Core Node Location Problem: Heuristic Solvability via Tabu Search

Darko Skorin-Kapov
skorin@adelphi.edu

Jadranka Skorin-Kapov
Jskorin@notes.cc.sunysb.edu

Hrvoje Podnar
podnarh1@southernct.edu

Adelphi University, Robert B. Willumstad School of Business

Working Paper Series:

SB-WP-2010-06

December 15, 2010, revised March 2, 2012

Abstract

We consider combinatorial problems arising in the design of Metro Core optical networks, dealing with the placement of specially equipped nodes capable of efficiently re-distributing the traffic. Two optimality criteria are considered: minimization of the maximal distance between two adjacent nodes, and minimization of the maximal path length. The paths are selected subject to the Quality of Service constraint implemented as the maximal hop length, and subject to survivability implemented as the request for having two edge disjoint paths. Integer programming formulations and a heuristic strategy based on tabu search are presented and solved either optimally using CPLEX 11.0 optimizer, or sub-optimally using a heuristic approach.

Key words: optical networks, integer programming formulations, tabu search, metaheuristic approach

1. Introduction

Metro networks usually span up to 200 km and serve as an intermediary network between access and backbone networks. Various design/service problems facing metro network administrators and designers include: traffic grooming to use resources more efficiently, sub-wavelength switching based on *GMPLS* (generalized multi-protocol label switching), and dynamic provisioning of resources, depending on the user's requests.

There are two parts in a metro network: *Metro Edge* and *Metro Core*. *Metro Edge* network consisting of edge (ingress and egress) nodes connects to outside access and backbone networks. The "interior" consist of the so-called *Metro Core* network that provides connectivity among edge nodes. The core nodes are equipped with various functionality making them expensive, hence, only a limited number of core nodes can be employed. Having a set of potential locations for core nodes, a designer has to decide where to place a limited number of core nodes in order to increase the quality of service, or survivability, or cost efficiency of a metro network. Hence, a challenging location problem is to decide which nodes in a metro network to equip with core capabilities. To illustrate the difference between various network structures, *Figure 1* presents connectivity among different networks (access, metro, and wan networks).

Today's optical network capabilities with respect to bandwidth exceed a single user's needs; hence, it is often economically justified to allocate partial bandwidth, and to groom the traffic to use the resources more efficiently. For that reason, metro networks need to have a capability for sub-wavelength granularity, which is accomplished via the *Multi Protocol Label Switch (MPLS)* mechanism. The process works as follows: starting at Label Edge Router (*LER*) (*ingress router*), packets are forwarded along *Label Switched Paths (LSPs)* according to created labels, forwarding to the next router, etc. Then, the last router in the path (*egress router*) removes the label and forwards the packet based on the header. Routers in between *Label Edge Routers* (*LER*s) are called *core* nodes or *Label Switching Routers (LSRs).*

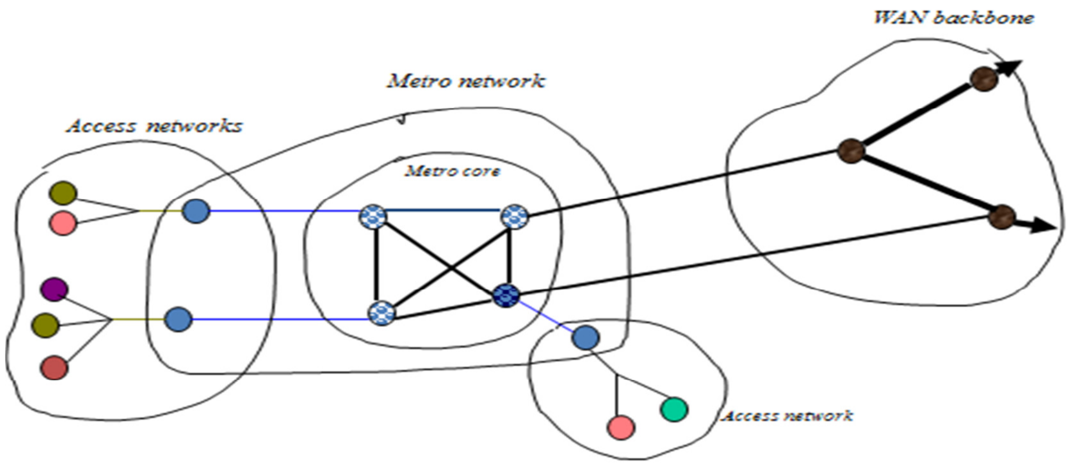

Figure 1: *Connectivity among Access, Metro, and Wan networks, and the difference between Metro Core and Metro Edge networks.*

The location of core nodes results with placement of intelligent switches performing various functions, including grooming and monitoring. Due to possible signal deterioration, such switches can also include amplifiers or regenerators. There is a difference since an amplifier (or repeater) amplifies both data and noise and it can be done optically, while regeneration involves o/e/o transmission. The regenerator location problem can be stated as follows: Given a network of *N* nodes and a set of lightpaths *L* with known distances, place the minimal number of regenerators so that a distance between regeneration points never exceeds a required distance.

Relevant literature dealing with placement of regenerators includes Kim et al. (2000) [6] who propose a minimum cost regenerator placement algorithm in all-optical multihop networks based on dynamic programming. Using spare transceivers in optical nodes, Ye et al. (2003) [9] propose a mixed integer linear program for maximizing the number of established connections in presence of regenerators. They propose a heuristic based on K-least-wavelength-weight-path routing. The environment of an *MPLS/WDM* network is considered in related papers by Gouveia et al. (2003, 2006) [3, 4]. In their Wavelength Division Multiplexing (*WDM*) network topology, the traffic matrix and end nodes which are *edge label switch routers (e-LSR)* are given, and the task is to decide on the number and on the placement of *core LSR (c-LSR)* so that selected lightpath routes satisfy *quality of service* (*QoS*) requirements implemented via the limited hop length. Subsequently, Chen et al. (2007) [1, 2] prove that the regenerator location problem is NP-Complete and propose three heuristics for solving it suboptimally.

Some optimization models relevant for the present work were considered in Skorin-Kapov and Skorin-Kapov (2008) [7]. There the authors stated that future work should include models taking into account survivability by requiring two link-disjoint paths between all source-destination pairs, and that for larger problems heuristic strategies should be developed. In this paper we precisely continue the work from Skorin-Kapov and Skorin-Kapov (2008) [7] by proposing new, augmented, mixed integer formulations and a tabu search heuristic strategy. The augmentation of the model is based on consideration of the requirement for survivability by implementing two disjoint link paths for each traffic request. The *QoS* is implemented via an upper limit on the number of hops that a path can have. It turned out that the requirement for adding backup paths made the models much more computationally demanding and necessitated the research towards development of a heuristic strategy to deal with larger problems.

In our study we consider two different objective functions: either minimizing the maximal length of a link, or minimizing the total path length. Minimizing the maximal length of a link contributes to the quality of a transmitted signal, less need for amplification and restoration, and better protection of a transported signal. Minimizing the maximal path length leads to minimizing delay. In addition, virtual topologies with a small number of hops are desirable from the economical point since that leads to the reduction of the cost of the station equipment, as argued in Stern and Bala (2000) [8]. In the following section we provide a formal model.

2. Formal Description of the Core Node Location Problem (CNLP) and MILP Formulations

We assume the following situation. Given is a complete graph $G = (V,E)$, and without loss of generality we assume that this is a graph G of shortest paths between nodes for the underlying physical network. A subset VE contains nodes that are endpoints to *LSP*s (label switch paths), or edge-*LSR*s (label switch routers). The rest of nodes, $VC=V\backslash VE$ are *candidate* nodes for the core-*LSR*s.

The assumption is that every pair of edge routers can communicate and that a label switched path can go either through edge-routers, or through core routers. The number of core routers is prescribed upfront since it depends on the available budget. For simplicity, we disregard possible constraints based on link capacities and take them to be large enough to accommodate the requested traffic. This seems to be a reasonable assumption in light of today's ever increasing bandwidth. However, we need to maintain a required quality of service (*QoS*). This is implemented through constraining the number of hops an *LSP* can take. In addition, to ensure survivability we mandate that two edge-disjoint *LSP* exist for every pair of edge routers. Given constrains on the number of possible core routers, the maximal hop distance for every *LSP*, and the two edge-disjoint *LPS*s for every pair of edge routers, the task is to place the prescribed number of core routers and to construct the *label switched paths* (LSPs) so as to minimize the maximal edge distance used. In another variant, we minimize the length of the maximal label switched path through the network. This scenario gives rise to the following mixed integer linear (MILP) models:

Dist_hop_backupMIP.mod	Minimize maximum link distance, given the prescribed number of hops and a requirement for backup edge-disjoint paths;

Path_hop_backupMIP.mod Minimize maximal path length, given the prescribed number of hops and a requirement for backup edge-disjoint paths.

We also use *MILP* formulations modeling the situation when core node locations are prescribed. These models are used as subroutines in a heuristic strategy, and are labeled as *CORE models:

Dist_hop_backupCORE.mod Minimize maximum link distance, given the prescribed number of hops and a requirement for backup edge-disjoint paths;

Path_hop_backupCORE.mod Minimize maximum link distance, given the prescribed number of hops and a requirement for backup edge-disjoint paths;

In the *Dist_hop_backupMIP.mod*, the variables are as follows:

DIST = continuous variable denoting the maximal link length

$$y1(s,t,i,j) = \begin{cases} 1 \text{ if } (s,t) \text{ pair uses the edge } (i,j) \\ 0 \quad \text{otherwise} \end{cases} : \quad s,t \in VE, s \neq t, \ i,j \in V$$

$$y2(s,t,i,j) = \begin{cases} 1 \text{ if } (s,t) \text{ pair uses the edge } (i,j) \\ 0 \quad \text{otherwise} \end{cases} : \quad s,t \in VE, s \neq t, \ i,j \in V$$

$$nc(j) = \begin{cases} 1 \text{ if node } j \text{ is a core LSR} \\ 0 \quad \text{otherwise} \end{cases} : \quad j \in \text{VC}$$

The formulation is then:

Minimize $DIST$ (1)

s.t.

$$D(i,j) * y1(s,t,i,j) \leq DIST \quad s,t \in VE, s \neq t, \ i,j \in V \qquad (2)$$

$$D(i,j) * y2(s,t,i,j) \leq DIST \quad s,t \in VE, s \neq t, \ i,j \in V \qquad (2a)$$

$$\sum_{j \varepsilon VC} nc(j) = N \qquad (3)$$

$$\sum_{i \in V} y1(s,t,i,j) \leq nc(j) \quad s,t \in VE, s \neq t, \ j \in VC \qquad (4)$$

$$\sum_{i \in V} y2(s,t,i,j) \leq nc(j) \quad s,t \in VE, s \neq t, \ j \in VC \qquad (4a)$$

$$\sum_{i \in V} y1(s,t,i,j) \leq 1 \quad s,t \in VE, s \neq t, \ j \in VE \qquad (5)$$

$$\sum_{i \in V} y2(s,t,i,j) \leq 1 \quad s,t \in VE, s \neq t, \ j \in VE \qquad (5a)$$

$$\sum_{j \in V} (y1(s,t,i,j) - y1(s,t,j,i)) = \begin{cases} 1 & \text{if } i = s \\ 0 & \text{if } i \neq s,t \\ -1 & \text{if } i = t \end{cases} \quad s,t \in VE, s \neq t, i \in V \qquad (6)$$

$$\sum_{j\,\epsilon\,V}(y2(s,t,i,j) - y2\,(s,t,j,i) = \begin{cases} 1 & if\ i = s \\ 0 & if\ i \neq s,t \\ -1 & if\ i = t \end{cases} \quad s,t\,\epsilon\,VE, s \neq t, i\,\epsilon\,V \qquad (6a)$$

$$\sum_{i\epsilon V, j\epsilon V} y1(s,t,i,j) \leq H \qquad s,t\,\epsilon\,VE, s \neq t \qquad (7)$$

$$\sum_{i\epsilon V, j\epsilon V} y2(s,t,i,j) \leq H \qquad s,t\,\epsilon\,VE, s \neq t \qquad (7a)$$

$$y1(s,t,i,j) + y2(s,t,i,j) \leq 1 \quad s,t\,\epsilon\,VE, s \neq t,\ i,j\,\epsilon\,V \qquad (8)$$

$$y1(s,t,i,j) + y1(s,t,j,i) \leq 1 \quad s,t\,\epsilon\,VE, s \neq t\ \ i,j\,\epsilon\,V \qquad (9)$$

$$y2(s,t,i,j) + y2(s,t,j,i) \leq 1 \quad s,t\,\epsilon\,VE, s \neq t\ \ i,j\,\epsilon\,V \qquad (9a)$$

The objective (1) minimizes the maximal length of a used link. D(i,j) denotes the length of the link (i,j), and constraints (2) and (2a) in fact make sure that DIST is the maximal length of any link used either by a primary path (2), or a secondary path (2a). Constraints (3) assure that we locate *N* core routers. Constraints (4) and (4a) dictate for both primary and secondary paths that if a link ending in a node from the potential core locations is used, than that node must be selected as a core node. Likewise, constraints (5) and (5a) ensure that a path from *s* to *t* can go over additional edge routers. Constraints (6) and (6a) are flow conservation constraints indicating that a path (*s,t*) has to start at *s*, and end at *t*. Constraints (7) and (7a) mandate that not more than *H* hops can be used on either primary (7) or secondary (7a) path. Constraints (8) make sure that the two paths are link-disjoint. Finally, constraints (9) and (9a) disallow cycling on a link used in a path.

If we want to minimize the length of the maximal path in the network, instead of variable *DIST*, we define the variable *PATH* as follows:

PATH = continuous variable denoting the maximal path length

Then, the objective function (1) and the constraints (2) and (2a) change to (1**, 2**, 2a**), and the rest of the constraints carries over resulting with the related MILP:

Minimize *PATH* (1**)

s.t.

$$\sum_{i\,\epsilon V, j\,\epsilon V} D(i,j) * y1(s,t,i,j) \leq PATH \quad s,t\,\epsilon\,VE, s \neq t \qquad (2\,**)$$

$$\sum_{i\,\epsilon V, j\,\epsilon V} D(i,j) * y2(s,t,i,j) \leq PATH \quad s,t\,\epsilon\,VE, s \neq t \qquad (2a\,**)$$

$plus\ constraints\ (3) - \ (9a).$

Since we are minimizing the maximal path length, in this case the idle cycling will not occur because that would add to the path length. Hence, in the *Path* version of the formulation we can disregard constraints (9) and (9a).

In the case of models with prescribed core node locations, i.e. *CORE models, we do not have variables *nc(j), j* ϵ *VC*. In *CORE formulations we define the set *VAC* as the n-cardinality set of *Actual Core* node locations, not a set of *potential* core node locations. (Hence, *VAC* is a subset of *VC*.) The *Dist_hop_backupCORE.mod* then becomes

Minimize *DIST* (1*)
s.t.

$$D(i,j) * y1(s,t,i,j) \leq DIST \quad s,t \in VE, s \neq t, \quad i,j \in VE \cup VAC \quad (2^*)$$

$$D(i,j) * y2(s,t,i,j) \leq DIST \quad s,t \in VE, s \neq t, \quad i,j \in VE \cup VAC \quad (2a^*)$$

$$\sum_{i \in VE \cup VAC} y1(s,t,i,j) \leq 1 \quad s,t \in VE, s \neq t, \ j \in VE \cup VAC \quad (5 *)$$

$$\sum_{i \in VE \cup VAC} y2(s,t,i,j) \leq 1 \quad s,t \in VE, s \neq t, \ j \in VE \cup VAC \quad (5a *)$$

$$\sum_{j \in VE \cup VAC} (y1(s,t,i,j) - y1(s,t,j,i)) = \begin{cases} 1 & if\ i = s \\ 0 & f\ i \neq s,t \\ -1 & if\ i = t \end{cases} \quad s,t \in VE, s \neq t, i \in VE \cup VAC \ (6 *)$$

$$\sum_{j \in VE \cup VAC} (y2(s,t,i,j) - y2(s,t,j,i)) = \begin{cases} 1 & if\ i = s \\ 0 & if\ i \neq s,t \\ -1 & if\ i = t \end{cases} \quad s,t \in VE, s \neq t, i \in VE \cup VAC \ (6a *)$$

$$\sum_{i \in VE \cup VAC, j \in VE \cup VAC} y1(s,t,i,j) \leq H \quad s,t \in VE, s \neq t \quad (7 *)$$

$$\sum_{i \in VE \cup VAC, j \in VE \cup VAC} y2(s,t,i,j) \leq H \quad s,t \in VE, s \neq t \quad (7a *)$$

$$y1(s,t,i,j) + y2(s,t,i,j) \leq 1 \quad s,t \in VE, s \neq t, \ i,j \in VE \cup VAC \quad (8 *)$$

$$y1(s,t,i,j) + y1(s,t,j,i) \leq 1 \quad s,t \in VE, s \neq t \ \ i,j \in VE \cup VAC \quad (9 *)$$

$$y2(s,t,i,j) + y2(s,t,j,i) \leq 1 \quad s,t \in VE, s \neq t \ \ i,j \in VE \cup VAC \quad (9a *)$$

In the Path*CORE model we replace (1*), (2*), and (2a*) with

Minimize *PATH* (1**)
s.t.

$$\sum_{i \in VE \cup VAC, j \in VE \cup VAC} D(i,j) * y1(s,t,i,j) \leq PATH \quad s,t \in VE, s \neq t \quad (2 **)$$

$$\sum_{i \in VE \cup VAC, j \in VE \cup VAC} D(i,j) * y2(s,t,i,j) \leq PATH \quad s,t \in VE, s \neq t \quad (2a **)$$

3. Tabu Search Strategy for the Core Node Location Problem (CNLP)

Due to the complexity of the problem, for larger problem instances it is mandatory to construct a heuristic strategy to solve it suboptimally. We experimented with a *tabu search* strategy whereby a search proceeds by evaluating a neighborhood of the current solution and selecting a best non-tabu solution. A feasible solution to the *CNLP* is given by the core node locations and the paths between every pair of edge nodes, satisfying the constraints given in the formulation. Our heuristic strategy works as follows. First, the initial solution is constructed in the following way:

a) <u>Initial selection of N core node locations</u>: For each j from *VC* (i.e. from the set of potential core node locations) calculate the total length (*TL*), i.e. the sum of lengths to all edge nodes:

$$\forall j \in VC, \sum_{i \in VE} D(i,j) = TL\ (j)$$

Order the nodes in *VC* in increasing order of *TL*. Select the first *N* nodes as the initial core-locations. The rationale is that the nodes with smaller total length are better positioned for serving as "hubs" or core-locations and will lead to smaller paths.

b) <u>Selecting initial paths (primary and secondary) having at most *H* hops</u>: Solve a subroutine that finds paths, given the locations of core nodes, i.e. solve the *CORE* subroutine. (I.e. depending on the objective, select either ***Dist_hop_backupCORE.mod*** or ***Path_hop_backupCORE.mod***.)

The initial solution is defined as the current solution and as the best found so far, i.e. the incumbent. In the improvement phase we attempt to find a better solution. First, we need to define the neighborhood of the current solution. Let *c* be the cardinality of the set *VC* (the set of potential core node locations), i.e. *c*=|*VC*|. There are $\binom{c}{N} = \frac{c!}{N!(c-N)!}$ possible subsets of *N* nodes in the set *VC*. However, for a *given* location of *N* core nodes and its respective paths obtained optimally via the *CORE* subroutine, we define its neighborhood as the set of all core-node locations that differ in only one node, plus their respective routes calculated optimally. Note that the evaluation of a neighborhood in which one core node is replaced by one non-core nodes from the set of potential core locations, implies evaluating *(c-N) N* exchanges (since there are *c-N* non-core nodes and each can replace each of the current *N* core nodes). The evaluation can be performed by calculating the respective paths using the **CORE* subroutine. However, given the extensive computational effort when evaluating all possible exchanges, we propose a reduced search as follows. For each exchange of a core node, say "*c-out*" with an incoming core node, say "*c-in*", we modify the *y* - variables as follows:

If *y1(s,t,c-out, j)* = 1 then *y1(s,t,c-out, j)* = 0 and *y1(s,t,c-in, j)* = 1 for all *s, t, j* ∈ V

If *y2(s,t,c-out, j)* = 1 then *y2(s,t,c-out, j)* = 0 and *y2(s,t,c-in, j)* = 1 for all *s, t, j* ∈ V

Calculate the objective function value (either *DIST* or *PATH*, depending on the model). For example, for the DIST model: *DIST*(new) = *max (D(i,j)*y1(s,t,i,j), D(i,j)* y2(s,t,i,j))* for all *s ≠ t, i, j* ∈ V. Among the exchanges, select the one with smallest *DIST* value (the ties are broken arbitrarily). When the exchange is selected, then run the *CORE* subroutine for the new set of core nodes, to get optimal paths and optimal objective function value.

The best neighbor is replacing the current solution, and if it improves the best known solution, it replaces the incumbent. The inverse exchange is forbidden for a number of subsequent iterations in order to prevent cycling. This number of iterations is given as the parameter *tabu list size (tl-size)*. The size of tabu list is an important parameter: if it is too big, it will constrain too much the search, if it is too small, it will not disable cycling. After some preliminary calculations, we decided that the size of a tabu list should be dependent on the problem size as a percentage of approximately 50% of non-core nodes in the set of possible core nodes, i.e. *1/2(c-N)*. In case that the inclusion of a tabu node in a set of core nodes gives a solution better than the best obtained thus far, its tabu status is overridden and the exchange is allowed. This is the so-called *aspiration criterion*. The stopping criterion is the number of iterations without improvement, and the preliminary results show that this number should be dependent on the neighborhood size. Since an iteration of tabu search is computationally quite demanding, we have to define the stopping criterion as a relatively small number of iterations without improvement. We set it to approximately 66% of the neighborhood size, i.e. as approximately *2/3*(c-N)N*. In order to diversify the search, we employ additional restarts based on the *long term memory* as follows. For each node from the set of potential core node locations we record the number of times it appeared in the selected core set. Say that node *i* appeared *k* times as a core location during the run of the algorithm. Then, its measure of total edge distances, *TL (i)* is increased for *10k%*, i.e. *TL(i) -> (1 + 10k/100)*TL(i)*. We then restart with the modified set of *TL* measures. This will penalize the frequently used nodes and will lead to a selection of a different set of initial core locations. We can invoke the long term memory a number of times (*LTM_restart*) and, due to the computational complexity of tabu search iterations, we decided to invoke it and restart three more times.

The pseudo code of the tabu search algorithm (*TABU_CNLP*) is as follows.

Step 1: Start Set the parameters for tabu list size (*tl_size*), the number of iterations without improvement (*Iter_no_impr*), and the number of long term memory restarts (*LTM_restart*). Set the starting solution: for each potential core node calculate the sum of lengths to all edge nodes (*TL*) and select *N* core nodes with the smallest sums. Perform the **CORE* subroutine to generate optimal paths and complete the initial solution.

Step 2: Tabu search heuristic *Iter* = 0, initial tabu list is empty. While the number of iterations without improvement is less than *Iter_no_impr*, do the following: at each iteration evaluate all single exchanges between a non-core and a core node by appropriately modifying the y-variables when re-directing the paths via the incoming core node, and away from the outgoing core node. Calculate the objective function value. Select the best exchange (giving the smallest objective value). Then, for a given set of core nodes, invoke the *CORE* subroutine and perform the best non-tabu exchange. Update the tabu list by recording the node that exited the set of core nodes and the respective iteration number. The inclusion of this node to the set of core locations is forbidden for *tl_size* subsequent iterations. The tabu status can be overridden if the inclusion the 'tabu' node in the core set would provide the solution better than the incumbent. Update the frequency of a node appearing in the solution for the long term memory purpose. Update the current solution, and if better solution is obtained, update the incumbent.

Step 3: Restart based on the Long Term Memory Until the number of restarts is less than *LTM_restarts*, modify the *TL* measures for potential core node locations using the long term memory. Go to step 1.

Step 4: Output Record the best obtained solution and its objective value.

4. Computational results

The computational results were performed on a Gateway PC with Intel Core2 CPU and with RAM of 2,038 MB (for optimal solvability). The tabu search heuristic tests were run on a Dell Precision T1500 with i7-870 quad core at 2.93GHz and 8GB of memory**.** CPLEX 11.0 solver was used to get optimal solutions for problems up to 20 nodes, and as a subroutine for the tabu search heuristic. In our computational study, we have used a number of randomly created data sets:

1. 15-, 16-,….,20- dimensional non-symmetric distance matrix, with $2 \leq D(i,j) \leq 100$
2. 25-dimensional data set with non-symmetric distance matrix, with $2 \leq D(i,j) \leq 200$
3. 30-dimensional data set with non-symmetric distance matrix, with $2 \leq D(i,j) \leq 200$

The first data set (ranging from 15 to 20 nodes) was also used in a companion paper by Skorin-Kapov and Skorin-Kapov (2008) where some simpler *MILP* models were considered: there was no requirement for survivability by implementing two disjoint link paths. The requirement for adding backup paths actually makes the models much more challenging and computationally demanding. To justify the development of a heuristic algorithm, in *Tables 1* and *2* we compare the optimal solutions and *CPU* times obtained via the CPLEX 11.0 solver for models with and without backup paths requirement. For example, the *CPU* time required for solving a problem of minimizing the maximal link distance in a network with 20 nodes, 5 core node locations, and 3 hops, was 114,028 seconds (or 31.7 hours) when the backup path requirement was implemented. The same model without backup paths and only the hop constraints run in 186 seconds. It is obvious that for problems with more than 20 nodes we need to devise a heuristic strategy.

Table 1.
Models Dist-hop-backupMIP.mod and Dist-hopMIP.mod with various parameters N (the number of core nodes) and H (the allowed hop distance)

Size n	Number of core nodes	H=3 $2 \leq D(i,j) \leq 100$ Dist_hop_backupMIP		Dist_hopMIP		H=5 $2 \leq D(i,j) \leq 100$ Dist_hop_backupMIP		Dist_hopMIP	
		CPU sec	**DIST**	**CPU sec**	**DIST**	**CPU sec**	**DIST**	**CPU sec**	**DIST**
15	**N=3**	979	41	35	31	1561	29	16	18
	N=2	1,431	41	38	31	1563	32	73	26
16	**N=4**	216	37	45	29	889	26	51	18
	N=2	1,609	41	41	31	10067	32	91	26
17	**N=4**	2,543	37	37	28	STOPPED	33	71	21
	N=2	7,167	44	54	37	STOPPED	41	130	21
18	**N=4**	5,267	37	92	28	STOPPED	30	80	21
	N=3	7,326	41	133	31	50,475	27	250	21
19	**N=4**	8,482	36	77	28	56,073	27	94	21
	N=3	21,115	38	141	28	STOPPED	98	109	21
20	**N=5**	114,028	33	186	28	112,601	21	371	18
	N=3	32,954	38	148	28	STOPPED	27	2,151	19

STOPPED= stopped after more than 2 days. In such a case the objective function is not provably optimal, it is just a best known solution.

Table 2.
Models Path_hop_backupMIP.mod and Path_hopMIP.mod with various parameters N (the number of core nodes) and H (the allowed hop distance)

Size n	Number of core nodes	H=3 $2 \leq D(i,j) \leq 100$ Path_hop_backupMIP		Path_hopMIP		H=5 $2 \leq D(i,j) \leq 100$ Path_hop_backupMIP		Path_hopMIP	
		CPU time	**PATH**	**CPU time**	**PATH**	**CPU time**	**PATH**	**CPU time**	**DIST**
15	**N=3**	1,634	65	2	56	1,210	65	3	52
	N=2	782	71	4	56	290	68	3	56
16	**N=4**	219	63	5	56	66	59	2	46
	N=2	15,206	71	3	56	STOPPED	71	5	56
17	**N=4**	8,099	67	5	56	STOPPED	64	10	54
	N=2	STOPPED	88?	3	59	STOPPED	106	4	56
18	**N=4**	4,374	67	8	56	STOPPED	70	7	54
	N=3	788	67	10	58	STOPPED	72	19	56
19	**N=4**	1,928	62	8	54	18,566	60	3	54
	N=3	STOPPED	272?	20	56	STOPPED	61	20	54
20	**N=5**	STOPPED	62?	41	50	STOPPED	62	3	45
	N=3	STOPPED	66?	38	56	STOPPED	77	10	53

STOPPED= stopped after more than 2 days. In such a case the objective function is not provably optimal, it is just a best known solution.

Tables 3 and 4 compare tabu search and optimal algorithm for models with backup paths. The column heads of Tables 3 and 4 are defined as follows. *Init DIST* (resp., *Init PATH*) denote the objective function value of the initial solution. For the tabu search strategy, *1st pass CPU* and the adjacent *DIST* (resp. *PATH*) columns indicate the time and the objective value reached without the long-term memory diversification. Finally, *CPU sec* and *DIST* (resp., *PATH*) indicate the overall time and the obtained objective value upon terminating the tabu search. For the respective MIP models (i.e. models attempted to solve optimally via CPLEX solver) we display the time (*CPU sec*) and the obtained objective value *DIST* (resp., *PATH*). From the results displayed in Table 3 (for the *DIST* model) for number of hops constrained to 3, (H=3), it is obvious that tabu search often finds the optimal solution in the beginning of the search, in much less time than needed for the optimal algorithm. When the number of hops is relaxed and constrained to 5, is seems that the problems become more difficult to solve. Indeed, in five instances, out of 12, the optimal algorithm had to be stopped after more than 2 days. In all instances that were stopped, tabu search provided better solutions, in reasonably smaller amount of time.

Table 4 reveals that the *PATH* models with backup paths tend to be easier than *DIST* models when solved with tabu search heuristic, however to solve them optimally is more challenging and many of the instances (especially with *H*=5) had to be stopped after running for more than two days. This is in contrast to the results for *DIST* and *PATH* modes without backup paths, as reported in Skorin-Kapov and Skorin-Kapov (2008) [7]. There the authors write that the model in which the maximal link distance (*DistMIP.mod* and *Dist_hopMIP.mod*) is minimized is much more computationally demanding than the model minimizing the maximal path length (*PathMIP.mod* and *Path_hopMIP.mod*). As a possible reason the authors state a more compact form of constraints for the path models, leading to better characterization of the feasible set. However, when models are augmented to take into account backup paths, it seems that a more challenging search has to be performed, resulting with the impossibility of an optimal algorithm to find a solution in a reasonable amount of time. In the backup enhanced models, the optimal strategy is less efficient for the PATH models. Interestingly, the heuristic strategy based on tabu search works better for the PATH models, as if again capitalizing on the compact form of the constraints. Comparing *DIST* and *PATH* models in the context of optical networking, is seems that the strategy of minimizing the maximal link distance is better suited for decisions regarding regeneration of optical signals, and the strategy of minimizing the maximal path distance diminishes the possibility of failures.

Table 3.

Model Dist-hop-backupMIP.mod solved to optimality and with Tabu Search heuristic (TABUDIST-CNLP)

Size n	**core nodes**	**H=3 2 ≤ D(i,j) ≤100**							**H=5 2 ≤ D(i,j) ≤100**						
		Tabu Search					**Dist_hop_ backupMIP**		**Tabu Search**					**Dist_hop_ backupMIP**	
		Init DIST	**1st pass CPU sec**	**DIST**	**CPU sec**	**DIST**	**CPU sec**	**DIST**	**Init DIST**	**1st pass CPU sec**	**DIST**	**CPU sec**	**DIST**	**CPU sec**	**DIST**
15	**N=3**	41✓	220	41✓	418	41✓	979	41	41	138	29✓	298	29✓	1561	29
	N=2	41✓	94	41✓	307	41✓	1431	41	41	49	32✓	112	32✓	1563	32
16	**N=4**	41	156	37✓	329	37✓	216	37	32	82	32	326	28×	889	26
	N=2	41✓	127	41✓	421	41✓	1609	41	41	54	32✓	134	32✓	10067	32
17	**N=4**	44	214	41	826	41×	2543	37	27**?**	195	27**?**	658	27**?**	STOPPED	33
	N=2	44✓	116	44✓	469	44✓	7167	44	41	172	38	521	33**?**	STOPPED	41
18	**N=4**	44	364	41	815	41×	5267	37	27**?**	553	27**?**	1100	27**?**	STOPPED	30
	N=3	44	337	41✓	750	41✓	7326	41	32**?**	174	32**?**	651	32**?**	50475	27
19	**N=4**	41	1006	38	1938	37×	8482	36	27✓	500	27✓	1675	27✓	56073	27
	N=3	41	280	41	920	38✓	21115	38	32**?**	251	32**?**	751	29**?**	STOPPED	98
20	**N=5**	41	1324	36	2532	36×	114028	33	27	920	27	2513	21✓	112601	21
	N=3	41	320	41	1270	38✓	32954	38	32	251	32	1093	27**?**	STOPPED	27

✓ = tabu is optimal; × = tabu is not optimal; **?** = best known; STOPPED = stopped after 2-4 days

Table 4.

Model Path_hop_backupMIP.mod solved to optimality and with Tabu Search heuristic (TABUPATH-CNLP)

Size n	**core nodes**	**H=3 2 ≤D(i,j) ≤100**							**H=5 2 ≤ D(i,j) ≤100**						
		Tabu Search					**Path_hop_ backupMIP**		**Tabu Search**					**Path_hop_ backupMIP**	
		Init PATH	**1st pass CPU sec**	**PATH**	**CPU sec**	**PATH**	**CPU sec**	**PATH**	**Init PATH**	**1st pass CPU sec**	**PATH**	**CPU sec**	**PATH**	**CPU sec**	**PATH**
15	**N=3**	73	24	68	127	65✓	1634	65	73	22	68	94	65✓	1210	65
	N=2	80	17	71✓	74	71✓	782	71	79	19	78	74	68✓	290	68
16	**N=4**	70	33	63✓	141	63✓	219	63	63	29	59✓	125	59✓	66	59
	N=2	80	18	71✓	78	71✓	15206	71	79	23	66**?**	79	66**?**	STOPPED	71
17	**N=4**	78	80	67✓	303	67✓	8099	67	63	65	61**?**	283	61**?**	STOPPED	64
	N=2	80	38	76**?**	157	76**?**	STOPPED	88	79	32	74	179	66**?**	STOPPED	106
18	**N=4**	78	86	67✓	422	67✓	4374	67	63	74	61**?**	365	61**?**	STOPPED	70
	N=3	78	61	68	313	67✓	788	67	67	102	65	316	64**?**	STOPPED	72
19	**N=4**	67	186	67	733	67×	1928	62	60✓	126	60✓	638	60✓	18566	60
	N=3	70	120	67**?**	549	67**?**	STOPPED	272	67	138	61**?**	541	61**?**	STOPPED	61
20	**N=5**	67	607	61**?**	1591	59**?**	STOPPED	62	60	230	59**?**	968	59**?**	STOPPED	62
	N=3	70	145	67	634	67×	STOPPED	66	67	154	61**?**	623	61**?**	STOPPED	77

✓ = tabu is optimal; × = tabu is not optimal; **?** = best known; STOPPED = stopped after 2-4 days

Finally, Tables 5 and 6 provide information on tabu search runs for larger problems, for *n* = 25 and 30. For these data we tested instances with various values of *H* and *N* as follows. For each data set, three values for the number of hops were tested: *H*= *n*/10, *n*/5, and *n*/3 (rounded down). The values for *N* (the number of core nodes) were selected as: *N*= |*VC*|/2 and |*VC*|/3 (rounded down). The column labeled *IDIST* displays the objective value of the initial solution. Since that value is used in the subsequent column to calculate the percentage improvement due to our heuristic strategy, we needed to get a sense of how good or bad the initial solution really is. To that end, we also generated initial solutions by *randomly* selecting a prescribed number of core nodes and then constructing the initial paths (primary and secondary) having at most H hops by using the *CORE subroutine. For the data tested, the *random strategy* in initially selecting the core node locations delivered *worse* initial objective values. Hence, the proposed initial solutions obtained via considerable computational effort are good starting solutions. As a consequence, the percentage improvement over those solutions seems to be a reasonable measure of the effectiveness of the proposed heuristic algorithm. In addition, due to the comparison with the optimal algorithm for smaller problems where the comparison is feasible, the proposed heuristic strategy proved very effective in delivering optimal or close to optimal solutions using considerably less time.

The results from Tables 5 and 6 reveal the following observations. *DIST* model shows greater variability in percentage of improvement, and it is much more time consuming. The *PATH* model shows less variability and greater improvement percentage on average, with much smaller standard deviation. In addition, *PATH* model seems to be less time consuming. The statistical comparison is presented in Table 7. The models are obviously computationally very demanding, especially the *DIST* model. The increased size of the network (number of nodes) results with considerably longer computational times. This seems not so for the *PATH* models. Hence, it seems that the heuristic strategy as proposed in this paper works better for the model with the objective to minimize the length of the maximal path, in effect diminishing the possibilities of failures.

Table 5.
Tabu search results for DIST model (TABUDIST-CNLP)

n	H	N	DIST	CPU sec	CPU to best solution	IDIST (Initial DIST)	% improvement 100*(IDIST-DIST)/DIST
25	**3**	**6**	62	23745	315	62	none
		4	64	13825	2711	68	6.25%
	5	**6**	47	60553	19611	50	6.38%
		4	48	18915	14280	50	4.17%
	8	**6**	42	37486	12470	50	19.05%
		4	49	113253	7227	50	2.04%
30	**4**	**7**	44	376694	240761	49	11.36%
		5	49	167895	6582	49	none
	6	**7**	37	217072	164143	42	13.51%
		5	37	161458	81116	49	32.43%
	10	**7**	42	171792	8208	42	none
		5	37	73720	35051	49	32.43%

The size of problem = *n*; the number of hops = *H*, the number of core nodes = *N*. The parameters of tabu search heuristic were set as: tabu list size (tl_size) = (*PC-N*)/2; number of iterations without improvement =25%* (*PC-N*)*N for *n*=25, =15%* (*PC-N*)*N for *n*=30, number of restarts=5.

Table 6.
Tabu search results for PATH model (TABUPATH-CNLP)

n	H	N	PATH	CPU sec	CPU to best solution	IPATH (Initial PATH)	% improvement 100*(IPATH-PATH)/PATH
25	**3**	**6**	106	3007	2650	115	8.49%
		4	106	2077	1462	115	8.49%
	5	**6**	92	2379	1735	106	15.22%
		4	97	1715	1399	108	11.34%
	8	**6**	89	2013	1409	106	19.10%
		4	99	2028	1558	108	9.09%
30	**4**	**7**	85	7882	2519	94	10.59%
		5	89	3723	1718	100	12.36%
	6	**7**	83	4886	1978	94	13.25%
		5	89	3695	1428	100	12.36%
	10	**7**	83	3093	966	94	13.25%
		5	89	3303	709	100	12.36%

The size of problem = *n*; the number of hops = H, the number of core nodes = *N*. The parameters of tabu search heuristic were set as: tabu list size (tl_size) = (*PC-N*)/2; number of iterations without improvement =25%* (*PC-N*)*N for *n*=25, =15%* (*PC-N*)*N for *n*=30, number of restarts=5.

Table 7.
Statistical Summary for larger problems (n=25, 30) adapted from Tables 5 and 6

	DIST			PATH		
	%improvement 100*(IDIST-DIST)/DIST	CPU (hours)	CPU to best (hours)	% improvement 100*(IPATH-PATH)/PATH	CPU (hours)	CPU to best (hours)
Average	10.6	33.2	13.7	12.2	0.9	0.5
St.Dev.	11.7	29.5	21.2	3.0	0.5	0.2
Max	32.4	104.6	66.9	19.1	2.2	0.7

5. Conclusions and Future Research

We presented MILP formulations and a heuristic search algorithm for a difficult combinatorial problem arising in management and design of optical networks. The problem is to decide on placement of a given number of specially equipped nodes, having the objective of either minimizing the maximal link length, or minimizing the maximal path length. This consideration resulted with two models, *DIST* and *PATH*, respectively. Due to modeling of constraints on the quality of service (via the maximal allowable hop distance), and the existence of backup paths (edge disjoint paths), the models appear to be very computationally demanding. A tabu search based heuristic strategy was developed and its merit was assessed first by comparison with the optimal algorithm from the CPLEX 11.0 solver for smaller problems, and next by the percentage of improvement with respect to the initial solution for larger problems. The strategy seems better suited for *PATH* models, resulting with bigger percentage of improvement over the initial solution, and more efficient computational times. Due to problem complexity, future research should investigate alternative heuristic strategies in hope to be able to tackle even larger problems.

References

1. Chen, S., & Raghavan, S. (2006). The Regenerator Location Problem. *Proceedings of the 8th INFORMS Telecommunication Conference.* Dallas, Texas.

2. Chen, S., Ljubic, I., & Raghavan, S. (2007). *The Regenerator Location Problem.* Report.

3. Gouveia, L., Patricio, P., & de Sousa, A. (2006). Hop-constrained node survivable network design: an application to MPLS over WDM. *Procedings of 8th INFORMS Teleommunication Conference* . Dallas, Texas.

4. Gouveia, L., Patricio, P., de Sousa, A., & Valadas, R. (2003). MPLS over WDM network design with packet level QoS constraints based on ILP models. *Proceedings of IEEE INFOCOM.*

5. (2008). *ILOG AMPL CPLEX System, Version 11.0.0* ILOG.

6. Kim, S.-W., Seo, S.-S., & Kim, S. (2000). Regenerator placement algorithms for connection establishment in all-optical networks. *GLOBECOM - Global Telecommunications Conference IEEE*, (pp. 1205-1209).

7. Skorin-Kapov, J., & Skorin-Kapov, D. (2008). MIPL formulations and Quantitative Analysis for the Core Node Location Problem. *International Journal of Quantitative Operations Management, 14 (1)*, 17-27.

8. T.E. Stern, K. Bala, Multiwavelength Optical Networks: A Layered Approach, *Addison-Wesley*, 2000

9. Ye, Y., Chai, T., Cheng, T., & Lu, C. (2003). Agorithms for the design of WDM translucent optical networks. *Optics Express , 11* (22), 2917-2926.