Access network design with capacity-dependent costs

Olivier Brun
LAAS-CNRS
Université de Toulouse
7, Avenue du Colonel Roche
F-31077 Toulouse, France
brun@laas.fr

Anouar Rachdi
LAAS-CNRS
Université de Toulouse
7, Avenue du Colonel Roche
F-31077 Toulouse, France
marachdi@laas.fr

Jean-Marie Garcia
LAAS-CNRS
Université de Toulouse
7, Avenue du Colonel Roche
F-31077 Toulouse, France
jmg@laas.fr

ABSTRACT

This article adresses the topological design problem of access networks. The topology of such networks is often designed assuming capacity-independent costs for the links and neglecting other equipment costs. However, with the massive deployment of optical fiber in all western countries, the cost of leasing transmission lines becomes cheaper and cheaper, and equipment costs are now a significant fraction of the total cost when designing a network. The main contribution of this paper is to integrate the cost of the equipements in the topological design of access networks. An exact algorithm and two heuristics are proposed to solve this problem.

1. INTRODUCTION

In a world where information technology (IT) is becoming pervasive, communication networks appear more and more as strategic resources. Near-optimal design of these networks is a critical issue since network design greatly affects the long-term network performances and it determines most of the investment cost. Since this investment cost is typically huge and since the return on investement cannot be expected before years, it is crucial to ensure that it is properly minimized.

Network operators as well as many governemental organisations operating in strategic areas (e.g. defense, air control, energy, etc.) have always been aware of the critical importance of efficient network design approaches. Disappointed by the lack of quality and flexbility of the VPN (Virtual Private Network) services offered by network operators, many large companies are now changing their strategy with respect to the governance of their IT infrastructure. The IT department of these companies are now investigating the opportunity to acquire their own communication infrastructure as well as the skills to design an optimized network architecture.

Therefore there is a renewed interest in efficient network design methods. The design of a WAN usually follows a (suboptimal) decomposition approach [1]. The first phase concerns the topological design of the access and backbone networks, i.e. the decisions where to install network equipments (nodes and links) in order to interconnect several sites distributed over a wide area. It is followed by the capacity-planning of the network architecture, i.e., the dimensionning of the communication links connecting the different sites and the equipment (routers, line cards) to be settled in these sites. Finally, the last phase deals with the definition of a routing strategy in the backbone network.

The rationale for the separation between topological design and equipement dimensionning is that the former incurs capacity-independent "fixed" costs which are several orders of magnitude larger than the equipment costs. These " fixed" costs typicaly represent the costs of digging trenches for optical fibers, site opening costs, or even equipment installation and configuration costs. As a consequence, it is frequently assumed that link costs are independent of the type of communication line that will effectively be installed and that other equipment costs (routers, communication cards) can be neglected in this first design phase.

This article adresses the topological design problem of access networks (for the backbone network design problem, refer to [5, 6, 7, 8]). Previous works on this problem include [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2, 4]. These works propose different approximate algorithms for the topological design of local and large-scale access networks. They consider different parameters and restrictions, including specific access network topologies (e.g. star, tree or rings) or a limit on the number of concentrators to be placed. Most of these works assume capacity-independent link costs, but some others assume modular link capacities and use capacity-dependent link costs. The optimization methods used in these works include Lagrangian relaxation with subgradient optimization [18], Simulated Annealing [17], Linear Programming Relaxation [10], Greedy Heuristics [16], Branch-and-Bound with Benders decomposition [19, 3], Neural Networks [9], Tabu Search [15] and Greedy Randomized Adaptive Search heuristics [4].

However, none of these previous works has considered the cost of the equipments (routers and line cards) to be settled in the selected access sites. In our opinion, this is a major issue. Indeed, with the massive deployment of optical fiber in all western countries, the cost of leasing transmission lines becomes cheaper and cheaper, and costs of routers and line cards are now a significant fraction of the total cost when designing a network. Therefore, there is an increasing need for an integrated approach of network design problems taking into account equipment costs in the early stages of the

ValueTools 2008, October 21-23, 2008, Athens, GREECE.

design process.

The main contribution of this paper is to integrate the cost of the equipements (link, line cards and routers) in the topological design of access networks. We first solve a dimensionning problem, where the equipment cost is computed for each possible equipment configuration of an access site. The solution of the dimensionning problem is then used to solve the access network topology problem, where we have to assign each terminal node to an access site. Several optimization algorithms are proposed. The first one is a Branch-and-Cut algorithm that can be used to obtain the optimal solution of small to medium access network design problems. The second one is an efficient heuristic combining a clustering algorithm and a local search algorithm. Results show that it often provides near-optimal solutions in very low computing times. Finally, the last algorithm is a tradeoff between the two previous ones based on the idea of limited discrepancy search: it can be used to improve the solutions provided by the clustering heuristic in limited computing times.

The paper is organized as follows. Section 2 states the problem. Section 3 is devoted to the dimensionning problem. Section 4 describes the Branch-and-Cut algorithm allowing to solve the problem to optimality, while sections 5 and 6 describe the approximation algorithms. Section 7 present the results of these algorithms on several problem instances. Finally, some conclusions are drawn in section 8.

2. PROBLEM STATEMENT

The problem can be stated as follows. We are given N terminal nodes to be connected by means of communication links to routers located in K possible access sites. The problem is to determine the sites to be opened, the equipments to be installed in each of the opened site and the terminal nodes connected to each site so that the total network cost is minimized.

2.1 Notations

Let I be the number of link/port types and let r_t be the bandwidth of the link/port type $t = 1, \ldots, I$. It is assumed that the traffic demand of each terminal node $i = 1, \ldots, N$, can be determined from the analysis of the traffic matrix and thus that a single link/port type can be assigned to each terminal node. In the following, Φ_t will denote the set of terminal nodes that have to be connected to the network using a link of type $t = 1, \ldots, I$.

A link of type $t = 1, \ldots, I$ needs to be connected to an interface card. Let p_t be the number of available ports on a line card of type $t = 1, \ldots, I$, and let ϕ_t be the cost of such a line card (including installation cost).

It is assumed that the routers to be installed in an opened site belong to a set of R router types. For router model $r = 1, \ldots, R$, let

- $\overline{s}_r$ be the number of available slots where line cards can be plugged.
- $\overline{T}_r$ be the maximum throughput of the router forwarding engine,
- ψ_r be the cost of such a router (including installation cost).

A salient feature of the proposed model is that, on the contrary to most previous works, the number of routers per opened site is not limited to one.

2.2 Feasible solutions

For each terminal node $i = 1, \ldots, N$ and each access site $j = 1, \ldots, K$, let us define the following 0-1 decision variable,

$$x_{i,j} = \begin{cases} 1 & \text{if terminal node } i \text{ is assigned to site } j \\ 0 & \text{otherwise} \end{cases}$$

A feasible assignment of terminal nodes to sites is a vector $\mathbf{x} = [x_{i,j}]$ such that,

$$\begin{aligned} \sum_j x_{i,j} &= 1 \quad i = 1 \ldots N \\ x_{i,j} &\in \{0,1\} \quad i = 1, \ldots, N,\ j = 1, \ldots, K \end{aligned}$$

Let X be the set of all feasible solutions. Given a solution $\mathbf{x} \in X$, let $a(\mathbf{x}, i)$ denotes the access site assigned to terminal node i, i.e.,

$$a(\mathbf{x}, i) = \sum_{j=1}^{K} j\, x_{i,j}$$

2.3 Objective Function

The cost of a solution $\mathbf{x} \in X$ is the sum of link costs, site opening costs and equipment costs.

2.3.1 Link costs

Let $c_{i,j}$ be the link cost (including the installation cost) of connecting the user i to site j. Therefore, the global link cost of solution $\mathbf{x} \in X$ is given by,

$$\sum_{i=1}^{N} \sum_{j=1}^{K} x_{i,j}\, c_{i,j}$$

Although no specific assumption on link costs is done in the following, these costs are typicaly capacity-dependent costs and are non-decreasing functions of the euclidean distance between the terminal node and the site. These costs are typically piecewise linear in the length and piecewise linear in the capacity and correspond, e.g., to a tariff system used by a provider of leased lines.

Note also that, as link costs are charged, e.g., on a monthly basis, whereas equipments have to be payed for once, we need an amortization period to make these two types of costs comparable.

2.3.2 Site opening costs

If at least one terminal node is assigned to site j, then an opening cost L_j is incurred. For each $\mathbf{x} \in X$ and each site j, let us define the following function,

$$u_j(\mathbf{x}) = \begin{cases} 1 & \text{if } \sum_{i=1}^{N} x_{i,j} \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

The function $u_j(\mathbf{x})$ indicates whether the site j has to be opened or not in the solution x. The site opening cost of solution $\mathbf{x}$ can then be stated as follows,

$$\sum_{j=1}^{K} u_j(\mathbf{x})\, L_j$$

2.3.3 Equipment costs

Moreover, if site j is opened, equipments (routers and line cards) need to be settled for this site. The key observation here is that the minimum cost of the equipments to be installed in a given site does not depend on the particular assignment considered, but only on the number of line cards of each type required to support users assigned to it.

Given a solution $\mathbf{x} \in X$, let $p_j^t(\mathbf{x})$ denotes the minimum number of cards of type $t = 1, \ldots, I$ required to support terminal nodes connected to site j, i.e.,

$$p_j^t(\mathbf{x}) = \left\lceil \frac{\sum_{i \in \Phi_t} x_{i,j}}{p_t} \right\rceil$$

where for any real-valued z, $\lceil z \rceil$ denotes the lowest integer n such that $z \leq n$.

Let $\mu^*(\mathbf{p}_j(\mathbf{x}))$ denotes the optimal equipment cost for site j in the solution $\mathbf{x} \in X$, where $\mathbf{p}_j(\mathbf{x})$ is the vector $\left[p_j^1(\mathbf{x}), \ldots, p_j^I(\mathbf{x})\right]$. The global equipment cost is then given by,

$$\sum_{j=1}^{K} \mu^* \left(\mathbf{p}_j(\mathbf{x})\right)$$

2.3.4 Objective function

Finally, the cost $\Gamma(x)$ associated to a solution $\mathbf{x} \in X$ is given by,

$$\Gamma(\mathbf{x}) = \sum_{i=1}^{N} \sum_{j=1}^{K} x_{i,j}\, c_{i,j} \;+\; \sum_{j=1}^{K} E_j(\mathbf{x})$$

where $E_j(\mathbf{x}) = u_j(\mathbf{x})\, L_j \;+\; \mu^* \left(\mathbf{p}_j(\mathbf{x})\right)$ is the cost associated to site j in solution $\mathbf{x} \in X$.

2.4 Mathematical model

The problem to be solved can be stated as follows

$$\min_{\mathbf{x} \in X} \; \Gamma(\mathbf{x}) = \sum_{i=1}^{N} \sum_{j=1}^{K} x_{i,j}\, c_{i,j} \;+\; \sum_{j=1}^{K} E_j(\mathbf{x}). \qquad (1)$$

Note that equipment cost and link cost depend on the number of access sites used: if we use many access sites, we will have to pay for many small equipments, which tend to be more expensive than fewer larger ones (economy of scale); so equipment cost will increase as the number of access sites increases. On the other hand, with few access sites, we need longer links to connect terminal nodes to access sites, leading to higher link cost. Obviously, there is a tradeoff and we need to find the number of (and assignment of terminal nodes to) access sites that minimizes total cost.

This problem can be formulated as a standard Integer Programming (IP) problem. Since there is no need for the designer to solve it on-line, one could argue that generic IP-based algorithms can give the optimal solution. However, such an approach would lead to days or even weeks of processing times before the optimal solution is obtained for most problem instances of interest. This is simply unacceptable for network designer. To reduce processing times, the structure of the problem has to be exploited in order to design algorithms specifically tailored for it.

Exact and heuristic algorithms to solve this problem will be presented in section 4, 5 and 6. The following section is devoted to the analysis of the function μ^*.

3. DIMENSIONNING PROBLEM

3.1 Card configurations

As stated above, the optimal cost of the equipments to be settled at site j in solution $\mathbf{x} \in X$ is a function of $\mathbf{p}_j(\mathbf{x})$. The latter is a vector of the form $\mathbf{s} = (s_1, \ldots, s_I)$ where s_t is a number of type t line cards. In the following, such a vector will be called a card configuration.

The maximum number of terminal nodes of type $t = 1, \ldots, I$ that can be assigned to a single site is $|\Phi_t|$. Therefore, the maximum number of cards of type t that needs to be considered is $h_t = |\Phi_t|/p_t$.

The set of all possible card configurations is thus given by,

$$\Lambda = \left\{ \mathbf{s} = \sum_{t=1}^{I} s_t\, \mathbf{e}_t \;\mid\; 0 \leq s_t \leq h_t\,,\; t = 1, \ldots, I \right\}$$

where $\mathbf{e}_t$ is the vector $(0, \ldots, 1, \ldots, 0)$ with 1 in position t and 0 elsewhere.

The total number of card configurations is $H = |\Lambda| = \prod_{t=1}^{I} (h_t + 1)$. Note that Λ contains the null configuration $(0, \ldots, 0)$ as well as the centralized configuration $(h_1, \ldots, h_I)$ allowing to connect all terminal nodes to a single site.

The card configurations can be numbered from 0 to $H - 1$, where the index $h(\mathbf{s})$ associated to configuration $\mathbf{s} = (s_1, \ldots, s_I)$ is given by,

$$h(\mathbf{s}) = s_I + \sum_{t=1}^{I-1} s_t \prod_{u=t+1}^{I} (h_u + 1) \quad \forall s \in \Lambda$$

This numbering scheme is illustrated in table 1 in the case $I = 3$ with $h_1 = h_2 = h_3 = 2$. In the following, $\mathbf{s}(h)$ will denote the configuration associated to index $h = 0, \ldots, H - 1$.

Table 1: Example of configurations in the case $I = 3$ with $h_1 = h_2 = h_3 = 2$.

Configuration $\mathbf{s}$	Index of $\mathbf{s}$
$(0, 0, 2)$	2
$(0, 1, 0)$	3
$(0, 2, 0)$	6
$(0, 2, 2)$	8
$(1, 0, 0)$	9
$(2, 2, 2)$	26

For each card configurations $\mathbf{s}, \mathbf{s}' \in \Lambda$, we say that $\mathbf{s} \leq \mathbf{s}'$ if and only if $s_t \leq s'_t$, $t = 1, \ldots, I$.

3.2 Configuration costs

In this section, our goal is to build the mapping $\mu^* : \Lambda \rightarrow \mathbb{R}$ such that $\mu^*(\mathbf{s})$ is the minimum cost of the equipements enabling to support the configuration $\mathbf{s} \in \Lambda$. This optimal cost has two components: the cost of the line cards and the minimum cost of a set of routers accomodating the cards. Since the cost of the cards is known in advance, $\mu^*(\mathbf{s})$ can be written as follows,

$$\mu^*(\mathbf{s}) = \sum_{t=1}^{I} s_t \, \phi_t \;+\; \mu_r^*(\mathbf{s}) \quad \forall \mathbf{s} \in \Lambda$$

where $\mu_r^*(\mathbf{s})$ denotes the cost of the set of routers associated to configuration $\mathbf{s} \in \Lambda$.

Let Λ_j be the set of card configurations that can be accomodated by a single router of type $j = 1, \ldots, R$,

$$\Lambda_j = \left\{ \mathbf{s} \in \Lambda \mid \sum_{t=1}^{I} s_t \leq \overline{s}_j \text{ and } \sum_{t=1}^{I} s_t \, p_t \leq \overline{T}_j \right\}$$

Thanks to Bellman optimality principle [1], the optimal cost of the set of routers associated to a card configuration $\mathbf{s} \in \Lambda$ can be written as follows,

$$\mu_r^*(s) = \min_{j=1,\ldots,R} \left[\min_{s_j \in \Lambda_j, s_j \leq s} (\psi_j \;+\; \mu_r^*(s - s_j)) \right]$$

This suggests that the function $\mu^*() : \Lambda \rightarrow \mathbb{R}$ can be computed using algorithm 1.

Algorithm 1 Algorithm to compute the optimal cost of each configuration.

procedure CONFIGURATIONCOST
  for $h = 0 \ldots H-1$ **do** ▷ compute sets Λ_j, $j = 1, \ldots, R$
    $\mathbf{s} = \mathbf{s}(h)$
    for $j = 1 \ldots R$ **do**
      if $\sum_{t=1}^{I} s_t \leq \overline{s}_j$ and $\sum_{t=1}^{I} s_t \, p_t \leq \overline{T}_j$ **then**
        $\Lambda_j = \Lambda_j \cup \{\mathbf{s}\}$
      end if
    end for
  end for
  $\mu_r^*(\mathbf{0}) = 0$ ▷ compute optimal router costs $\mu_r^*(\mathbf{s})$, $\mathbf{s} \in \Lambda$
  for $h = 0 \ldots H-1$ **do**
    $\mathbf{s} = \mathbf{s}(h)$, $\mu_r^*(\mathbf{s}) = \infty$
    for $\mathbf{s}' \in \Lambda$, $\mathbf{s}' \neq \mathbf{0}$ and $\mathbf{s}' \leq \mathbf{s}$ **do**
      for $j = 1, \ldots, R$ **do**
        if $\mathbf{s}' \in \Lambda_j$ **then**
          $\mu_r^*(\mathbf{s}) = \min\left(\mu_r^*(s)\,,\, \psi_j + \mu_r^*(\mathbf{s} - \mathbf{s}')\right)$
        end if
      end for
    end for
    $\mu^*(\mathbf{s}) = \mu_r^*(\mathbf{s}) \;+\; \sum_{t=1}^{I} s_t \, \phi_t$
  end for
end procedure

Table 2: Computing times to build the mapping $\mu^* : \Lambda \rightarrow \mathbb{R}$.

$\lvert\Phi_1\rvert$	$\lvert\Phi_2\rvert$	$\lvert\Phi_3\rvert$	$\lvert\Phi_4\rvert$	# configurations	CPU time (s)
60	30	20	0	84	0.0008
60	30	40	0	140	0.002
60	60	40	0	245	0.005
120	60	40	0	455	0.014
60	30	20	16	252	0.005
60	30	40	16	420	0.011
60	60	40	16	735	0.035
120	60	40	16	1365	0.080
60	30	20	32	420	0.012
60	30	40	32	700	0.029
60	60	40	32	1225	0.067
120	60	40	32	2275	0.180

3.3 A simple example

In order to demonstrate the efficiency of the above algorithm, let us consider the following very simple example. Let us assume that there are 4 types of line cards available:

- Type 1: p_1=10 ports, r_1=2 Mbps, ϕ_1= 3,000 €,
- Type 2: p_2=10 ports, r_2=10 Mbps, ϕ_2= 10,000 €,
- Type 3: p_3=10 ports, r_3=100 Mbps, ϕ_3= 80,000 €,
- Type 4: p_3=8 ports, r_3=256 Mbps, ϕ_3= 100,000 €.

There are also 3 router models available:

- Model 1: $\overline{s}_1$=15 slots, $\overline{T}_1$= 1 Gbps, ψ_1=100,000 €,
- Model 2: $\overline{s}_2$=8 slots, $\overline{T}_1$= 3 Gbps, ψ_1=140,000 €,
- Model 3: $\overline{s}_2$=4 slots, $\overline{T}_1$= 4 Gbps, ψ_1=155,000 €.

We considered several scenarios with different values of the number $|\Phi_i|$ of type i=1,2,3,4 terminals. Table 2 reports the total number of configurations for each scenario as well as the computing time to solve all the configurations for each scenario. These results show clearly that the dimensionning subproblem can be solved efficiently.

3.4 Extensions

For ease of presentation, we have assumed that each router can accomodate any card model, provided its forwarding engine has enough bandwidth. In practice, one has to deal with compatibility constraints that do not allow to plug some line cards on a router. These compatibility constraints can easily be integrated in our model by incorporating them in the definition of the set Λ_j of card configurations that can be accomodated by a router of type j: if line card t is not supported by this router model, then all card configurations $\mathbf{s}$ such that $s_t > 0$ do not belong to Λ_j.

We have also assumed a one-to-one correspondance between link rates and line card rates. In practice, it often happens that the same link rate is provided by several line cards, with different numbers of ports and different costs. Our model can be extended to handle this case by considering link configurations in addition to card configurations.

4. EXACT ALGORITHM FOR THE ASSIGNMENT PROBLEM

Assuming that the mapping μ^* has been computed, we now consider the assignment problem 1. This problem can be solved using the branch-and-cut algorithm 2 (note that $E_j(x)$ is used here to denote the cost of site j for any partial solution x).

This recursive algorithm takes as input the current partial solution x, the next terminal node i to be assigned, the cost of the current partial solution and an upper bound ub on the optimal cost. It performs a tree search using an upper bound and a lower bound to prune the solution tree. The upper bound is updated each time a complete solution with a cost lower than the upper bound is found. A lower bound on the optimal cost-to-go is generated in each node of the tree. Moreover, a specific value-ordering method is used to vist the more promising solutions first and thus to lower the upper bound as fast as possible. Finally, distance cuts are used in order to avoid the exploration of sub-trees containing only non optimal solutions.

Algorithm 2 Branch-and-cut algorithm for assignment problem

```
1: procedure BC(x, i, ub)
2:     if i > N then                                  ▷ no terminal left
3:         if Γ(x) < ub then
4:             ub=Γ(x) and x* = x
5:         end if
6:         return 0
7:     end if
8:     bestCost=∞
9:     for j ∈ Sites(x, i) do
10:        Let x'_{i,j} = 1 and x'_{n,k} = x_{n,k}   k = 1...K, n < i
11:        costToGo = c_{i,j} + E_j(x') − E_j(x)
12:        lb = costToGo+lowerBound(x');
13:        if Γ(x)+lb < ub then
14:            costToGo = costToGo + BC(x',i + 1, ub);
15:        else
16:            costToGo = lb;
17:        end if
18:        if costToGo < bestCost then
19:            bestCost=costToGo
20:        end if
21:    end for
22:    return bestCost
23: end procedure
```

In the following, we describe the details of this algorithm.

4.1 Initial upper bound

The branch-and-cut algorithm is started with an initial value of the upper bound. It is of course very important that the initial value be as tight as possible in order to prune the search tree as much as possible. In our implementation, we have used the cost returned by the heuristic described in section 5. As it will be shown in section 7, this heuristic often provides near-optimal solutions in very low computing times.

As described in Algorithm 2, the upper bound is updated each time an improving complete solution is discovered.

4.2 Lower bound

A lower bound on the optimal cost-to-go is generated in each node of the search tree. Let us consider a partial solution x such that terminal nodes $1,\ldots,i$ have already been assigned. Let $Subtree(x)$ be the set of complete solutions in the subtree rooted at x, i.e.,

$$Subtree(x) = \left\{x' \in X | a(x',k) = a(x,k) \quad k = 1,\ldots,i\right\}.$$

A lower bound $lb(x)$ on the optimal cost-to-go starting from partial solution x is such that,

$$\Gamma(x') \geq \Gamma(x) + lb(x) \quad x' \in Subtree(x).$$

The cost of any solution $x' \in Subtree(x)$ can be written as follows,

$$\begin{aligned}\Gamma(x') &= \Gamma(x) + \sum_{k=i+1}^{N}\sum_{j} c_{k,j}x'_{k,j} \\ &\quad + \sum_{j}\left[u_j(x') - u_j(x)\right] L_j \\ &\quad + \sum_{j}\left[\mu_r^*(\mathbf{p}_j(x') - \mu_r^*(\mathbf{p}_j(x)\right] \\ &\quad + \sum_{j}\sum_{t=1}^{I}\left[p_j^t(x') - p_j^t(x)\right]\phi_t\end{aligned}$$

Therefore, a lower bound can be obtained on a term-by-term basis:

$$\begin{aligned}\sum_{k=i+1}^{N}\sum_{j} c_{k,j}x'_{k,j} &\geq \sum_{k=i+1}^{N}\min_j c_{k,j} \\ \sum_{j}\left[u_j(x') - u_j(x)\right] L_j &\geq 0 \\ \sum_{j}\left[\mu_r^*(\mathbf{p}_j(x') - \mu_r^*(\mathbf{p}_j(x)\right] &\geq 0 \\ \sum_{j}\sum_{t=1}^{I}\left[p_j^t(x') - p_j^t(x)\right]\phi_t &\geq \sum_{t=1}^{I} max\left(h_t - \sum_j p_j^t(x)\ ,\ 0\right)\phi_t\end{aligned}$$

The last inequality is obtained by observing that,

$$\sum_j p_j^t(x') \geq max\left(h_t, \sum_j p_j^t(x')\right)$$

since the number of line cards of type t in solution x' is obviously greater than in partial solution x and it cannot be lower than the number of line cards h_t in a centralized allocation.

It yields the following lower bound,

$$lb(x) = \sum_{k=i+1}^{N}\min_j c_{k,j} + \sum_{t=1}^{I} max\left(h_t - \sum_j p_j^t(x)\ ,\ 0\right)\phi_t.$$

Note that this lower bound can be reached starting from solution x, if, when assigning each remaining terminal to its nearest site, the number of opened sites is not increased and if no more line cards need to be plugged on already installed routers.

If $\Gamma(x) + lb(x)$ is greater than the current upper bound, then no improving solution is present in $Subtree(x)$ and this branch of the search tree need no be explored.

4.3 Candidate sites

We will use the following proposition.

Proposition 1. *Let i and $i' \neq i$ be two terminal nodes such that $i, i' \in \Phi_t$, $t = 1, \ldots, I$. Let $\mathbf{x} \in X$ be a solution such that i and i' are assigned to access sites k' and $k \neq k'$, respectively. If,*

$$c_{i'k} + c_{ik'} > c_{ik} + c_{i'k'}$$

then, solution $\mathbf{x}$ is not optimal.

Proof
Let us consider a solution $\mathbf{x}$ such that $a(\mathbf{x}, i) = k'$ and $a(\mathbf{x}, i') = k$. Let $\mathbf{x}' \in X$ be the solution obtained by swapping the assignments of terminals i and i' in such a way that $a(\mathbf{x}', i) = k$ and $a(\mathbf{x}', i') = k'$. The assigment of the other terminal nodes is not changed.

The assignments of terminals to access sites $j \neq k, k'$ has not been changed, and therefore the equipment costs are identical for these sites in solutions $\mathbf{x}$ and $\mathbf{x}'$. Moreover, since $i, i' \in \Phi_t$, swapping the assignment of terminal i and i' does not require a configuration change on sites k and k'. Therefore,

$$\sum_j E_j(\mathbf{x}) = \sum_j E_j(\mathbf{x}')$$

Beside,

$$\begin{aligned}
\sum_{u=1}^{N} \sum_j x_{uj}\, c_{uj} &= c_{i'k} + c_{ik'} + \sum_{u \neq i,i'} \sum_j x_{uj}\, c_{uj} \\
&> c_{ik} + c_{i'k'} + \sum_{u \neq i,i'} \sum_j {x'}_{uj}\, c_{uj} \\
&> \sum_u \sum_j {x'}_{uj}\, c_{uj}
\end{aligned}$$

It yields $\Gamma(\mathbf{x}) > \Gamma(\mathbf{x}') \geq \Gamma^*$. Solution $\mathbf{x}$ is therefore not optimal.

The previous proposition can be used to reduce the exploration of the search tree. Let us consider a partial solution $\mathbf{x}$ such that terminal nodes $1, \ldots, i-1$ have already been assigned. According to the previous proposition, the set of access sites $Sites(\mathbf{x}, i)$ to which terminal node $i \in \Phi_t$ can be assigned is the set of sites j such that,

$$\forall i' < i, i' \in \Phi_t,\ c_{i,j} + c_{i',a(\mathbf{x},i')} \leq c_{i,a(\mathbf{x},i')} + c_{i',j}$$

4.4 Value ordering and variable ordering

Let us consider a partial solution $\mathbf{x}$ such that terminal nodes $1, \ldots, i-1$ have already been assigned. The set of candidate sites to which terminal node i can be assigned is restricted to $Sites(\mathbf{x}, i)$. The order in which the assignments of terminal i to sites $j \in Sites(\mathbf{x}, i)$ are considered is important since an efficient value-ordering scheme can enable to improve the upper bound faster and thus to prune the search tree much more.

A natural ordering scheme is to consider the sites $j \in Sites(\mathbf{x}, i)$ in order of increasing link costs $c_{i,j}$. In the following, we assume that this value ordering scheme is used.

The order in which the assigments of terminal nodes to access sites is performed is also important. In our implementation, we have chosen to consider the terminal nodes in order of increasing bandwidth requirement, i.e. to consider first the terminal nodes belonging to Φ_I, then the terminal nodes belonging to Φ_{I-1}, etc.

5. CLUSTERING HEURISTIC FOR THE ASSIGNMENT PROBLEM

A standard approach to approximately solve problems such as (1) is to use lagrangian relaxation and subgradient optimization (see chapter 5 of [1]). Unfortunately, such algorithms performs badly for our problem due to a strong duality gap. We propose in this section another optimization approach combining a clustering algorithm and a local search procedure.

Clustering algorithms have often been used for network design problems because they capture the fundamental geographic nature of such problems. Indeed, terminal nodes are often grouped into clusters and assigned to their nearest access sites in order to minimize the distance cost.

The main ideas underlying the proposed heuristic are the following:

- a problem instance can always be divided into two separate sub-problems by clustering terminal nodes and access sites into two regions,
- if in a problem instance there is a single access site, then there is a single feasible solution,
- merging the solutions of the problem for two separate regions can be done efficiently.

The proposed heuristic is a recursive procedure (see algorithm 3). This algorithm takes as inputs a subset $\mathcal{T} \subseteq \{1, \ldots, N\}$ of terminal nodes and a subset $\mathcal{S} \subseteq \{1, \ldots, K\}$ of access sites, where the equalities hold for the first call to this procedure. It computes the min-cost solution $\mathbf{x}$ for the allocation of terminal nodes $i \in \mathcal{T}$ to access sites $j \in \mathcal{S}$.

The first step of the algorithm is to compute the best centralized allocation, i.e. the solution $\mathbf{x}^{\text{center}}$ such that $a(\mathbf{x}^{\text{center}}, i) = j^*$, $i \in \mathcal{T}$, where

$$j^* = argmin_{j \in \mathcal{S}} \left[\sum_{i \in \mathcal{T}} c_{i,j} + L_j \right].$$

Note that all centralized allocations have the same equipment cost and hence j^* can be computed by considering only distance costs and the site opening cost.

If there are more than one access site and more than one terminal node, i.e. if a distributed allocation is achievable, then the algorithm splits the set of nodes $\mathcal{T} \cup \mathcal{S}$ into two separate regions. These regions are computed on the basis of the node geographical locations using an efficient implementation of the k-Means clustering algorithm. Hereafter, these two regions will be called the "left" region and the "right" region. Let $\mathcal{T}^{\text{left}}$ and $\mathcal{S}^{\text{left}}$ (respectively $\mathcal{T}^{\text{right}}$ and

$\mathcal{S}^{\text{right}}$) be the set of terminal nodes and access sites for the "left" (respectively "right") region.

The next step is to check if feasible allocations are acheivable for the left and right regions. A feasible allocation is achievable for the left region if $\mathcal{T}^{\text{left}} \neq \emptyset$ and $\mathcal{S}^{\text{left}} \neq \emptyset$. Similar conditions are used to check the existence of a feasible solution for the right region. If both regions admit a feasible allocation, then the algorithm is recursivly called on the left and right regions, resulting in two separate allocations: $\mathbf{x}^{\text{left}}$ for the left region and $\mathbf{x}^{\text{right}}$ for the right region.

Algorithm 3 Clustering heuristic for assignment problem

```
1: procedure SplitAndFusion(T, S, x)
2:     j* = argmin_{j∈S} [Σ_{i∈T} c_{i,j} + L_j]
3:     x^center : a(x^center, i) = j*, ∀i ∈ T
4:     x^merge = x^center
5:     if |S| > 1 and |T| > 1 then
6:         (T^left, S^left, T^right, S^right) = Kmean(T ∪ S,2);
7:         if T^left ≠ ∅, S^left ≠ ∅, T^right ≠ ∅ and S^right ≠
           ∅ then
8:             SplitAndFusion(T^left, S^left, x^left);
9:             SplitAndFusion(T^right, S^right, x^right);
10:            x^merge = x^left + x^right
11:            LocalSearch(x^merge);
12:        end if
13:    end if
14:    if Γ(x^center) ≤ Γ(x^merge) then
15:        x = x^center
16:    else
17:        x = x^merge
18:    end if
19: end procedure
```

Let $\mathbf{x}^{\text{merge}} = \mathbf{x}^{\text{left}} + \mathbf{x}^{\text{right}}$ be the solution vector obtained by merging the two separate allocations:

$$a(\mathbf{x}^{\text{merge}}, i) = \begin{cases} a(\mathbf{x}^{\text{left}}, i) & i \in \mathcal{T}^{\text{left}} \\ a(\mathbf{x}^{\text{right}}, i) & i \in \mathcal{T}^{\text{right}} \end{cases}, \; i \in \mathcal{T}$$

The solution $\mathbf{x}^{\text{merge}}$ is a feasible solution for the allocation of sites $i \in \mathcal{T}$ to sites $j \in \mathcal{S}$. The next step is to optimize this solution using a local search procedure.

Finally, the algorithm compares the cost of the best centralized allocation $\mathbf{x}^{\text{center}}$ with the cost of the optimized distributed allocation $\mathbf{x}^{\text{merge}}$ and returns the min-cost allocation.

The local search procedure for the optimization of $\mathbf{x}^{\text{merge}}$ is described in Algorithm 4. The neighborhood $\mathcal{N}(\mathbf{x})$ of a solution $\mathbf{x}$ is defined as follows,

$$\mathcal{N}(\mathbf{x}) = \mathcal{N}^{\text{left}}(\mathbf{x}) \cup \mathcal{N}^{\text{right}}(\mathbf{x}),$$

where $\mathcal{N}^{\text{left}}(\mathbf{x})$ is the set of solutions $\mathbf{x}'$ such that it exists a unique $i \in \mathcal{T}^{\text{left}}$ such that $a(\mathbf{x}, i) \in \mathcal{S}^{\text{left}}$ and $a(\mathbf{x}', i) \in \mathcal{S}^{\text{right}}$, while $\mathcal{N}^{\text{right}}(\mathbf{x})$ is the set of solutions $\mathbf{x}'$ such that it exists a unique $i \in \mathcal{T}^{\text{right}}$ such that $a(\mathbf{x}, i) \in \mathcal{S}^{\text{right}}$ and $a(\mathbf{x}', i) \in \mathcal{S}^{\text{left}}$.

In other words, a neighbor solution belonging to $\mathcal{N}^{\text{left}}(\mathbf{x})$ is obtained by re-assigning a terminal node allocated to an access site in the left region to an access site in the right region. On the contrary, a neighbor solution belonging to $\mathcal{N}^{\text{right}}(\mathbf{x})$ is obtained by re-assigning a terminal node allocated to an access site in the right region to an access site in the left region.

Algorithm 4 Local search optimization of distributed allocation

```
1: procedure LocalSearch(x)
2:     oldCost = ∞
3:     while Γ(x) <oldCost do
4:         oldCost = Γ(x)
5:         x* = argmin_{x'∈N(x)} Γ(x')
6:         if Γ(x*) < Γ(x) then
7:             x = x*
8:         end if
9:     end while
10: end procedure
```

6. LDS HEURISTIC FOR THE ASSIGNMENT PROBLEM

The concept of *Limited Discrepancy Search* (LDS) was initially proposed by Harvey and Ginsberg for Constraint Satisfaction Problem (CSP) [20]. In practice, many such problems have spaces that are too large to search exhaustively. However, the key observation made by Harvey and Ginsberg is that one can often find solutions while searching only a small fraction of the space by relying on carefully tuned heuristics to guide the search toward regions of the space that are likely to contain solutions.

Assume that we know an efficient heuristic to solve a CSP. The heuristic will lead directly to a solution most of the time, but it may also fails. The intuition behind LDS is that, when the heuristic fails, the heuristic probably would have lead to a solution if and only if it had not made one or two "wrong" decisions that got it off track. The idea is then to systematically follow the heuristic at all but a limited number of decision points (which are called "discrepancies").

The concept of LDS can easily be adapted to our problem. Since we observe that the clustering heuristic often leads to "good" solutions, it makes sense to expect that most of the terminal assignments it performs are those that an optimal solution would have chosen, except for a limited number of wrong decisions.

Let $\overline{\mathbf{x}} \in X$ be the solution obtained with the clustering heuristic and $\mathbf{x} \neq \overline{\mathbf{x}}$ be another feasible solution. We define the distance between $\mathbf{x}$ and $\overline{\mathbf{x}}$ as follows,

$$d(\mathbf{x}, \overline{\mathbf{x}}) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{K} |x_{ij} - \overline{x}_{ij}|$$

In other words, $d(\mathbf{x}, \overline{\mathbf{x}}) = k$ if and only if there are k terminal nodes that are not assigned to the same site in $\mathbf{x}$ and $\overline{\mathbf{x}}$. For a given integer parameter β, let $X_\beta(\overline{\mathbf{x}})$ be the subset of solutions $\mathbf{x} \in X$ such that $d(\mathbf{x}, \overline{\mathbf{x}}) \leq \beta$.

Table 3: Router models.

Router Model	# Slots	Throughput	Cost (€)
A	12	384 Mbps	6,000
B	16	1 Gbps	15,000
C	8	10 Gbps	18,000

The proposed heuristic is described by algorithm 5. It explores the region $X_\beta(\overline{\mathbf{x}})$ of the solution space. The algorithm is very similar to the branch-and-bound algorithm 2: it performs a tree search using upper and lower bounds as well as distance cuts to prune the solution tree. The main difference is that a subtree is not explored if it is rooted at a partial solution $\mathbf{x}$ such that $d(\mathbf{x}, \overline{\mathbf{x}}) > \beta$

Algorithm 5 LDS heuristic for assignment problem

```
 1: procedure LDS(x, i, ub)
 2:     if i > N then
 3:         if Γ(x) < ub then
 4:             ub=Γ(x) and x* = x
 5:         end if
 6:         return 0
 7:     end if
 8:     bestCost=∞
 9:     for j ∈ Sites(x, i) do
10:         Let x'_{i,j} = 1 and x'_{n,k} = x_{n,k}   k = 1...K, n < i
11:         Let x'' such that a(x'', n) = a(x', n) for n ≤ i
                and a(x'', n) = a(x̄, n) for n > i
12:         if d(x'', x̄) ≤ β then
13:             costToGo = c_{i,j} + E_j(x') − E_j(x)
14:             lb = costToGo+lowerBound(x');
15:             if Γ(x)+lb < ub then
16:                 costToGo = costToGo + LDS(x',i+1, ub);
17:             else
18:                 costToGo = lb;
19:             end if
20:             if costToGo < bestCost then
21:                 bestCost=costToGo
22:             end if
23:         end if
24:     end for
25:     return bestCost
26: end procedure
```

7. RESULTS

7.1 Equipments

The results below have been produced using the router models, line card models and link costs presented in Tables 3, 4 and 5. Note the economy of scale in favor of large equipments.

In the following, the site opening cost is fixed to 15000 €, and link costs are evaluated for a period of one year.

7.2 Scenarios

To assess the performances of the proposed algorithms, we consider 10 scenarios. The main features of each scenario are presented in table 6. This table gives the number N of terminal nodes, the number K of access sites, as well as the number $|\Phi_t|$ of terminals per access rate t and the resulting number of possible card configurations. Note that in these scenarios, the number of card configurations are quite modest. This is due to the fact that we have used line cards with many ports. However, as seen in section 3.3, the number of card configurations has a very limited impact on computing times.

Table 4: Line card models.

Line Card Model	# Ports	Throughput (Mbps)	Cost (€)
1	16	16 × 2	2,000
2	12	12 × 10	3,000
3	8	8 × 100	5,000
4	4	4 × 1000	8,000

Table 5: Link costs.

Capacity	Cost/Km per year(€)
2 Mbps	275
10 Mbps	400
100 Mbps	575
1 Gbps	725

In order to precise the relative weights of link costs and other equipment costs, table 7 gives the average cost per link over a one year period when terminals are connected to nearest access sites.

7.3 Results

We compare below the cost obtained with the exact Branch-and-Cut algorithm (BC), with the clustering heuristic (CH) and with the LDS heuristic (limited to 10 discrepancies) with those obtained with the following heuristics:

- Nearest site assignment (NSA) of terminals,
- Minimum cost centralized assignment (MCCA) of terminals,
- Assignment of terminal nodes obtained using the standard practice (SP) for access network design. The first step is to compute the optimal access network topology

Table 6: Main features of the scenarios used to assess the performances of the proposed algorithms.

Scenario	N	K	H	$\lvert\Phi_0\rvert$	$\lvert\Phi_1\rvert$	$\lvert\Phi_2\rvert$	$\lvert\Phi_3\rvert$
1	8	5	8	1	2	5	0
2	16	5	16	1	6	5	4
3	11	6	2	11	0	0	0
4	13	6	16	1	3	5	4
5	18	8	16	2	7	6	3
6	14	9	2	14	0	0	0
7	28	10	36	3	13	7	5
8	25	11	24	4	8	10	3
9	28	13	36	3	13	7	5
10	40	8	24	0	0	22	18

Table 7: Average costs per link when terminals are assigned to nearest access sites.

Scenario	Average Cost per link (€)
1	18,427
2	15,204
3	794
4	12,556
5	8,167
6	1,199
7	10,128
8	8,474
9	9.308
10	6,367

Table 8: Cost of optimal solutions and gap in % with the costs obtained with the heuristic algorithms.

Scenario	Optimal Cost (€)	NSA (%)	MCCA (%)	SP (%)	CH (%)	LDS (%)
1	262,968	7	14.7	7	5.7	0
2	400,221	9.25	9.8	4.4	6.3	0
3	41,223	200.2	0	0	0	0
4	306,900	17.4	11.6	2.3	0	0
5	317,090	26.1	34.1	0.25	10.7	0.25
6	73,465	173.3	3.25	0	0	0
7	564,956	21.5	34.9	7.4	0.06	0
8	461,453	25	33	3.8	9.5	1.6
9	559,881	33.9	36.1	8.4	0.97	0.4
10	596,921	10.2	70.4	5.0	0.8	0.4

taking into account capacity-dependent link costs and site opening costs only, but not router and card costs (they are set to 0). The next step is the dimensionning of the routers and line cards to be settled in the opened access sites. At the end of this step, the global cost of the access network can be computed taking into account equipment costs.

Table 8 gives the optimal cost for each scenario and the gap in percent with the costs obtained with the various heuristic algorithms. It should be noted that the proposed approach allows a significant reduction in cost by integrating traffic demands in the access network design. Note also that the clustering heuristic often provides a very good solution (in 6 of the 10 scenarios), and that this solution is always improved by the LDS heuristic to obtain a near-optimal design.

Table 9 show the computing times for scenarios 7, 8, 9 and 10 (for other scenarios, all algorithms complete within 1 second). It can be noted that, despite they diverge, the computing times of the Branch-and-Cut algorithms are quite reasonable even for problem instances of practical interest (8 access sites and 40 terminal nodes in scenario 10). The LDS heuristic allows to keep reasonable computing times, while providing near-optimal solutions. However, for very large problem instances (more than 100 terminal nodes), network designers will probably have to resort to the clustering heuristic or to the standard practice.

Table 9: Computing times.

Scenario	BC	SP	CH	LDS
7	1 min 40 s	0.71 s	0.52 s	48.7 s
8	7 min 25 s	1 min 5 s	0.62 s	6 min 18 s
9	53 min 43 s	35.5 s	0.57 s	15 min 9 s
10	74 min 28 s	34.5 s	0.56 s	15 min 21 s

8. CONCLUSION

Since equipment costs are now a significant fraction of the total cost when designing a network, it can be expected that important cost-savings can be achieved if equipment costs are taken into account in the early stages of the design process. The main contribution of this paper is to integrate the cost of the equipements in the topological design of access networks.

We have proposed a branch-and-cut algorithm that can be used to obtain the optimal solution of small to medium problem instances. Two different heuristics have been proposed for large problem instances. The clustering heuristic often provides fairly good solutions in very low computing times. The LDS heuristic can be used to improve the solutions obtained with the clustering heuristic, and results show that it provides near-optimal solutions with a significant reduction of computing times with respect to the branch-and-cut algorithm. It should be noted that the Limited Discrepancy Search could also be used to define the neighbourhood of a solution. We are currently investigating tabu search algorithms exploiting this neighbourhood structure.

9. REFERENCES

[1] M. Pióro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks*, Morgan Kaufmann Series in Networking, 2004.

[2] J. G. Klincewicz *Hub location in backbone/tributary network: a review* Location Science 6, pp 307-335, 1998.

[3] C. D. Randazzo, H. P. L. Luna and P. Mahey, *Benders Decomposition for Local Access Network Design with Two Technologies* Discrete Mathematics and Theoretical Computer Science 4, 2001, pp 235-246

[4] F. R. Amoza, *GRASP heuristics for Wide Area Network design*, PhD Thesis, IRISA, 2005.

[5] H. N. Gabow, M. X. Goemans, and D. P. Williamson. *An efficient approximation algorithm for the survivable network design problem.* In Proceedings of the 3rd MPS Conference on Integer Programming and Combinatorial Optimization, pages 57-74, 1993.

[6] M. Grotschel, C. L. Monma, M. Stoer, *Design of Survivable Networks*, Volume on "Networks, Handbook in Operations Research and Management Science, 1993.

[7] B. Fortz, *Design of Survivable Networks with Bounded Rings*, PhD Thesis, Université Libre de Bruxelles, 1999.

[8] M. Stoer. *Design of survivable networks*, volume 1531 of Lecture Notes in Mathematics. Springer Verlag, 1992.

[9] K. Altinkemer and A. Chaturvedi. *Neural networks for topological design of local access tree networks* In Proceeding of Telecommunication Systems, Modelling and Analysis Conference, pp 256-263, 1993.

[10] M. Andrews and L. Zhang. *The access network*

design problem In Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science, 1998

[11] R. T. Berger and S. Raghavan. *Long-Distance Access Network Design*. Management Science, 50:309-325, 2004

[12] L. F. Frantzeskakis and H. Luss. *The network redesign problem for access telecommunications networks*. Naval Research Logistics, 46:487-506, 1999

[13] B. Gavish. *Topological design of telecommunication networks - local access design methods*. Annals of Operations Research, 33:17-71, 1991.

[14] B. Gavish and K. Altinkemer. *Parallel savings heuristics for the topological design of local access tree networks*. In Proceedings of IEEE INFOCOM'86. Fifth Annual Conference on Computers and Communications Integration Design, Analysis, Management, pp 130-139, 1986.

[15] A. Girard, B. Sanso , and L. Dadjo. *A tabu search algorithm for access network design*. Annals of Operations Research, 106(1-4):229-262, 2001 .

[16] L. Gouveia and M. J. Lopes. *Using generalized capacitated trees for designing the topology of local access networks*. Telecommunications Systems, 7(4):315-337, 1997

[17] B. Lukic. *An approach of designing local access network using Simulated Annealing method*. In ConTEL'99 - 5th International Conference on Telecommunications, pages 241-247, 1999.

[18] G. R. Mateus and R. V. L. Franqueira. *Model and heuristics for a generalized access network design problem*. Telecommunication Systems - Modelling, Analysis, Design and Management, 15(3-4):257-271, 2000.

[19] C. D. Randazzo and H. P. L. Luna. *A comparison of optimal methods for local access uncapacited network design*. Annals of Operations Research, 106:263-286, 2001.

[20] W. D. Harvey, M. L. Ginsberg, *Limited Discrepancy Search*. Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95); Vol. 1, 1995.