comparative graph showing both the optimal values and the results from the

approximation algorithm.   
  
  
code:-

function compare\_gap12\_greedy\_vs\_ilp()

datasetIndex = 12;

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

fileId = fopen(fileName, 'r');

if fileId == -1

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

end

% Read number of problem instances

numProblems = fscanf(fileId, '%d', 1);

% Initialize result arrays

greedyCosts = zeros(numProblems, 1);

ilpCosts = zeros(numProblems, 1);

% Process each problem

for problemIndex = 1:numProblems

numServers = fscanf(fileId, '%d', 1);

numUsers = fscanf(fileId, '%d', 1);

costMatrix = fscanf(fileId, '%d', [numUsers, numServers])';

resourceMatrix = fscanf(fileId, '%d', [numUsers, numServers])';

serverCapacities = fscanf(fileId, '%d', [numServers, 1]);

% --- Greedy Solution ---

greedyAssign = greedy\_gap\_assignment(numServers, numUsers, costMatrix, resourceMatrix, serverCapacities);

greedyCosts(problemIndex) = sum(sum(costMatrix .\* greedyAssign));

% --- ILP Optimal Solution ---

ilpAssign = solve\_gap\_max(numServers, numUsers, costMatrix, resourceMatrix, serverCapacities);

ilpCosts(problemIndex) = sum(sum(costMatrix .\* ilpAssign));

fprintf('Problem %2d | Greedy: %6d | Optimal (ILP): %6d\n', ...

problemIndex, round(greedyCosts(problemIndex)), round(ilpCosts(problemIndex)));

end

fclose(fileId);

% --- Plot the comparison graph ---

figure;

bar([greedyCosts, ilpCosts]);

title('GAP12: Greedy vs Optimal (ILP) Cost Comparison');

xlabel('Problem Instance');

ylabel('Total Cost');

legend({'Greedy Approximation', 'Optimal (ILP)'}, 'Location', 'northwest');

grid on;

set(gca, 'FontSize', 12);

end

% Greedy assignment function (already defined in your code)

function assignmentMatrix = greedy\_gap\_assignment(numServers, numUsers, costMatrix, resourceMatrix, serverCapacities)

assignmentMatrix = zeros(numServers, numUsers);

remainingCapacities = serverCapacities(:);

for userIndex = 1:numUsers

bestCost = Inf;

bestServer = -1;

for serverIndex = 1:numServers

if resourceMatrix(serverIndex, userIndex) <= remainingCapacities(serverIndex) && ...

costMatrix(serverIndex, userIndex) < bestCost

bestCost = costMatrix(serverIndex, userIndex);

bestServer = serverIndex;

end

end

if bestServer > 0

assignmentMatrix(bestServer, userIndex) = 1;

remainingCapacities(bestServer) = remainingCapacities(bestServer) - resourceMatrix(bestServer, userIndex);

end

end

end

% ILP assignment function (already defined in your code)

function assignmentMatrix = solve\_gap\_max(numServers, numUsers, costMatrix, resourceMatrix, serverCapacities)

f = -costMatrix(:);

Aeq = kron(eye(numUsers), ones(1, numServers));

beq = ones(numUsers, 1);

Aineq = zeros(numServers, numServers \* numUsers);

for i = 1:numServers

for j = 1:numUsers

Aineq(i, (j-1)\*numServers + i) = resourceMatrix(i, j);

end

end

bineq = serverCapacities;

lb = zeros(numServers \* numUsers, 1);

ub = ones(numServers \* numUsers, 1);

intcon = 1:(numServers \* numUsers);

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

x = intlinprog(f, intcon, Aineq, bineq, Aeq, beq, lb, ub, options);

assignmentMatrix = reshape(x, [numServers, numUsers]);

end

output:-

