

remainingCapacity = capacityVector;

for index = sortedIndices'

[machine, task] = ind2sub([numMachines, numTasks], index);

if remainingCapacity(machine) >= demandMatrix(machine, task)

allocationMatrix(machine, task) = 1;

remainingCapacity(machine) = remainingCapacity(machine) - demandMatrix(machine, task);

end

end

end

function displayFormattedResults(formattedOutput, totalFiles)

filesPerRow = 4;

maxInstances = max(cellfun(@length, formattedOutput));

for startIdx = 1:filesPerRow:totalFiles

endIdx = min(startIdx + filesPerRow - 1, totalFiles);

for fileIdx = startIdx:endIdx

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

end

fprintf('\n');

for caseIdx = 1:maxInstances

for fileIdx = startIdx:endIdx

if caseIdx <= length(formattedOutput{fileIdx})

fprintf('%s\t', formattedOutput{fileIdx}{caseIdx});

else

fprintf('\t');

end

end

fprintf('\n');

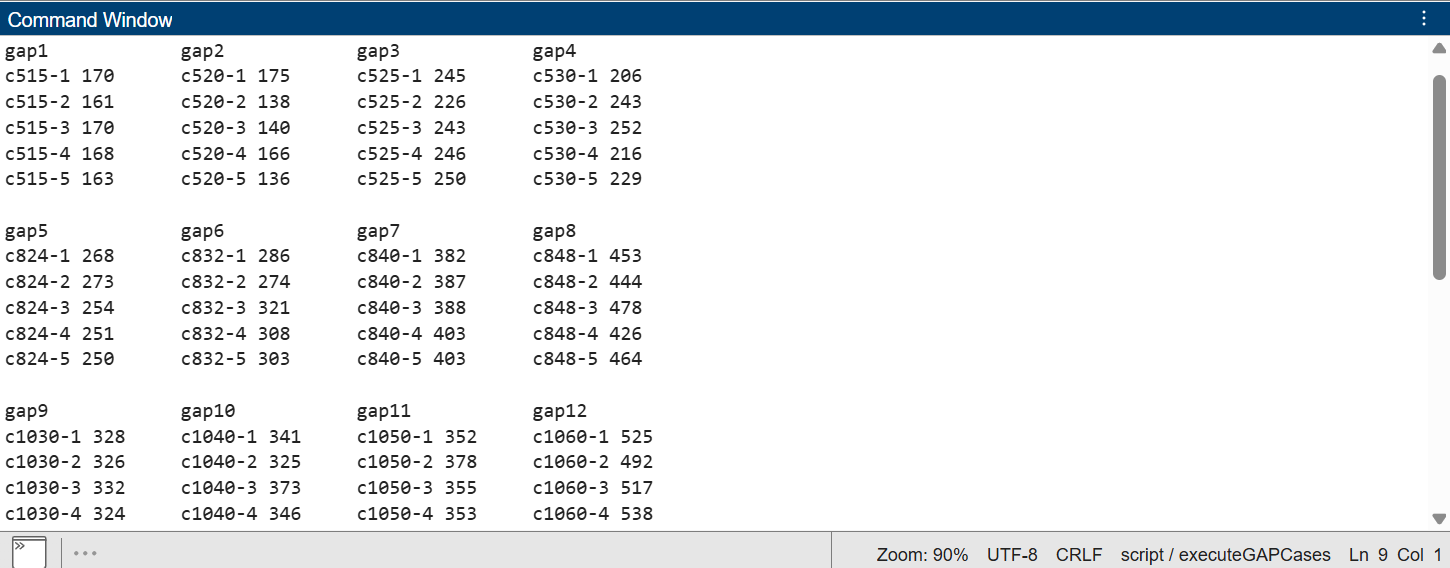
end

fprintf('\n');

end

end

executeGAPCases();

OUTPUT:  
  
  
Comparison Code:  
function convergence\_comparison()

% Load data from files

optimalData = readtable('results\_optimal.txt', 'Delimiter', ',', 'VariableNamingRule', 'preserve');

approxData = readtable('results\_approx.txt', 'Delimiter', ',', 'VariableNamingRule', 'preserve');

% Extract Instance IDs and values

optimalIDs = optimalData.InstanceID;

optimalValues = optimalData.OptimalCost;

approxIDs = approxData.InstanceID;

approxValues = approxData.Profit;

% Match IDs between optimal and approx

[commonIDs, idxOptimal, idxApprox] = intersect(optimalIDs, approxIDs, 'stable');

% Extract corresponding profits

matchedOptimal = optimalValues(idxOptimal);

matchedApprox = approxValues(idxApprox);

% Plot only the comparison of Optimal vs Approximate

figure;

plot(1:length(commonIDs), matchedOptimal, '-o', 'LineWidth', 2);

hold on;

plot(1:length(commonIDs), matchedApprox, '-x', 'LineWidth', 2);

xlabel('Instance Index');

ylabel('Profit');

title('Optimal vs Approximate Profit Comparison');

legend('Optimal', 'Approximate','Location','northwest');

grid on;

end

Output:  
