Integration of Equipment Constraints in the Capacity-planning Process

C. Fortuny, O. Brun, Z. Ben-Hamouda and J.M. Garcia
LAAS-CNRS
7, Avenue du Colonel Roche,
Toulouse, France.
Email: {cfortuny, brun, zbenhamo, jmg}@laas.fr

***Abstract*—This paper addresses the network capacity planning (or dimensioning) problem. The assignment of links capacity is often made by taking into account link costs and neglecting other equipment costs such as routers and line cards. 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. The main contribution of this paper is to integrate the cost of the equipments in the capacity planning process. An exact algorithm and a local search-based heuristic are proposed to solve this problem. Numerical results show that significant cost-savings can be achieved when equipment costs are taken into account in the capacity planning process.**

I. Introduction

Internet has become pervasive in the last decade thanks to a succession of drivers, starting with the rise of the World Wide Web, followed by business adoption of Internet applications, and more recently, the use of peer-to-peer file-sharing software. The Internet is now progressively evolving towards a global communication infrastructure supporting new real-time services in addition to traditional document-retrieval applications.

With the Internet getting more and more present in our daily activities, network outages or even significant degradations of the quality of service become less and less tolerable. To avoid network congestions and the resulting service degradations, Internet Service Providers (ISP) need to properly dimension the core network and trunk lines giving the subscribers access to the Internet. In the current competitive context, ISPs cannot afford installing excessive amounts of capacity and therefore need efficient capacity planning methods. The goal of such methods is to ensure a healthy network that can grow to meet future needs.

A. Previous Work

This article addresses the capacity planning problem of IP networks. This problem has been fairly discussed in the literature and several models have been proposed (see section 4.3 of [10] for an introduction).

Kleinrock[1] was among the first to study a version of the network capacity planning problem. The author aim was to dimension a network such that the average end-to-end delay is minimized. Kleinrock assumes in his model that the total network cost is constant. However, it often happens that, minimizing the average delay cannot ensure a limited delay for each network flow. To overcome this limitation, Meister et Al. have considered in [7], [6] a capacity planning model minimizing the weighted sum of the average delay. K. Maruyama et al. have extended the last model by involving the paradigm of service class [4], [3], [5].

Runggeratigul et Al. propose a capacity assignment model dealing with a piecewise linear concave cost function in order to well represent the relation between link's capacity and link's cost [11] . The authors propose an heuristic algorithm derived from the Lagrange multiplier method to solve the problem. In [9], Mahey et al. consider the capacity and flow assignment problem in data networks and propose a method base on generalized Benders decomposition. Other recent works include [13], [8].

To the extend of our knowledge, none of the previous works has considered the cost and constraints of the line cards. In our opinion, this is a major issue. There are two main motivations for the integration of line card constraints in the capacity-planning process. The first one is related to the network cost. 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 line cards are now a significant fraction of the total cost of a network. The second motivation is related to the feasibility of the solution obtained. Indeed, if the capacity-planning process is conduced without taking into account router and line card constraints, this might lead to a solution which cannot be implemented in practice without changing some of the routers. Therefore, there is an increasing need for an integrated approach of the capacity-planning problem taking into account line card costs and constraints.

B. Our Contribution

The main contribution of this paper is to integrate the equipment constraints and costs in the capacity-planning problem. We first solve a subproblem to compute the optimal

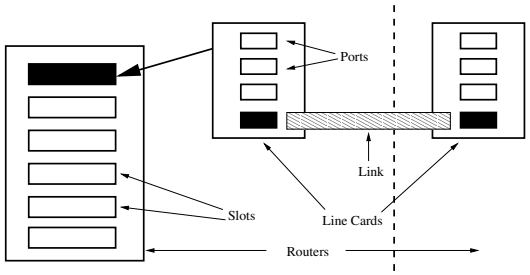

Fig. 1. A link is plugged on a port of a compatible line card at each of its extremities, and the line card is in turn plugged on a slot of the router.

line card configuration for each possible link configuration of a router. The solution of this subproblem is then used to solve the capacity-planning problem, where we have to assign a capacity (from a modular set of capacities) to each network link. Several optimization algorithms are proposed. The first one is a Branch-and-Bound algorithm that can be used to obtain the optimal solution for small to medium network sizes. The second one is an efficient local-search heuristic.

The paper is organized as follows. Section II introduces our notations and states the problem. Section III is devoted to the analysis of optimal card configurations. Section IV describes the Branch-and-Bound algorithm allowing to find the exact solution of the problem, while section V describes the approximation algorithm. Section VI present the results of these algorithms on several problem instances. Finally, some conclusions are drawn in section VII.

II. Problem Statement

We consider a network composed of a set $\mathcal{N}$ of IP routers interconnected by a set $\mathcal{L}$ of links. We let L denote the number of links. Each link is plugged on a port of a compatible line card at each of its extremities. The line cards are in turn plugged on a router slot. The number of ports of a line card and the number of slots of a router are limited and these constraints have to be taken into account when dimensioning the network.

In words, the problem is to find the minimum cost changes on the network configuration that will allow to meet some performance constraints. In the following, we precisely define what is a network configuration, what are the performance constraints to be satisfied, what is the cost induced by a change of network configuration, and finally state our capacity-planning problem.

A. Network Configuration

Let $\mathcal{I}$ be the set of link/port types, numbered $1, \ldots, I$, and let r_t be the bandwidth of the link/port type $t \in \mathcal{I}$ (note that here the term link refers to a network interface).

Definition 1: Let us introduce the following binary decision variables,

$$x_{l,t} = \begin{cases} 1 & \text{if link } l \text{ is of type } t \\ 0 & \text{otherwise} \end{cases}$$

A link configuration is a vector $\mathbf{x} = (x_{l,t}) \in \{0,1\}^L$ such that,

$$\sum_{t \in \mathcal{I}} x_{l,t} = 1 \quad , \; l \in \mathcal{L}$$

A link of type t needs to be connected to a Physical Interface Card (PIC) at each of its extremities. Let $\mathcal{C}$ be the set of all card types and $\mathcal{C}_t \subset \mathcal{C}$ be the set of line cards with port type t. We assume that each link type t can be connected to at least one card type, i.e. $|\mathcal{C}_t| \geq 1$. Let t_c be the port type of a line card of type c, p_c be its number of available ports and $R_c = p_c\, r_{t_c}$ be its total rate.

Definition 2: A card configuration for router $n \in \mathcal{N}$ is a vector $\mathbf{y}_n = (y_{n,c})_{c \in \mathcal{C}}$, where $y_{n,c}$ denotes the number of type c line cards plugged on router n. A network-wide card configuration is a vector $\mathbf{y} = (\mathbf{y}_n)_{n \in \mathcal{N}}$, where $\mathbf{y}_n$ is the card configuration of router $n \in \mathcal{N}$.

We assume that the router models are known and fixed, which means that we do not consider the possibility of changing the model of a router in the course of the capacity-planning process. For each router $n \in \mathcal{N}$, let S_n be the number of available slots where line cards can be plugged, T_n be the maximum throughput of the router forwarding engine and $\Omega_n \subset \mathcal{L}$ be the set of links connected to this router.

Definition 3: A card configuration $\mathbf{y}_n$ is feasible on router $n \in \mathcal{N}$ if and only if $\mathbf{y}_n \in \Lambda(S_n, T_n)$, where,

$$\Lambda(S_n, T_n) = \left\{ \mathbf{y}_n \in \mathbb{N}^C : \sum_{c \in \mathcal{C}} y_{n,c} \leq S_n, \sum_{c \in \mathcal{C}} y_{n,c}\, R_c \leq T_n \right\}$$

In other words, a card configuration is feasible on router n if and only if the number of cards does not exceed the number of slots and if the aggregated rate of the cards does not exceed the capacity of the forwarding engine. We are now in position to define a network configuration.

Definition 4: A network configuration is a pair $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ is a link configuration and $\mathbf{y}$ is a network-wide card configuration. A network configuration is feasible if and only,

$$\sum_{l \in \Omega_n} x_{l,t} \leq \sum_{c \in \mathcal{C}_t} y_{n,c}\, p_c \;, \; t \in \mathcal{I}, n \in \mathcal{N} \tag{1}$$

$$\mathbf{y}_n \in \Lambda(S_n, T_n) \quad , \; n \in \mathcal{N} \tag{2}$$

Constraints 1 state that the card configuration of router n has to be compatible with the configuration of the links connected to this node: the card configuration of this router has to provide enough type-t ports to connect the links of the

same type.

In the following, we let $(\mathbf{x}^0, \mathbf{y}^0)$ denote the initial network configuration and we assume that this initial configuration does not meet the performance constraints stated below.

B. Performance Constraints

Assume that the network configuration $(\mathbf{x}, \mathbf{y})$ is chosen for the network. Then, the capacity of link l is given by,

$$C_l(\mathbf{x}) = \sum_{t \in \mathcal{I}} x_{l,t}\, r_t$$

Let $\mathcal{F}$ be the set of all traffic flows in the network. For each flow, we let s_f and t_f denote the source and destination nodes, respectively, and d_f be the traffic demand.

The routing strategy is assumed to be known and fixed. Let $0 \leq \pi^f_{u,v} \leq 1$ be the proportion of the traffic flow f flowing from nodes u to v. These routing parameters should satisfy the following constraint for each node $u \in \mathcal{N}$,

$$\sum_v \pi^f_{u,v} - \sum_v \pi^f_{v,u} = \begin{cases} 1 & u = s_f \\ -1 & u = t_f \\ 0 & \text{otherwise} \end{cases}$$

Then the total traffic flowing from u to v is just,

$$Y_{u,v} = \sum_{f \in \mathcal{F}} \pi^f_{u,v}\, d_f$$

Clearly, the utilization rate of each link has to be strictly lower than 1, and hence the network configuration $(\mathbf{x}, \mathbf{y})$ has to be such that $Y_{u,v} < C_l(\mathbf{x})$ if $l = (u, v)$, i.e.,

$$Y_{u,v} < \sum_{t \in \mathcal{I}} x_{l,t}\, r_t \quad ,\ l = (u, v),\ u, v \in \mathcal{N} \tag{3}$$

In addition to these constraints on the utilization rates of the links, we also consider constraints on the end-to-end flow-delay. We consider K classes of traffic and let $\mathcal{F}_k$ denote the set of traffic flows of class $k = 1, \ldots, K$ (with $\mathcal{F} = \cup_k \mathcal{F}_k$). Let τ_k be the maximum admissible value for the delay of class-k flows. Assuming the M/M/1 queueing model for each network interface, the constraints on the flow-delays can be written as follows [2],

$$\sum_{u,v \in \mathcal{N}} \frac{\pi^f_{u,v}}{C_{u,v}(\mathbf{x}) - Y_{u,v}} < \tau_k \quad ,\ f \in \mathcal{F}_k, k = 1, \ldots, K \tag{4}$$

Note that, although it has not been done for ease of presentation, propagation delays of links can be readily introduced in the above constraints.

C. Cost of a network configuration change

Assume that the network configuration $(\mathbf{x}^0, \mathbf{y}^0)$ is changed in order to satisfy the performance constraints (3) and (4). Let $(\mathbf{x}, \mathbf{y})$ be the new network configuration and assume that this network configuration is selected for a period of T months.

The cost of the network configuration change has two components. The first one is related to a possible change of the card configuration, while the second one is related to the change of the link configuration. Let ϕ_c denote the cost of a type-c line card. Then, the cost of the evolution from the initial card configuration $\mathbf{y}^0_n$ to the new card configuration $\mathbf{y}_n$ is given for each router n by,

$$F^C_n(\mathbf{y}_n) = \sum_{c \in \mathcal{C}} \max(0, y_{n,c} - y^0_{n,c})\, \phi_c \quad ,\ n \in \mathcal{N} \tag{5}$$

In other words, if the new card configuration has less type-c cards than the initial card configuration, it costs nothing (we can remove cards for free). However, if the new card configuration of node n has more type-c cards than the initial one, then we have to pay for each one of the added cards.

Let us now consider the cost associated to the change of the link configuration of link $l \in \mathcal{L}$. We assume that the cost of a link is an increasing function of its capacity. More precisely, the cost of a type-t link has two components. The first one, α_t, is a fixed installation cost, to be paid for once. The other one, β^t_l is a recurrent cost to be paid each month. Note that β^t_l is typically a piecewise linear increasing function of the euclidean distance between the end nodes, as usual with tariff systems used by providers of leased lines. The cost of the link configuration change is given for each link $l \in \mathcal{L}$ by,

$$F^L_l(\mathbf{x}_l) = \begin{cases} T \sum_{t \in \mathcal{I}} x_{l,t}\, \beta^t_l & \mathbf{x}_l = \mathbf{x}^0_l \\ \sum_{t \in \mathcal{I}} x_{l,t}\, (\alpha_t \ +\ T\, \beta^t_l) & \text{otherwise} \end{cases} \tag{6}$$

Note that the installation cost is taken into account only if the type of the link is changed. Note also that the above cost is always strictly positive, even if the type of the link is not changed. Finally, the cost of the network configuration change is given by,

$$F(\mathbf{x}, \mathbf{y}) = \sum_{l \in \mathcal{L}} F^L_l(\mathbf{x}_l) \ +\ \sum_{n \in \mathcal{N}} F^C_n(\mathbf{y}_n) \tag{7}$$

D. Mathematical model

The problem to be solved can be stated as follows

$$\min_{(\mathbf{x},\mathbf{y})} \; F(\mathbf{x},\mathbf{y})$$

Subject to:

$$\begin{aligned}
\sum_{l\in\Omega_n} x_{l,t} \le \sum_{c\in\mathcal{C}_t} y_{n,c}\, p_c \qquad &, \; t\in\mathcal{I}, n\in\mathcal{N} \\
\mathbf{y}_n \in \Lambda(S_n,T_n) \qquad &, \; n\in\mathcal{N} \\
Y_{u,v} < \sum_{t\in\mathcal{I}} x_{l,t}\, r_t \;\; &, \; l=(u,v),\; u,v\in\mathcal{N} \\
\sum_{u,v\in\mathcal{N}} \frac{\pi^f_{u,v}}{C_{u,v}(\mathbf{x}) - Y_{u,v}} < \tau_k \;\; &, \; f\in\mathcal{F}_k, k=1,\ldots,K \\
\sum_{t\in\mathcal{I}} x_{l,t} = 1 \qquad &, \; l\in\mathcal{L} \\
x_{l,t}\in\{0,1\} \qquad t\in\mathcal{I},\; l\in\mathcal{L}
\end{aligned}$$

III. Optimal Card Configurations

A. Link Configurations of a Router

Assume that a link configuration $\mathbf{x}$ has been selected and let us consider the optimal card configuration of a router $n\in\mathcal{N}$. This card configuration has to be such that it provides a number of type-t ports greater than or equal to $g^t_n(\mathbf{x}) = \sum_{l\in\Omega_n} x_{l,t}$. It is easy to see that the optimal card configuration for the router n does depend on the link configuration $\mathbf{x}$ only throught the vector $\mathbf{g}_n(\mathbf{x}) = (g^t_n)_{t\in\mathcal{I}}$. In the following, such a vector will be called a link configuration for router n.

Let $L_n = |\Omega_n|$ denote the number of links connected to router n. It is easy to see that the set of all possible link configurations $\mathbf{g}_n$ for router n is,

$$G_n = \left\{ \mathbf{g}_n = \sum_{t\in\mathcal{I}} g^t_n\, \mathbf{e}_t \;:\; \sum_{t\in\mathcal{I}} g^t_n = L_n \right\} ,$$

where $\mathbf{e}_t$ is the vector $(0,\ldots,1,\ldots,0)$ with 1 in position t and 0 elsewhere. Note that the number of such link configurations is the number of weak compositions of L_n into exactly I parts,

$$|G_n| = \binom{L_n + I - 1}{I-1}$$

In practice, the number of link configurations for a router is quite limited. For instance, if we assume a degree L_n=5 for router n and I=6 link/port types, then the number of link configurations is just 252. This suggests that the optimal card configuration can be computed beforehand for each possible link configuration $\mathbf{g}_n\in G_n$ and for each router $n\in\mathcal{N}$. We present below a very efficient dynamic-programming solution to achieve this task.

B. Optimal Card Configuration

Let us consider a link configuration $\mathbf{g}_n\in G_n$. The optimal card configuration $\mathbf{y}^*_n(\mathbf{g}_n)$ for this link configuration is the solution of the following optimization problem, if any,

$$\begin{aligned}
\text{Min}_{\;\mathbf{y}_n\in\Lambda(S_n,T_n)} \; &\sum_{c\in\mathcal{C}} \max(0, y_{n,c} - y_{n,c}(0))\, \phi_c \\
\text{subject to :} \qquad & \\
g^t_n \le \sum_{c\in\mathcal{C}_t} y_{n,c}\, p_c \quad &, t\in\mathcal{I}
\end{aligned}$$

Let $\mu^*(\mathbf{g}_n, S_n, T_n)$ be the cost of an optimal solution to the above problem, if any. By convention, $\mu^*(\mathbf{g}_n, S_n, T_n) = \infty$ otherwise. Let us now assume, that this problem admits one or more feasible solutions.

The optimal solution $\mathbf{y}^*_n(\mathbf{g}_n)$ is composed of at least one line card. If this line card is of type $c\in\mathcal{C}$, then clearly we can connect p_c links of type t_c to this line card. In this case, the problem amounts to find the optimal card configuration for the remaining links, i.e. for the vector $\mathbf{g}_n - p_c\mathbf{e}_{t_c}$, with at most $S_n - 1$ slots and a maximum throughput $T_n - R_c$. This leads to the following dynamic programming equation,

$$\begin{aligned}
\mu^*(\mathbf{g}_n, S_n, T_n) \;=\; &\min_{c\in\mathcal{C}} [\epsilon_c\, \phi_c \; + \\
&\mu^*(\mathbf{g}_n - p_c\, \mathbf{e}_{t_c}, S_n - 1, T_n - R_c)]
\end{aligned}$$

where,

$$\epsilon_c = \begin{cases} 0 & \text{if } y^*_{n,c}(\mathbf{g}_n - p_c\, \mathbf{e}_{t_c}) < y_{n,c}(0) \\ 1 & \text{otherwise} \end{cases}$$

The above dynamic programming equation allows to compute the optimal card configuration for each possible value of $g_n\in G_n$. In the following, for ease of notations, we will write $\mu^*_n(\mathbf{g}_n(\mathbf{x}))$ instead of $\mu^*(\mathbf{g}_n(\mathbf{x}), S_n, T_n)$.

IV. Exact Algorithm

Assuming that the mapping μ^*_n has been computed for each node $n\in\mathcal{N}$, we now consider the global problem. It can now be formulated as follows,

$$\min_{\mathbf{x}} \; F(\mathbf{x}) = \sum_{l\in\mathcal{L}} F^L_l(\mathbf{x}_l) \;+\; \sum_{n\in\mathcal{N}} \mu^*_n(\mathbf{g}_n(\mathbf{x}))$$

Subject to:

$$\begin{aligned}
Y_{u,v} < \sum_{t\in\mathcal{I}} x_{l,t}\, r_t \;\; &\; l=(u,v),\; u,v\in\mathcal{N} \\
\sum_{u,v\in\mathcal{N}} \frac{\pi^f_{u,v}}{C_{u,v}(\mathbf{x}) - Y_{u,v}} < \tau_k \;\; &\; f\in\mathcal{F}_k, k=1,\ldots,K \\
\sum_{t\in\mathcal{I}} x_{l,t} = 1 \qquad &\; l\in\mathcal{L} \\
x_{l,t}\in\{0,1\} \qquad t\in\mathcal{I},\; l\in\mathcal{L}
\end{aligned}$$

This problem can be solved using the branch-and-bound algorithm 1. This recursive algorithm takes as input the current partial solution $\mathbf{x}$, the next link l to be assigned and an upper bound ub on the optimal cost. We assume that when

the algorithm is started, each link has an infinite capacity and for each router n, $\mathbf{y}_n = \mathbf{0}$. The algorithm 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 visit the more promising solutions first and thus to lower the upper bound as fast as possible.

algorithm 1 Branch-and-bound algorithm

```
1:  procedure BB(x, l, ub)
2:      if l > |L| then                                  ▷ no link left
3:          if F(x) < ub then
4:              ub=F(x) and x* = x
5:          end if
6:          return 0
7:      end if
8:      bestCost=∞
9:      for t ∈ T do
10:         Let x'_l = e_t and x'_k = x_k   k ≠ l
11:         g'_u = g_u + e_t and g'_v = g_v + e_t        ▷ l = (u, v)
12:         for f ∈ F such that π^f_{u,v} > 0 do
13:             delay(f,x')=delay(f,x)+π^f_{u,v}/(r_t − Y_{u,v})
14:         end for
15:         if x' is feasible then
16:             costToGo = F^L_l(x'_l) + μ*_u(g'_u) − μ*_u(g_u)
17:             costToGo = costToGo + μ*_v(g'_v) − μ*_v(g_v)
18:             lb = costToGo+lowerBound(x');
19:             if F(x)+lb < ub then
20:                 costToGo = costToGo + BB(x',l + 1, ub);
21:             else
22:                 costToGo = lb;
23:             end if
24:             if costToGo < bestCost then
25:                 bestCost=costToGo
26:             end if
27:         end if
28:     end for
29:     return bestCost
30: end procedure
```

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

A. Feasibility Test

Let $\mathbf{x}'$ be the partial solution obtained from $\mathbf{x}$ when assigning type t to link l. The feasibility test amounts to check that $\mathbf{x}'$ yields a feasible network configuration, and that it satisfies the performance constraints.

Let nodes u and v be the endpoints of l. Then the constraint on the utilization rate of link l is satisfied if and only if $Y_{u,v} < r_t$ and $Y_{v,u} < r_t$.

Let delay$(f, \mathbf{x}')$ denote the end-to-end delay of flow f in the partial solution $\mathbf{x}'$. Then,

$$\text{delay}(f, \mathbf{x}') = \begin{cases} \text{delay}(f, \mathbf{x}) + \pi^f_{u,v}/(r_t - Y_{u,v}) & \pi^f_{u,v} > 0 \\ \text{delay}(f, \mathbf{x}) & \pi^f_{u,v} = 0 \end{cases}$$

Therefore, flow-delay constraints are satisfied in solution $\mathbf{x}'$ if and only if for each flow $f \in \mathcal{F}$ such that $\pi^f_{u,v} > 0$, we have,

$$\text{delay}(f, \mathbf{x}') < \tau_k \quad f \in \mathcal{F}_k \text{ s. t. } \pi^f_{u,v} > 0, k = 1, \ldots, K$$

Finally, we have to check that partial solution $\mathbf{x}'$ yields a feasible network-wide card configuration. It is obvious that $\mathbf{g}_n(\mathbf{x}') = \mathbf{g}_n(\mathbf{x})$ for $n \neq u, v$ and thus $\mathbf{x}'$ yields a feasible card configuration for each router $n \neq u, v$. Moreover, $\mathbf{g}_u(\mathbf{x}') = \mathbf{g}_u(\mathbf{x}) + \mathbf{e}_t$ and $\mathbf{g}_v(\mathbf{x}') = \mathbf{g}_v(\mathbf{x}) + \mathbf{e}_t$. Therefore, we only need to check that these router link configurations yield feasible card configurations for routers u and v, i.e. that $\mu^*_u(\mathbf{g}_u(\mathbf{x}')) < \infty$ and $\mu^*_v(\mathbf{g}_v(\mathbf{x}')) < \infty$.

B. Upper and Lower Bounds

The algorithm use an upper bound and a lower bound to prune the solution tree. The upper bound is initialized with a greedy heuristic and it is updated each time an improving complete solution is discovered.

A lower bound on the optimal cost-to-go is generated in each node of the search tree. Let us consider a partial solution $\mathbf{x}$ such that a link/port type has already been selected for links $1, \ldots, l$. Let $Subtree(\mathbf{x})$ be the set of complete solutions in the subtree rooted at $\mathbf{x}$, i.e.,

$$Subtree(\mathbf{x}) = \{\mathbf{x}' \ : \ \mathbf{x}'_k = \mathbf{x}_k \quad , k = 1, \ldots, l\}.$$

A lower bound $lb(\mathbf{x})$ on the optimal cost-to-go starting from partial solution $\mathbf{x}$ is such that $F(\mathbf{x}') \geq F(\mathbf{x}) + lb(\mathbf{x})$ for each $\mathbf{x}' \in Subtree(\mathbf{x})$. In our case, a lower bound is easily obtained by summing the costs of the links $k > l$, assuming that for each of these links we select the capacity directly greater than the load such that it yields a feasible card configuration on each endpoint.

Let us now here again consider the partial solution $\mathbf{x}'$ obtained from $\mathbf{x}$ when assigning type t to link l. Any solution in the subtree rooted at $\mathbf{x}'$ will have a cost greater than or equal to $F(\mathbf{x}') + lb(\mathbf{x}')$. The cost $F(\mathbf{x}')$ of the partial solution $\mathbf{x}'$ is obtained from the cost of $\mathbf{x}$ by adding the immediate cost induced by the selection of type t for link l,

$$\begin{aligned} F(\mathbf{x}') \ = \ & F(\mathbf{x}) + F^L_l(\mathbf{x}'_l) + \mu^*_u(\mathbf{g}_u(\mathbf{x}')) - \mu^*_u(\mathbf{g}_u(\mathbf{x})) \\ & + \mu^*_v(\mathbf{g}_v(\mathbf{x}')) - \mu^*_v(\mathbf{g}_v(\mathbf{x})) \end{aligned}$$

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

C. Value ordering and variable ordering

The order in which the assignment of link/port types to links is 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 types t in the order of increasing capacity r_t, starting from the minimal capacity such that the constraint on the utilization rate is satisfied. In the following, we assume that this value ordering scheme is used.

The order in which we consider the links is also important. In our implementation, we have chosen to consider the links in the order of decreasing loads, i.e. we first consider the links with the highest costs.

V. Local Search Algorithm

The exact algorithm described above can be used only for small to moderate problem instances. As a consequence, the design of an efficient heuristic is of paramount importance for dealing with problems of larger sizes. In this section, we describe a local-search heuristic that allows to solve large problem instances in limited computing times.

Local search methods are well known approximation techniques for combinatorial optimization. Such methods belong to the class of iterative optimization procedures. The general step of an iterative procedure consists in constructing from a current solution $\mathbf{x}$ a next solution $\mathbf{x}'$ and in checking whether one should stop there or perform another step. Local search methods are iterative procedures in which a neighborhood $N(\mathbf{x})$ is defined for each feasible solution $\mathbf{x}$, and the next solution $\mathbf{x}'$ is searched among the solutions in $N(\mathbf{x})$. The most famous local search method which has been used for finding approximations to minimization problems is the so-called descent method, where the selected neighbor $\mathbf{x}'$ is the one with the minimum cost in $N(\mathbf{x})$.

Our local search method is described in algorithm 2. It is a descent method defining a small size neighborhood for each solution. Efficient techniques are proposed to evaluate the feasibility and the cost of neighbors of a solution. This enables the rapid convergence towards a local minimum. A wider neighborhood structure is also defined in order to escape from local minimum. We describe below the details of the algorithm.

algorithm 2 Local Search Algorithm

```
1:  procedure LOCAL SEARCH
2:      x(0) : initial feasible solution
3:      k = 0 and x* = x(0); STOP=false;
4:      while STOP=false do
5:          x(k+1) : min cost feasible solution in N(x(k))
6:          if F(x(k+1)) > F(x(k)) then
7:              x(k+1) : min cost feasible solution in
                    N_β(x(k))
8:          end if
9:          if F(x(k+1)) > F(x(k)) then
10:             STOP=true
11:         else if F(x(k+1)) < F(x*) then
12:             x* = x(k+1)
13:         end if
14:     end while
15: end procedure
```

A. Neighborhood Structure

Let $\mathbf{x}$ be a solution to our problem and let $N(\mathbf{x})$ be its neighborhood. A link configuration $\mathbf{x}'$ belongs to $N(\mathbf{x})$ if and only if it differs from $\mathbf{x}$ only with respect to the type of a single link. In other words, $\mathbf{x}' \in N(\mathbf{x})$ if and only if there is a unique $l \in \mathcal{L}$ such that,

$$\mathbf{x'}_l \neq \mathbf{x}_l \quad \text{and} \quad \mathbf{x'}_k = \mathbf{x}_k \ , k \neq l, k \in \mathcal{L} \tag{8}$$

The size of this neighborhood is $L\,(I-1)$.

B. Feasibility of a Neighbor

Let $\mathbf{x}'$ be a neighbor of $\mathbf{x}$ and let l be the unique link such that $\mathbf{x'}_l \neq \mathbf{x}_l$. Let t be the type of link l in solution $\mathbf{x}'$ (i.e. $x'_{l,t} = 1$). Finally, let nodes u and v be the endpoints of l.

The constraint on the utilization rate of link l is satisfied if and only if $Y_{u,v} < r_t$ and $Y_{v,u} < r_t$. Regarding the flow-delay constraint, we have for each flow f such that $\pi^f_{u,v} > 0$,

$$\text{delay}(f, \mathbf{x}') = \text{delay}(f, \mathbf{x}) - \frac{\pi^f_{u,v}}{C_l(\mathbf{x}) - Y_{u,v}} + \frac{\pi^f_{u,v}}{C_l(\mathbf{x}') - Y_{u,v}}$$

The above equation allows a fast update of the flow delays. To check that the flow-delay constraints are satisfied for class k in solution $\mathbf{x}'$, we just have to consider all flows $f \in \mathcal{F}_k$ such that $\pi^f_{u,v} > 0$, and to check that $\text{delay}(f, \mathbf{x}') < \tau_k$.

Finally, observe that $\mathbf{g}_n(\mathbf{x}') = \mathbf{g}_n(\mathbf{x})$ for $n \neq u, v$. Moreover, if z denotes the type of link l in solution $\mathbf{x}$, we have:

$$\mathbf{g}_u(\mathbf{x}') = \mathbf{g}_u(\mathbf{x}) + \mathbf{e}_t - \mathbf{e}_z$$
$$\mathbf{g}_v(\mathbf{x}') = \mathbf{g}_v(\mathbf{x}) + \mathbf{e}_t - \mathbf{e}_z$$

The above equations allow a fast update of the link configurations of routers u and v. To check that solution $\mathbf{x}'$ correspond to a feasible network configuration we then just have to check that $\mu^*_u(\mathbf{g}_u(\mathbf{x}')) < \infty$ and $\mu^*_v(\mathbf{g}_v(\mathbf{x}')) < \infty$.

C. Evaluation of Neighbors

Using the same notations as in section V-B, the cost of solution $\mathbf{x}'$ can be written as follows,

$$\begin{aligned} F(\mathbf{x}') = F(\mathbf{x}) + F_l^L(\mathbf{x'}_l) - F_l^L(\mathbf{x}_l) \\ +\mu_u^*(\mathbf{g}_u(\mathbf{x}')) - \mu_u^*(\mathbf{g}_u(\mathbf{x})) \\ +\mu_v^*(\mathbf{g}_v(\mathbf{x}')) - \mu_v^*(\mathbf{g}_v(\mathbf{x})) \end{aligned}$$

The above equation allows a very efficient evaluation of the neighbors of solution $\mathbf{x}$.

D. LDS-based Neighborhood

The above descent algorithm generally converges very fast towards a local minimum. In order to escape from local minimum, we propose to use a wider neighborhood. This neighborhood is generated using a *Limited Discrepancy Search* (LDS) algorithm.

The concept of LDS was initially proposed by Harvey and Ginsberg for Constraint Satisfaction Problem (CSP) [12]. 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. Let $\overline{\mathbf{x}}$ be a local minimum reached using the local search procedure and $\mathbf{x} \neq \overline{\mathbf{x}}$ be another feasible solution. We define the distance between $\mathbf{x}$ and $\overline{\mathbf{x}}$ as follows,

$$d(\mathbf{x}, \overline{\mathbf{x}}) = \frac{1}{2} \sum_{l \in \mathcal{L}} \sum_{t \in \mathcal{I}} |x_{l,t} - \overline{x}_{l,t}|$$

In other words, $d(\mathbf{x}, \overline{\mathbf{x}}) = k$ if and only if there are k links which do not have the same type in $\mathbf{x}$ and $\overline{\mathbf{x}}$. For a given integer parameter β, let $N_\beta(\overline{\mathbf{x}})$ be the subset of solutions $\mathbf{x}$ such that $d(\mathbf{x}, \overline{\mathbf{x}}) \leq \beta$. $N_\beta(\overline{\mathbf{x}})$ defines a neighborhood of the local minimum $\overline{\mathbf{x}}$. Note that when $\beta = 1$, this neighborhood reduces to the neighborhood defined in section V-A.

This neighborhood is generated with an algorithm which is very similar to the branch-and-bound algorithm 1: it performs a tree search using upper and lower bounds as well as a feasibility test 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 $d(\mathbf{x}, \overline{\mathbf{x}}) > \beta$.

It is easy to show that the size of the neighborhood $N_\beta(\overline{\mathbf{x}})$ is given by,

$$|N_\beta(\overline{\mathbf{x}})| = \sum_{k=1}^{\beta} \binom{L}{k} (I-1)^k$$

Model		OC-1	OC-3	OC-12
Capacity (Kbps)		50725	152174	608698
Γ_t^L (€)		4500	4500	4500
Γ_t^F (€/month)	1-10 Km	2100	2600	5000
	>10Km	2000	2500	4700
Γ_t^D (€/Km/month)	1-10 Km	0	0	0
	>10Km	25	25	25

TABLE I
LINK'S MODEL, CAPACITY, AND COST USED FOR TESTS

Cards	**Model**	**#Ports**	**Cost (K€)**
1	OC-1	1	30
2	OC-1	2	40
3	OC-1	4	60
4	OC-3	1	40
5	OC-3	2	60
6	OC-3	4	80
7	OC-12	1	60
8	OC-12	2	80
9	OC-12	4	120

TABLE II
TYPES OF CARD USED FOR TESTS

In the experiments presented below, we have used a value of β such that $N_\beta(\overline{\mathbf{x}})$ contains at least 1 % of all the solutions. Since this might lead to very long computing times for some networks, we also limit the exploration of the neighborhood to a maximum number of neighbors (for instance, at most 500,000 neighbors).

VI. TESTS AND RESULTS

In this section we present the experimental results for the heuristic mentioned in the previous section. The results below have been produced using the link models and line card models presented in Tables I, II. We used for our tests one router's model with 8 slots and a maximal rate of 1e+08 Kbps. Tests were made for random problems, with vertices uniformly generated. We first assess our dimensioning algorithm under simple topologies with a few number of nodes and having trivial solutions. We then present results for networks with nodes varying from 20 to 100 nodes and compare our approach to the equipment-independent approach.

A. Performance Assessment of our Algorithm

The topology we studied is depicted in figure 2. Table III shows initial link's models, traffic's demands and link's utilizations.

The solution consists on replacing link models on all the overloaded links (utilization around 197%) with link model OC-3. By doing so, a line card of type OC-3 is required in all destination routers except $D1$, to support links connected to that nodes. Moreover, one card of model OC-3 with 4 ports and another one of the same model, but with only 1 port, should be added on router S. The algorithm introduced

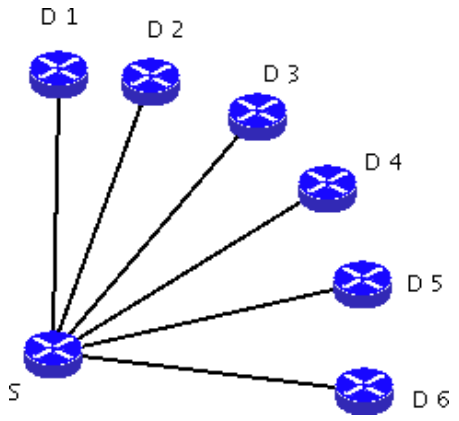

Fig. 2. Test 1 : Topology

	Link's Model	Traffic's Demand (Kbps)	Utilization (%)
S → D 1	OC-1	20000	39.4
S → D 2	OC-1	100000	197.1
S → D 3	OC-1	100000	197.1
S → D 4	OC-1	100000	197.1
S → D 5	OC-1	100000	197.1
S → D 6	OC-1	100000	197.1

TABLE III
TEST 1 : INITIAL VALUES

before finds this solution in less than 20 seconds.

Let us now consider the same previous scenario, except that the number of ports on router S is limited to 2 instead of 8. It is clear that, the above solution is not feasible. The solution provided by our algorithm is such that two cards are used on S : one card OC-3 with 4 ports and one card OC-3 with 2 ports. The OC-3 model is installed on all links. This demonstrates the ability of our algorithm to provide solutions which are feasible with respect to the equipment constraints.

B. Our Model Usefulness

We have submitted our approach to a series of tests in order to determine its efficiency and responsiveness in relation to the numbers of nodes. We have compared, on the other hand, the cost obtained under our approach with that obtained with the equipment-independent approach. By equipment-independent model, we denote the approach where network is dimensioned without taking into account the cost of the equipments to be installed on nodes.

The most interesting results obtained are shown in table IV. In this table, we denote by C_{indep} the total cost obtained with the equipment-independent approach, and C_{dep} the total cost obtained by our algorithm. It is clear that significant cost-savings can be achieved by integrating equipment costs in the dimensioning problem. Clearly this demonstrates the usefulness of our design model.

Tests	1	2	3
# Nodes	22	36	84
# Links	36	56	145
C_{dep} (K€)	1689	3223	4487
C_{indep} (K€)	2015	3404	5977
Gain (K€)	326	181	1490
Gain (%)	**16.2**	**5.3**	**24.9**

TABLE IV
EXAMPLE OF MAJOR ECONOMIC GAINS THANKS TO PROPOSED ALGORITHM

VII. CONCLUSION

The main contribution of this paper is to introduce a comprehensive capacity planning model that integrates the cost of the equipments. Exact and heuristic algorithms have been proposed to solve this problem. Numerical results show that significant cost-savings can be achieved when equipment costs are taken into account in the capacity planning process.

REFERENCES

[1] L. Kleinrock. *Queueing Systems*, volume Vol 2 : Computer Applications. Wiley-Interscience, 1976.
[2] L. Kleinrock. *Queueing Systems*, volume Vol 1 : Theory. Wiley-Interscience, 1976.
[3] K. Maruyama, L. Fratta, and D. T. Tang. Heuristic design algorithm for computer communication networks with different classes of packets. *j-IBM-JRD*, 21(4):360–369, jul 1977.
[4] K. Maruyama and D. T. Tang. Discrete link capacity assignment in communication networks. In *ICCC*, pages 92–97, 1976.
[5] K. Maruyama and D. T. Tang. Discrete link capacity and priority assignments in communication networks. *IBM Journal of Research and Development*, Vol 21:p.254–263, May 1977.
[6] B. Meister, H. Muller, and H. Rudin. On the optimization of message-switching networks. *Communications, IEEE Transactions on [legacy, pre - 1988]*, Vol 20, no.1:p. 8–14, Feb 1972.
[7] B. Meister, H. Muller, and H. Rudin. New optimization criteria for message-switching networks. *Communications, IEEE Transactions on [legacy, pre - 1988]*, Vol 19 Issue: 3:p.256–260, Jun 1971.
[8] B. J. Oommen and T. D. Roberts. Continuous learning automata solutions to the capacity assignment problem. *IEEE Trans. Comput.*, 49(6):608–620, 2000.
[9] A. B. e. F. B. P. Mahey. Capacity and flow assignment of data networks by generalized benders decomposition. *Journal of Global Optimization*, 20–2:173–193, 2001.
[10] M. Pioro and D. Medhi. *Routing, Flow and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Series in Networking, 2004.
[11] S. Runggeratigul and S. Tantaratana. Link capacity assignment in packet-switched networks: The case of piecewise linear concave cost function, 1999.
[12] M. L. G. William D. Harvey. Limited discrepancy search. In C. S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95); Vol. 1*, pages 607–615, Montréal, Québec, Canada, August 20-25 1995. Morgan Kaufmann, 1995.
[13] H. H. Yen and F. Y. S. Lin. Delay constrained routing and link capacity assignment in virtual circuit networks(network). *IEICE transactions on communications*, 88(5):2004–2014, May 2005.