OPTIMAL PLACEMENT OF ADD/DROP MULTIPLEXERS: HEURISTIC AND EXACT ALGORITHMS

Alain SUTTER [*] François VANDERBECK [†] Laurence WOLSEY [‡]

December 1994
(Revised January 1996)

Subject Classification: (Tele)communications, Assignment of Equipment, Comparison of Safety Policies, Integer Programming, Heuristics and Decomposition Algorithms.

[*]CNET, France Telecom, Issy-les-Moulineaux, France
[†]Judge Institute of Management Studies, Cambridge University, England
[‡]CORE, Université Catholique de Louvain, Louvain-la-Neuve, Belgium

Abstract

We study a problem that has arisen recently in the design of telecommunications transmission networks at France Telecom. Given a set of centers in a city or conglommeration linked together on a ring architecture, given the expected demands between the centers and an essentially unlimited availability of rings of fixed capacity on the network, assign demand pairs and corresponding add/drop multiplexers to the rings so as to satisfy the demands and minimize the number of "costly" multiplexers installed.

Heuristics based on simulated annealing have been developed for the basic problem and several variants. France Telecom is particularly interested in validating the effectiveness of the heuristics. An exact algorithm based on integer column generation is shown to provide tight performance guarantees, and in most cases to give provable minimum cost solutions. Column generation is used even though the subproblems are strongly NP-hard and are solved by standard mixed integer programming software. In addition the Master Problems involve both integer columns and integer variables, whereas these are $0-1$ in most reported applications.

Introduction

The introduction of new technology has raised a number of new and interesting theoretical and practical issues in the modelling and optimization of the design of telecommunication networks. France Telecom has recently decided to adopt a so-called ring architecture for all their local, main and sectorial networks. Here we consider the problem of assigning telecommunication traffic on such a network made of a cable of optic fibers that visits telecommunication centers in a circular fashion. Given a set of centers in a city or conglommeration linked together by individual circular optic fibers called rings with an essentially unlimited availability of rings of fixed capacity on the network, and given the expected demands between pairs of centers, assign demands to the rings subject to the ring capacities. On each ring it is then necessary to install an add/drop mutiplexer at every center incident to one of the assigned demands. Given the high cost of the multiplexers, the objective is to find an assignment requiring the smallest total number of multiplexers.

The practical object of this study is to develop fast and effective heuristics for the problem outlined above. Speed and simplicity are important both because it is a subproblem of a much larger design problem (the subproblem treated here is solved very frequently), and because the heuristics are to be used by the operators managing the networks. However given the significance of the multiplexer costs, it is also important to have performance guarantees for some real life subproblems. After briefly presenting the heuristics, the main goal in this paper is to explain how the quality of the heuristics was validated using optimization methods. In particular, we show that:

i) The problem considered is one for which attempting to solve the initial integer programming with commercial mixed integer programming software is unsuccessful because of the weakness of the bounds and the symmetry of the solution space.

ii) Somewhat contrary to the prevailing wisdom that limits the use of IP column generation (IPCG) to problems in which the subproblems can be solved rapidly by specialized dynamic programming algorithms, IPCG is shown to be useful when the subproblem is difficult (strongly NP-hard), and is solved by standard MIP software.

iii) When the Master Problem has columns and variables that are integer and not just $0-1$ as in the majority of applications, IPCG can still be implemented using the generalized branching scheme proposed in [16].

iv) The combination of heuristics and IPCG is effective, and permits the company to validate the quality of its heuristics.

v) The resulting optimization tool can be modified to treat different scenarios, such as the costs of increasing network reliability by splitting demands, etc.

In Section 1 below, we discuss the technical and economic background, before defining the problem to be tackled more precisely. In Section 2, we present two formulations of the problem and its variants as integer programs. The first is perhaps the more natural in that it closely matches our initial description of the problem. The second, a partitioning problem with an enormous number of columns, each corresponding to a feasible ring layout, is the one we have found useful. This is essentially the standard problem reformulation obtained by applying Dantzig-Wolfe decomposition. Section 3 is devoted to a brief description of the heuristics used, as well as the integer programming column generation algorithm we use to solve the partitioning problem. In the last section, we present computational results on a set of real life problems, slightly modified for reasons of confidentiality, and examine how the results might help answer the different questions posed in practice.

1 Modelling Issues

The advent of CCITT Synchronous Digital Hierarchy (SDH) as the new transmission technology is leading to important changes in European Networks. The SDH offers a considerably better solution to networking than that provided by the old Plesiochronous Digital Hierarchy (PDH). Byte Visibility at all Bit Rates means that only simple multiplexing is needed to extract one tributary and re-insert another (it is much simpler to extract information as there is direct rather than sequential access to the information). Note that previously with PDH, a mountain of multiplexers was needed to access the information. The removal of this multiplex mountain brings significant decreases in power requirements and costs. Moreover, the normalization and the simplicity of standard interfaces imply that the operating companies are now free to make commercial decisions regarding equipment sourcing. Possibilities for efficient management of the networks are increased, and the increased flexibility permits the development of new services for customers.

Equipment for SDH networks is standardized and well-developed. It offers high levels of functional integration; for example, the line termination function no longer has to be physically separated from a multiplexer allowing easy point to point transmission. Add-Drop Multiplexers (ADMs) are likely to be used in ring topologies. In its basic form , an ADM multiplexes sixty-three Virtual -Channel VC-12 (2048 kbit/s) in one Synchronous Transport Module STM-1 (155 Mbit/s). Other variants could employ higher rate aggregate ports as for example: VC-12 (or and) STM-1 tributaries in one STM-4 (620 Mbit/s).

Point-to-point network architectures are being replaced in the public network by ring topologies. France Telecom (FT) has decided on this architecture for all the local, main local and sectorial networks. This type of arrangement gives high protection in case of a simple node or edge failure i.e. when equipment or cable fails. All the traffic is sent in two opposite directions around the ring and therefore provides protection through route diversity. They are Self Healing Rings (SHRs), and they guarantee automatic restoration of service for all simple failures. The benefits from a ring structure include simplicity, flexibility and fast restoration time. Analysis indicates that rings are the best economic alternative compared to existing point-to-point architectures.

Different SHRs exist. In particular, one can find unidirectional and bidirectional SHRs with two or four fibers, see Bars et al.[2] and Newall[12] for a detailed discussion of the technology. FT has decided to use path protection SHRs only. With this architecture, the sending node duplicates the signal to be transmitted on both the working and protection fibers. During normal operation, only the working signal is selected by the path switch at the receiving node. With path protection, the signal consumes bandwith at each node on the ring. Thus it requires capacity equal to the sum of the point-to-point demands. On the one hand their management is very simple and automatic restoration for a virtual channel is very fast (a few m-s). On the other hand, compared to bidirectional rings with link protection, they can be suboptimal in terms of bandwidth use, especially for a full mesh demand pattern where all the traffic demands between nodes are quasi-uniform.

The problem of finding an optimal assignment of Add-Drop-Multiplexers on Self-Healing-Rings is of major interest for France Telecom. ADMs are expensive pieces of equipment, for example one ADM STM-1 costs something like $50000 US . The French transmission Network has approximatively 40 Sectorial Networks, 400 Main Local Networks and 9000 local Networks. Roughly speaking, each Network has from 1 to 5 clusters and each cluster has a set of SHRs (from 1 to 10) with the same physical routing path. Moreover, in the general European context of deregulation, cost saving is a strategic issue for FT.

The basic problem to be solved can be summarized as follows:

As input, we are given a cluster which can be viewed as a set of Self-Healing-Rings (circular fibers) with the same routing path. Cables with 10, 20, 30 or more fibers are installed. The traffic demand between all pairs of centers is known.

As output, we give the number of SHRs and for each ring the set of demands assigned to the ring with the assignment of Add-Drop-Multiplexers on that ring.

The constraints are:

- for each ring, the sum of all traffic does not exceed the maximum capacity of the ring (all STM-1, or all STM-4, ...),
- if the demand between centers i and j is routed on (assigned to) a ring, then ADMs are installed on that ring at centers i and j,
- all demands are routed (if a demand exceeds the capacity, we must also specify how the demand is broken up).

Note that in this problem, once the cables are installed, the fiber is supposed to be free. The only objective is to minimize the cost of the ADM equipment.

Three other practical variants of the problem are also modeled and discussed.

i) Suppose that for reasons of security in case of an ADM failure, it is decided to split each demand (i, j) into two with each half necessarily being transmitted on a different ring. What is the increase of ADM cost?

ii) Suppose on the other hand that we allow arbitrary integer splitting of each demand (i, j) between one or more rings. How much is gained, if any, from this increased flexibility?

iii) Finally suppose that we allow arbitrary integer splitting, but for security reasons at most half of any demand can be assigned to a single ring. How much is gained relative to *i*) or lost relative to *ii*)?

Kling and Sutter (1992) [10] contains a first presentation of the basic model discussed here. They show results with a greedy heuristic and suggest the use of the partitioning formulation presented below.

2 Formulations

An instance of the problem can be represented as a graph $G = (V, E)$ where the $n = |V|$ nodes are the telecommunication centers in the cluster and the $m = |E|$ edges denote pairs of centers between which there is a traffic demand, i.e. $d_e > 0$ for $e \in E$. Let $C > 0$ denote the capacity of each ring. Note that even though we talk of "rings", for our problem this does not imply an ordering of the nodes of the graph. For demands with $d_e \geq C$, we use $\lfloor \frac{d_e}{C} \rfloor$ rings at full capacity, and then treat the remaining demand as d_e. Thus without loss of generality we assume that $d_e < C$ for all $e \in E$.

Letting

$y_e^k = 1$ if demand e is assigned to ring k, and $y_e^k = 0$ otherwise, and

$x_i^k = 1$ if a multiplexer is installed at node i for ring k, and $x_i^k = 0$ otherwise,

we obtain a natural integer programming formulation (F):

$$z = \min \sum_{k=1}^{K} \sum_{i \in V} x_i^k \tag{1}$$

$$\sum_{k=1}^{K} y_e^k = 1 \text{ for } e \in E \tag{2}$$

$$y_e^k \leq x_i^k, y_e^k \leq x_j^k \text{ for } e = (i,j) \in E, k = 1, \ldots, K \tag{3}$$

$$\sum_{e \in E} d_e y_e^k \leq C \text{ for } k = 1, \ldots, K \tag{4}$$

$$y_e^k, x_i^k \in \{0,1\} \; e \in E, i \in V, k = 1, \ldots, K \tag{5}$$

Unfortunately applying a standard mixed integer programming package directly to this formulation is very ineffective. A good a priori limit K on the number of rings needed is hard to obtain, and the bound on the objective value obtained from the linear programming relaxation is very weak (setting $y_e^k = x_i^k = 1/K$ for all i, e, k shows that its value does not exceed $|V|$). In addition all the rings have identical capacity, so each feasible solution is repeated an enormous number of times just by permuting the numbering of the rings, which leads to well-known difficulties in branch and bound. Some results presented in the computational section show that, even after tightening, the formulation (1)-(5) is very weak and is ineffective in practice.

The problem variants can be formulated similarly.

Variant 1. The demand d_e must be split, so if $d_e < 2C$, we replace $e = (i,j)$ by two parallel edges $e^1 = e^2 = (i,j)$ with $d_{e^1} = \lfloor \frac{d_e}{2} \rfloor, d_{e^2} = \lceil \frac{d_e}{2} \rceil$. We then replace the constraint (3) for edge e by constraints $y_{e^1}^k + y_{e^2}^k \leq x_i^k, y_{e^1}^k + y_{e^2}^k \leq x_j^k$ for all k. If $d_e \geq 2C$, we use $\lfloor \frac{d_e}{C} \rfloor$ rings at full capacity and treat the remainder as a demand d_e that does not have to be split.

Variant 2. The demand d_e can be split and any integer amount allocated to each ring. Here we can treat demands with $d_e > C$ directly. Now we let $y_e^k \in Z_+$ be the amount of demand e assigned to ring k.

The resulting modified formulation is:

$$\sum_{k=1}^{K} y_e^k \;=\; d_e \text{ for } e \in E$$

$$\begin{aligned} y_e^k \le \tilde{d}_e x_i^k, y_e^k &\le \tilde{d}_e x_j^k \text{ for } e=(i,j)\in E, k=1,\dots,K \\ \sum_{e\in E} y_e^k &\le C \text{ for } k=1,\dots,K \\ y_e^k \in Z^+, x_i^k &\in \{0,1\}\ e\in E, i\in V, k=1,\dots,K \end{aligned}$$

where $\tilde{d}_e = \min\{d_e, C\}$.

Variant 3. The formulation is identical to that of Variant 2 except that $\tilde{d}_e = \min\{\lceil \frac{d_e}{2} \rceil, C\}$.

Another view of the basic problem is that on any potential ring layout q, the set of demands (edges) E^q and associated multiplexers (nodes) $V^q = V(E^q)$ must form an edge induced subgraph of G of cost $\mid V^q \mid$. This leads to an alternative formulation as an edge partition problem:

Find an edge partition $\{E^1,\dots,E^p\}$ of E such that $\sum_{e\in E^i} d_e \le C$ for $i=1,\dots,p$, and $\sum_{i=1}^p \mid V^i \mid$ is minimized.

This can then be written as an integer programming partitioning problem:

$$z = \min \sum_{q\in Q} c_q \lambda_q \tag{6}$$

$$\sum_{q\in Q} a_{eq}\lambda_q = b_e \text{ for } e\in E \tag{7}$$

$$L \le \sum_{q\in Q} \lambda_q \le K \tag{8}$$

$$0 \le \lambda_q \le 1 \quad q\in Q \tag{9}$$

$$\lambda_q \text{ integer } \quad q\in Q \tag{10}$$

where $\{E^q\}_{q\in Q}$ is a complete list of feasible edge assignments for a ring, a_{eq} is 1 if $e\in E^q$ and is 0 otherwise, b_e=1 for all e, $c_q = |V^q|$, and λ_q is the number of times E^q with incidence vector a_q is chosen. Without loss of generality we have added constraint (8) requiring a lower bound L and an upper bound K on the number of rings used. Feasible columns are precisely the solutions of $X = \{(x,y)\in\{0,1\}^n \times \{0,1\}^m : \sum_e d_e y_e \le C, y_e \le x_i, y_e \le x_j \text{ for } e=(i,j)\in E\}$.

The variants can also be handled by generalizing this formulation slightly.

Variant 1. To handle the split demands, we treat the cases where d_e is even or odd slightly differently. (If not, the number of partition constraints is immediately doubled). When d_e is even, edge e has demand $\lfloor \frac{d_e}{2} \rfloor$ twice. $a_{eq}=1$ means that ring q contains one of these split demands, and we set $b_e = 2$ in the corresponding partition constraint (7):

$$\sum_{q\in Q} a_{eq}\lambda_q = 2 \text{ for } e \text{ with } d_e \text{ even }.$$

When d_e is odd, constraint (7) associated with row e is replaced by two constraints corresponding to rows e^1 and e^2 both with right hand side equal to 1, and with the additional restriction on the columns that $a_{e^1q} + a_{e^2q} \leq 1$. Feasible columns are precisely the solutions of

$$X = \{(x,y) \in \{0,1\}^n \times \{0,1\}^{m'} :$$
$$\sum_{e:d_e even} \frac{d_e}{2} y_e + \sum_{e:d_e odd} (d_{e^1} y_{e^1} + d_{e^2} y_{e^2}) \leq C,$$
$$y_e \leq x_i, y_e \leq x_j \text{ for } d_e \text{ even } e \in E$$
$$y_{e^1} + y_{e^2} \leq x_i, y_{e^1} + y_{e^2} \leq x_j \text{ for } d_e \text{ odd } e \in E\}$$

where $m' = m + \mid \{e \in E : d_e \text{ odd}\} \mid$.

Variants 2 and 3. Here we change the scale so that $0 \leq a_{eq} \leq \tilde{d}_e$ with $a_{eq} \in Z_+$ represents the number of units of demand d_e assigned to pattern q. Now the right hand side of the partitioning constraint (7) becomes $b_e = d_e$. Feasible columns are precisely the solutions of $X = \{(x,y) \in \{0,1\}^n \times Z^m : \sum_e y_e \leq C,\ y_e \leq \tilde{d}_e\, x_i,\ y_e \leq \tilde{d}_e\, x_j \text{ for } e = (i,j) \in E\}$, where $\tilde{d}_e = \min\{d_e, C\}$ in Variant 2 and $\tilde{d}_e = \min\{\lceil \frac{d_e}{2} \rceil, C\}$ in Variant 3.

Observe that the set-partitioning formulation is precisely the formulation obtained by applying Dantzig-Wolfe decomposition to the original formulation (F) with constraints (2) kept in the Master.

3 Heuristic and Exact Algorithms

Here we describe briefly the approaches we have developed to obtain

i) good feasible solutions quickly (which provide upper bounds on the optimal value),

ii) tight lower bounds which permit us to check whether these feasible solutions are close to optimal,

iii) exact optimal solutions.

The good feasible solutions are found initially by using simulated annealing heuristics. Tight lower bounds are then obtained using a column generation approach to solve the linear programming relaxation (6)-(9). The final step to obtain optimal solutions involves the insertion of the column generation approach into a branch-and-bound algorithm. The goal below is not to describe the algorithms in detail, but to indicate the important choices made and to provide appropriate references.

3.1 Heuristic Algorithms

Heuristic 1

A simulated annealing algorithm has been implemented for the basic problem. A "solution" is an assignment of edge demands to rings. The 'neighborhood" is implicitly defined by the two moves described below:

- **transfer**: Select an edge e and a ring k with e not assigned to k in the current solution. Transfer edge e to ring k.
- **exchange**: Select edges e_1 and e_2 that are assigned to different rings k_1 and k_2 respectively in the current solution. Exchange the edges by assigning e_1 to ring k_2 and e_2 to ring k_1.

The cost of a solution is just the number of multiplexers required if the assignment is feasible, and a solution is immediately rejected otherwise.

Variant 1 can be handled by a minor modification that takes into account the splitting of the demands.

Heuristic 2

A second simulated annealing algorithm has been developed for Variants 2 and 3. Here a "solution" is an assignment of multiplexers to rings. One can test if a given multiplexer assignment permits one to partition all the demands among the rings by solving a max flow problem: Given a multiplexer assignment $x^k \in \{0,1\}^{|V|}, k = 1, \ldots K$, define a network with node set $N = E \cup \{1, \ldots, K\} \cup \{s, t\}$, arcs from s to e with capacity d_e for all $e \in E$, arcs from e to k with capacity $\tilde{d}_e \min\{x_i^k, x_j^k\}$ for $e = (i,j) \in E$ and $k = 1, \ldots, K$, and arcs form k to t for $k = 1, \ldots, K$; the current assignment x is feasible if the associated maximum $s - t$ flow is equal to $\sum_e d_e$. More specifically, we solve:

$$\begin{array}{rcl}
\zeta(x) & = & \max \sum_k \sum_e y_e^k \\
\sum_k y_e^k & \leq & d_e \quad e \in E \\
\sum_e y_e^k & \leq & C \quad k = 1, \ldots, K \\
y_e^k & \leq & \tilde{d}_e \min\{x_i^k, x_j^k\} \;\; e = (i,j) \in E, k = 1, \ldots, K \\
y_e^k & & \text{integer} \quad e \in E, k = 1, \ldots, K
\end{array}$$

where y_e^k is the amount of demand on edge e assigned to ring k. We know that in an extreme optimal flow y_e^k is integer as the data $\tilde{d}_e, C$ are integer. The assignment x is feasible if and only if $\zeta(x) = \sum_e d_e$.

The "neighborhood" of a solution defined by an assignment x is implicitly given by the two following types of moves:

- **flip ADM**: Select a node i and a ring k. If $x_i^k = 1$, set $x_i^k = 0$. Otherwise, if $x_i^k = 0$, set $x_i^k = 1$.

- **exchange ADM**: Select a node i and two rings k_1 and k_2 with $x_i^{k_1} \neq x_i^{k_2}$. Exchange the values of $x_i^{k_1}$ and $x_i^{k_2}$.

The cost $F(x)$ of a solution x is the ADM installation cost plus a penalty associated with the flow infeasibility

$$F(x) = \sum_i \sum_k x_i^k \; + \; \alpha \; (\sum_e d_e - \zeta(x)),$$

where $\alpha > 0$ is a scalar parameter that specifies the relative weight of the penalty in the cost function.

3.2 LP Column Generation

A natural approach for solving linear partitioning problems such as (6)-(9) with an enormous number of columns is to generate columns if and when they are needed, and handle the others implicitly. Indeed column generation algorithms for linear programs date back to the 1960s (Dantzig-Wolfe (1960)). The cutting stock problem, treated by Gilmore and Gomory (1961,1963), is a well-known example in which the columns are obtained as feasible solutions to an integer program. For the basic problem and its variants, the standard approach has been adopted:

i) Solve a *Restricted LP Master Problem* (linear programming relaxation (6)(7) and (9) with a subset of the columns).

ii) Take optimal dual variables π associated with the constraints (7), and search for a missing column with a negative reduced cost by solving the *column generation subproblem*:

$$\omega = \min\{\sum_i x_i - \sum_e \pi_e y_e : (x, y) \in X\} \tag{11}$$

Note that adding cardinality constraints (8) would simply amount to adding a constant to column reduced costs.

iii) If $\omega = 0$, the LP Master Problem with value z^{LP} has been solved, and $\lceil z^{LP} \rceil$ is a lower bound on the optimal integer value z.

iv) Otherwise add one or more columns with negative reduced cost to the Restricted LP Master, and iterate.

For the basic problem, the subproblem (11) takes the form:

$$\omega = \min \sum_i x_i - \sum_e \pi_e y_e$$

$$\begin{aligned} y_e \leq x_i, y_e \leq x_j \quad & e = (i,j) \in E \\ & \sum_{e \in E} d_e y_e \leq C \\ & y_e, x_i \in \{0,1\} \end{aligned}$$

Taking $d_e = \pi_e = 1$ for $e \in E$ and $C = k(k-1)/2$, it is easily checked that $\omega = k - k(k-1)/2$ if and only if the graph contains a k-clique [3]. It follows that the subproblem is strongly NP-hard. For the model variants, similar restrictions (taking $d_e = 2$ for $e \in E$ in Variants 1 and 3) demonstrate the complexity of the subproblem. Therefore as the subproblem is difficult, and no special purpose software was available, we chose to use a commercial MIP code. For the basic problem, an attempt is also made to find (possibly several) negative reduced cost columns using a greedy heuristic. The subproblem is only solved exactly when the heuristic is unsuccessful.

In designing a successful implementation of this algorithm, attention also needs to paid to several points such as:

i) the choice of artificial variables to introduce in Phase 1 of the simplex algorithm and

ii) the tailing-off effect (in which often 10-20% of the iterations are required for the last 1% increase in objective value) .

In our implementation phase 1 of the simplex algorithm is handled by the introduction of a single artificial variable, a vector of 1's whose cost is modified in the course of the algorithm [15], [17]. The tailing-off effect is tackled by the explicit use of early termination criteria based on the observation that the optimal value z is integer. Using duality, this allows us to terminate column generation once $\lceil z^{LP} \rceil$ is known, even though z^{LP} has not been determined exactly. The early termination criteria can also be used to determine a value $\epsilon < 0$ such that all subproblem solutions with value exceeding ϵ can be cut off. Details can be found in Vanderbeck and Wolsey [16]. Finally as suggested in the literature, the master problem is solved with $\geq$ rather than $=$ constraints for reasons of stability.

3.3 IP Column Generation

Work on exact column generation algorithms for 0-1 integer programs originated largely in the 80's with an extensive body of work on routing and scheduling, in which the subproblems are restricted shortest path problems of various types. More recently other researchers have started using the approach, which has also been called branch-and-price. Barnhart et al. [1], Desaulnier et al. [6] and Desrosiers et al. [7] present recent surveys. The basic difficulty is to find an effective way to embed LP column generation in a branch and bound scheme. The important questions concern the choice of branching rules, and the resulting modification of the column generation subproblem at nodes of the branch and bound tree. Several successful applications have been reported. In most of these applications, either there are K non-indentical subproblems with associated convexity constraints of the form $\sum_{q\in Q^k} \lambda_q^k = 1$ for each subproblem $k = 1, \ldots, K$ or the coefficients of the constraint matrix (7) satisfy $a_{eq} \in \{0, 1\}$ and the right hand side vector b is a vector of 1's. In the first case, branching is straightforward as the branching constraints can be expressed in terms of the variables of the subproblems and therefore branching does not require any modifications to the subproblems [6]. In the second case, a branching scheme due to Ryan and Foster [13],[14] is used. When the master variables are not restricted to 0-1 as is the case in our variants 1-3, a more elaborate branching scheme such as that of Vance et al. [14], Vanderbeck and Wolsey [16] or Vanderbeck [17] is required. Even when not required, the latter branching schemes are in certain cases more efficient, yielding fewer nodes in the enumeration tree.

The Basic Problem .

Here $a_{eq}, b_e \in \{0, 1\}$, so the Ryan and Foster branching scheme is theoretically sufficient to eliminate any fractional solution ([14], [15]). However, we also use more "global" branching rules that are helpful given the symmetric structure of the solution space. Our implementation incorporates three specific choices of branching rules:

i) In an integer solution the number of rings used (the sum of all the variables) must be integer, i.e.

$$\sum_{q\in Q} \lambda_q \in Z_+ \tag{12}$$

ii) The number of ADMs needed at a node must be integer, i.e.

$$\sum_{q\in Q: a_{i\,q}=1} \lambda_q \in Z_+, \tag{13}$$

where we introduce the notation $a_{i\,q} = 1$ if $a_{e\,q} > 0$ for some edge e incident to node i.

iii) The Ryan and Foster scheme consists here in partitioning the set of solutions into those in which two specific edges lie on different rings, and those in which they lie on the same ring, i.e.

$$\sum_{q\in Q: a_{e\,q}=a_{f\,q}=1} \lambda_q = 0 \text{ or } 1 \text{ respectively.} \tag{14}$$

where $e, f \in E$ are the specified pair of edges.

The above rules amount to defining a subset $\tilde{Q} \subseteq Q$ of variables and enforcing that the sum of these variables is integer. If $\sum_{q\in\tilde{Q}} \lambda_q = \alpha \notin Z_+$, we branch on $\sum_{q\in\tilde{Q}} \lambda_q \leq \lfloor\alpha\rfloor$, and $\sum_{q\in\tilde{Q}} \lambda_q \geq \lceil\alpha\rceil$. Each rule must be completed by introducing additional constraints or variables in the subproblem, so that the column generated satisfies the branching constraints, or the column reduced costs are appropriately modified to reflect the branching constraints. In case *i*), the branching constraints take the form (8) and hence modify the column reduced costs by a constant. In case *ii*) the cost associated with the node concerned by the branching constraint is amended in the subproblem. In case *iii*), branching results in adding a constraint of the form $x_e + x_f \leq 1$ or $x_e = x_f$ in the subproblem. In each case the Phase 1 and early termination procedures described above are unaffected by the modifications resulting from branching.

Variant 1.

In this case $a_{eq} \in \{0,1\}$, $b_e \in \{1,2\}$, and $\lambda_q \in \{0,1,2\}$, and we apply the scheme proposed in Vanderbeck and Wolsey [16]. In principle this means that one might have to branch on a subset of variables defined by fixing the assignment of up to three edges, so as to eliminate all fractional solutions. However, for the instances reported on in the next section, a generalization of rule *iii*) involving only two edges has sufficed. Branching rule (14) is only valid if either e or f, say e, is an odd demand edge, i.e. $b_e = 1$. Then branching results in adding either $x_e + x_f \leq 1$ or $x_e \leq x_f$ in the subproblem. If both e and f are even demand edges, i.e. $b_e = b_f = 2$, $\sum_{q \in Q: a_{eq} = a_{fq} = 1} \lambda_q \in \{0,1,2\}$, then branching results in adding a binary variable in the subproblem that takes value 1 if and only if both e and f are selected in the subproblem solution and that comes in the expression of the column reduced cost with a weight equal to the master dual variable associated with the branching constraint.

Variants 2 and 3.

The coefficients a_{eq} are integers varying between 0 and d_e. The branching rules used here are specific cases of the general branching framework of [16] and [17]. In Variants 2 and 3, branching aims at enforcing integrality of the ADM assignments. Indeed, once an optimal integer assignment of the multiplexers is known, an associated optimal assignment of the demands can be obtained by solving a max-flow problem. Therefore we successively enforce the integrality of the number of rings used (see (12)), the total number of ADMs used (adding a cut in the master that rounds up the objective value), the number of ADMs at any given node (see (13)), the number of rings with ADMs at two specific nodes i and j (i.e. $\sum_{q \in Q: a_{i\,q} = a_{j\,q} = 1} \lambda_q \in Z_+$), and the number of rings sharing 3 specific nodes i, j, and l (i.e. $\sum_{q \in Q: a_{i\,q} = a_{j\,q} = a_{l\,q} = 1} \lambda_q \in Z_+$). The modifications resulting from the last two branching rules involve adding one binary variable in the subproblem whose value is a function of the node assignment variables concerned.

We have also implemented primal heuristics in an attempt to contruct an improved incumbent solution. For the basic problem and Variant 1, it is a simple greedy algorithm for a partitioning problem, using the existing set of columns. For Variants 2 and 3, we round up the current ADM assignments and solve a max-flow problem in the search for a feasible integer solution.

4 Computational Results

Here we report on computational experience with the heuristic and exact algorithms for the basic problem and its three variants. The first simulated annealing heuristic is described in more detail in Kling and Sutter(1992). The second with arbitrary splitting of the demands uses the solution and neighborhood described in Section 3.1. The LP and IP column generation algorithms described above were initially implemented using MINTO (Nemhauser and Savelsberg (1993)). They have now been implemented directly in C using the CPLEX subroutine library Version 3.0 [4]. CPLEX is used to solve the linear programming master problems, and CPLEXMIP to solve the subproblems.

The test instances have been adapted from a series of real data sets. The number of nodes and edges(demands) varies from $(7, 21)$ to $(15, 72)$, and in many instances the graphs are almost complete. The instances are run with several different capacities, namely 50, 60,100, 200, 214 and 240. The name of each instance is $n.m.C$ where n is the number of nodes, m the number of edges and C is the capacity, with a final letter b if two instances have identical parameters.

In Table 1 we present the bounds and running times (on a HP 9000/712/80) obtained using both the heuristics and exact IP column generation algorithms. There are four columns for each of the four models: the basic problem and Variants 1, 2 and 3 respectively.

In the first column (RB) we give the initial lower bound $\lceil z^{LP} \rceil$ obtained by solving the LP Master problem at the root node of the branch and bound tree and in the third column (HR) z^H is the initial upper bound obtained from the simualated annealing heuristic. In the second ([L,U]) we give the final interval $[\underline{z}, \overline{z}]$ where $\underline{z}$ is the best lower bound obtained and $\overline{z}$ is the best upper bound after running the IP Column Generation Algorithm for at most 4 hours. Note that the values are nondecreasing from left to right with $\lceil z^{LP} \rceil \leq \underline{z} \leq \overline{z} \leq z^H$. If the interval $[\underline{z}, \overline{z}]$ is a single point, the instance has been solved to optimality and $z = \underline{z} = \overline{z}$. The fourth column shows the CPU time required to obtain the final interval $[\underline{z}, \overline{z}]$, excluding that for the simulated annealing heuristic. When optimality is proven, it is the total run time. All other instances were run for a total of 4 hours, but no further improvement of the bounds was observed.

Table 1 shows that the heuristics perform remarkably well. In fact $z^H \leq z+1$ for all instances that have been solved to optimality and $z^H = z$ for most of them. What is more for the large majority of instances, if one just runs the heuristic and the LP column generation algorithm *without* branch and bound, one obtains $z^H \leq \lceil z^{LP} \rceil + 1$, namely a feasible solution guaranteed to be within one of optimal. For Variant 2 and 3, the gaps between the lower and upper bounds $\lceil z^{LP} \rceil$ and z^H tend to be larger. In such cases we are not always able to prove optimality within the allocated CPU time.

Table 1 also permits us to compare the optimal values obtained from the basic problem and its two variants. Although Variant 2 never requires more multiplexers than the basic problem, we observe that the values of Variant 2 and the basic problem are almost identical. However imposing that the demands be split into two (Variant 1) increases the multiplexer costs by between 20 and 50%. With Variant 3 we see that when security constraints (split demands) are imposed, increased flexibility again fails to significantly improve the multiplexer costs.

Table 2 gives details and statistics concerning the exact IP column algorithm so as to give an idea of how such an algorithm behaves. Here there are five columns for each model: *nod* denotes the total number of nodes in the branch-and-bound tree, *nbSP* is the total number of subproblems solved, *time* here represents the total run time, *mas* is the percentage of time spent solving the restricted LP Master problems, and *spr* is the percentage of time spent solving the subproblems. The table shows clearly that in most cases the time spent in solving the subproblems is a crucial factor.

We have also attempted to solve two of the easier instances of the basic problem with the initial mixed integer programming formulation (1)-(5) tightened by symmetry-breaking constraints. Instance 7.21.60b is solved in 26M:45S with CPLEX 3.0. on a Sun 10, model 51. Instance 8.28.60b ran for 12 hours and produced lower and upper bounds $[L, U] = [26, 44]$. This compares with solution times of 2 and 27 seconds respectively with column generation, see Table 1.

5 Conclusions

We believe there are several tentative conclusions that can be drawn from this paper:

i) Even if fast heuristics are developed for use in practice, it is possible in certain circumstances to use optimisation methods to verify that the solutions obtained are close to optimal, and thus strengthen both the modeler's and the enduser's confidence in the use of the heuristic. For many of the instances treated here, this is achieved by using column generation, but without having to resort to branch and bound.

ii) Exact optimal solutions can sometimes be found to IP column generation problems even when the master coefficients and variables can take integer values not restricted to 0-1, as in Variants 1, 2 and 3, and the subproblems are themselves difficult integer programs. This provides a new problem class where integer programming column generation is effective, adding to the numerous successful applications reported in [],[],[]. This also indicates that for certain problems the theoretical framework introduced in Vanderbeck and Wolsey can be usefully put into practice. ??Moreover, the simplest of these branching rules appears to be the most useful.??

iii) Our efforts to solve to optimality instances with more than fifteen nodes per cluster have failed in most cases because of the excessive amount of time needed to solve the column generation subproblem. However, France Telecom clusters typically involve less than 10 nodes. Improved algorithms for the optimal / heuristic construction of a single ring appear to be crucial to further progress.

iv) More generally, the modelling of telecommunications networks of the future is an iterative process as the technology evolves, and the practical performance of material is observed. One such aspect is a recent request to incorporate additional constraints on the capacity of certain multiplexers. The column generation approach has the advantage that such a constraint can be easily and rapidly incorporated. Results for this new variant are not presented here.

Acknowledgements: This research was financed in part by the CNET, France Telecom under research contract no. 94 5b, entitled: Optimisation de trajectoires d'installation des Multiplexeurs d'Insertion Extraction dans des anneaux de transmissions synchrones.

References

[1] BARNHART, C., JOHNSON, E.L., NEMHAUSER, G.L. and M.W.P. SAVELSBERGH. 1994. Branch and Price: Column Generation for Solving Huge Integer Programs. Computational Optimization Center COC-94-03, Georgia Institute of Technology, Atlanta.

[2] BARS G., JONEOUR, G. and T. STEPHANT. 1991. Présentation générale de la hiérarchie numérique synchrone. NT/LAR/SMR/70 CNET, France Telecom, Lannion, 1991.

[3] D. Bienstock 1996. Private communication.

[4] *Using the CPLEX Linear Optimizer*, version 3.0. (1994). CPLEX Optimization, Inc., Suite 279, 930 Tahoe Blvd., Bldg. 802, Incline Village, NV 89451-9436. (702) 831-7744.

[5] DANTZIG, G.B. AND P. WOLFE. 1960. Decomposition Principle for Linear Programs. *Operations Research* **8**, 101-111.

[6] DESAULNIER G., J. DESROSIERS, I. IOACHIM, M.M. SOLOMON, and F.SOUMIS. 1994. A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems, Les Cahiers du GERAD, G-94-46.

[7] DESROSIERS, J., DUMAS, Y., SOLOMON, M.M. and F. SOUMIS. 1994. Time Constrained Routing and Scheduling, Chapter 11 in *Handbooks in Operations Research and Managament Science: Networks*, Ball M.E., Magnanti, T.L., Monma C. and G.L. Nemhauser (eds.). Amsterdam, North-Holland.

[8] GILMORE, P.C. and R.E. GOMORY. 1961. A Linear Programming Approach to the Cutting Stock Problem. *Operations Research* **9**, 849-859.

[9] GILMORE, P.C. and R.E. GOMORY. 1963. A Linear Programming Approach to the Cutting Stock Problem: Part II. *Operations Research* **11**, 863-888.

[10] KLING, O. and A. SUTTER. 1992. Affectation et Optimisation des M.I.E. dans les Anneaux SDH Heuristique et Validation. DE/ATR/119.92, CNET, France Télécom.

[11] NEMHAUSER, G.L. and M. SAVELSBERGH. 1993. MINTO Users guide, Georgia Institute of Technology.

[12] NEWALL, C. (ed.) 1992. Synchronous Transmission Systems. DOC GII9, Issue 3, Northern Telecom, London.

[13] RYAN, D.M. and B.A. FOSTER. 1981. An Integer Programming Approach to Scheduling. *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, A. Wren (ed.). Amsterdam, North-Holland, 269-280.

[14] VANCE, P.H., C. BANHARD, E.L. JOHNSON, and G.L. NEMHAUSER 1994. Solving Binary Cutting Stock Problems by Column Generation and Branch-and-bound. *Computational Optimization and Applications* **3**, p. 111-130.

[15] VANDERBECK, F. 1994. Decomposition and Column Generation for Integer Programs. Ph.D. Thesis. Faculté des Sciences Appliqées, Université Catholique de Louvain, Louvain-la-Neuve.

[16] VANDERBECK, F. and L.A. WOLSEY. 1994. Exact IP Column Generation. CORE Discussion Paper 9419. Université Catholique de Louvain, Louvain-la-Neuve.

[17] VANDERBECK, F. 1995. On Integer Programming Decomposition and Ways to Enforce Integrality in the Master. *Research Papers in Management Studies*, 1994-1995 N0 29, University of Cambridge.