1- The Logical Topology Design Problem

1-1- Introduction

The Logical Topology Design (LTD) problem is one of the NP- hard problems and it is not easy to find an optimal solution for it. We will try to discuss the formulas of this problem and convert the formulas to Mosel language.

First of all, we have to answer this question: What is the goal of this lab?

The objective of this lab is to know the existing traffic between the nodes in a physical topology, we will try to minimize the maximum flow among the links by finding a logical topology between the nodes and try to minimize network congestion by routing the traffic on the logical topology.

1-2- Formalization

For solving any Np- hard problem, we have to answer these questions:

- What is the objective function?
- What constraints we have?
- What is the variables that we have?

and at the end

- What are the inputs?

For this problem, we have two different formulation. In the first one, the traffic can be split between different links, while in the second one splitting the traffic is not allowed.

1-2-1- Splitting is Allowed

In this section, we will explain some parameters and formulations related to LTD problem where splitting the flow is allowed, also we will write the related Mosel code after explaining the mathematical formulation.

- Input Parameters
  - Number of Nodes:
    - *Node = 1..8*
  - Number of lasers and photodiodes in every node: δ
    - *Delta := 3*
  - Traffic matrix: t^{sd}
    - *TRAFFIC: array(Node,Node) of real*
    - *for all (i,j in Node | i <> j) do*
      TRAFFIC(i,j):= 1+(random(10-1))*

- Variables
  - Existing light path between nodes i and j : $b_{ij} \in \{0,1\}$
    - *b_ij: array (Node,Node) of mpvar*
    - *for all (i , j in Node) b_ij(i,j) is_binary*
  - The flows between source (s) and destination (d) over light path between nodes i and j: f_{ij}^{sd}
    - *f_ij_sd: array (Node,Node,Node,Node) of mpvar*
    - *for all (i ,j,s ,d in Node) f_ij_sd(i,j,s,d) is_semcont 0*
  - The total flows on the light path between nodes i and j : f_{ij}

- f_ij: array (Node,Node) of mpvar
- forall (i , j in Node) f_ij(i,j) is_semcont 0

- Topology constrains:
  - Total flows going out from i to j (#lasers) :
    - for all (i in Node) sum (j in Node) b_ij(i,j) <= Delta $\sum_j bij \leq \delta o$
  - Total flows from i entering to j (#photodiodes) :
    - for all (j in Node) sum (i in Node) b_ij(i,j) <= Delta $\sum_j bji \leq \delta i$

- Flow constrains:
  - Total traffic entering to the d is equal to the traffic between s and d and total flows exiting from d is zero.
    - for all (s in Node, d in Node | s<>d) sum (j in Node| j <> d) f_ij_sd(j,d,s,d) =TRAFFIC(s,d)

      $$\sum_j f_{jd}^{sd} = t^{sd}$$

    - for all (s , d in Node | s<>d) sum(j in Node | j <> d) f_ij_sd(d,j,s,d) = 0

      $$\sum_j f_{dj}^{Sd} = 0 \quad \forall i$$

  - Total traffic exiting from s is equal to the traffic between s and d and total flows entering to the s is zero.

    $$\sum_j f_{sj}^{sd} = t^{sd} \qquad \sum_j f_{js}^{sd} = 0 \quad \forall i$$

    - for all (s , d in Node | s<>d) sum (j in Node| j <> s) f_ij_sd(s,j,s,d) =TRAFFIC(s,d)
    - for all (s , d in Node | s<>d) sum (j in Node | j <> s) f_ij_sd(j,s,s,d) = 0
  - The total traffic exiting from j is equal to the total traffic entering to the j

    $$\sum_j f_{ij}^{sd} - \sum_j f_{ji}^{sd} = 0 \quad \forall i \neq sd, s, d$$

    - forall (s , d , i in Node | i<>s and i<>d and s<>d) sum(j in Node|j<>s and j<>i and j<>d) (f_ij_sd(i,j,s,d) -f_ij_sd(j,i,s,d))=0
  - The flow on the light path between nodes i and j should be greater or equal to zero

    $$f_{ij} \geq 0 \quad \forall i, j$$

    - forall (i , j in Node | i<>j) f_ij(i,j) >=0
  - The flow exiting from s to d should be greater or equal to zero

    $$f_{ij}^{sd} \geq 0 \quad \forall i, j, s, d$$

    - forall (s ,d,i,j in Node | i<>j and s<>d) f_ij_sd(i,j,s,d)>=0
  - The maximum flow should be greater or equal to zero
    - $f_{max} \geq 0$

- Feasibility constraints:
  - The traffic can only path from the links that exist, otherwise the flow will be zero

    $$f_{ij}^{sd} \leq b_{ij} \times t^{sd} \quad \forall i, j, s, d$$

- forall (s ,d ,i ,j in Node | i<>j and s<>d) f_ij_sd(i,j,s,d)<=b_ij(i,j)*TRAFFIC(s,d)

- Objective function:
  - Minimizing maximum flow $\min f_{max}$
  - ObjectiveFunction := fmax
  - minimize(ObjectiveFunction)

1-2-2- Splitting is Not Allowed

The above formulas are not realistic since the flow can be split. For the other case where flow splitting is not allowed we have to change and add following formulas which will make our problem more complex:

- Variables
  - $f_{ij}^{sd} \in \{0,1\}$
    - forall (i ,j ,s ,d in Node) f_ij_sd(i,j,s,d) is_binary

- Constraints:
  - $f_{ij} = \sum_{sd} t^{sd} \times f_{ij}^{sd}$
    - forall (i , j in Node | i<>j) f_ij(i,j)= sum(s , d in Node | s<>d) TRAFFIC(s,d) * f_ij_sd(i,j,s,d)
  - $\sum_j f_{sj}^{sd} = 1$
    - forall(s , d in Node | s<>d) sum(j in Node| j <> s) f_ij_sd(s,j,s,d) = 1
  - $\sum_j f_{jd}^{sd} = 1$
    - forall (s , d in Node | s<>d) sum(j in Node| j <> d) f_ij_sd(j,d,s,d) = 1

- Feasibility constraints:
  - $f_{ij}^{sd} \leq b_{ij} \qquad \forall i,j,s,d$
    - forall (s ,d ,i ,j in Node | i<>j and s<>d) f_ij_sd(i,j,s,d)<=b_ij(i,j)

1-3- Solving the problem for N = 10 and Δ = 3

In this section, we are going to analyze the LTD problem for both formulations when the number of nodes is equal to 10 and the delta is restricted to 3. We will Consider a uniform traffic matrix in which the traffic sent from any source to any destination is a uniform random variable in the range [1,10] and the maximum timeout for the solver is set to10 minutes.

The result is shown in figure1, as we can see in the chart, by changing the seeds the value of the Fmax changes slightly and for both formulations the best result has been found when the seed is equal to 8. For this test we set the maximum timeout to 10 minutes.

The numerical results are demonstrated in table1. When the splitting is allowed, the solver can find the best solution very fast, but in the other case it takes 10 minutes for the solver to find at least one good solution. By having this results, we can understand that this problem belongs to the NP-hard problems.

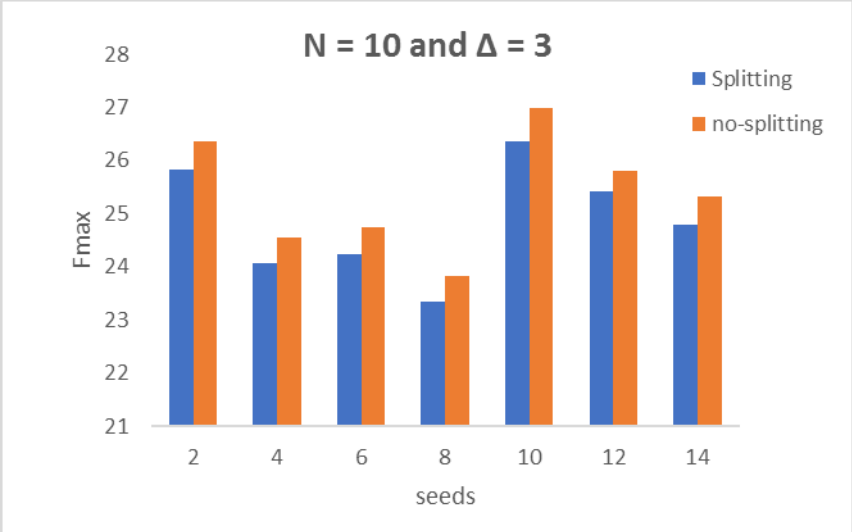

Figure 1: Fmax and seed

Seeds	2	4	6	8	10	12	14
Splitting -time(s)	3.5	2.6	7.5	3.2	3.6	3.9	3.5
no-splitting-time(s)	600	600	600	600	600	600	600

Table 1: Fmax and times

1-4- Solving the problem when Δ = 3

In this section, we are going to consider the LTD problem in both scenarios when there are different number of nodes (4 to 20) and the delta is fixed to 3. We are going to plot and comment average, minimum, maximum values for computation time and gap by running the program with different random seeds from 1 to 5. The maximum time is fixed to 10 minutes.

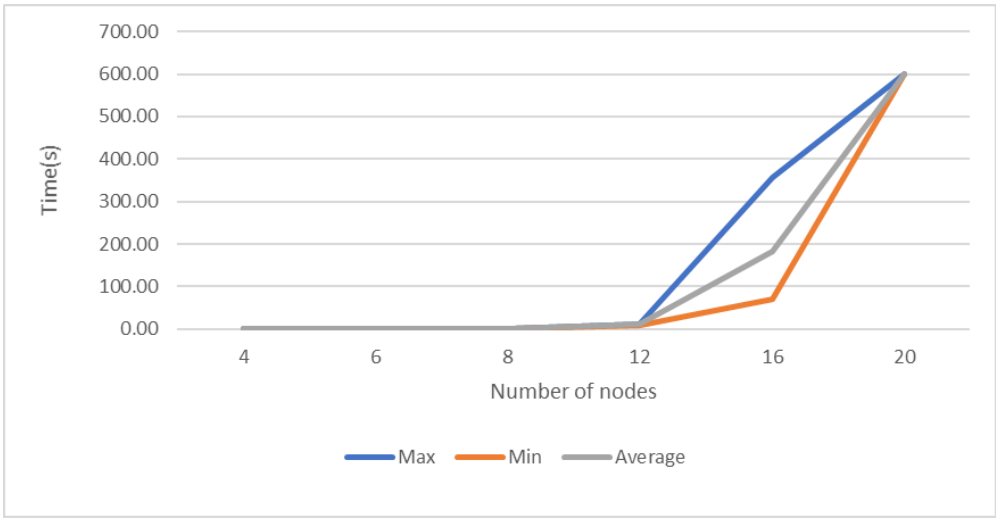

Figure 2: Max-Min-Avg of time splitting is allowed

Graphs in figure2 and figure3 show the results of maximum, minimum and average when splitting the flow is allowed. By changing the number of the nodes, the time that the solver needs to run the pro-

gram, increases. By this result we can understand that by having higher number of nodes, the complexity of the problem will increase, and it is going to take a longer time for finding the solution, and we can see that we have outlier when the number of the nodes is equal to 20.

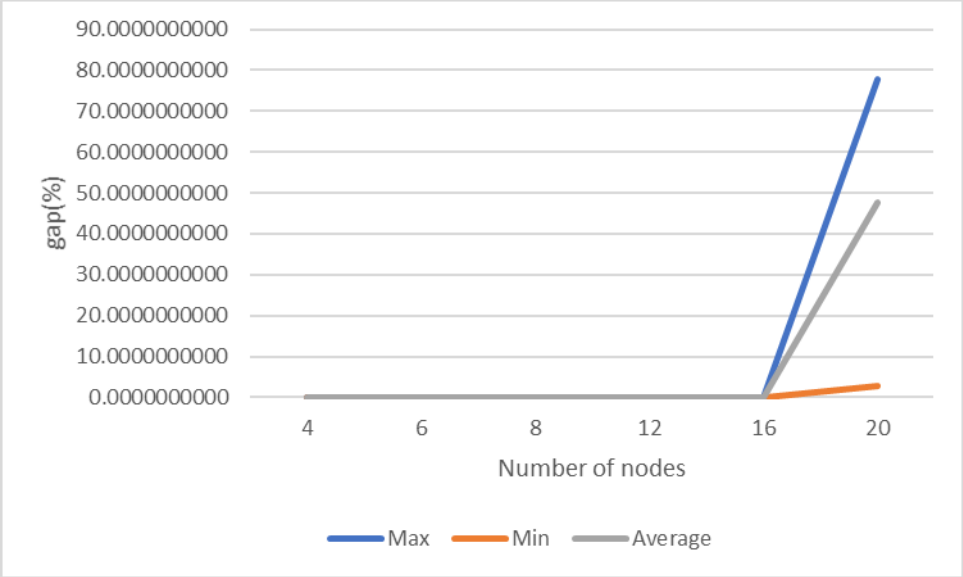

Figure 3: Max-Min-Avg of gap splitting is allowed

Now we are going to consider the same situation for running the code when the flow splitting is not allowed.

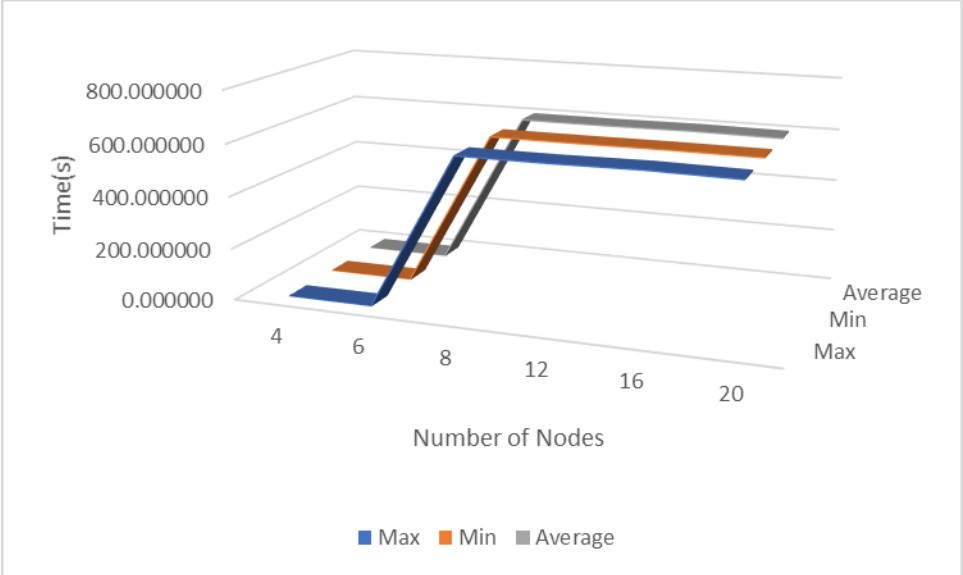

Figure 4: Max-Min-Avg of time splitting is not allowed

As you can see in figure 4 and 5 when we consider the splitting is not allowed, the problem becomes more complex and by increasing the number of nodes the time and the gap will increase. Consequently, when the number of nodes are 8 or more, the solver cannot find the optimal solution in given a time.

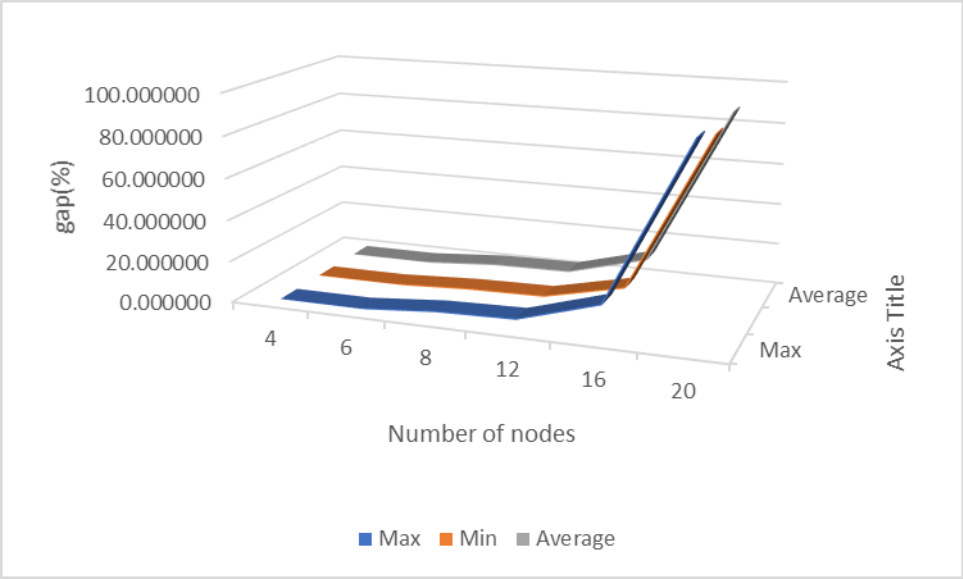

Figure 5: Max-Min-Avg of gap splitting is not allowed

By increasing the number of nodes the complexity of the problem is incremented and consequently, the time needed to solve the problem is increased. The effect of mentioned case can be seen clearly in the above figures.

1-5- Solving the problem when N = 8

In this part, the same experiment is repeated when there are different numbers of delta (1 to 5) and the number of nodes is fixed to 8. We are going to plot and comment average, minimum and maximum values for computation time and gap by running the program with different random seeds from 1 to 5. The maximum time is fixed to 10 minutes.

First, when splitting the flow is allowed, we take a look at the computed maximum, minimum and the average values for both time and gap.

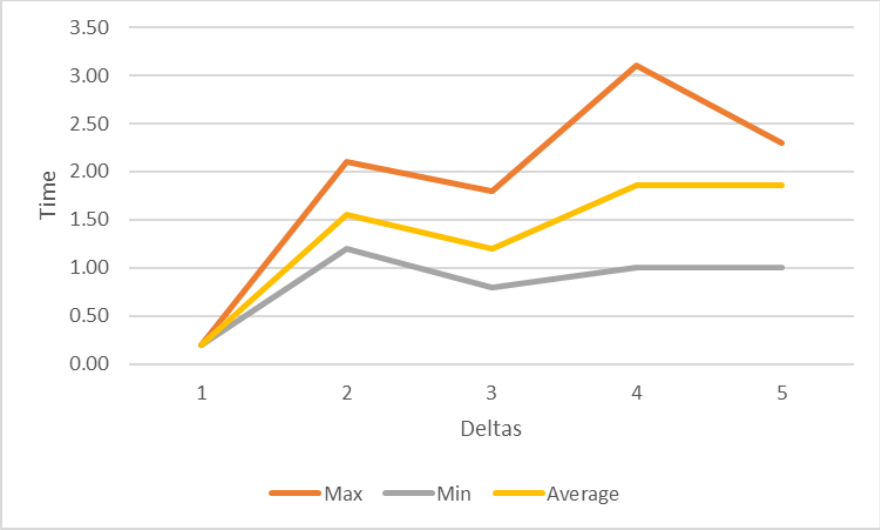

Figure 6: Max-Min-Avg of gap splitting is allowed

By looking at figure6 we can realize that when the value of delta increments, the time that we need to find the optimal solution will increase. But we can see that we have a pick when the delta is even, it shows that in this case the problem in more complex. And by looking at figure7 we can see that we have same situation in gap when we increase the delta but the value of the gap is very low, which means that Xpress provides us an optimal solution in this case.

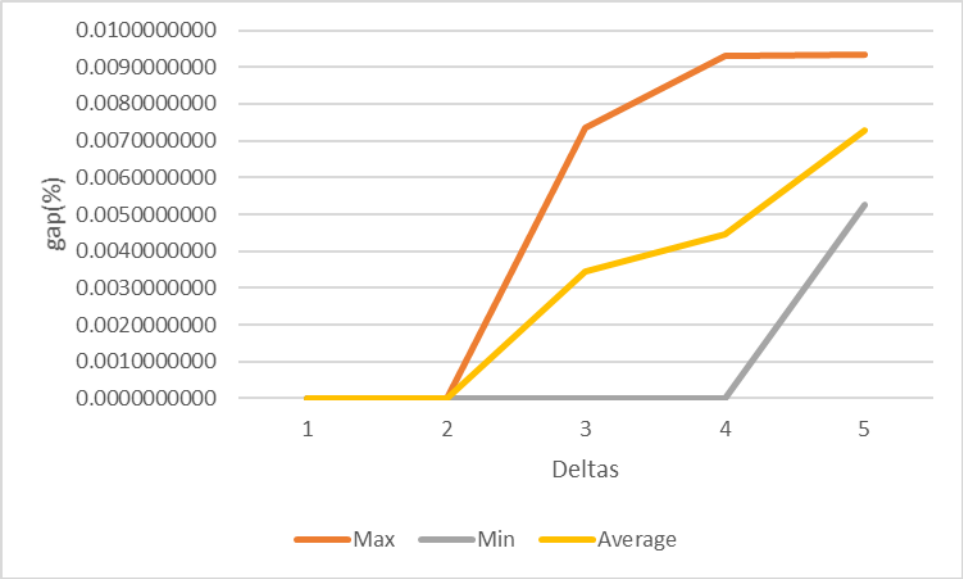

Figure 7: Max-Min-Avg of gap splitting is allowed

Then we repeat this part with same situation for the case when the flow splitting in not allowed. Line the previous time by increasing the delta we are expected to see that the time and gap will increase. The graph in figures 8 and 9 will show this. When the delta is equal to 1, the Xpress can find an optimal solution but when we increase the delta the problem become more complex and time that we sat for the running the problem will be over, but in the case that the delta is equal to 4 when the random seed is equal to 1 or 3 luckily it can find the best solution for our problem.

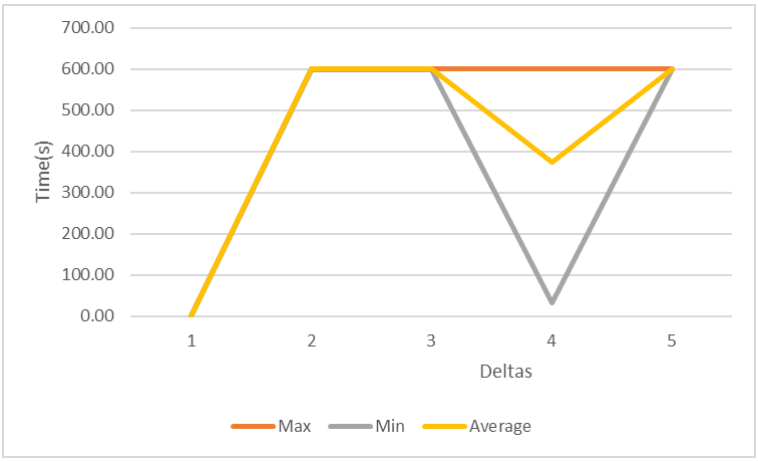

Figure 8: Max-Min-Avg of time splitting is not allowed

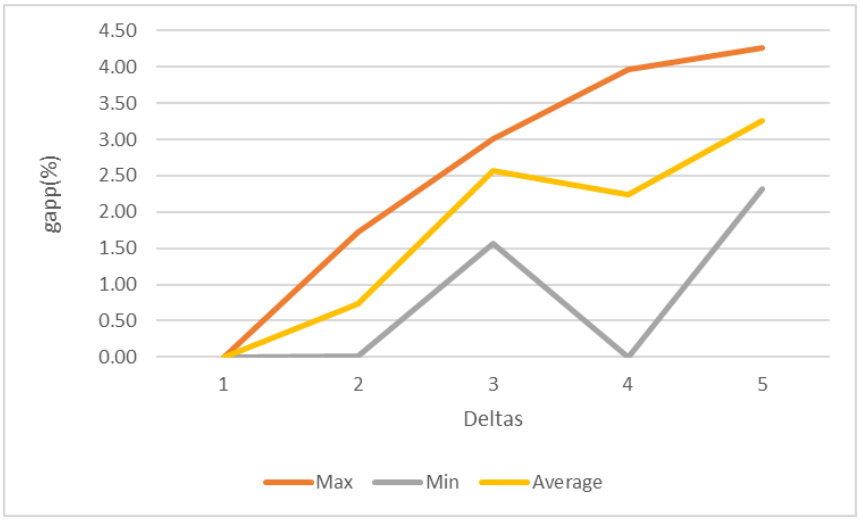

Figure 9: Max-Min-Avg of gap splitting is not allowed

By increasing the Delta, more logical topologies will be created which leads to having more transmitters and receivers and consequently more complicated topology which results increasing the calculation time, while decreasing in Fmax.

1-6- Manhattan Topology

In this part of our experiment we have to put the number of nodes equal to 16 and the number of delta equal to 4. We have to restrict our topology to bidirectional Manhattan topology and the traffic matrix is created with a random uniform distributed in range [1,10]. The traffic can be sent from any source to any destination. In figure 10 you can see the result for Fmax and in table 2 you can see the time that it takes to run.

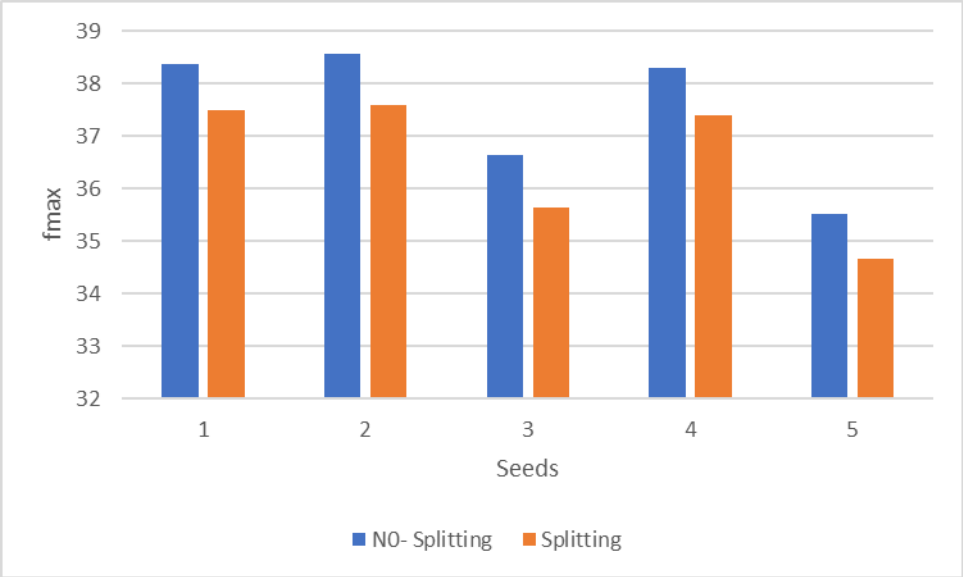

Figure 10: Fmax and seed

As we can see in figure 10 like the other topology that we tested before, the value of Fmax in not splitting mode is higher than the other one in all seeds, and if we compare the times, we can see that the Xpress can find the best solution easily in the case that the splitting is allowed, but in the other one it can find only one optimal solution in 10 minutes.

seeds	1	2	3	4	5
No-Splitting, time(s)	599.5	599.8	599.3	599.9	600.1
Splitting, time(s)	0.3	0.3	0.3	0.3	0.3

Table 2: Fmax and times

1-7- Aggregated flows f_{ij}^{s}

In this part, when flow splitting is allowed, we have to reformulate our problem by using aggregated flows. The result can be seen in figure 11.

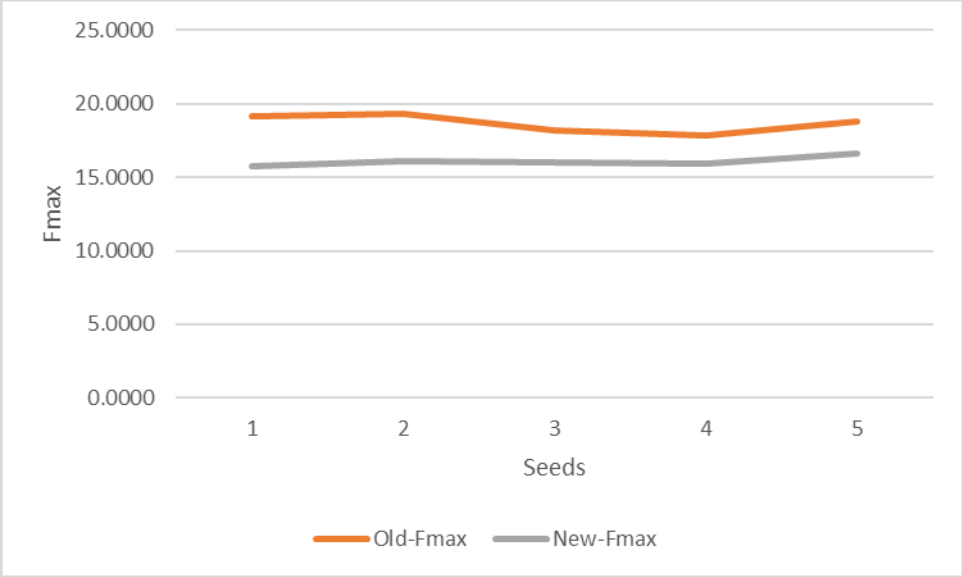

Figure 11: Aggregated flow

We tested the new formulation with old one only when the number of nodes is equal to 10 and the delta is equal to 3. As you can see in the graph, the aggregated formulation can find the better result but there is no big differences.

1-8- Unbalanced traffic matrix

For this part, we are going to change our traffic matrix to an unbalanced one and the new traffic matrix can be generated by using these formulation:

tsd=Uniform [10,20] for (1< s ≤ N/2, 1< d ≤ N/2) and (N/2 < s ≤N, N/2 < d ≤N)

tsd=Uniform [1,2] for (1< s ≤ N/2, N/2< d ≤ N) and (N/2< s≤ N, 1< d ≤ N/2)

Moreover, we are going to repeat our experiments with the situations that we had before in sections 1-4 and 1-6 and compare the results.

For the first part, we are going to set the delta to 3 and change the number of nodes from 4 to 20 with different random seeds between 1 and 5, and at the end we will calculate the minimum, maximum for time and gap.

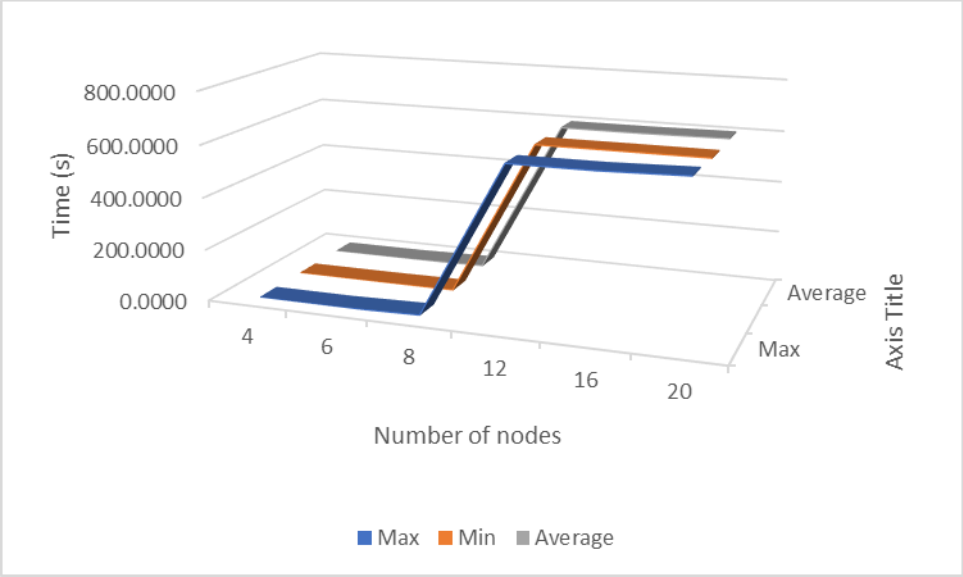

Figure 12: Max-Min-Avg of time splitting is allowed with unbalanced traffic matrix

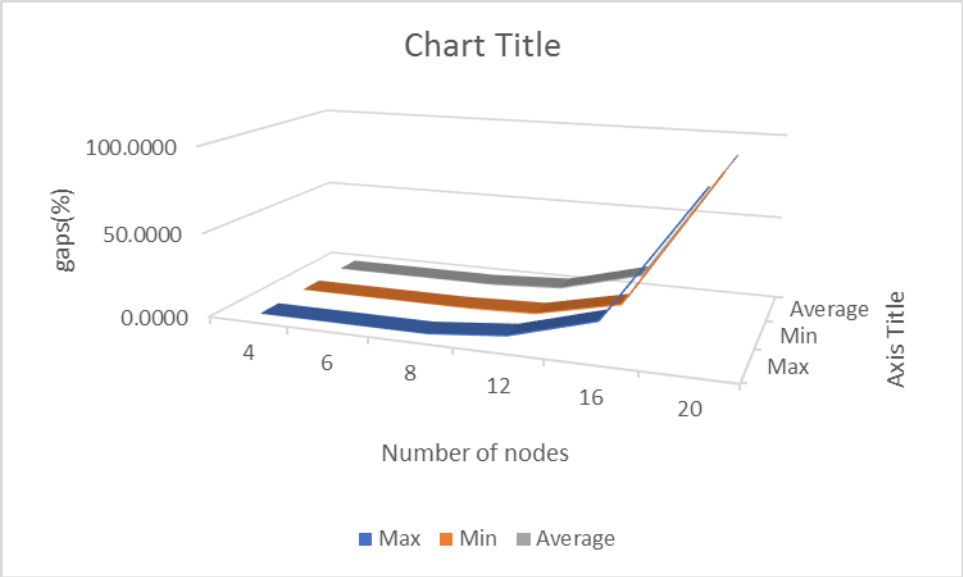

Figure 13: Max-Min-Avg of gap splitting is allowed with unbalanced traffic matrix

As you can see in figures 12 and 13, by increasing the number of nodes the time and the gap will increase. Especially when the number of nodes are greater than 8. This is different from the result that we have in section 1-4. In the previous case the only outlier that we had was when the number of nodes was equal to 20. So, we can conclude that by having unbalanced traffic matrix the problem will become more complex and it takes more time for Xpress to find at least one lower bound for our problem.

Now if we compare the result with the result that we found in section 1-6 we will see that as we expected, the value of Fmax is higher in the unbalanced case because of the traffic that we have. And this is true for both case where the splitting the flow is allowed or not.

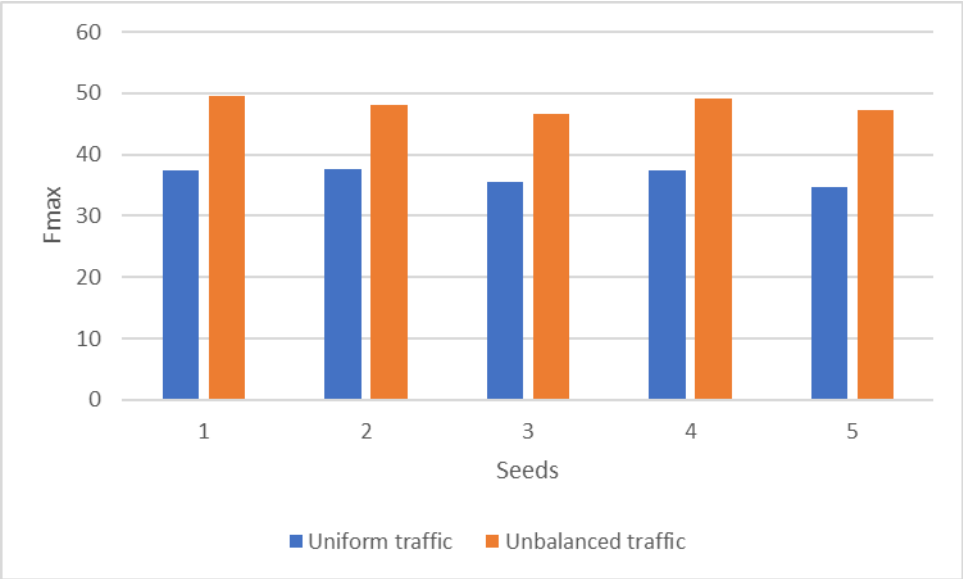

Figure 14: Fmax – Balanced vs Unbalanced traffic splitting is allowed

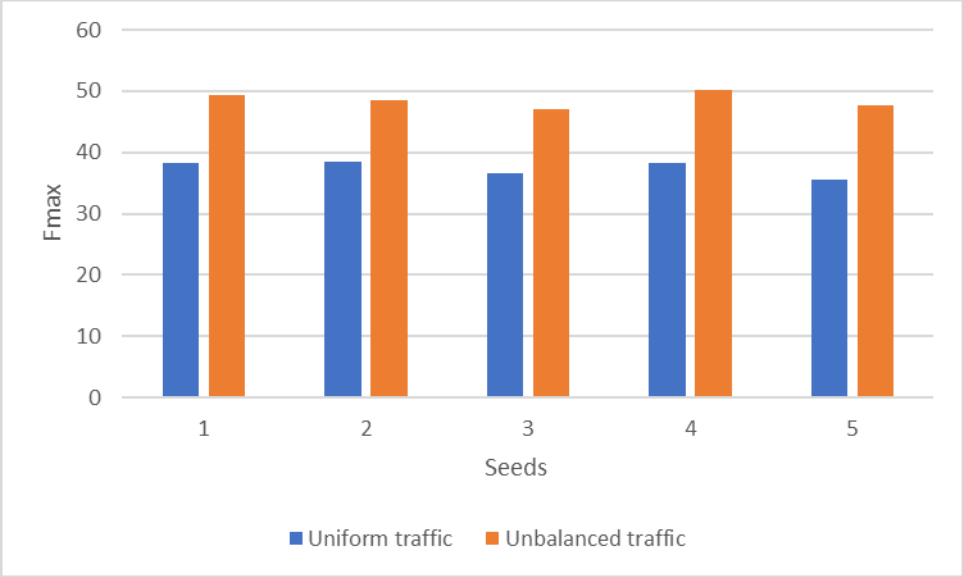

Figure 15: Fmax – Balanced vs Unbalanced traffic splitting is not allowed

2- Greedy heuristic For LTD Problem

2-1- Introduction

In this lab, we are going to design a greedy heuristic algorithm for solving the LTD problem which we saw in the previous section. We used python language for this implementation.
First, what is the greedy heuristic approach?
The greedy heuristic algorithm is an algorithm that is capable of solving NP- hard problems. For interpreting how it works, roughly we can say that in every state, first it selects a locally optimal choice expecting that it finds the best solution in this way and after making the choice, it will not return to that state. So, it works fast but there is a possibility of getting stuck in a local optimal instead of discovering the global optimal solution for the given problem.

2-2- Proposed Greedy Heuristic

Our proposed algorithm works as follows:
First the user defines the number of nodes, deltas and the number of simulations to repeat. Then, it generates a uniform random traffic matrix in the range [0.5, 1.5], and by employing this matrix it creates a vector. Each element of this vector has a sub vector which includes: the source, destination and the traffic that is exchanged between them.
Then in the next step, the vector is sorted in a decreasing way. By taking the first element of the vector, if source has a laser available and destination has a phototroph available, it creates the link between source and destination (This is for the case that source and destination are directly connected), but when there is no direct link between source and destination, it should find the multi-hop path on the logical topology.
The traffic is routed following this procedure: first it checks the neighbors of source to see if they have a direct link to the destination, if it finds a link between the neighbor and the destination it adds the flow on that related link, else it goes to the neighbor and checks its neighbors to find the destination and repeats this procedure in a recursive mode to find the destination.
However, there is a possibility that the number of nodes, number of destinations and the complex topology does not grant the algorithm to find the way. In this case, the whole procedure is restarted.
The stopping condition is applied when all the elements in the vector has been considered. The algorithm will repeat these steps for a number equal to the number of simulations that the user has defined and at the end it will print the average for all “Fmax”s that is found during the running.

2-3- Check with Random Input

For evaluating the proposed algorithm, we have to compare our algorithm with the random approach. The algorithm that works in the random mode, selects an element from the vector that contains source, destination and flows randomly and then tries to find the path between all available links.
In table 3 you can find the Fmax for different number of nodes and deltas that are calculated with our proposed approach and the random one. The number of simulations here is equal to 100. As you can see in the table, our algorithm operates better in most of the cases except the

cases with a red color. But by increasing the number of the nodes the result that our algorithm prepare improves.

			Greedy							Random				
Deltas	2	3	4	5	6	7	8	2	3	4	5	6	7	8
Nodes														
4	3.2289	1.4248	1.4318	1.4317	1.4239	1.4328	1.4149	3.6443	1.415	1.415	1.4243	1.4146	1.4224	1.4241
6	9.4099	5.6246	3.0766	1.4683	1.466	1.4706	1.4659	10.2811	6.6142	3.9633	1.4645	1.4584	1.4682	1.464
8	20.0574	12.6457	8.682	5.2793	3.0099	1.4856	1.4862	21.7573	15.3392	10.7075	7.0623	3.9939	1.4832	1.4817
10	35.2529	25.7362	18.2882	12.4998	8.5516	5.3057	3.0288	37.3974	28.8132	21.7354	16.1879	11.5184	7.2255	4.0968
12	53.2456	41.6123	32.0712	23.8471	17.7413	12.6227	8.4198	60.35727	45.6248	35.9665	28.7875	21.9316	17.4874	11.8198
16	105.606	88.9087	75.0104	58.9548	49.2134	37.1142	31.7842	110.1166	98.7342	79.8796	71.4216	56.2776	47.7079	39.5617
20	175.6724	165.3003	137.383	116.8068	95.0439	78.4025	68.2391	186.957	169.1529	150.339	123.8168	113.7921	95.3916	83.3952

Table 3: Greedy Algorithm vs Random One

2-4- Unbalanced Traffic

In this case we have an unbalanced traffic matrix. The traffic that is exchanged among the nodes can belong to two possible classes:

Low traffic: tsd=Uniform [0.5,1.5]

High traffic: tsd =Uniform [5,15]

And we are going to consider that 10% of the traffic demands belongs to the high-traffic class.

Table 4 shows the achieved results, Fmax in this table is average of 20 simulations. Again, like before our algorithm works better in most of the cases but in some cases random one works better. Furthermore, it can be seen that by increasing the number of nodes, the result of our algorithm becomes better. This happens since by increasing the number of the nodes, random algorithm has a lower chance to start from good point in the beginning.

			Greedy							Random				
Deltas	2	3	4	5	6	7	8	2	3	4	5	6	7	8
Nodes														
4	9.6061	10.0549	8.5067	8.2779	7.4752	8.2684	9.018	8.9259	8.7256	9.0334	7.2878	9.6707	8.0411	6.7137
6	16.1585	15.3639	11.3997	13.1045	11.546	11.4459	12.1309	28.02407	18.182	15.74	12.1912	12.1807	11.1683	10.3912
8	30.9299	19.7953	15.8319	13.944	12.7819	13.6941	13.0577	45.4803	35.3349	26.0431	19.8328	14.9566	13.3286	13.6498
10	48.8849	32.4379	23.3953	17.4319	15.5045	14.9943	14.562	84.1663	61.6549	44.0219	37.0927	26.9516	19.517	15.0705
12	73.9853	50.5986	36.298	28.2463	22.3014	17.6219	16.254	113.6631	91.1145	79.5427	58.8572	40.75024	37.5163	28.0188
16	156.3528	105.8579	79.6099	67.8489	52.5615	40.7253	34.3676	210.1673	196.5059	159.0685	139.6198	119.5164	96.0126	75.1551
20	231.3791	184.2523	149.5352	121.8994	105.1513	85.2849	72.0783	371.0533	320.9036	294.3924	257.9995	207.6905	184.249	177.9417

Table 4: Greedy Algorithm vs Random One with unbalance traffic

2-5- Manhattan Topology

In this part, we are going to design a new greedy algorithm where the topology is a bidirectional Manhattan and nodes are smartly placed. Our algorithm works in this way:

After creating the traffic matrix, it calculates the amount of traffic that each node is going to exchange (Both incoming and outgoing traffic). Now it creates a vector that each element of it has sub elements. In each sub element, you can discover the total traffic that the node is going to transfer and the also node number. Then, it sorts the vectors in a decreasing way. Starting from the first element of the vector, it finds a free place and inserts it in Manhattan topology where there is a highest free neighboring place. Next, for those free places in the neighborhood it finds and inserts the nodes with the highest amount of traffic. After injecting all the nodes in Manhattan, the algorithm will stop. This algorithm tries

to put flows with higher values near each other. We ran this algorithm for 1000 iterations and the average for Fmax is calculated equal to 13.7471764629.

2-6- Improving the Algorithm

For enhancing the algorithm that we introduced in the previous step, we decided to implement the simulated annealing algorithm. It is a metaheuristic to approximate global optimization in a large search space. The method models the physical process of heating a material and then slowly lowering the temperature to decrease defects, as a result, minimizing the system energy. It sets the big number as a temperature and in each step, it reduces the temperature and then it stops when the temperature reaches to zero. In each annealing simulation we find the new solution for the problem and it decides to keep the new solution or not. It also decides to keep the solution with the probability and reaped the experiment until reaching to the stopping condition. These probabilities help the algorithm to avoid getting stuck in the local optimal and help it to find the global optimum.

Our approach to work in these way, is that first, after running the algorithm in steps two until five and reaching to the result, it randomly swaps two nodes and recomputes the Fmax, every time that it algorithm finds a better result it saves it in the "best result" variable, and continues the procedure to reach to the stopping condition. If it couldn't find a better result, it accepts the result by a probability that it is calculated every time with respect to the temperature that is stored in the system, the probability is calculated in this way:

$$P = \exp((fmax - new\_fmax) / temperature)$$

And the temperature decreases by the rate of 0.003.

In table 5 you can find the results for different runs with different temperatures.

Temperature	FirstFmax	Second Fmax
10	13.9024	13.6014
100	14.5822	13.0319
1000	13.3375	12.3579
10000	14.0385	12.6124
100000	13.7175	12.8794
1000000	14.4027	12.4669

Table 5: simulated annealing vs greedy

As you can see in the table, our simulated annealing approach can find a better solution every time. Generally, we can say that by increasing the temperature we can reach to better results.

3- Green Network

Energy consumption is an aspect which is becoming very important in ICT and in particular in network developing. In fact a lot of energy is required not only to create the network elements, but also to keep them turned on. As a result, ISPs are investing a lot of resources in "Green Network" developing.
Optimize the power consumption of a network is the objective of this laboratory.

3-1- Power consumption in a network

In a network the power is consumed by links and nodes, the power consumed by nodes is much higher than the one consumed by links.
The power consumed by the elements is given by 2 parameters: one is constant in time and is the power consumed if the element is on, the second one depends on the flow that pass through the element.
As we can see in Figure 16, the main part of the energy consumption is given by the first parameter; so the main objective is trying to turn off as many devices as possible and optimize the on-devices use.

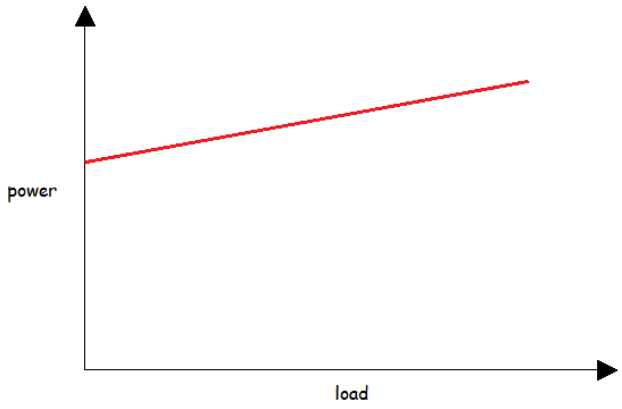

Figure 16: Power consumed by loads

Now we can introduce the formulations:
Regarding the nodes, the total power consumption is given by the summation of the following terms:
1. $PNode = \sum_{i=1}^{N} Pny_i;$
2. $PNodeFlow = \sum_{j=1}^{N} 2P_{flow} f_{ji} y_i;$

The first one in not related to the loading, "Pn" gives the power that the "router i" needs in order to be on. y is a binary number, its value is 1 if the "router i" is turned on, otherwise it is 0. The second term is instead connected with the loading of the nodes, in particular, the parameters on this formulation have the following meaning:

- f_{ji} is the flow directed to the node i, it has to be multiplied by 2 in order to consider the power consumed by managing the flow: it may be sent towards another node or, if it is the addressed, handled internally;
- P_{flow} is a constant that adapts the flow into power;
- Again y is a Boolean, 1 if the router i is on, 1 otherwise.

Now we need to analyse the formulas connected to links, again it should be made by two terms:

1. $PLink = \sum_{i=1}^{N}\sum_{j=1}^{N} \quad x_{ij} Pl_{ij}$
2. $PLinkFlow = \sum_{i=1}^{N}\sum_{j=1}^{N} \quad f_{ij} P_{flowlink}$

The first formula considers if link(ij) is turned on or not. If it is, its consumption is given by Pl, where this value may be different for each link; theoretically it depends on different parameters including length, speed and capacity. In our simulation we use two different methods to fix these values, we are going to talk about in the next paragraph.

PLinkFlow is the term connected with the flow on the link, the idea is explained for PnodeFlow, but in this case we are considering the flows on links and not the nodes.

So the power consumption of a network is given by the sum of two terms, one depends on the flow (the sum of PnodeFlow and PlinkFlow) and the other one depends only on on/off status (sum of Pnode and Plink).

3-2- Optimization

In order to obtain different results, we tried to run the simulation in "Mosel" with different topologies and parameters; first we considered a traffic matrix in which all nodes generate or receive traffic. In this situation, obviously we cannot turn off nodes (the traffic matrix is randomly created). This type of scenario has been tested with two different topologies with different number of nodes:

A random mesh and a full mesh, both with 12, 15, 18 and 20 nodes. In figure 1 we have the constraints of the model.

$$\sum_{j=1}^{N} f_{ij}^{sd} - \sum_{j=1}^{N} f_{ji}^{sd} = \begin{cases} t^{sd}, & \forall s,d,i = s \\ -t^{sd}, & \forall s,d,i = d \\ 0, & \forall s,d,i \neq s,d \end{cases}$$

$$f_{ij} = \sum_{s=1}^{N}\sum_{d=1}^{N} f_{ij}^{sd} \qquad \forall i,j$$

$$f_{ij} \leq \alpha c_{ij} x_{ij} \qquad \forall i,j$$

$$\sum_{j=1}^{N} x_{ij} + \sum_{j=1}^{N} x_{ji} \leq M y_i \qquad \forall i.$$

We described above the meaning of fij, but we need to spend few words on xij, yi, alpha and cij:

- xij is a matrix of binary numbers: 1 if the link ij is on, 0 otherwise; it should be different from the initial topology because in the optimization some links should be turned off.
- cij is the capacity of the topology.
- yi is a binary array; 1 if the node i is on, 0 otherwise;
- alpha is an overall parameters it gives the percentage of links capacity that can be used, its value is (0,1).

The computation time in our simulation was 300s, so the information about the gap with the optimal solution refers to this value. We used this value because we noted that the differences of gaps were very small with a computation time of 600s.

3-2-1- Random topology

This topology has been created by rounding a number $i = k + 0.1\ where k \in (0; 1)$ into 0 or 1 (1 if the link exists, otherwise not), we have chosen to add the value 0.1 in order to permit traffic flow from and towards each node. Following this idea we obtain that around the 60% of possible links exist. Using this topology we fixed two settings:

- Different parameters of power consumption: for each node and link we assigned a particular power consumption, Pnode from 200 to 250 and Plink depending on the capacity, in particular multiplying it by 0.05; since the capacity on a link is randomly chosen from 200 to 250, Plink:(10,12,5);
- Parameters equal and constant: Plink=11, Pnode=200.

All these values are chosen totally randomly and they do not have any relationship with real values. In this model we used some constant values: Alpha=0.8, Pflow=0.01 and Pflowlink=0,001. We chose this values in order to make flow dependent components very lower than the others. The capacity of the links was randomly set (cij (200,250)). Since the values of the two different settings are similar we obtained similar results. In particular after the optimization we had that a part of the links were turned off.

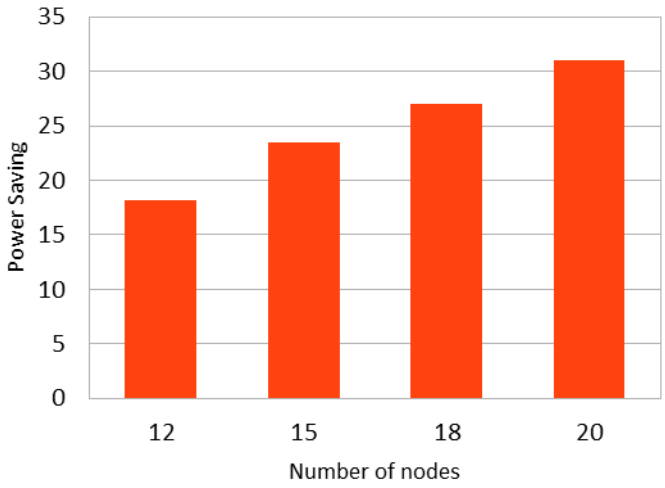

Figure 17: Power saving according to the number of nodes

We obtained that the number of links which were on were the 20%-30% of the initial links, this result brings a great power saving.

Comparing the result, we note that the on links were almost the same in the two configurations, with a little difference in terms of energy given by the randomly choice of the power. This optimization was tried with different random seeds and the results were very similar. Obviously the traffic increases with the number of nodes and so consequently the power consumed connected with it increases.In the graph in Figure 17, we can see the mean energy saving (in %) for different number of nodes.

The problem is an NP-hard problem but we obtained good values of gap from the optimal solution, about the 5% for different seeds and different number of nodes.

3-2-1-1- Considering alpha and capacity

This type of result is strongly connected with the alpha parameter and the capacity of the links. In fact, reducing these two variables significantly, we could not have feasible solutions or a much lower power saving because the model needs more links to allow the traffic flow between nodes. If we have a big capacity we can route a large part of the flows on few links and turn off the others.

3-2-2- Full mesh topology:

All nodes are connected to each other and the parameters were set in the same as the previous topology.

The values of power consumption are very similar to the previous one; because the traffic matrix was made in the same way and the consumption of the elements are roughly the same. The main difference is the comparison with the initial topology. In fact, a full mash topology has a larger power consumption than the random one because there are more links exist. So energy saved is much higher because there are a larger number of turned off links. Obviously, the full mesh is stronger than the previous topology since it supports a traffic matrix with larger values; moreover, in this way it is possible to choose a lot of different paths for the flows and then choose the cheaper one in terms of power.

3-2-3- Real values

After simulating the first part with random values, we tried to change them in order to obtain some more realistic results; from the paper we found some data about energy consumption in real topologies. This paper considers that an ISP has different types of routers with different levels of energy needed. In our model we used a mean value for nodes and link:

- Node: 4000W,
- Link: 600W.

The capacity was set 4500 for each link.

In the paper we also found that the simulations were made with alpha=0.5, so we decided to use that value.

We did not find data about power consumed by elements considering the flow on them, hence we adapted them and in particular we set the following values:

- Pflow=0.7
- Pflowlink=0.2

We chose a full mash topology to try the system with these new values.

The results of the number of links turned off are similar to the previous full mash topology, the differences are in the order of 1-3 links more or less. On the other hand, we obtained some difference in the “distribution” of power consumption: the following graphs in figure 18 show that with real values links consume a larger part of the total power used; this happen because the previous setting was not realistic, we just assumed that nodes consume more than links.

If we increase traffic matrix values or decrease the capacity of the links, we should have again another type of distribution. This note just to mention that in real topologies the total power consumption connected with links reaches the 45% of the total consumption.

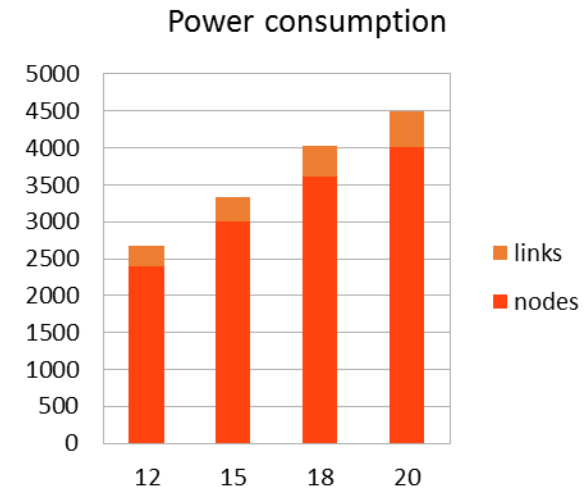

Figure 18: Power Consumption of links and nodes, according to the number of nodes

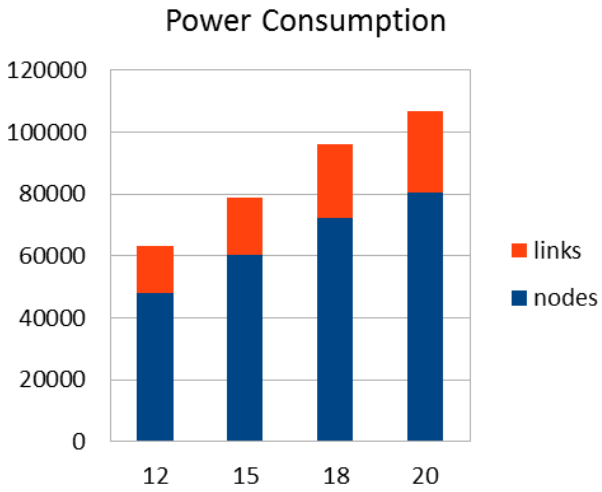

Figure 19: Power Consumption of links and nodes, according to the number of nodes

3-3- Nodes optimization

In the previous models we could turn off only links because nodes generate or receive traffic. Now we try to set the traffic towards and from some nodes in the topology at "0". For this experiment we started from a full mesh topology with the real values used before.

As said before, the power consumed by the flows are much less than the others, so our results will be quite good.

So in this model, in the formulations, our objective function does not consider PnodeFlow and PlinkFlow.

In order to initialize the problem, we chose two different numbers from 1 and N (number of nodes) and we zeroed the corresponding nodes traffic.

Since the full mesh topology allows a great number of possible paths, after the optimization we noted that the nodes without traffic were turned off. This is a very important result because nodes consume a

lot of energy, in particular in our case for each node turned off we save 4kW (It is a large number in small networks).

3-4- Considerations

In order to obtain good results it is important that the links of the network have enough capacity to host some flows, only in this case it is possible to turn off some links. Turn off nodes is more difficult because they should be without traffic. In real world we would like to have flexible networks, they should manage networks in order to minimize the power in particular during time slots in which resources are not crowed (for example during the night).

The problem that we tried to solve with Mosel programming is a NP-hard problem, so we had a gap from the optimal solution, this gap changed modifying values. In particular, for the first examples, with random values, gap was around 5-6%; realistic values made the problem more difficult to solve and the gap raised in values around 10-15%.

The first examples were not with real values so we omitted the unit of measurement.