Departamento de Eletrónica Telecomunicações e Informática

8240 - Mestrado Integrado em Engenharia de Computadores e Telemática

Modelação e Desempenho de Redes e Serviços

Year 2021/2022

Mini-Project 2:

TRAFFIC ENGINEERING OF TELECOMMUNICATION NETWORKS

Authors:
Lúcia Sousa 93086
Rodrigo Martins 93264

January, 2022

Contents

1 Network and service description

Considering the MPLS (Multi-Protocol Label Switching) network of an ISP (Internet Service Provider) with the following topology composed by 10 nodes and 16 links and defined over a rectangle with 600 Km by 400 Km, depicted in the figure below:

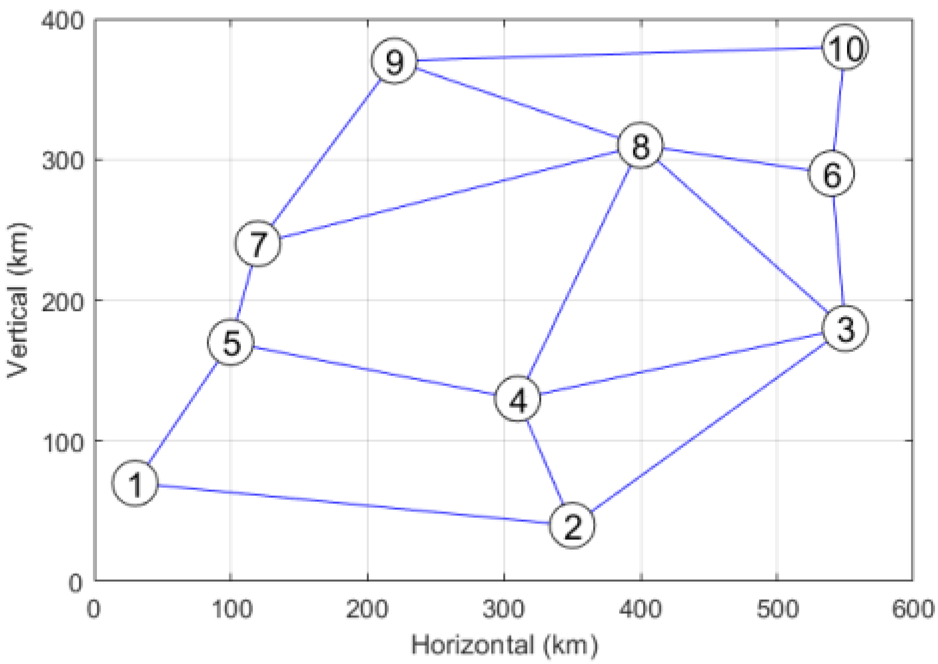


Figure 1: ISP Network

The length of all links is provided by the square matrix L. The capacity of all links is 10 Gbps in each direction. Considering a unicast service defined with the following 9 flows (throughput values bt and bt in Gbps) depicted in the figure below:

t	o_t	d_t	b_t	$\underline{b}_t$
1	1	3	1.0	1.0
2	1	4	0.7	0.5
3	2	7	2.4	1.5
4	3	4	2.4	2.1
5	4	9	1.0	2.2
6	5	6	1.2	1.5
7	5	8	2.1	2.2
8	5	9	1.6	1.9
9	6	10	1.4	1.6

Figure 2: Flows

2 Task 1

In this task, the aim is to compute a symmetrical single path routing solution to support the unicast service which minimizes the resulting worst link load. The results and conclusions of the computation mentioned above will be presented in the next paragraphs.

2.1 Experiment 1.a

The objective of this exercise is using the k-shortest path algorithm (using the lengths of the links), compute the number of different routing paths provided by the network to each traffic flow.

2.1.1 Results

The obtained results can be seen in the table below:

Flow number	Routing Paths
1	32
2	32
3	38
4	24
5	36
6	37
7	25
8	41
9	28

```
% Compute up to n paths for each flow:
n= inf;
[sP nSP]= calculatePaths(L,T,n);

for i=1:nFlows
    fprintf('No. of different routing paths for flow %d = %d\n',i,length(sP
        {1,i}));
end
```

2.2 Experiment 1.b

The objective of this exercise is to run a random algorithm during 10 seconds in three cases:

- using all possible routing paths
- using the 10 shortest routing paths
- using the 5 shortest routing paths

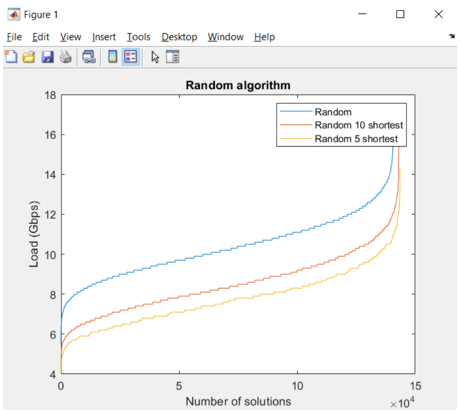
For each case, register the worst link load value of the best solution, the number of solutions generated by the algorithm and the average quality of all solutions. On a single figure, plot for the three cases the worst link load values of all solutions in an increasing order and finally to take conclusions on the influence of the number of routing paths in the efficiency of the random algorithm.

```matlab
%Optimization algorithm resorting to the random strategy:
t= tic;
bestLoad= inf;
worstLoad= inf;
sol= zeros(1,nFlows);
allValues= [];
while toc(t)<tempo
    for i= 1:nFlows
        % for 10 shortest
        % n = min(10,nSP(i));
        % for 5 shortest
        % n = min(5,nSP(i));
        sol(i)= randi(n);
    end
    Loads= calculateLinkLoads(nNodes,Links,T,sP,sol);
    load= max(max(Loads(:,3:4)));
    allValues= [allValues load];
    if load<bestLoad
        bestSol= sol;
        bestLoad= load;
    else
        worstLoad=load;
    end
end
```

2.2.1 Results

In the random algorithm a random routing path is selected to each flow. In the random 5 and 10 shortest algorithm the pool of paths to select from is reduced to the 5 and 10 shortest paths available for that flow respectively.

With this in mind we espected that the random 5 shortest algorithm would produce the best results followed by the random 10 shortest and finally the random algorithm. This results are to be expected because if we reduce the available paths for each flow to the 5 or 10 shortest this would inevitably produce shortest and with less load complete paths.
This results can be observed in the following figures:



(a) Graph for the random algorithm

```
RANDOM:
   Worst load = 9.00 Gbps
   Best load = 5.50 Gbps
   No. of solutions = 140680
   Av. quality of solutions = 10.32 Gbps
RANDOM 10 SHORTEST:
   Worst load = 6.70 Gbps
   Best load = 4.40 Gbps
   No. of solutions = 143020
   Av. quality of solutions = 8.51 Gbps
RANDOM 5 SHORTEST:
   Worst load = 7.30 Gbps
   Best load = 4.00 Gbps
   No. of solutions = 143542
   Av. quality of solutions = 7.75 Gbps
```

(b) Random algorithm results

After analyzing the results it is possible to say that our expectations regarding the load of the paths were met. It can be seen in the figures that the random 5 shortest algorithm generates better(less load) and more solutions than the other algorithms, as expected.

2.3 Experiment 1.c

The objective of this experiment is to repeat experiment 1.b but now using a greedy randomized algorithm instead of the random algorithm and to take conclusions on the influence of the number of routing paths in the efficiency of the greedy randomized algorithm.

```
%Optimization algorithm with greedy randomized:
t= tic;
bestLoad= inf;
allValues= [];
while toc(t)<tempo
    ax2= randperm(nFlows);
    sol= zeros(1,nFlows);
    for i= ax2
        k_best= 0;
        best= inf;
        % for 10 shortest
        % n = min(10,nSP(i));
        % for 5 shortest
        % n = min(5,nSP(i));
        for k= 1:n
            sol(i)= k;
            Loads= calculateLinkLoads(nNodes,Links,T,sP,sol);
            load= max(max(Loads(:,3:4)));
            if load<best
                k_best= k;
                best= load;
            end
        end
        sol(i)= k_best;
    end
    load= best;
    allValues= [allValues load];
    if load<bestLoad
        bestSol= sol;
        bestLoad= load;
    end
end
```

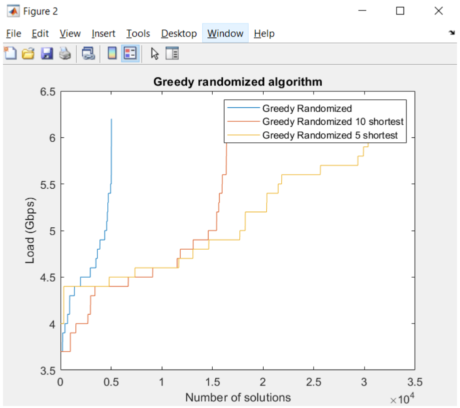
2.3.1 Results

In the greedy randomized algorithm a random order to the flows is choosen to select their routing paths and the greedy strategy is applied to the previously selected order(select the best flow for each path).

In the greedy randomized 5 and 10 shortest algorithm the pool of paths to select from is reduced to the 5 and 10 shortest paths available for that flow respectively and then a random order for those flows is selected.

With this in mind we espected that the greedy randomized 5 shortest algorithm would produce the best results followed by the greedy randomized 10 shortest and finally the greedy randomized algorithm. This results are to be expected because if we reduce the available paths for each flow to the 5 or 10 shortest this would inevitably produce shortest and with less load complete paths.

This results can be observed in the following figures:



(a) Graph for greedy randomized algorithm

```
GREEDY RANDOMIZED:
   Best load = 3.70 Gbps
   No. of solutions = 5033
   Av. quality of solutions = 4.54 Gbps
GREEDY RANDOMIZED 10 SHORTEST:
   Best load = 3.70 Gbps
   No. of solutions = 16392
   Av. quality of solutions = 4.52 Gbps
GREEDY RANDOMIZED 5 SHORTEST:
   Best load = 4.00 Gbps
   No. of solutions = 32562
   Av. quality of solutions = 5.07 Gbps
```

(b) Greedy randomized algorithm results

After analyzing the results is possible to say that our expectations regarding the load of the paths weren't met for all cases, this means that we were expecting that the greedy randomized 5 shortest would produce paths with less load than the others, but this isn't valid because the greedy randomized 10 shortest produced paths with less load than the greedy randomized 5 shortest at least in the beginning. The greedy randomized also produced better paths in the beginning together with the greedy randomized 10 shortest, but after the middle of the graph de greedy randomized 5 shortest started producing more solutions to the same load making this algorithm better to find alternative paths with the same load.

2.4 Experiment 1.d

The objective of this experiment is to repeat experiment 1.b but now using a multi start hill climbing algorithm instead of the random algorithm and to take conclusions on the influence of the number of routing paths in the efficiency of the multi start hill climbing algorithm.

```
%Optimization algorithm with multi start hill climbing:
t= tic;
bestLoad= inf;
allValues= [];
contadortotal= [];
while toc(t)<tempo
    %GREEDY RANDOMIZED:
    ax2= randperm(nFlows);
    sol= zeros(1,nFlows);
    for i= ax2
        k_best= 0;
        best= inf;
        % for 10 shortest
        % n = min(10,nSP(i));
        % for 5 shortest
        % n = min(5,nSP(i));
        for k= 1:n
            sol(i)= k;
            Loads= calculateLinkLoads(nNodes,Links,T,sP,sol);
            load= max(max(Loads(:,3:4)));
            if load<best
                k_best= k;
                best= load;
            end
        end
        sol(i)= k_best;
    end
    Loads= calculateLinkLoads(nNodes,Links,T,sP,sol);
```

```
    load= best;

    %HILL CLIMBING:
    continuar= true;
    while continuar
        i_best= 0;
        k_best= 0;
        best= load;
        for i= 1:nFlows
            % for 10 shortest
            % n = min(10,nSP(i));
            % for 5 shortest
            % n = min(5,nSP(i));
            for k= 1:n
                if k~=sol(i)
                    aux= sol(i);
                    sol(i)= k;
                    Loads= calculateLinkLoads(nNodes,Links,T,sP,sol);
                    load1= max(max(Loads(:,3:4)));
                    if load1<best
                        i_best= i;
                        k_best= k;
                        best= load1;
                    end
                    sol(i)= aux;
                end
            end
        end
        if i_best>0
            sol(i_best)= k_best;
            load= best;
        else
            continuar= false;
        end
    end
    allValues= [allValues load];
    if load<bestLoad
        bestSol= sol;
        bestLoad= load;
    end
end
```

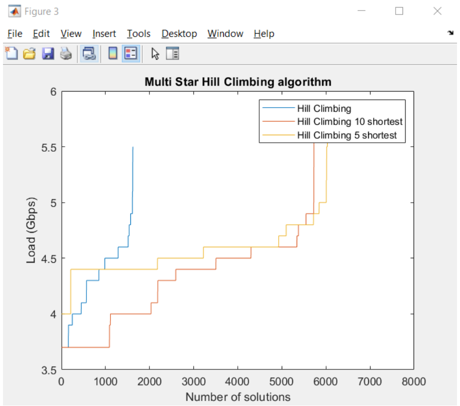
2.4.1 Results

In the multi star hill climbing algorithm the program will iterate through the paths of the solutions until it finds the best solution at the moment (i.e. the path with fewer load).

In the multi star hill climbing algorithm 5 and 10 shortest algorithm the pool of paths to select from is reduced to the 5 and 10 shortest paths available for that flow respectively.

With this in mind we espected that the multi star hill climbing algorithm 5 shortest algorithm would produce the best results followed by the multi star hill climbing algorithm 10 shortest and finally the multi star hill climbing algorithm. This results are to be expected because if we reduce the available paths for each flow to the 5 or 10 shortest this would inevitably produce shortest and with less load complete paths.

The obtained results can be observed in the following figures:



(a) Graph for multi star hill climbing algorithm

```
MULTI START HILL CLIMBING:
   Best load = 3.70 Gbps
   No. of solutions = 1620
   Av. quality of solutions = 4.29 Gbps
MULTI START HILL CLIMBING 10 SHORTEST:
   Best load = 3.70 Gbps
   No. of solutions = 5728
   Av. quality of solutions = 4.27 Gbps
MULTI START HILL CLIMBING 5 SHORTEST:
   Best load = 4.00 Gbps
   No. of solutions = 7493
   Av. quality of solutions = 4.75 Gbps
```

(b) Multi star hill climbing algorithm results

After analyzing the results is possible to say that our expectations regarding the load of the paths weren't met. The multi star hill climbing algorithm 10 shortest was the one that produced more solutions with less load followed by the multi star hill climbing algorithm and finally the multi star hill climbing algorithm 5 shortest. Only at the end of the graph the multi star hill climbing algorithm 5 shortest was better than the others by producing more solutions with the same load as the multi star hill climbing algorithm 10 shortest. With this we can conclude that multi star hill climbing algorithm 5 and 10 shortest can produce more solutions than the normal version, but the normal version can produce better or equal paths, this can be explained by saying that is not always worth taking the shortest path in each iteration because in the end it might end up creating a bigger path than a path with some "worst" paths in the middle.

2.5 Experiment 1.e

The objective of this experiment is to compare the efficiency of the three heuristic algorithms based on the results obtained in 1.b, 1.c and 1.d.

2.5.1 Results

By comparing all of the three heuristics is possible to say that the best one is the multi star hill climbing algorithm and the worst is the random algorithm. The multi star hill climbing algorithm for the same amount of time produces much less and better solutions than the random algorithm being much more efficient than it. The greedy randomized can also produce on average good solutions but is still less efficient than the multi star hill climbing but much more efficient than the randomized algorithm.

3 Task 2

Considering that the energy consumption of each link is proportional to its length and that a link not supporting traffic in any of its direction can be put in sleeping mode with no energy consumption. In this task, the aim is to compute a symmetrical single path routing solution to support the unicast service which minimizes the energy consumption of the network. The results and conclusions of the computation mentioned above are will be presented in the next paragraphs.

The objective of this exercise is to run a random algorithm during 10 seconds in three cases:

- using all possible routing paths
- using the 10 shortest routing paths
- using the 5 shortest routing paths

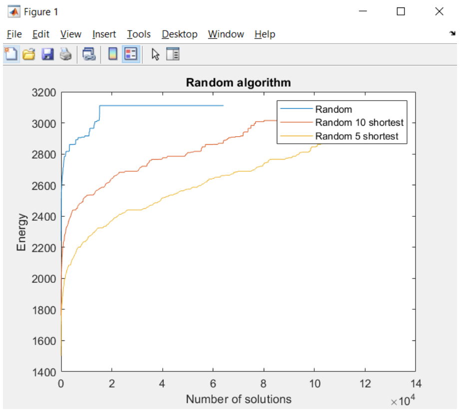
For each case, register the energy consumption value of the best solution, the number of solutions generated by the algorithm and the average quality of all solutions. On a single figure, plot for the three cases the worst link load values of all solutions in an increasing order and finally take conclusions on the influence of the number of routing paths in the efficiency of the random algorithm.

```
%random algorithm
t= tic;
bestEnergy= inf;
sol= zeros(1,nFlows);
allValues= [];
while toc(t)<10
    for i= 1:nFlows
        % for 10 shortest
        % n = min(10,nSP(i));
        % for 5 shortest
        % n = min(5,nSP(i));
        sol(i)= randi(n);
    end
    Loads= calculateLinkLoads(nNodes,Links,T,sP,sol);
    load= max(max(Loads(:,3:4)));
    if load<=10
        energy = 0;
        for a=1:nLinks
            %loads(a,1) loads(a,2) are the nodes
            %loads(a,3) loads(a,4) are the values of the loads that pass through
            %link (between the nodes) in both directions
            if Loads(a,3)+Loads(a,4)>0 %is supporting traffic
                energy = energy + L(Loads(a,1),Loads(a,2)); %the energy equals the energy + the link length
            end
        end
    else
        energy=inf; % guarantees that it does not choose if the capacity/load is greater than 10 gigabits
    end
    allValues= [allValues energy];
    if energy<bestEnergy
        bestSol= sol;
        bestEnergy= energy;
    end
end
```

3.0.1 Results

In the random algorithm a random routing path is selected to each flow. In the random 5 and 10 shortest algorithm the pool of paths to select from is reduced to the 5 and 10 shortest paths available with less energy consuption for that flow respectively. With this in mind we expected that the random 5 shortest algorithm would produce the best results followed by the random 10 shortest and finally the random algorithm. This results are to be expected because if we reduce the available paths for each flow to the 5 or 10 shortest this would inevitably produce shortest and with less energy consumption complete paths.

The obtained results can be observed in the following figures:



(a) Graph for random algorithm

```
RANDOM:
   Best energy = 2242.0
   No. of solutions = 141389
   Av. quality of solutions = Inf
RANDOM 10 SHORTEST:
   Best energy = 1761.0
   No. of solutions = 142657
   Av. quality of solutions = Inf
RANDOM 5 SHORTEST:
   Best energy = 1502.0
   No. of solutions = 142125
   Av. quality of solutions = Inf
```

(b) Random algorithm results

After analyzing the results is possible to say that our expectations regarding the energy consumption of the paths were met. It can be seen in the figures that the random 5 shortest algorithm generates better(less energy consumption) and more solutions than the other algorithms, as expected.

3.1 Experiment 2.b

The objective of this experiment is to repeat experiment 2.a but now using a greedy randomized algorithm instead of the random algorithm and take conclusions on the influence of the number of routing paths in the efficiency of the greedy randomized algorithm.

```
%Optimization algorithm with greedy randomized:
t= tic;
bestEnergy= inf;
allValues= [];
while toc(t)<tempo
    continuar= true;
    while continuar
        continuar= false;
        ax2= randperm(nFlows);
        sol= zeros(1,nFlows);
        for i= ax2
            k_best= 0;
            best= inf;
            % for 10 shortest
            % n = min(10,nSP(i));
            % for 5 shortest
            % n = min(5,nSP(i));
            for k= 1:n
                sol(i)= k;
                Loads= calculateLinkLoads(nNodes,Links,T,sP,sol);
                load= max(max(Loads(:,3:4)));
                if load <= 10
                    energy= 0;
                    for a= 1:nLinks
                        if Loads(a,3)+Loads(a,4)>0
                            energy= energy + L(Loads(a,1),Loads(a,2));
                        end
                    end
                else
                    energy= inf;
                end
                if energy<best
                    k_best= k;
                    best= energy;
                end
            end
            if k_best>0
                sol(i)= k_best;
            else
                continuar= true;
                break;
            end
        end
    end
    energy= best;
    allValues= [allValues energy];
    if energy<bestEnergy
        bestSol= sol;
        bestEnergy= energy;
    end
end
```

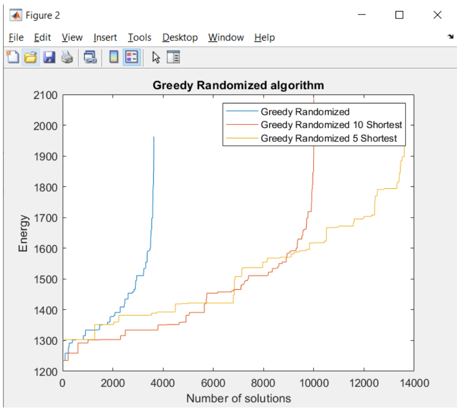
3.1.1 Results

In the greedy randomized algorithm a random order to the flows is choosen to select their routing paths and the greedy strategy is applied to the previously selected order(select the lesser energy consumption flow for each path).

In the greedy randomized 5 and 10 shortest algorithm the pool of paths to select from is reduced to the 5 and 10 shortest paths available for that flow respectively and then a random order for those flows is selected.

With this in mind we espected that the greedy randomized 5 shortest algorithm would produce the best results followed by the greedy randomized 10 shortest and finally the greedy randomized algorithm. This results are to be expected because if we reduce the available paths for each flowto the 5 or 10 shortest this would inevitably produce shortest and with less load complete paths.

The obtained results can be observed in the following figures:



(a) Graph for greedy randomized algorithm

```
GREEDY RANDOMIZED:
   Best energy = 1235.0
   No. of solutions = 3632
   Av. quality of solutions = 1394.9
GREEDY RANDOMIZED 10 SHORTEST:
   Best energy = 1235.0
   No. of solutions = 10000
   Av. quality of solutions = 1414.7
GREEDY RANDOMIZED 5 SHORTEST:
   Best energy = 1303.0
   No. of solutions = 13654
   Av. quality of solutions = 1511.8
```

(b) Greedy randomized algorithm results

After analyzing the results is possible to say that our expectations regarding the energy consumption of the paths weren't met for all cases, this means that we were expecting that the greedy randomized 5 shortest would produce paths with less load than the others, but this isn't valid because the greedy randomized 10 shortest produced paths with less load than the greedy randomized 5 shortest at least in the beginning. The greedy randomized also produced better paths in the beginning together with the greedy randomized 10 shortest, but after the middle of the graph de greedy randomized 5 shortest started producing more solutions to the same load making this algorithm better to find alternative paths with the same load.

3.2 Experiment 2.c

The objective of this experiment is to repeat experiment 2.a but now using a multi start hill climbing algorithm instead of the random algorithm and take conclusions on the influence of the number of routing paths in the efficiency of the greedy randomized algorithm.

```
%Optimization algorithm with multi start hill climbing:
t= tic;
bestEnergy= inf;
allValues= [];
contadortotal= [];
while toc(t)<tempo
    %GREEDY RANDOMIZED:
    continuar= true;
    while continuar
        continuar= false;
        ax2= randperm(nFlows);
```

```
            sol= zeros(1,nFlows);
            for i= ax2
                k_best= 0;
                best= inf;
                % for 10 shortest
                % n = min(10,nSP(i));
                % for 5 shortest
                % n = min(5,nSP(i));
                for k= 1:n
                    sol(i)= k;
                    Loads= calculateLinkLoads(nNodes,Links,T,sP,sol);
                    load= max(max(Loads(:,3:4)));
                    if load <= 10
                        energy= 0;
                        for a= 1:nLinks
                            if Loads(a,3)+Loads(a,4)>0
                                energy= energy + L(Loads(a,1),Loads(a,2));
                            end
                        end
                    else
                        energy= inf;
                    end
                    if energy<best
                        k_best= k;
                        best= energy;
                    end
                end
                if k_best>0
                    sol(i)= k_best;
                else
                    continuar= true;
                    break;
                end
            end
        end
        energy= best;

        %HILL CLIMBING:
        continuar= true;
        while continuar
            i_best= 0;
            k_best= 0;
            best= energy;
            for i= 1:nFlows
                % for 10 shortest
                % n = min(10,nSP(i));
                % for 5 shortest
                % n = min(5,nSP(i));
                for k= 1:n
                    if k~=sol(i)
                        aux= sol(i);
                        sol(i)= k;
                        Loads= calculateLinkLoads(nNodes,Links,T,sP,sol);
                        load1= max(max(Loads(:,3:4)));
                        if load1 <= 10
                            energy1= 0;
                            for a= 1:nLinks
                                if Loads(a,3)+Loads(a,4)>0
                                    energy1= energy1 + L(Loads(a,1),Loads(a,2));
                                end
                            end
                        else
                            energy1= inf;
```

```
                    end
                    if energy1<best
                        i_best= i;
                        k_best= k;
                        best= energy1;
                    end
                    sol(i)= aux;
                end
            end
        end
        if i_best>0
            sol(i_best)= k_best;
            energy= best;
        else
            continuar= false;
        end
    end
    allValues= [allValues energy];
    if energy<bestEnergy
        bestSol= sol;
        bestEnergy= energy;
    end
end
```

3.2.1 Results

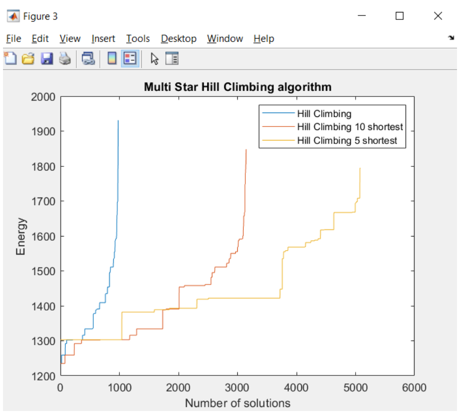
In the multi star hill climbing algorithm the program will iterate through the paths of the solutions until it finds the best solution at the moment (i.e. the path with less energy consumption).

In the multi star hill climbing algorithm 5 and 10 shortest algorithm the pool of paths to select from is reduced to the 5 and 10 shortest paths available for that flow respectively.

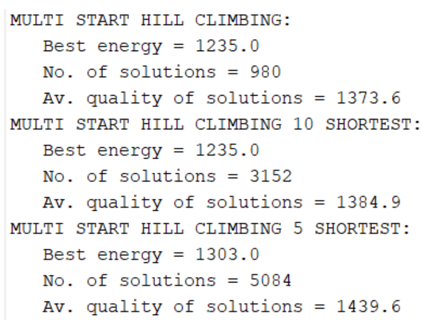
With this in mind we espected that the multi star hill climbing algorithm 5 shortest algorithm would produce the best results followed by the multi star hill climbing algorithm 10 shortest and finally the multi star hill climbing algorithm.

This results are to be expected because if we reduce the available paths for each flow to the 5 or 10 shortest this would inevitably produce shortest and with less energy consumption complete paths.

The obtained results can be observed in the following figures:



(a) Graph for multi star hill climbing algorithm



```
MULTI START HILL CLIMBING:
   Best energy = 1235.0
   No. of solutions = 980
   Av. quality of solutions = 1373.6
MULTI START HILL CLIMBING 10 SHORTEST:
   Best energy = 1235.0
   No. of solutions = 3152
   Av. quality of solutions = 1384.9
MULTI START HILL CLIMBING 5 SHORTEST:
   Best energy = 1303.0
   No. of solutions = 5084
   Av. quality of solutions = 1439.6
```

(b) Multi star hill climbing algorithm results

After analyzing the results is possible to say that our expectations regarding the energy consumption of the paths weren't met. The multi star hill climbing algorithm 10 shortest was the one that produced more solutions with less energy consumption followed by the multi star hill climbing algorithm and finally the multi star hill climbing algorithm 5 shortest. Only at the end of the graph the multi star hill climbing algorithm 5 shortest was better than the others by producing more solutions with the same load as the multi star hill climbing algorithm 10 shortest. With this we can conclude that multi star hill climbing algorithm 5 and 10 shortest can produce more solutions than the normal version, but the normal version can produce better or equal paths, this can be explained by saying that is not always worth taking the lesser energy consumption path in each iteration because in the end it might end up creating a bigger path than a path with some higher energy consumption paths in the middle.

3.3 Experiment 2.d

The objective of this experiment is to compare the efficiency of the three heuristic algorithms based on the results obtained in 2.a, 2.b and 2.c.

3.3.1 Results

By comparing all of the three heuristics is possible to say that the best one is the multi star hill climbing algorithm and the worst is the random algorithm. The multi star hill climbing algorithm for the same amount of time produces much less and better solutions than the random algorithm being much more efficient than it. The greedy randomized can also produce on average good solutions but is still less efficient than the multi star hill climbing but much more efficient than the randomized algorithm.

4 Task 3

Assuming that all routers are of very high availability (i.e., their availability is 1.0). Compute the availability of each link based on the length of the link assuming the model considered in J.-P. Vasseur, M. Pickavet and P. Demeester, "Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS", Elsevier (2004). In this task, the aim is to compute a pair of symmetrical routing paths to support each flow of the unicast service.

```
for i= 1:nFlows
    fprintf('Flow %d:\n',i);
    fprintf('   First path: %d',sP1{i}{1}(1));
    for j= 2:length(sP1{i}{1})
        fprintf('-%d',sP1{i}{1}(j));
    end
    if ~isempty(sP2{i}{1})
        fprintf('\n   Second path: %d',sP2{i}{1}(1));
        for j= 2:length(sP2{i}{1})
            fprintf('-%d',sP2{i}{1}(j));
        end
    end
    fprintf('\n   Availability of First Path= %.5f%%\n',100*a1(i));
    sum1 = sum1 + 100*a1(i);
    if ~isempty(sP2{i}{1})
        fprintf('   Availability of Second Path= %.5f%%\n',100*a2(i));
        sum2 = sum2 + 100*a2(i);
    end
end

fprintf('\nAverage Availability of First Path= %.5f%%\n',sum1/nFlows);
fprintf('Average Availability of Second Path= %.5f%%\n',sum2/nFlows);
```

```

% 1+1 protection
fprintf('\n1+1 protection:\n');
fprintf('Bandwidth on each direction of each link\n');
Loads= calculateLinkLoads1plus1(nNodes,Links,T,sP1,sP2)
fprintf('Total Bandwidth\n');
totalLoad= sum(sum(Loads(:,3:4)))

% links without enough capacity
bt = Loads(:,3);
bt_2 = Loads(:,4);

for i=1:nLinks
    if bt(i) > 10 || bt_2(i) > 10
        fprintf('Link %d without enough capacity\n',i);
    end
end

% 1:1 protection
fprintf('\n1:1 protection:\n');
fprintf('Bandwidth on each direction of each link\n');
Loads= calculateLinkLoads1to1(nNodes,Links,T,sP1,sP2)
fprintf('Total Bandwidth\n');
totalLoad= sum(sum(Loads(:,3:4)))

% links without enough capacity and the highest bandwidth value required
bt = Loads(:,3);
bt_2 = Loads(:,4);
maxLoad= max(max(Loads(:,3:4)));
for i=1:nLinks
    if bt(i) > 10 || bt_2(i) > 10
        fprintf('Link %d without enough capacity\n',i);
    end
end
fprintf('Highest Bandwidth required= %.5f\n',maxLoad);
```

4.1 Experiment 3.a

The objective of this exercise is to compute for each flow one of its routing paths given by the most available path.

4.1.1 Results

The obtained results can be seen in Figure 9.

4.2 Experiment 3.b

The objective of this experiment is to compute for each flow another routing path given by the most available path which is link disjoint with the previously computed routing path. Compute the availability provided by each pair of routing paths and finally present all pairs of routing paths of each flow and their availability and also present the average service availability (i.e., the average availability value among all flows of the service).

The obtained results can be observed in the following figure:

4.2.1 Results

```
Flow 1:
   First path: 1-2-3
   Second path: 1-5-4-3
   Availability of First Path= 99.65085%
   Availability of Second Path= 99.63803%
Flow 2:
   First path: 1-5-4
   Second path: 1-2-4
   Availability of First Path= 99.78969%
   Availability of Second Path= 99.73937%
Flow 3:
   First path: 2-4-5-7
   Second path: 2-3-8-7
   Availability of First Path= 99.75688%
   Availability of Second Path= 99.54719%
Flow 4:
   First path: 3-4
   Second path: 3-2-4
   Availability of First Path= 99.84802%
   Availability of Second Path= 99.78606%
Flow 5:
   First path: 4-8-9
   Second path: 4-5-7-9
   Availability of First Path= 99.75631%
   Availability of Second Path= 99.71684%
Flow 6:
   First path: 5-7-8-6
   Second path: 5-4-3-6
   Availability of First Path= 99.68533%
   Availability of Second Path= 99.64530%
Flow 7:
   First path: 5-7-8
   Second path: 5-4-8
   Availability of First Path= 99.77394%
   Availability of Second Path= 99.74175%
Flow 8:
   First path: 5-7-9
   Second path: 5-4-8-9
   Availability of First Path= 99.84980%
   Availability of Second Path= 99.62348%
Flow 9:
   First path: 6-10
   Second path: 6-8-9-10
   Availability of First Path= 99.94159%
   Availability of Second Path= 99.58959%

Average Availability of First Path= 99.78360%
Average Availability of Second Path= 99.66973%
```

Figure 9: Most available routing paths with respective availability

4.3 Experiment 3.c

Recalling that the capacity of all links is 10 Gbps in each direction. The objective of this experiment is to compute how much bandwidth is required on each direction of each link to support all flows with 1+1 protection using the previously computed pairs of link disjoint paths and to also compute the total bandwidth required on all links and finally registering which links do not have enough capacity.

The obtained results can be observed in the following figure:

4.3.1 Results

```
1+1 protection:
Bandwidth on each direction of each link

Loads =

    1.0000    2.0000    1.7000    1.5000
    1.0000    5.0000    1.7000    1.5000
    2.0000    3.0000    5.5000    4.9000
    2.0000    4.0000    5.5000    4.1000
    3.0000    4.0000    4.9000    4.3000
    3.0000    6.0000    1.2000    1.5000
    3.0000    8.0000    2.4000    1.5000
    4.0000    5.0000   10.8000   10.3000
    4.0000    8.0000    4.7000    6.6000
    5.0000    7.0000    8.3000    9.6000
    6.0000    8.0000    2.9000    2.8000
    6.0000   10.0000    1.4000    1.6000
    7.0000    8.0000    4.8000    6.4000
    7.0000    9.0000    2.6000    4.1000
    8.0000    9.0000    4.0000    5.7000
    9.0000   10.0000    1.4000    1.6000

Total Bandwidth

totalLoad =

  131.8000

Link 8 without enough capacity
```

Figure 10: 1+1 protection results

4.4 Experiment 3.d

The objective of this experiment is to compute how much bandwidth is required on each link to support all flows with 1:1 protection using the previously computed pairs of link disjoint paths and also to compute the total bandwidth required on all links and finally registering which links do not have enough capacity and the highest bandwidth value required among all links.

The obtained results can be observed in the following figure:

4.4.1 Results

```
1:1 protection:
Bandwidth on each direction of each link

Loads =

    1.0000    2.0000    1.7000    1.5000
    1.0000    5.0000    1.7000    1.5000
    2.0000    3.0000    3.4000    3.4000
    2.0000    4.0000    4.8000    3.6000
    3.0000    4.0000    3.9000    3.3000
    3.0000    6.0000    1.2000    1.5000
    3.0000    8.0000    2.4000    1.5000
    4.0000    5.0000    6.9000    5.6000
    4.0000    8.0000    4.7000    6.6000
    5.0000    7.0000    8.3000    9.6000
    6.0000    8.0000    2.9000    2.8000
    6.0000   10.0000    1.4000    1.6000
    7.0000    8.0000    4.8000    6.4000
    7.0000    9.0000    2.6000    4.1000
    8.0000    9.0000    2.6000    4.1000
    9.0000   10.0000    1.4000    1.6000

Total Bandwidth

totalLoad =

  113.4000

Highest Bandwidth required= 9.60000
```

Figure 11: 1:1 protection results

4.5 Experiment 3.e

The objective of this experiment is to compare the results of 3.c and 3.d and justify the differences.

4.5.1 Results

In the 1+1 protection the flow is sent duplicated by the two routes, this means that in this type of protection the recovery time of failures is very short but requests a lot of resources from the network.

In the 1:1 protection the flow is sent by one of the routes and the other route is only used for backup in case the main route fails, this means that the recovery time in failures is much bigger because the source node must be notified of the failure to start transmiting through the backup route. One advantage of this protection is that the backup route resources can be shared between other flows.

After analysing the results is possible to affirm that the 1:1 protection is the best one for this network because the highest required bandwidth is 9.6 and the network capacity is up to 10Gbps and the 1+1 protection requires more capacity of link 8 than what is provided for that link (10.8 $>$ 10 and 10.3 $>$ 10 (Gbps)). We can also conclude that the 1:1 protection makes the network require less link load which is to be expected(i.e. backup routes resources can be shared among other routes).