A Heuristic Method for Discrete Capacity Tree Access Network Design

André Girard
GERAD and INRS-Télécommunications
C.P. 644, Montréal (Québec) H5A 1C6, Canada
andre@inrs-telecom.uquebec.ca

Brunilde Sansò
GERAD and École Polytechnique de Montréal
C.P. 6079, Succ. Centre-ville
Montréal (Québec) H3C 3A7, Canada
bruni@crt.umontreal.ca

Linda Dadjo
GERAD and INRS-Télécommunications
C.P. 644, Montréal (Québec) H5A 1C6, Canada
dadjo@inrs-telecom.uquebec.ca

April, 2001

Les Cahiers du GERAD
G–2001–17

Abstract

This paper solves the problem of designing an access tree network for which the users are connected to the switches through SONET channels on fiber optics links and ADM equipment at the nodes. The economies of scale provided by the SONET hierarchy are specifically taken into account. A fast tabu search method is proposed and its details are thoroughly discussed. Results are provided for a set of random cases as well as for a real network.

Résumé

Dans ce article nous proposons une méthode de résolution rapide basée sur l'heuristique taboue pour le problème de la conception de réseau d'accès en arbre. Dans notre cas les usagers sont reliés à travers des ADM installés aux noeuds aux commutateurs par de la fibre optique avec la technologie SONET. Nous tenons essentiellement compte de l'économie d'échelle qui peut être réalisée par l'utilisation de la hiérarchie SONET. Les tests sont faits aussi bien sur des réseaux générés aléatoirement que sur des données provenant de l'industrie.

1 Introduction

The design of transmission networks have generally been done in two phases: first the design of the access and then of the backbone. Recently, there has been a number of authors that have examined the integrated design (see [6] and the articles that followed [4, 5, 2, 3] as well as the excellent review by Klincewicz [13]). In the integrated design, the switch location, their capacities and number of ports as well as the links between switches are obtained as solution of the backbone design. On the other hand, such a backbone solution will depend on the solution of the access network where the topology, line capacities and homing pattern are selected. In fact, the solution of the access determines the traffic matrix that will serve as input to the backbone design. Typically, the solution algorithm will iterate between the backbone and the access solution until it reaches a suitable integrated solution.

Many configurations are possible for the access design but a star is a very popular choice for many type of networks [6]. However, when the multiplexing of user demands might yield cost reductions, an access tree with concentrator equipment might be more appropriate, as shown on Fig. 1.

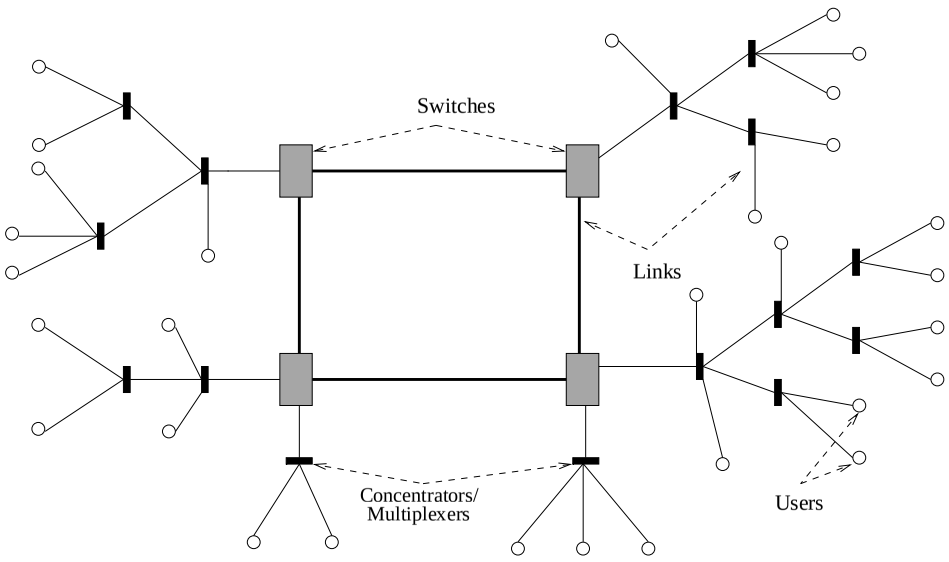

Figure 1: Tree Access Network-Ring Backbone

Unfortunately, the design of access networks with a tree topology is extremely time consuming, much more so than with a star topology. In [7], a first attempt was made to solve such problem under realistic conditions by a simple tabu heuristic based on the work of Sharaiha et al [14]. Numerical results were disappointing since the CPU time required for solving a small access network turned out to be much too large to be usable within an integrated optimization heuristic. Clearly, in order to provide good solutions in reasonable time, the tabu search would have to be closely tailored to the problem under consideration. In this paper these adaptations are summarized. We show how effective

they were in reducing the computation time sometimes by a factor of 100 over the simple tabu heuristic.

This paper is divided as follows. In Section 2, the problem and the mathematical model will be discussed. The heuristic will be presented in Section 3 whereas numerical results and conclusions follow in Sections 4 and 5, respectively.

2 The Model

In this Section, we first present the problem and discuss modeling features. Next, we recall how the access design relates to the integrated network design. This will give us an indication of what parameters are supposed to be given in an integrated procedure. The last subsection contains a description and discussion of the proposed optimization problem.

2.1 The Access Network Design Problem

In all known models for access network design [8, 12, 11, 1, 9] only concentration of the signal at the nodes is taken into account. Information is received on many links and is retransmitted on a single line towards the center. Also, only one link can be used between each user and a concentrator.

In our model, we consider that there is ADM equipment at the nodes. The ADMs act as concentrators and multiplexers and allow the installation of more than one line between two nodes. The problem we treat can be theoretically classified as a Multi Center Capacitated Spanning Tree with different capacity of the ports at the central site, many access link types and ADM nodes. As far as we know, this problem is a generalization of the models described in [8, 12, 11, 1, 9] and has not been examined in the literature.

Even though the model can handle multiple transmission rates, we nevertheless have to make a strong simplification in our traffic model: We assume that the total demand of a user site can be arbitrarily partitioned on any combination of SONET channels. This is acceptable if the traffic in the network layer is packetized but not very realistic whenever a particular demand may be composed of aggregated streams, some of which may have to be carried entirely on a single channel for performance reasons. In that case, an even more complex model would be required.

An important feature of our model relates to the functioning of the ADMs themselves. ADMs are able to inject into and extract signals from the SONET frame without having to demultiplex the complete signal. This means that we can readily add signals as we go up the tree but also that all outgoing signals need not be on the same SONET channel and can be split on lower-rate channels if this turns out to be more economical. This is a feature of our model that is unique as far as we know and that can lead to significant cost reductions if properly used.

2.2 Integrated Design Problem

We based the modeling of our access problem on the integrated design algorithm of [6] which is a tabu search that iterates between a backbone design stage and an access network design

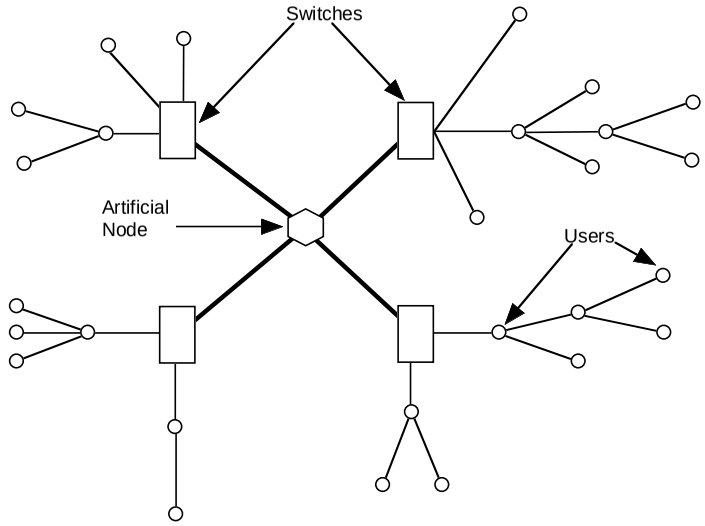

Figure 2: Artificial Node

subproblem. At the backbone design level, it is necessary to choose the switch location and type as well as the connections among switches. These two features determine how many slots are available to install access cards. Another decision variables are the port rates installed in each slot. Thus, for our access design problem, we assume that switch location and type and the number and capacity of ports available are provided by the backbone design.

2.3 Mathematical Model

Before presenting the formulation, we make the following assumptions:

- switch location, users location, user demand and characteristics are given,
- all user sites have ADMs,
- the cost of links between a user and the ADM includes the ADM cost,
- the cost of links from a user to a switch includes the port and the multiplexer cost of that switch,
- we assume that there is an artificial node to which all switches are connected by links of zero cost and infinite capacity. This is useful to model the problem as a tree design instead of a forest, as it is shown in figure 2.

Let us also define the following notation:

$M = \{1, \ldots, |M|\}$ the set of users,
$N = \{1, \ldots, |N|\}$ the set of switches,

n_k the capacity of port $k \in N$,

τ_j the rate of customer $j \in M$. As we said earlier, we assumed that this demand can be arbitrarily separated on different SONET channels.

The decision variables are

$$x_{ij} = \begin{cases} 1 & \text{if terminal } i \in M \text{ is connected to} \\ & \text{terminal } j \in M \\ 0 & \text{otherwise,} \end{cases}$$

$$y_i^k = \begin{cases} 1 & \text{if terminal } i \in M \text{ is connected to} \\ & \text{port } k \in N \\ 0 & \text{otherwise.} \end{cases}$$

We also define the auxiliary variable z_i^k, a function of x_{ij} and y_i^k given by

$$z_i^k = \begin{cases} 1 & \text{if terminal } i \in M \text{ is in the tree rooted} \\ & \text{at port } k \in N \\ 0 & \text{otherwise.} \end{cases}$$

Note that since we are considering undirected links, we do not distinguish between x_{ji} and x_{ij} and we will always use x_{ij} for $i < j$. To complete the notation, let us define the following:

$c_{ij}(\mathbf{x}, \mathbf{y}, \tau)$**:** cost of connecting user $i \in M$ to user $j \in M$ (for $i < j$), the ADM cost being included in $c_{i,j}$,

$c'_{ij}(\mathbf{x}, \mathbf{y}, \tau)$**:** cost of connecting user $i \in M$ to switch $j \in N$.

The actual cost of a link is determined by its length and the amount of demand it carries. The most economical set of capacities that will carry the demand on a link is installed based on the costs of the transmission systems as described in [10]. It makes full use of the economy of scale present in SONET equipment cost. The access design problem can then be written as

$$\min_{\mathbf{x},\mathbf{y}} z = \sum_{i \in M} \sum_{j \in M} c_{ij} + \sum_{i \in M} \sum_{j \in N} c'_{ij} \tag{1}$$

$$|M| = \sum_{i \in M} \sum_{j \in M} x_{ij} + \sum_{i \in M} \sum_{k \in N} y_i^k \tag{2}$$

$$1 = \sum_{k \in N} z_i^k \quad (\forall i \in M) \tag{3}$$

$$n_k \geq \sum_{i \in M} \tau_i z_i^k \quad (\forall k \in N) \tag{4}$$

and $\mathbf{x}$ and $\mathbf{y}$ are 0-1 variables.

The objective function (1) is simply the total connection cost. Restrictions (2) and (3) indicate that the resulting solution must be a tree and Eq. (4) is the capacity constraint on the ports. Note that the introduction of variables **z** simplifies the presentation but would not necessarily be the best one for solving the problem by an ILP method.

Because of the form of the cost function, the problem is too difficult to tackle by a generic ILP procedure. For this reason, we have chosen Tabu search as a solution method. The specific features of our algorithm will be given in the next Section.

3 Tabu Search Heuristic

This Section is divided as follows. In the first Subsection we define the general features of the tabu search procedure. We devote the second Subsection to the explanations of a simple tabu search against which we will compare our work. The proposed method is outlined in the last Subsection.

3.1 Basic features

Solutions and neighbors: First of all, the solution technique is such that at any given iteration, the current solution is a tree. A neighbor of that solution can be any of the trees obtained by a cut-and-paste operation that still keeps a tree topology, i.e., constraints (2–3) are always feasible. In fact, infeasibility is only allowed with respect to the port capacity constraints (4).

Quality of the solution: For feasible solutions, the quality of the solution is simply evaluated by the objective function z. If the solutions are infeasible, a penalized objective $\tilde{z}$ is defined as follows. Let n_k be the capacity of port k and let Q_i be the cumulative demand of a tree rooted at node i (this is the sum of the demands for all the users attached to the tree). The amount of constraint violation for this port is defined as $Q_k^+ = \max(0, Q_k - n_k)$. Given a penalty coefficient α, the penalized cost of the spanning tree, denoted $\widetilde{z}$, is given by

$$\widetilde{z} = z + \alpha \sum_{k=1}^{m} Q_k^+ \tag{5}$$

where z is the current value of the objective function.

Moves: A move is defined by a triplet of node numbers (i, j, k). The link between i and its father in the current tree is removed and the link (j, k) is added to reconnect the tree. Based on this, we define the move variables as

$m^*(i, j, k)$ the move that would yield a feasible tree with a cost z lower than that of the best feasible tree so far,

$m_b(i, j, k)$ the move not on the tabu list that produces the infeasible tree with the least penalized cost $\tilde{z}$ and

$m_g(i, j, k)$ the tabu move that yields the lowest penalized cost.

3.2 Simple tabu search

The starting point against which our work will be compared is a straightforward solution technique used in [7] that we now briefly describe.

For every iteration, the algorithm tries out *all* possible cut-and-paste operations as suggested in [14]. Following the recommendations in [14], a move is declared tabu simply by stating that the two links involved in the cut and paste operation are themselves tabu for k iterations where k is randomly chosen in a given interval. Thus, there is no distinct tabu list and the tabu character of a link is stored in the data structure. It is not possible to really control this parameter in order to improve the running time.

Two solutions are kept for every iteration: the best current feasible solution, which is updated only when a better feasible solution is found, and the working solution which is transformed by potentially infeasible moves. Then, for every iteration, we get two kind of moves: one for feasible and one for infeasible trees. Another feature worth mentioning is that as soon as a link improves the solution, it looses its tabu status. This increases the risk of cycling since it is equivalent to using a tabu list of length one. We can summarize the main handicaps of the method as follows: search over all the neighborhood, random nature of the tabu list and choice of the move at each iteration. These and other elements are modified in the proposed method described in the next subsection.

3.3 Proposed Method

The simple heuristic described above is extremely time-consuming. We now describe the changes that are required to reduce the computation time sufficiently that the method can be used within an integrated network design procedure.

3.3.1 Neighborhood and Moves Numerical results indicated that most, if not all, cost reductions were obtained for a class of moves denoted $m(i, k)$ where a sub-tree rooted at i is cut by removing the link between i and its father and is reconnected to node k by link (i, k). The difference in the moves allowed by the original and the proposed method are illustrated in Figs. 3 and 4. On these figures, the link that is removed is shown in dotted lines and the link that is added is in bold. With the modified neighborhood, the move described in the first of these figures is no longer possible and only the moves similar to the one described in the second figure are now allowed.The traffic carried on those links where there is a change is indicated both before and after the move.

The most important advantage of such a move is that it is not necessary to update the topology or the traffic of the subtree at i since it is moved as a whole. This produces substantial reductions in computation time. The dimensioning of the network after the move is much simpler as well since only the portion above nodes i and k has to be recalculated.

We underline, however, that this class of moves explores only a subset of the neighborhood and could miss out some solutions. Nevertheless, we will later see that this rarely occurs and that the cost penalty incurred from limiting all explorations to moves of this type seems acceptable.

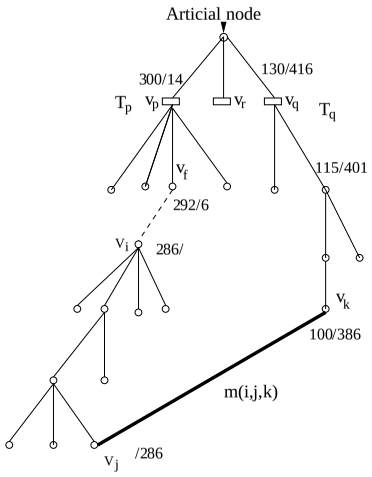

Figure 3: Move in Simple Heuristic

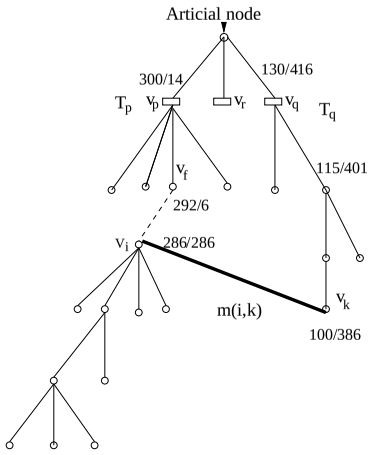

Figure 4: Moves allowed in Modified Heuristic

3.3.2 Tabu Lists For each move $m(i,k)$, we keep the inverse move, $m(i,f)$ (f being the father of i before the move), in a list of size L so that link (f,i) cannot be put back into the network for the next L iterations. One advantage of this deterministic tabu list is that it gives better control over the time during which the link stays tabu. This is important if we want to minimize CPU time. Another point is that we do not scan all network links at each iteration to update their tabu status, which had to be done in the previous method.

3.3.3 Aspiration and Diversification The aspiration criterion is whether the use of a link yields a feasible solution of lower cost than the currently best feasible solution. When a link meets this aspiration criterion, it is removed from the list.

Concerning solutions diversification, generic TS heuristics often use a long-term memory to force the search into new unexplored regions. Two techniques are usually used: 1) producing initial solutions from preceding search statistics or 2)use a second tabu list that contains moves that occur frequently.

In our case, recall that the network that we are producing will be used within a general search for the integrated design. For that reason, we feel that a high accuracy is not needed and that restarts with new initial solutions will not be very useful. We tried instead to force diversification within a given search.

This has been implemented as a second tabu criterion based on the frequency with which the father of a given node is changed. We use a counter that indicates how many times the node has changed father since the last feasible solution was found. When this counter exceeds a given threshold, no further move is allowed for this node. The only exception is when the move would lead to a better feasible solution as explained by the aspiration criterion.

3.3.4 Data Structure There are two operations that are frequently encountered during the procedure. One is to find the end points of a link given its number and the other is the opposite, find the link number given the end points. When a linked list representation of the tree is used, it is necessary to traverse the tree at least partially in order to find this information. A substantial reduction in overall computation time can be obtained if this information is stored into an $n \times n$ node-arc matrix where a_{ij} is the number of the arc from i to j and n is the total number of nodes including the ports. Although this matrix needs an update, the modification is much faster than the repeated traversals required in the simple implementation.

The addition of a second tabu list mentioned in Subsection 3.3.3 requires another modification of the data structure: a field has been added to the node data to indicate the number of times this node has been reconnected to a different father.

3.3.5 Stopping Rule The algorithm stops if there have been $K = 100 \times n$ iterations without improving the feasible solution. The advantage of this simple rule is that it depends on the size of the problem. Therefore, we have less chance of overlooking a good solution in a large problem than if we used a fixed iteration count as the stopping rule.

3.3.6 Algorithm The algorithm can be described as follows, with s being the number of iterations without feasible solution.

1. Initialization.
2. Search in the current neighborhood and calculate $m^*(i,j,k)$, $m_b(i,j,k)$ with $i = j$.
3. If $m^*(i,j,k)$ exists, update the current solution and $s = 0$. This is equivalent to making a move that improves the cost while leading to a feasible tree. This move is made even if it is tabu.
4. Otherwise, apply the move $m_b(i,j,k)$, increase s. In other words, if we cannot improve the solution by a feasible move, we make the best penalized move.
5. Update tree and tabu list.
6. Stop when $s = 100 \times N$
7. End.

4 Numerical Results

Two types of tests networks have been used to evaluate our procedure: First, a group of 20, 30 and 40 node networks randomly constructed and second, a real access network from a carrier. Data for these test networks can be obtained from the first author. We present results to estimate the impact of the choice of parameters on the computation time and the solution quality. Complete results are given in [10]. We will also see that the actual

Initial Solution: Cost= 5786, Penalized Cost= 11686			First Feasible Solution Cost= 7373	
Heuristic	Final Solution	CPU Time (sec)	Improvement	No Iterations
HE	5706	432	22.6%	14
HP	5657	32	23.27%	81
HPD	5571	18	24.44%	121

Table 1: 20-node Network

computation time is sufficiently small to use the heuristic in the general integrated design method.

In all that follows, we denote by HE the Tabu Search described in [7] and by HP our proposal. The variant with diversification will be denoted by HPD.

4.1 Random Test Networks

We present some results showing how each modification to the simple tabu search has improved either the computation time, the value of the solution or both.

4.1.1 Neighborhood Reduction and Fixed-length Tabu List The reduction proposed in Section 3.3.1 combined with a fixed-length Tabu List gives algorithm HP. The results obtained for the three test networks results are described in Tables 1, 2 and 3.

The impact of using the reduced neighborhood and a FIFO Tabu List can clearly be seen in these tables. These results confirm what we said before to the effect that the reduction of the neighborhood does decrease significantly the overall CPU time, in the case of the 40-node network, by a factor of more than 100. We would expect that this gain would grow larger as the network size increases. Also, note that in the case of the 30-node network, the number of iterations has actually increased significantly but because each iteration is so much simpler, the overall effect is still a drastic reduction in CPU time. There is, of course, a price to pay for this since we are systematically neglecting certain moves. In the case of the 40-node network, the cost is almost 9% over what could be obtained by the exploration of the full neighborhood.

4.1.2 Diversification Diversification turned out to be a very important improvement to the procedure, as can be seen from Tables 1, 2 and 3. In fact, it provides a substantial reduction both of the number of iterations and of the overall CPU time while offering an improvement of the final value of the objective over what is found by the algorithm without diversification. In these tables, the relative improvements are given with respect to the first feasible solution.

4.2 Carrier Network

The network provided by a carrier has 46 users and 4 switches and we have assumed that there was a single port per switch. The user demands vary widely from a low value of 9

Initial Solution: Cost= 31806, Penalized Cost= 33156			First Feasible Solution Cost= 31860	
Heuristic	Final Solution	CPU Time (sec)	Improvement	No Iterations
HE	27815	2650	12.69%	39
HP	26558	52	16.64%	18
HPD	25329	60	20.50%	156

Table 2: 30-Node Network

Initial Solution: Cost= 21075, Penalized Cost= 33975			First Feasible Solution Cost= 18418	
Heuristic	Final Solution	CPU Time (sec)	Improvement	No Iterations
HE	15622	16380	15.18%	13
HP	15893	155	13.71%	96
HPD	15487	172	15.91%	191

Table 3: 40-Node Network

up to a maximum of 625 Mb/sec. The current topology, that includes a backbone ring connecting 4 switches, is shown on Fig. 5.

Algorithm HE got stuck in the first local minimum and found a solution quite similar to the initial (infeasible) solution found by Kruskal's algorithm as can be seen from Figs. 7 and 8. On these figures, the ring has been replaced by the artificial node that connects the 4 switches. We can compare the network topology obtained by the HP and HPD algorithms on Figs. 9 and 10. By examining the network of Fig. 9, one may question the presence of some links such as LVCT-LCRN or CHSB-WMDJ. In fact, an analysis of the actual costs and demands indicated that connecting LVCT to VLDR would exceed the switch capacity of 2100 since it already has a flow of 2082 and the LVCT user requires 22 units. The same argument holds for the CHSB to WMDJ connection. Even if the AMOS switch could take the flow from CHSB, the cost of the CHSB-VLDR path is lower than the cost of the CHSB-AMOS path.

On the other hand, no justification can be found for not replacing the JUTL-AMOS and CHSB-CHPS links by JUTL-LSAR and RDSN-CHPS. In fact, these links are cheaper and there is some available capacity at the VLDR switch. As can be appreciated from Table 4, diversification can produce a significant improvement. We see from Fig. 10 that the algorithm can find these solutions only by using diversification.

We now want to compare the current topology with ring backbone to the one found by our method. Since our solution produces a tree rooted at the artificial node, we should somehow remove some links from the ring to have a tree. This is done by introducing an artificial node and constructing a MST spanning all the sites with the arc length as the metric. The links ROYN-CDLC DPQT-CRCY LCRN-VLDR TSCH-AMOS are thus removed from the current topology and give us a tree. The links are then dimensioned

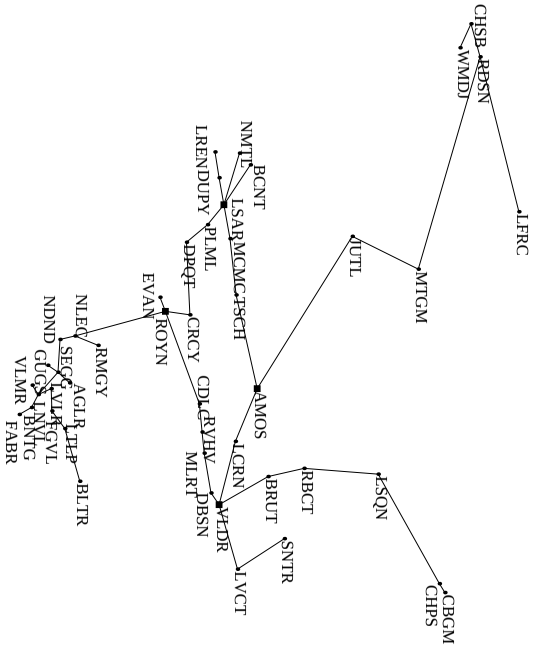

Figure 5: Topology of Carrier Network

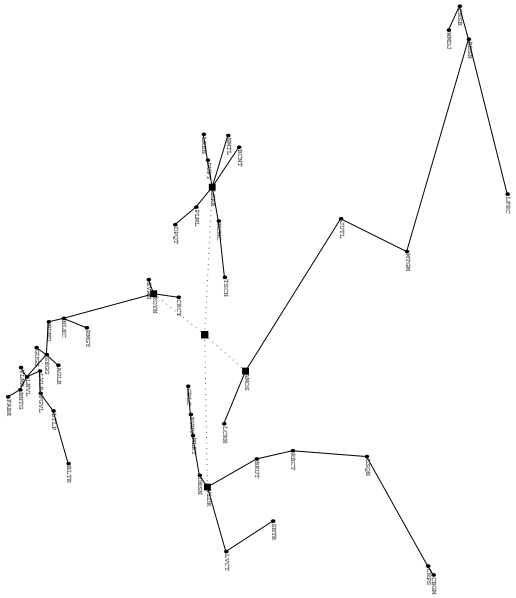

Figure 6: Network without Backbone

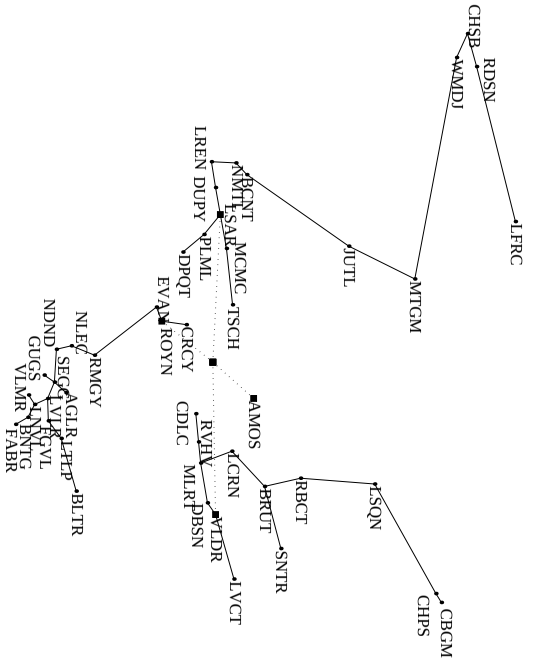

Figure 7: Kruskal's Solution for Carrier Network

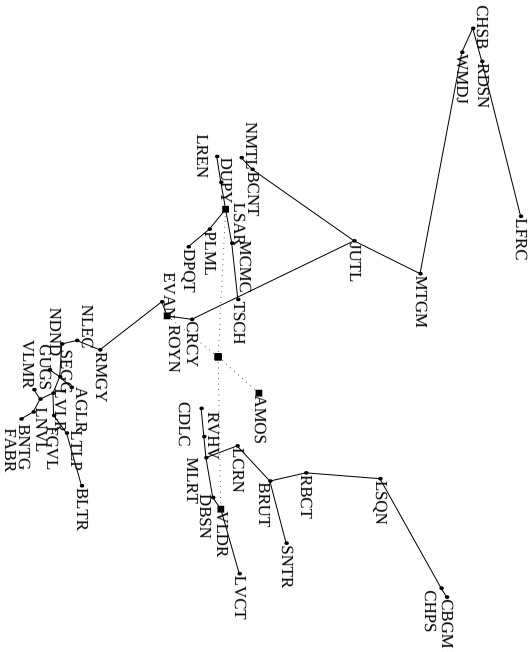

Figure 8: HE Solution for Carrier Network

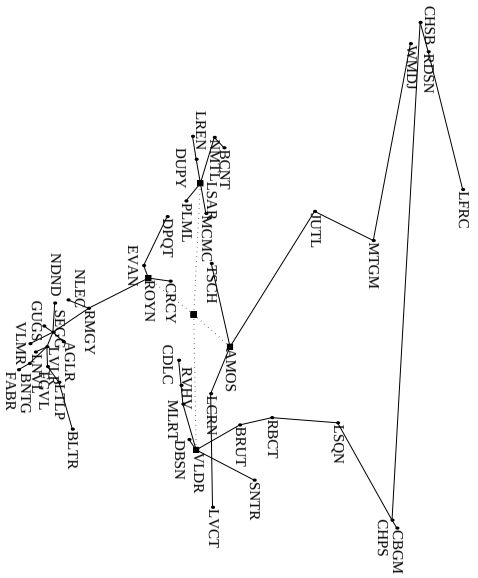

Figure 9: HP Solution for Carrier Network

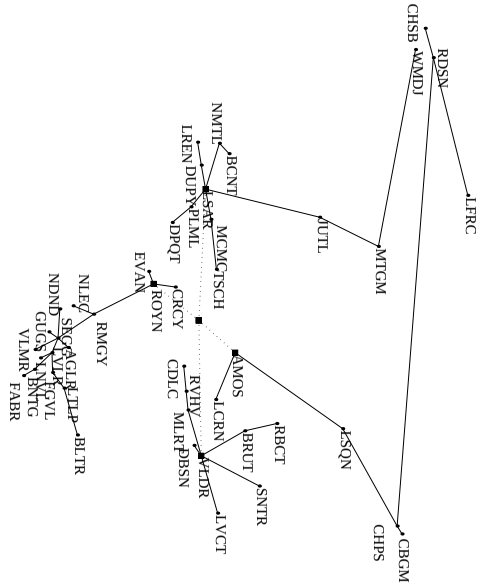

Figure 10: HPD Solution for Carrier Network

Initial Solution: Cost= 199416 Penalized Cost = 214635			1st Feasible Solution Cost= 207605	
Heuristic	Final Solution	CPU Time (sec)	Improvement	No Iterations
HE	204901	20199	1.3%	13
HP	184937	162	10.92%	133
HPD	184840	137	10.97%	215

Table 4: Summary of Results for Real Network

MST on the Current Network. Cost= 197296.19		
Heuristic	Final Solution	Improvement
HE	204901	−3.85%
HP	184937	6.26%
HPD	184840	6.31%

Table 5: Cost Comparison with the Current Network

taking into account the real cost function. The cost comparison is shown in Table 5 and the topology is shown on Fig. 6. We see that the simple heuristic HE actually produces a solution of larger cost than the current network but that the two improved heuristics manage to obtain a small cost reduction of about 6%.

5 Conclusions

We have proposed a Tabu Search method for an access network design problem. The model is new since it considers the use of ADM equipment at the nodes so that the demand can be sent on different SONET channels if that reduces costs. First, we briefly explained a simple version of the procedure and underlined its shortcoming in terms of computation time. With this in mind, and given that our method will be used in an integrated network design, we proposed specific changes to improve the procedure. The most important were the neighborhood reduction, a fast data structure, a modification of the exploration strategy as well as a diversification strategy. All these led to significant improvements in CPU times, sometimes by a factor of over 100 over the simple heuristic, as well as to improved solutions for most cases.

Concerning further work, there is always the nagging question of the optimality of the solution. Current work is being done to answer this question. And since the problem was framed in the context of an integrated backbone/access design problem, the integration of the algorithm presented here into the overall method still remains to be done.

References

[1] K. Altinkemer and H. Pirkul. Heuristics with constant error guarantees for the multi center capacitated minimum spanning tree problem. *Journal of Information & Optimization Sciences*, 13(1):49–71, 1992.

[2] S. Chamberland and B. Sansò. Heuristics for the topological design problem of two-level multitechnology telecommunication networks with modular switches. *INFORMS Journal on Computing*, to appear in 2001, GERAD G97-69.

[3] S. Chamberland and B. Sansò. Update of two-level networks with modular switches Proceedings of ITC16, 1009-1018, 1999.

[4] S. Chamberland, B. Sansò, and O. Marcotte. Heuristics for ring network design when several types of switches are available. In *Proc. IEEE International Communications Conference*, 570-574, 1997.

[5] S. Chamberland and B. Sansò. Expansion of Multiple Ring MANs. Proceedings of Globecom 98, 2501-2507, 1998.

[6] S. Chamberland, B. Sansò, and O. Marcotte. Topological design of two-level telecommunication networks with modular switches. *Operations Research*, Vol 48, No 5, pp. 745–760, 2000

[7] M. El-Hafa, A. Girard, and B. Sansò. Design of core and edge broadband networks with tree access topology. In *Optimization Days*, 1999.

[8] L.R. Esau and K.C. Williams. On teleprocessing system design. *IBM System Journal*, 5:142–147, 1966.

[9] B. Gavish. Topological design of telecommunication networks—local access design methods. *Annals of Operations Research*, 33:17–71, December 1991.

[10] A. Girard, B. Sansò, and L. Dadjo. A tabu search algorithm for access network design. Submitted to Annals of Operations Research, 2000. GERAD G2000-48

[11] M. Goldstein and B. Rothfarb. The one-terminal telpack problem. *Operations Research*, 19:156–169, 1971.

[12] A. Kershenbaum. Computing capacitated minimal spanning trees efficiently. *Networks*, 4:299–310, 1974.

[13] J.G. Klincewicz. Hub location in backbone/tributary network design. *Location Science*, to appear.

[14] Y. M. Sharaiha, M. Gendreau, G. Laporte, and I. H. Osman. A tabu search algorithm for the capacitated shortest spanning tree problem. *Networks*, 29(3):161–171, 1997.