# ASSIGNMENT : - 1


## NAME - NIRAJ KUMAR
## REG.NO - 2024PGCSCS07

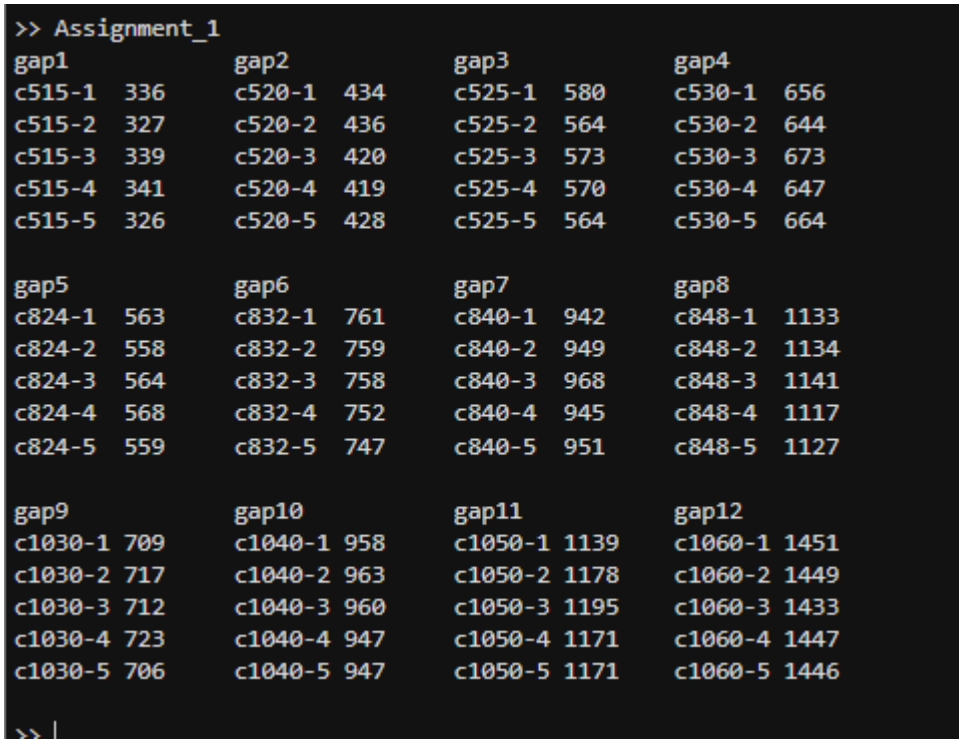

**QUESTION_1 - Find the optimal value for the GAP-based Resource Allocation problem mentioned above and verify the optimal fitness value for each instance in the GAP dataset (GAP1–GAP12). Additionally, plot a graph representing the optimal fitness value for each instance.**

```
function optimal_large_gap_modified()
   num_files = 12;
   all_results = cell(num_files, 1);
   all_objective_values = [];
   % Iterate through gap1 to gap12
   for g = 1:num_files
       filename = sprintf('gap%d.txt', g);
       fid = fopen(filename, 'r');
       if fid == -1
           error('Error opening file %s.', filename);
       end
       % Read the number of problem sets
       num_problems = fscanf(fid, '%d', 1);
       results = cell(num_problems, 1);
       for p = 1:num_problems
           % Read problem parameters
           m = fscanf(fid, '%d', 1);
           n = fscanf(fid, '%d', 1);
           % Read cost and resource matrices
           c = fscanf(fid, '%d', [n, m])';
           r = fscanf(fid, '%d', [n, m])';
           % Read server capacities
           b = fscanf(fid, '%d', [m, 1]);
           % Solve the problem
           x_matrix = solve_gap_max(m, n, c, r, b);
           objective_value = sum(sum(c .* x_matrix));
           results{p} = sprintf('c%d-%d\t%d', m*100 + n, p, round(objective_value));
           all_objective_values = [all_objective_values; objective_value];
       end
       fclose(fid);
       all_results{g} = results;
   end
```

```
    % Display results side by side
    files_per_row = 4;
    for start_idx = 1:files_per_row:num_files
        end_idx = min(start_idx + files_per_row - 1, num_files);
        for g = start_idx:end_idx
            fprintf('gap%d\t\t', g);
        end
        fprintf('\n');
        max_problems = max(cellfun(@length, all_results(start_idx:end_idx)));
        for p = 1:max_problems
            for g = start_idx:end_idx
                if p <= length(all_results{g})
                    fprintf('%s\t', all_results{g}{p});
                else
                    fprintf('\t\t');
                end
            end
            fprintf('\n');
        end
        fprintf('\n');
    end
    % Plot the optimal fitness values
    figure;
    plot(1:length(all_objective_values), all_objective_values, 'bo-', 'LineWidth', 1.5,
'MarkerSize', 6);
    title('Optimal Fitness Value per Problem Instance');
    xlabel('Instance Index');
    ylabel('Optimal Fitness Value');
    grid on;
end
% ✅ This is the sub-function that solves the GAP problem
function x_matrix = solve_gap_max(m, n, c, r, b)
    f = -c(:); % Convert to column vector for maximization
    % Constraint 1: Each user assigned exactly once
    Aeq_jobs = kron(eye(n), ones(1, m));
    beq_jobs = ones(n, 1);
    % Constraint 2: Server resource constraints
    Aineq_agents = zeros(m, m * n);
    for i = 1:m
        for j = 1:n
            Aineq_agents(i, (j-1)*m + i) = r(i,j);
        end
    end
    bineq_agents = b;
    % Define variable bounds (binary decision variables)
    lb = zeros(m * n, 1);
    ub = ones(m * n, 1);
    intcon = 1:(m*n);
    % Solve using intlinprog
```

```
   options = optimoptions('intlinprog', 'Display', 'off');
   x = intlinprog(f, intcon, Aineq_agents, bineq_agents, Aeq_jobs, beq_jobs, lb, ub,
options);
   % Reshape into m × n matrix
   x_matrix = reshape(x, [m, n]);
end
optimal_large_gap_modified();
```

## OUTPUT :-

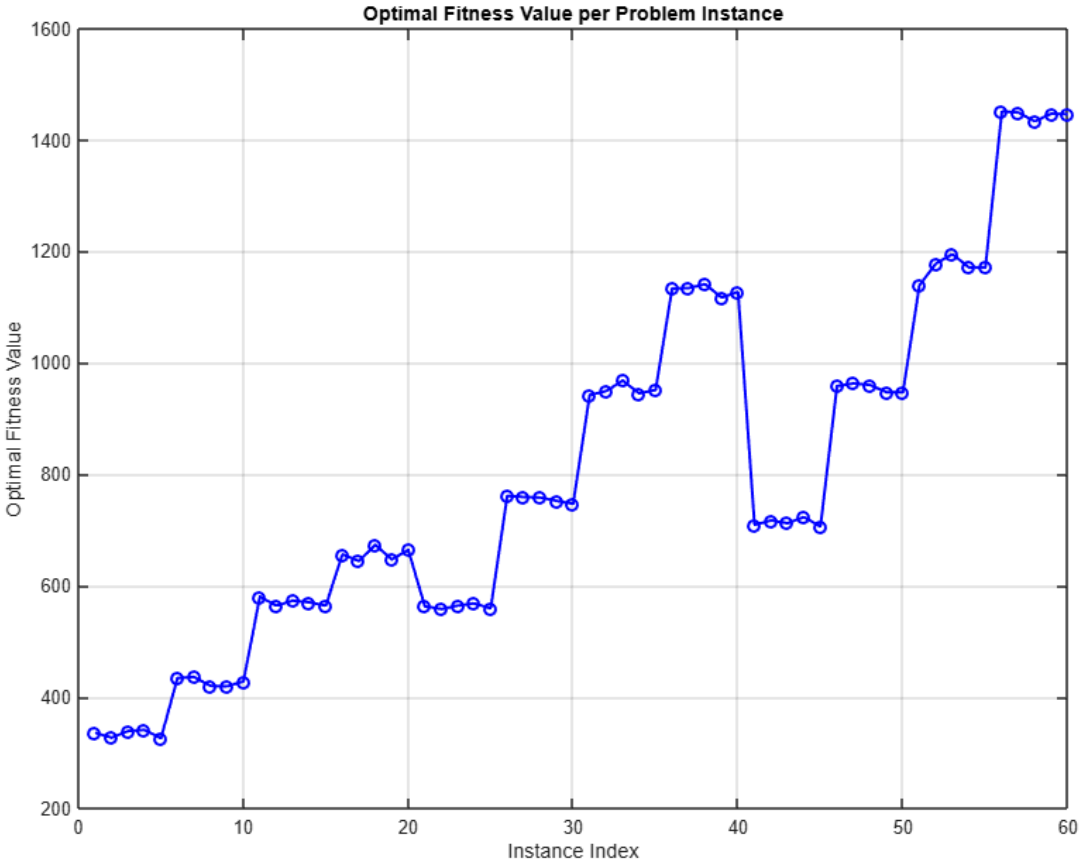


```
>> Assignment_1
gap1            gap2            gap3            gap4
c515-1  336     c520-1  434     c525-1  580     c530-1  656
c515-2  327     c520-2  436     c525-2  564     c530-2  644
c515-3  339     c520-3  420     c525-3  573     c530-3  673
c515-4  341     c520-4  419     c525-4  570     c530-4  647
c515-5  326     c520-5  428     c525-5  564     c530-5  664

gap5            gap6            gap7            gap8
c824-1  563     c832-1  761     c840-1  942     c848-1  1133
c824-2  558     c832-2  759     c840-2  949     c848-2  1134
c824-3  564     c832-3  758     c840-3  968     c848-3  1141
c824-4  568     c832-4  752     c840-4  945     c848-4  1117
c824-5  559     c832-5  747     c840-5  951     c848-5  1127

gap9            gap10           gap11           gap12
c1030-1 709     c1040-1 958     c1050-1 1139    c1060-1 1451
c1030-2 717     c1040-2 963     c1050-2 1178    c1060-2 1449
c1030-3 712     c1040-3 960     c1050-3 1195    c1060-3 1433
c1030-4 723     c1040-4 947     c1050-4 1171    c1060-4 1447
c1030-5 706     c1040-5 947     c1050-5 1171    c1060-5 1446

>>
```


Optimal Fitness Value per Problem Instance
Optimal Fitness Value
Instance Index
1600
1400
1200
1000
800
600
400
200
0
10
20
30
40
50
60