1 Instances generation

Experimentations for the MSOND problem will be based on two types of instances : aleatory and realistic ones. Both instances are generated (with some modifications) from some known libraries' instances.
The instances will be in the following form : Name_N2_N1_D, where Name is the name of the instance, often to design the reference from the library used (like berlin from the TSP for instance), N2 is the number of nodes of the optical layer, N1 is the one on the logical layer and D the number of the demands. We will suppose that N2 is always greater than N1 which is the most near to reality.
In all of the instances, we choose to have $N1 \simeq 0.75N2$ [1]
All instances for the moment will be euclidian and the cost of the objective function is calculated through the euclidian distances. Other types of cost could after be considered : the TSP includes for instance manhattan distances, maximum distances, geographical distances, pseudo-euclidian instances, etc... [2]

1.1 Aleatory instances

- These instances are generated from the TSP euclidian instances,
- according to the size of the instance, we choose the $N2$ first nodes of the original instance and among them the $N1$ first nodes for the logical layer,
- choice of the demands : based on the number of the demands, the choice of the origins and destinations of these demands is completely aleatory (we can also think about generating this points aleatory but it will not give the real impact of augmenting the size of an instance on the resolution),
- the two paths routing the demands : the paths are choosen as following. We first begin by choosing the number of terminals of the first path and after we generate this number from the nodes of the graph (except the origin and destination). We do the same for the second path. We choose the number of terminals and generate these ones from the nodes that have not been used yet. Here the aleatory aspect is first ensured by the number of terminals but also by the choice of terminals themselves,
- for each size of the instance, we will try it on five examples. These examples are different since based on different instances from the TSP (let us say for instance berlin52, eil51, etc...)
- for each size of the logical layer (N2_N1), we generate demands with the following density (25%, 50%, 70%),

1. I tried also to be the most near to reality but I am not sure of this proportion.
2. But As far as I am concerned, I think that the euclidian is always the nearest to reality (since costs of optical fibers depend actually on the distance between two points).

- for these instances, we can also, impose the number of terminals of each demand (however the choice itself of the terminals is always kept aleatory). This can help us compare the impact of augmenting the number of terminals on the resolution.

1.2 Realistic instances

1.2.1 Gravitory model

This model was used by Sylvie Borne on a french network containing some french cities (see her Phd Thesis P167). It is briefly based on the following ideas :

- we choose the $N2$ most populated cities and among them the $N1$ most populated ones as well,
- based on a gravitory formula, we calculate for all the possible demands (all possible combinations of pairs of nodes $N1$) a quantity indicating its volume (cf. formula PhD Sylvie). we then choose among them the D most important demands,
- to generate the two paths, we forbid the direct edge between the origin and destination and some edges of the graph. We calculate two shortest paths to stay the nearest to reality.

1.2.2 Instances based on the SNDlib

- we choose the $N2$ first nodes of the original instance and among them the $N1$ first nodes for the logical layer,
- we choose the D most important demands (volume of demands are given in the instances).