MINI-PROJECT 2:

***TRAFFIC ENGINEERING OF TELECOMMUNICATION NETWORKS***

Pedro Caio (105127) and Miguel Ferreira (93419)

Auto-evaluation: 50 % / 50 %

17/12/2024

Contents

[Task 1 (evaluation: 6.0 values) 2](#_Toc185270260)

[1.a. 2](#_Toc185270261)

[1.b 3](#_Toc185270262)

[1.c 4](#_Toc185270263)

[1.d 6](#_Toc185270264)

[1.e 7](#_Toc185270265)

[1.f 8](#_Toc185270266)

[Task 2 (evaluation: 8.0 values) 10](#_Toc185270267)

[2.a. 10](#_Toc185270268)

[2.b. 10](#_Toc185270269)

[2.c. 10](#_Toc185270270)

[2.d. 10](#_Toc185270271)

[2.e. 10](#_Toc185270272)

[References 10](#_Toc185270273)

[Appendix: 10](#_Toc185270274)

[Exercise 1: 10](#_Toc185270275)

[createPathFlows 10](#_Toc185270276)

[c paragraph code: 11](#_Toc185270277)

[d paragraph code: 12](#_Toc185270278)

[e paragraph code: 13](#_Toc185270279)

[f paragraph code: 14](#_Toc185270280)

# Task 1 (evaluation: 6.0 values)

In Task 1, consider the solutions where all traffic flows of both unicast services ( and ) are routed through the shortest propagation delay path.

## 1.a.

Compute the worst round-trip delay and the average round trip delay of each of the 3 services (presenting all values in milliseconds) if the anycast nodes of the anycast service () are network nodes and .

**Answer:**

Anycast nodes= 3 10

Worst round-trip delay (unicast service s=1) = 9.04 ms

Average round-trip delay (unicast service s=1) = 5.42 ms

Worst round-trip delay (unicast service s=2) = 11.07 ms

Average round-trip delay (unicast service s=2) = 5.89 ms

Worst round-trip delay (anycast service) = 6.16 ms

Average round-trip delay (anycast service) = 3.43 ms

The code that generated these results was:

load("InputDataProject2.mat", "L", "Links", "Nodes", "T");

nNodes= size(Nodes,1);

nLinks= size(Links,1);

nFlows= size(T,1);

nFlows\_unicast = nnz(T(:,1) == 1 | T(:,1) == 2);

nFlows\_unicast1 = nnz(T(:,1) == 1);

v = 2e5;

D = L/v; % propagation delay

anycastNodes = [3, 10];

unicastservices = [1, 2];

[sP,nSP,totalCosts\_unicast\_a, totalCosts\_anycast\_a,T] = createPathFlows(D,T,1,unicastservices,anycastNodes);

totalCosts\_unicast1\_a = totalCosts\_unicast\_a(1:nFlows\_unicast1);

totalCosts\_unicast2\_a = totalCosts\_unicast\_a(nFlows\_unicast1:end);

fprintf("Anycast nodes= %s\n", num2str(anycastNodes))

fprintf("Worst round-trip delay (unicast service s=1) = %.2f ms\n", max(totalCosts\_unicast1\_a\*2)\*1e3)

fprintf("Average round-trip delay (unicast service s=1) = %.2f ms\n", mean(totalCosts\_unicast1\_a\*2)\*1e3)

fprintf("Worst round-trip delay (unicast service s=2) = %.2f ms\n", max(totalCosts\_unicast2\_a\*2)\*1e3)

fprintf("Average round-trip delay (unicast service s=2) = %.2f ms\n", mean(totalCosts\_unicast2\_a\*2)\*1e3)

fprintf("Worst round-trip delay (anycast service) = %.2f ms\n", max(totalCosts\_anycast\_a\*2)\*1e3)

fprintf("Average round-trip delay (anycast service) = %.2f ms\n", mean(totalCosts\_anycast\_a\*2)\*1e3)

## 1.b

Determine the link loads of all links and the worst link load of the network of the previous solution.

**Answer:**

Worst link load = 98.20

{ 1 - 2}: 15.00 15.20

{ 1 - 5}: 20.30 26.20

{ 1 - 7}: 0.00 0.00

{ 2 - 3}: 48.00 52.00

{ 2 - 4}: 33.10 34.20

{ 2 - 5}: 47.80 48.90

{ 3 - 6}: 31.50 3.00

{ 3 - 8}: 33.10 31.60

{ 4 - 5}: 36.00 21.00

{ 4 - 8}: 34.40 35.20

{ 4 - 9}: 13.40 15.60

{ 4 - 10}: 28.30 46.90

{ 5 - 7}: 47.80 48.90

{ 6 - 8}: 11.60 9.90

{ 6 - 14}: 38.90 27.80

{ 6 - 15}: 8.50 0.90

{ 7 - 9}: 55.50 71.10

{ 8 - 10}: 76.80 88.00

{ 8 - 12}: 14.80 14.10

{ 9 - 10}: 70.30 98.20

{10 - 11}: 85.60 65.50

{11 - 13}: 61.70 46.50

{12 - 13}: 15.00 17.10

{12 - 14}: 14.80 14.10

{13 - 14}: 40.90 40.80

{13 - 16}: 0.00 0.00

{14 - 15}: 29.30 29.40

{15 - 16}: 0.00 0.00

The code that generated these results was:

sol= ones(1,nFlows);

Loads\_b= calculateLinkLoads(nNodes,Links,T(:,2:end),sP,sol);

W\_b = max(max(Loads\_b(:,3:4)));

fprintf("Anycast nodes= %s\n", num2str(anycastNodes))

fprintf("Worst link load = %.2f\n", W\_b)

for i=1:size(Loads\_b,1)

fprintf('{%2d - %2d}: %5.2f %5.2f \n', Loads\_b(i,1), Loads\_b(i,2), Loads\_b(i,3), Loads\_b(i,4))

end

## 1.c

Consider that you can freely select the anycast nodes of the anycast service. Try all possible combinations of nodes and select the one that minimizes the worst link load. Indicate the two selected nodes, the worst round-trip delay and the average round trip delay of each service.

**Answer:**

Anycast nodes= 1 6

Worst round-trip delay (unicast service s=1) = 9.04 ms

Average round-trip delay (unicast service s=1) = 5.42 ms

Worst round-trip delay (unicast service s=2) = 11.07 ms

Average round-trip delay (unicast service s=2) = 5.89 ms

Worst round-trip delay (anycast service) = 6.41 ms

Average round-trip delay (anycast service) = 3.02 ms

Worst link load = 76.60

The code that generated these results was:

n\_anycastNodes = 2;

% Optimization loop

W\_best = inf;

sol= ones(1,nFlows);

for anycastNodes = nchoosek(1:nNodes,n\_anycastNodes)'

[sP,nSP,totalCosts\_unicast, totalCosts\_anycast,T] = createPathFlows(D,T,1,unicastservices,anycastNodes);

% minimizing the worst link load

Loads= calculateLinkLoads(nNodes,Links,T(:,2:end),sP,sol);

W\_i = max(max(Loads(:,3:4)));

if W\_i < W\_best

anycastNodes\_best\_c = anycastNodes;

W\_best = W\_i;

sP\_best = sP;

nSP\_best = nSP;

totalCosts\_unicast\_best\_c = totalCosts\_unicast;

totalCosts\_anycast\_best\_c = totalCosts\_anycast;

T\_best = T;

end

end

totalCosts\_unicast1\_best\_c = totalCosts\_unicast\_best\_c(1:nFlows\_unicast1);

totalCosts\_unicast2\_best\_c = totalCosts\_unicast\_best\_c(nFlows\_unicast1:end);

fprintf("Anycast nodes= %s\n", num2str(anycastNodes\_best\_c'))

fprintf("Worst round-trip delay (unicast service s=1) = %.2f ms\n", max(totalCosts\_unicast1\_best\_c\*2)\*1e3)

fprintf("Average round-trip delay (unicast service s=1) = %.2f ms\n", mean(totalCosts\_unicast1\_best\_c\*2)\*1e3)

fprintf("Worst round-trip delay (unicast service s=2) = %.2f ms\n", max(totalCosts\_unicast2\_best\_c\*2)\*1e3)

fprintf("Average round-trip delay (unicast service s=2) = %.2f ms\n", mean(totalCosts\_unicast2\_best\_c\*2)\*1e3)

fprintf("Worst round-trip delay (anycast service) = %.2f ms\n", max(totalCosts\_anycast\_best\_c\*2)\*1e3)

fprintf("Average round-trip delay (anycast service) = %.2f ms\n", mean(totalCosts\_anycast\_best\_c\*2)\*1e3)

Loads\_c= calculateLinkLoads(nNodes,Links,T\_best(:,2:end),sP\_best,sol);

W\_c = max(max(Loads\_c(:,3:4)));

fprintf("Worst link load = %.2f\n", W\_c)

## 1.d

Again, consider that you can freely select the anycast nodes of the anycast service. Try all possible combinations of nodes and select the one that minimizes the worst round-trip delay of the anycast service. Indicate the two selected nodes, the worst round-trip delay and the average round trip delay of each service.

**Answer:**

Anycast nodes= 4 12

Worst round-trip delay (unicast service s=1) = 9.04 ms

Average round-trip delay (unicast service s=1) = 5.42 ms

Worst round-trip delay (unicast service s=2) = 11.07 ms

Average round-trip delay (unicast service s=2) = 5.89 ms

Worst round-trip delay (anycast service) = 4.42 ms

Average round-trip delay (anycast service) = 2.90 ms

Worst link load = 76.60

As the structure of the code remained the same as the previous exercise only the optimization code will be shown, for full code check the [respective appendix](#_d_paragraph_code:).

% Optimization loop

W\_best = inf;

for anycastNodes = nchoosek(1:nNodes,n\_anycastNodes)'

[sP,nSP,totalCosts\_unicast, totalCosts\_anycast,T] = createPathFlows(D,T,1,unicastservices,anycastNodes);

% minimizing the worst round-trip delay of the anycast service

W\_i = max(totalCosts\_anycast\*2);

if W\_i < W\_best

anycastNodes\_best\_d = anycastNodes;

W\_best = W\_i;

sP\_best = sP;

nSP\_best = nSP;

totalCosts\_unicast\_best\_d = totalCosts\_unicast;

totalCosts\_anycast\_best\_d = totalCosts\_anycast;

T\_best = T;

end

end

## 1.e

Again, consider that you can freely select the anycast nodes of the anycast service. Try all possible combinations of 2 nodes and select the one that minimizes the average round trip delay of the anycast service. Indicate the two selected nodes, the worst round-trip delay and the average round trip delay of each service.

**Answer:**

Anycast nodes= 5 14

Worst round-trip delay (unicast service s=1) = 9.04 ms

Average round-trip delay (unicast service s=1) = 5.42 ms

Worst round-trip delay (unicast service s=2) = 11.07 ms

Average round-trip delay (unicast service s=2) = 5.89 ms

Worst round-trip delay (anycast service) = 4.90 ms

Average round-trip delay (anycast service) = 2.52 ms

Worst link load = 76.60

As the structure of the code remained the same as the previous exercise only the optimization code will be shown, for full code check the [respective appendix](#_e_paragraph_code:).

% Optimization loop

W\_best = inf;

sol= ones(1,nFlows);

for anycastNodes = nchoosek(1:nNodes,n\_anycastNodes)'

[sP,nSP,totalCosts\_unicast, totalCosts\_anycast,T] = createPathFlows(D,T,1,unicastservices,anycastNodes);

% minimizing the average round-trip delay of the anycast service

W\_i = mean(totalCosts\_anycast\*2);

if W\_i < W\_best

anycastNodes\_best\_e = anycastNodes;

W\_best = W\_i;

sP\_best = sP;

nSP\_best = nSP;

totalCosts\_unicast\_best\_e = totalCosts\_unicast;

totalCosts\_anycast\_best\_e = totalCosts\_anycast;

T\_best = T;

end

end

## 1.f

Compare the results obtained in all experiments of Task 1 and draw all possible conclusions.

**Answer:**

c\_means= [5.425 5.8933 3.0233]

d\_means= [5.425 5.8933 2.8983]

e\_means= [5.425 5.8933 2.525]

c\_max= [9.04 11.07 6.41]

d\_max= [9.04 11.07 4.42]

e\_max= [9.04 11.07 4.9]

c\_max-load= 76.60

d\_max-load= 76.60

e\_max-load= 76.60

**Uma imagem com texto, captura de ecrã, diagrama, Tipo de letra

Descrição gerada automaticamente**

It's possible to observe that the unicast delays are the same across all optimizations, this is because the routing for unicast services is fixed and not influenced by the anycast optimizations. The paths and delays for unicast services are determined independently of the anycast service optimizations, leading to consistent delays regardless of the changes made for the anycast service, the only change is on the loads which is not evaluated.

Minimizing the worst link load (**1.c**) results in higher delays for the anycast service because the optimization focuses on balancing the load across the network rather than optimizing for delay. This can lead to longer paths being chosen to avoid congestion, increasing the round-trip delays.

Minimizing the worst round-trip delay of the anycast service (**1.d**) significantly improves both the worst and average delays for the anycast service having, expectantly, the best worse round-trip delay of all optimizations, because of the optimization objective. This approach can be seen as a prioritization of the worst-case scenario, leading to slightly higher mean delay to all users in comparison to the (**1.e**) optimization.

Minimizing the average round-trip delay of the anycast service (**1.e**) achieves the best average delay performance for the anycast service, expectantly, because of the objective of the optimization. This approach can be seen as a prioritization of the average scenario, which may be useful if the slightly worse worst case is admissible in the problem at stake.

In summary, the different optimization objectives lead to varying delay performances. Minimizing the worst link load can increase delays due to load balancing, while minimizing the worst or average round-trip delays for the anycast service directly improves delay performance by prioritizing shorter and more efficient paths. The unicast delays are unaffected because they are determined independently of the anycast service optimizations.

# Task 2 (evaluation: 8.0 values)

Consider the optimization problem of computing a single routing path to each traffic flow of both unicast services, aiming to minimize the worst link load of the network. Use a k-shortest path algorithm (using the propagation delays of the links) to determine the candidate routing paths for each flow of each unicast service.

To solve this problem, use a Multi Start Hill Climbing algorithm with initial Greedy Randomized solutions and with a stopping criterion defined by a given running time. Besides the found best solution, the algorithm must also output the following performance parameters:

1. total number of cycles run
2. the running time at which the algorithm obtains its best solution
3. the number of cycles at which the algorithm obtains its best solution.

The code that was used to run the simulation was:

% Initial variables  
load('InputDataProject2.mat');  
nNodes = size(Nodes,1);  
  
anycastNodes = [3 10];  
unicastServices = [1 2];  
  
% [3 10]  
% [1 6]  
% [4 12]  
% [5 14]  
  
v= 2e5;  
D = L/v;  
k = 6;  
[sP,nSP,totalCosts\_unicast, totalCosts\_anycast,T] = createPathFlows(D, T, k, unicastServices, anycastNodes);  
  
  
timeLimit = 40;  
  
[bestSol,tbestSol,bestObjective,totalNCycles, bestNCycles,avObjective] = AlgorithmD(nNodes,Links,T,sP,nSP,timeLimit, unicastServices);

The simulation relies on the calls of two functions *createPathFlows* and *AlgorithmD.* In the first function, the shortest paths are obtained as well as the total costs for each service, which in this case will be related to the propagation delays. The *T* matrix is also returned as the destination nodes for the anycast service are obtained.

Immediately, the performance parameters (i), (ii) and (iii) are obtained from variables *totalNCycles*, *tbestSol* and *bestNCycles*, respectively.

function [bestSol,tbestSol,bestObjective,totalNCycles, bestNCycles,avObjective] = AlgorithmD(nNodes,Links,T,sP,nSP,timeLimit, unicastServices)  
 t= tic;  
 nFlows= size(T,1);  
 nFlows\_unicast = nnz(ismember(T(:,1),unicastServices));  
  
 bestObjective= inf;  
 totalNCycles= 0;  
 aux= 0;  
 while toc(t) < timeLimit  
 % Greedy random algorithm  
 sol= zeros(1,nFlows);  
 sol= BuildSolution(sol, nNodes,Links,T,sP,nSP,nFlows, nFlows\_unicast);  
 sol= HillClimbing(sol, nNodes,Links,T,sP,nSP, nFlows\_unicast);  
 %  
 Loads= calculateLinkLoads(nNodes,Links,T(:,2:end),sP,sol);  
 load= max(max(Loads(:,3:4)));  
 totalNCycles= totalNCycles+1;  
 aux= aux+load;  
 if load<bestObjective  
 bestNCycles = totalNCycles;  
 bestSol= sol;  
 bestObjective= load;  
 tbestSol= toc(t);  
 end  
 end  
 avObjective= aux/totalNCycles;  
end  
  
function sol = BuildSolution(sol, nNodes,Links,T,sP,nSP,nFlows, nFlows\_unicast)  
 sol(nFlows\_unicast+1:nFlows) = 1;  
 Flow\_order = randperm(nFlows\_unicast);  
  
 for f=Flow\_order  
 best = inf;  
 for i=1:nSP(f)  
 sol(f) = i;  
 Loads= calculateLinkLoads(nNodes,Links,T(:,2:end),sP,sol);  
 load= max(max(Loads(:,3:4)));  
 if load < best  
 best= load;  
 i\_best= i;  
 end  
 end  
 sol(f)= i\_best;  
 end  
end

For *AlgorithmD*, the most important observation is the optimization of the initial solution that was made in the function *BuildSolution*. For the anycast service the solution can already be assumed since there is only one path for each of its flows. Therefore we only need to randomize the order of the unicast flows, as shown in the declaration of *Flow\_order*.

function sol = HillClimbing(sol, nNodes,Links,T,sP,nSP, nFlows\_unicast)  
 %nFlows = length(sol);  
 improving = true;  
 while improving  
 % Get current best solution and respective load  
 current\_sol = sol;  
 Loads= calculateLinkLoads(nNodes,Links,T(:,2:end),sP,current\_sol);  
 load= max(max(Loads(:,3:4)));  
 best = load;  
 bestmove = [];  
 % Improvement loop onto neighboring solutions  
 for f=1:nFlows\_unicast  
 for i=1:nSP(f)  
 % The current solution is not a neighbor of itself  
 if i==current\_sol(f)  
 continue  
 end  
 sol(f) = i;  
 Loads= calculateLinkLoads(nNodes,Links,T(:,2:end),sP,sol);  
 load= max(max(Loads(:,3:4)));  
 if load < best  
 best= load;  
 i\_best= i;  
 bestmove = [f, i\_best];  
 end  
 end  
 % recover the starting solution  
 sol(f)= current\_sol(f);  
 end  
 if isempty(bestmove)  
 improving = false;  
 else  
 sol(bestmove(1))= bestmove(2);  
 bestmove = [];  
 end  
 end  
end

Naturally, in the HillClimbing algorithm, we will now only need to iterate through the unicast flows.

## 2.a.

Run the algorithm for 30 seconds with k = 6 candidate paths for each unicast flow when the anycast nodes of the anycast service are network nodes 3 and 10. Indicate the worst round-trip delay and the average round trip delay of each service (presenting all values in milliseconds) and the worst link load of the network. Indicate also the algorithm performance parameters.

The code that generated these results was:

nFlows\_unicast = nnz(ismember(T(:,1),unicastServices));  
nServices = length(unicastServices);  
worstDelay = zeros(nServices, 1);  
averageDelay = zeros(nServices, 1);  
for f=1:nFlows\_unicast  
  
 averageDelay(T(f, 1)) = averageDelay(T(f, 1)) + totalCosts\_unicast(f, bestSol(f));  
  
 if totalCosts\_unicast(f, bestSol(f)) > worstDelay(T(f, 1))  
 worstDelay(T(f, 1)) = totalCosts\_unicast(f, bestSol(f));  
 end  
  
end  
  
% Convert to milisseconds  
averageDelay = averageDelay \* 2 \* (10^3);  
worstDelay = worstDelay \* 2 \* (10^3);  
  
  
nFlows\_s1 = 12; nFlows\_s2 = 8; %TODO  
  
fprintf("Worst round-trip delay (service 1) = %.2f ms\n", worstDelay(1));  
fprintf("Average round trip delay (service 1) = %.2f ms\n", averageDelay(1)/nFlows\_s1);  
  
fprintf("Worst round-trip delay (service 2) = %.2f ms\n", worstDelay(2));  
fprintf("Average round trip delay (service 2) = %.2f ms\n", averageDelay(2)/nFlows\_s2);  
  
fprintf("Worst round-trip delay (service 3) = %f ms\n", max(totalCosts\_anycast) \* 2 \* (10^3));  
fprintf("Average round trip delay (service 3) = %f ms\n", mean(totalCosts\_anycast) \* 2 \* (10^3));  
  
fprintf("Worst link load of the network = %.2f Gbps\n", bestObjective);

From the totalCosts variables obtained previously by the createPathFlows function, we obtain the delay times since the best solution is now known.

**Answer:**

Total nº Cycles = 9680

Running time at which the algorithm obtains its best solution = 0.39 s

The number of cycles at which the algorithm obtains its best solution. = 100

Worst round-trip delay (service 1) = 10.13 ms

Average round trip delay (service 1) = 6.41 ms

Worst round-trip delay (service 2) = 11.07 ms

Average round trip delay (service 2) = 6.52 ms

Worst link load of the network = 74.60 Gbps

## 

## 2.b.

Repeat experiment 2.a with the anycast nodes selected in experiment 1.c.

**Answer:**

Total nº Cycles = 8374

Running time at which the algorithm obtains its best solution = 0.03 s

The number of cycles at which the algorithm obtains its best solution. = 3

Worst round-trip delay (service 1) = 9.04 ms

Average round trip delay (service 1) = 5.64 ms

Worst round-trip delay (service 2) = 11.07 ms

Average round trip delay (service 2) = 6.17 ms

Worst link load of the network = 60.00 Gbps

## 2.c.

Repeat experiment 2.a with the anycast nodes selected in experiment 1.d.

**Answer:**

Total nº Cycles = 6508

Running time at which the algorithm obtains its best solution = 0.06 s

The number of cycles at which the algorithm obtains its best solution. = 9

Worst round-trip delay (service 1) = 9.04 ms

Average round trip delay (service 1) = 5.68 ms

Worst round-trip delay (service 2) = 11.07 ms

Average round trip delay (service 2) = 6.34 ms

Worst link load of the network = 62.60 Gbps

## 2.d.

Repeat experiment 2.a with the anycast nodes selected in experiment 1.e.

**Answer:**

Total nº Cycles = 7731

Running time at which the algorithm obtains its best solution = 0.02 s

The number of cycles at which the algorithm obtains its best solution. = 1

Worst round-trip delay (service 1) = 9.04 ms

Average round trip delay (service 1) = 5.68 ms

Worst round-trip delay (service 2) = 11.07 ms

Average round trip delay (service 2) = 6.52 ms

Worst link load of the network = 60.00 Gbps

## 2.e.

Compare the results obtained in all experiments of this Task 2 and of the previous Task 1 and draw all meaningful conclusions both concerning the differences between the solutions and the differences between the performance of the algorithm.

# References

# Appendix:

Reatualizar o código novamente

## Exercise 1:

### createPathFlows

function [sP,nSP,totalCosts\_unicast, totalCosts\_anycast,T] = createPathFlows(costMatrix,T,k,unicastservices,anycastNodes)

nFlows= size(T,1);

nFlows\_unicast = nnz(ismember(T(:,1),unicastservices));

n\_anycastNodes = length(anycastNodes);

totalCosts\_unicast = zeros(nFlows\_unicast, 1);

totalCosts\_anycast = zeros(nFlows - nFlows\_unicast, 1);

% Possible anycast paths

possible\_shortestPath = cell(n\_anycastNodes,1);

possible\_totalCost = zeros(n\_anycastNodes,1);

k= 1;

sP= cell(1,nFlows);

nSP= zeros(1,nFlows);

for f = 1:nFlows

if ismember(T(f,1), unicastservices)

% Flow is unicast

[shortestPath, totalCost] = kShortestPath(costMatrix,T(f,2),T(f,3),k);

sP{f}= shortestPath;

nSP(f)= length(totalCost);

totalCosts\_unicast(f) = totalCost;

else

% Flow is anycast

if ismember(T(f,2), anycastNodes)

sP{f} = {T(f,2)};

nSP(f) = 1;

T(f,3) = T(f,2);

% totalCost is already initialize to zero

else

for node = 1:n\_anycastNodes

[possible\_shortestPath(node), possible\_totalCost(node)] = kShortestPath(costMatrix,T(f,2),anycastNodes(node),k);

end

[M, I] = min(possible\_totalCost);

sP{f} = possible\_shortestPath(I);

nSP(f) = length(possible\_totalCost(I));

% index of array must begin in 1

totalCosts\_anycast(f - nFlows\_unicast) = possible\_totalCost(I);

T(f,3) = anycastNodes(I);

end

end

end

end

### c paragraph code:

% Optimization loop

W\_best = inf;

for anycastNodes = nchoosek(1:nNodes,n\_anycastNodes)'

[sP,nSP,totalCosts\_unicast, totalCosts\_anycast,T] = createPathFlows(D,T,1,unicastservices,anycastNodes);

% minimizing the worst round-trip delay of the anycast service

W\_i = max(totalCosts\_anycast\*2);

if W\_i < W\_best

anycastNodes\_best\_d = anycastNodes;

W\_best = W\_i;

sP\_best = sP;

nSP\_best = nSP;

totalCosts\_unicast\_best\_d = totalCosts\_unicast;

totalCosts\_anycast\_best\_d = totalCosts\_anycast;

T\_best = T;

end

end

totalCosts\_unicast1\_best\_d = totalCosts\_unicast\_best\_d(1:nFlows\_unicast1);

totalCosts\_unicast2\_best\_d = totalCosts\_unicast\_best\_d(nFlows\_unicast1:end);

fprintf("Anycast nodes= %s\n", num2str(anycastNodes\_best\_d'))

fprintf("Worst round-trip delay (unicast service s=1) = %.2f ms\n", max(totalCosts\_unicast1\_best\_d\*2)\*1e3)

fprintf("Average round-trip delay (unicast service s=1) = %.2f ms\n", mean(totalCosts\_unicast1\_best\_d\*2)\*1e3)

fprintf("Worst round-trip delay (unicast service s=2) = %.2f ms\n", max(totalCosts\_unicast2\_best\_d\*2)\*1e3)

fprintf("Average round-trip delay (unicast service s=2) = %.2f ms\n", mean(totalCosts\_unicast2\_best\_d\*2)\*1e3)

fprintf("Worst round-trip delay (anycast service) = %.2f ms\n", max(totalCosts\_anycast\_best\_d\*2)\*1e3)

fprintf("Average round-trip delay (anycast service) = %.2f ms\n", mean(totalCosts\_anycast\_best\_d\*2)\*1e3)

Loads\_d= calculateLinkLoads(nNodes,Links,T\_best(:,2:end),sP\_best,sol);

W\_d = max(max(Loads\_d(:,3:4)));

fprintf("Worst link load = %.2f\n", W\_d)

### d paragraph code:

% Optimization loop

W\_best = inf;

for anycastNodes = nchoosek(1:nNodes,n\_anycastNodes)'

[sP,nSP,totalCosts\_unicast, totalCosts\_anycast,T] = createPathFlows(D,T,1,unicastservices,anycastNodes);

% minimizing the worst round-trip delay of the anycast service

W\_i = max(totalCosts\_anycast\*2);

if W\_i < W\_best

anycastNodes\_best\_d = anycastNodes;

W\_best = W\_i;

sP\_best = sP;

nSP\_best = nSP;

totalCosts\_unicast\_best\_d = totalCosts\_unicast;

totalCosts\_anycast\_best\_d = totalCosts\_anycast;

T\_best = T;

end

end

totalCosts\_unicast1\_best\_d = totalCosts\_unicast\_best\_d(1:nFlows\_unicast1);

totalCosts\_unicast2\_best\_d = totalCosts\_unicast\_best\_d(nFlows\_unicast1:end);

fprintf("Anycast nodes= %s\n", num2str(anycastNodes\_best\_d'))

fprintf("Worst round-trip delay (unicast service s=1) = %.2f ms\n", max(totalCosts\_unicast1\_best\_d\*2)\*1e3)

fprintf("Average round-trip delay (unicast service s=1) = %.2f ms\n", mean(totalCosts\_unicast1\_best\_d\*2)\*1e3)

fprintf("Worst round-trip delay (unicast service s=2) = %.2f ms\n", max(totalCosts\_unicast2\_best\_d\*2)\*1e3)

fprintf("Average round-trip delay (unicast service s=2) = %.2f ms\n", mean(totalCosts\_unicast2\_best\_d\*2)\*1e3)

fprintf("Worst round-trip delay (anycast service) = %.2f ms\n", max(totalCosts\_anycast\_best\_d\*2)\*1e3)

fprintf("Average round-trip delay (anycast service) = %.2f ms\n", mean(totalCosts\_anycast\_best\_d\*2)\*1e3)

Loads\_d= calculateLinkLoads(nNodes,Links,T\_best(:,2:end),sP\_best,sol);

W\_d = max(max(Loads\_d(:,3:4)));

fprintf("Worst link load = %.2f\n", W\_d)

### e paragraph code:

% Optimization loop

W\_best = inf;

sol= ones(1,nFlows);

for anycastNodes = nchoosek(1:nNodes,n\_anycastNodes)'

[sP,nSP,totalCosts\_unicast, totalCosts\_anycast,T] = createPathFlows(D,T,1,unicastservices,anycastNodes);

% minimizing the average round-trip delay of the anycast service

W\_i = mean(totalCosts\_anycast\*2);

if W\_i < W\_best

anycastNodes\_best\_e = anycastNodes;

W\_best = W\_i;

sP\_best = sP;

nSP\_best = nSP;

totalCosts\_unicast\_best\_e = totalCosts\_unicast;

totalCosts\_anycast\_best\_e = totalCosts\_anycast;

T\_best = T;

end

end

totalCosts\_unicast1\_best\_e = totalCosts\_unicast\_best\_e(1:nFlows\_unicast1);

totalCosts\_unicast2\_best\_e = totalCosts\_unicast\_best\_e(nFlows\_unicast1:end);

fprintf("Anycast nodes= %s\n", num2str(anycastNodes\_best\_e'))

fprintf("Worst round-trip delay (unicast service s=1) = %.2f ms\n", max(totalCosts\_unicast1\_best\_e\*2)\*1e3)

fprintf("Average round-trip delay (unicast service s=1) = %.2f ms\n", mean(totalCosts\_unicast1\_best\_e\*2)\*1e3)

fprintf("Worst round-trip delay (unicast service s=2) = %.2f ms\n", max(totalCosts\_unicast2\_best\_e\*2)\*1e3)

fprintf("Average round-trip delay (unicast service s=2) = %.2f ms\n", mean(totalCosts\_unicast2\_best\_e\*2)\*1e3)

fprintf("Worst round-trip delay (anycast service) = %.2f ms\n", max(totalCosts\_anycast\_best\_e\*2)\*1e3)

fprintf("Average round-trip delay (anycast service) = %.2f ms\n", mean(totalCosts\_anycast\_best\_e\*2)\*1e3)

Loads\_e= calculateLinkLoads(nNodes,Links,T\_best(:,2:end),sP\_best,sol);

W\_e = max(max(Loads\_e(:,3:4)));

fprintf("Worst link load = %.2f\n", W\_e)

### f paragraph code:

% Calculate means and max values

c\_means = [mean(totalCosts\_unicast1\_best\_c), mean(totalCosts\_unicast2\_best\_c), mean(totalCosts\_anycast\_best\_c)] \* 2 \* 1e3;

d\_means = [mean(totalCosts\_unicast1\_best\_d), mean(totalCosts\_unicast2\_best\_d), mean(totalCosts\_anycast\_best\_d)] \* 2 \* 1e3;

e\_means = [mean(totalCosts\_unicast1\_best\_e), mean(totalCosts\_unicast2\_best\_e), mean(totalCosts\_anycast\_best\_e)] \* 2 \* 1e3;

c\_max = [max(totalCosts\_unicast1\_best\_c), max(totalCosts\_unicast2\_best\_c), max(totalCosts\_anycast\_best\_c)] \* 2 \* 1e3;

d\_max = [max(totalCosts\_unicast1\_best\_d), max(totalCosts\_unicast2\_best\_d), max(totalCosts\_anycast\_best\_d)] \* 2 \* 1e3;

e\_max = [max(totalCosts\_unicast1\_best\_e), max(totalCosts\_unicast2\_best\_e), max(totalCosts\_anycast\_best\_e)] \* 2 \* 1e3;

% Labels for the x-axis

labels = {'Unicast Service 1', 'Unicast Service 2', 'Anycast Service'};

% Bar plot with means

x = 1:length(labels); % the label locations

width = 0.25; % the width of the bars

fcomparison\_1f = figure;

hold on;

bar(x - width, c\_means, width, 'DisplayName', 'Question 1.c');

bar(x, d\_means, width, 'DisplayName', 'Question 1.d');

bar(x + width, e\_means, width, 'DisplayName', 'Question 1.e');

% Adding points for the max values

plot(x - width, c\_max, 'ro', 'HandleVisibility', 'off');

plot(x, d\_max, 'ro', 'HandleVisibility', 'off');

plot(x + width, e\_max, 'ro', 'HandleVisibility', 'off');

% Add some text for labels, title and custom x-axis tick labels, etc.

xlabel('Services');

ylabel('Round-Trip Delay (ms)');

title('Round-Trip Delays by Service and Result');

set(gca, 'XTick', x, 'XTickLabel', labels);

legend('show');

hold off;

fprintf("c\_means= [%s]\n", num2str(c\_means))

fprintf("d\_means= [%s]\n", num2str(d\_means))

fprintf("e\_means= [%s]\n", num2str(e\_means))

fprintf("c\_max= [%s]\n", num2str(c\_max))

fprintf("d\_max= [%s]\n", num2str(d\_max))

fprintf("e\_max= [%s]\n", num2str(e\_max))

fprintf("c\_max-load= %.2f\n", W\_c)

fprintf("d\_max-load= %.2f\n", W\_d)

fprintf("e\_max-load= %.2f\n", W\_e)

% saveas(fcomparison\_1f, fullfile('doc/assets', '1f\_comparison.png')) % does not work

saveas(fcomparison\_1f, fullfile('doc/assets', '1f\_comparison.svg'))