Capacity Planning Under Uncertain Demand in Telecommunications Networks

LISSER, A., *et al*.

Abstract

This paper deals with the sizing of telecommunications networks offering private line service to a few clients. The clients ask for some transfer capacity between some pair of nodes, but their demand is uncertain. In case of high demand and insufficient capacity, some clients may be denied the transfer; the telecommunications company pays a penalty cost for that. The network has a fixed topology. In planning the network capacity, the company wants to balance the investment cost with the total expected penalty cost. The planning situation is modeled as a stochastic programming problem. The scenarios are built under the assumption that the clients have independent demands. The solution method is based on Benders decomposition coupled with the analytic center solution method. We solve some large size problem instances. For one problem instance, we perform sensitivity analysis and draw the trade-of cost curve vs. the unitary penalty cost. Finally, we run the algorithm on a parallel computing platform.

Capacity Planning Under Uncertain Demand in Telecommunications Networks *

A. Lisser[†], A. Ouorou[‡], J.-Ph. Vial[‡] and J. Gondzio[§]

October 8, 1999

Abstract

This paper deals with the sizing of telecommunications networks offering private line services to a few clients. The clients ask for some transfer capacity between some pair of nodes, but their demand is uncertain. In case of high demand and insufficient capacity, some clients may be denied the transfer; the telecommunications company pays a penalty cost for that.

The network has a fixed topology. In planning the network capacity, the company wants to balance the investment cost with the total expected penalty cost. The planning situation is modeled as a stochastic programming problem. The scenarios are built under the assumption that the clients have independent demands. The solution method is based on Benders decomposition coupled with the analytic center cutting plane method. We solve some large size problem instances. For one problem instance, we perform sensitivity analysis and draw the trade-off cost curve vs. the unitary penalty cost. Finally, we run the algorithm on a parallel computing platform.

Key words: Telecommunication network design, stochastic programming, decomposition, network flows.

*Research funded by CNET-France Telecom

[†]France Telecom, CNET, 38-40 rue de Général Leclerc, F-92131 Issy-les-Moulineaux, France.

[‡]Logilab, HEC, Section of Management Studies, University of Geneva, 102 Bd Carl-Vogt, CH-1211 Geneva 4, Switzerland.

[§]Department of Mathematics & Statistics, The University of Edinburgh, Mayfield Road, Edinburgh EH9 3JZ, United Kingdom.

1 Introduction

The investment for link capacities in network design is a basic problem in the telecommunications industry. In the present paper, we consider its important variant in which the demands are uncertain. This variant is relevant to virtual networks aiming to serve few important clients whose demands are liable to vary from one period of time to the next one.

Demands consist of bandwidth requests between some origin-destination nodes in transport network. A request is successful if the requested bandwidth is allocated to it. The allocation remains in force until a disconnection request is received. Since bandwidth requests vary from one period to another, the telecommunications company must routinely face the problem of allocating bandwidths to the clients and determining the routing of the messages. In the event of specially high demands, the existing network capacity may not suffice to meet all demands. The company must decide which customers will not be served; in case of failure to meet some demand, the company would pay a penalty cost. The obvious strategy to decrease the risk of failures and the amount of unserved demand consists in installing extra link capacity. The main issue in this paper is to find a reasonable tradeoff between the capacity investment cost and the expected cost of unserved demands.

Since the company may influence demands by offering different kind of contracts with different service qualities and different penalty costs, there is much fuzziness in the choice of a satisfactory capacity investment. There is a need for a decision support system to analyze various options and find where capacity is most needed and by which amount. The present study emphasis on the arbitrage between investment for capacity and costs of not serving the demand.

The problem is naturally formulated as a two stage stochastic programming problem with recourse. The first stage deals with investment for link capacities. The second stage, or *recourse* problem, concerns the bandwidth allocation and the associated routing. The recourse problem is typically a multicommodity flow, a problem well studied in the literature [3, 15, 6, 9]. Assuming that the cost of unserved demand is linear, the optimal value of the recourse problem becomes a convex function of the right-hand side of its capacity constraints. However, it is well-known that this function is piecewise linear, thus nondifferentiable. If we assume linear capacity expansion costs —a reasonable working assumption in the planning phase in search for arbitrages— we obtain that the total cost, the investment cost plus the expected failure cost, is convex. The problem involves a few variables only, but it is clearly nondifferentiable.

The literature abounds with methods for solving nondifferentiable problems. In the present study, we use the *Analytic Center Cutting Plane Method*, in short ACCPM, introduced in [11], with its specialized implementation for stochastic programming [4]. The method is related to the classic Benders decomposition method, but it differs from it in the following way. Instead of solving to optimality the current relaxed master problem, ACCPM computes an approximate analytic center of the current localization set, a polyhedral set made of all cuts previously generated. This approach has a desirable stabilization effect and prevents the occasionally slow and unpredictable behavior of Benders decomposition. ACCPM has been used with success in large-scale convex optimization [12, 4, 16, 17]. It is robust and efficient, making it possible to use it on a production basis.

Our study is similar to an earlier study of Sen et al. [20]. In their work, the authors deal with network design subject to capacity budget constraint and the objective of minimizing the expected number of unserved demands. We aim at cost optimal capacity assignment in order to minimize the expectation of rejection penalty costs. Some interesting discussions on uncertainty can be found in [22, 23] in the framework of distributed communications networks.

The paper is organized as follows. The capacity planning problem with random demand is formulated in section 2. The solution method is presented in section 3. In section 4 we discuss practical implementation issues. Section 5 presents computational results and reports on a parallel implementation of our decomposition scheme. We give our conclusion in the last section.

2 Mathematical Model

In this section, we present a mathematical model for the capacity planning under uncertainty problem. We first state the notation used throughout the paper.

Consider $G = (V, E)$ a non-oriented connected graph that represents a network with m nodes and n links. We denote by $\mathrm{b} \geq 0$ a vector in $\mathbb{R}^n$ whose components correspond to the (possibly zero) capacities that already equip the links. Let $x_j^{\min} \geq 0$ and $x_j^{\max} \geq 0$ be respectively the lower and upper capacity bounds available for link j, and c_j the unit cost for capacity investment on the link.

Let $\mathcal{K}$ denote the index set of customers. The bandwidth requirements of a customer $k \in \mathcal{K}$ are represented by a multicommodity defined by a set Q_k of single commodity flows between origin-destination pair $(O_i, D_i),\ i \in Q_k$. The demands are considered fractionable, hence for each $i \in Q_k$, let us consider $P_k(i)$ a set of (elementary) paths that can be used between O_i and $D_i, i \in Q_k$. Define P_j as the set of paths using link j and for each path p, let y_p be the flow through p.

The capacity planning problem may be formulated as a two stage problem. The first stage deals with the choice of link expansion capacities. In the second stage, the link capacity vector $\mathrm{b} + x$ is given and we seek for the most efficient utilization of the network capacity. The overall objective is to minimize the sum of the investment cost and the expected rejection costs. We assume that uncertainty in the bandwidth requests can be modeled by random variables with known probability distributions. Let $X = \{x \mid x_j^{\min} \leq x_j \leq x_j^{\max},\ j \in E\}$ be the set determined by the range capacity constraints. The problem is mathematically formulated as:

$$\begin{aligned} \min \quad & c^\top x + \mathcal{E}[\mathcal{D}(x, \tilde{d})] \\ \text{s.t.} \quad & x \in X \end{aligned} \tag{1}$$

In the above formulation, $\tilde{d}$ is a multidimensional random variable associated with the customers demands provided by the network in a study period T; $\mathcal{E}[.]$ denotes the expectation function and $\mathcal{D}$ is a random variable representing the total cost of rejection penalties. $\mathcal{D}$ is the optimal value of the recourse problem:

$$\begin{aligned} \mathcal{D}(x,d) = \quad \min \quad & \sum_{k \in \mathcal{K}} \pi_k \Big(\sum_{i \in Q_k} s_{ki} \Big) \\ \text{s.t.} \quad & \sum_{p \in P_j} y_p \leq \mathrm{b}_j + x_j, \quad j \in E, \\ & \sum_{p \in P_k(i)} y_p + s_{ki} = d_{ki}, \quad k \in \mathcal{K},\ i \in Q_k, \\ & y_p \geq 0, \ \text{for each } p, \\ & s_{ki} \geq 0, \ k \in \mathcal{K}, \quad i \in Q_k. \end{aligned} \tag{2}$$

The objective function of the above problem involves coefficients representing a customer unit rejection penalty cost $\pi_k,\ k \in \mathcal{K}$ for not satisfying the customer's demand. The outcome of the demand between an OD pair i of a customer k, is denoted by d_{ki}, then the demand vector d belongs to $\prod_{k\in\mathcal{K}} \mathbb{R}^{|Q_k|}$ (d is a possible realization of the random variable $\tilde{d}$).

The first set of constraints are the capacity constraints; they express that the capacities assigned to the links cannot be exceeded (note that the additional capacity vector x is assumed given at this stage). The slack variable s_{ik} represents the part of demand i of customer k that cannot be satisfied.

The second-stage problem (2) is a linear multicommodity flow problem: it has received considerable attention in the literature, see for instance [3, 15, 6].

Within the context of our current model, it may be desirable to consider routing costs using for instance the Kleinrock average delay function. However this function, which depends both on the total link flow and capacity, is not jointly convex in its arguments, making the problem extremely difficult from the theoretical and practical point of view. As pointed out in [19], convexity assumption is fundamental if one wants to solve very large stochastic programming problems. The method we develop and implement is general enough to handle convex routing costs provided fast algorithms for convex multicommodity flow subproblems are used, see [18] for a survey.

One of the main difficulties in stochastic programming lies in the evaluation of random functions and their expectations. Approximation techniques are used for practical solutions [14, 5]. Too many additional factors concur in the final design of a telecommunications network, especially those pertaining to technologies. Due to the fast innovation in telecommunication technologies, it is difficult to make valid costs and capacities predictions. The purpose of our model is rather to contribute to the learning process of the decision-makers, in displaying solutions which achieve good hedging against the uncertain demand.

Assume that the random demand variable is discrete, attaining only a finite number of values d_t with probability $p_t > 0,\ t = 1, \ldots, T$ (total number of outcomes), where $\sum_{t=1}^{T} p_t = 1$. Then, the capacity planning problem (1) becomes

$$\begin{aligned} \min \quad & c^\top x + \sum_{t=1}^{T} p_t \mathcal{D}(x, d_t) \\ \text{s.t.} \quad & x \in X. \end{aligned} \tag{3}$$

Concerning the data, the basic assumption is that, the capacity expansion costs c_j, the initial link capacity b_j and the link capacity bounds $x_j^{\min}$ and $x_j^{\max}$, are well-known. Though π_k may be given any value, in our test problems we use the same value

$$\pi_k = \kappa \max_{j \in E} c_j \tag{4}$$

for all the customers.

The major random factor driving uncertainty is customers demand and their probability distributions. We model the uncertainty as follows. We assume that the clients are independent. A scenario is the outcome of C independent random trials, where C is total number of customers. For simplicity, we assume that every customer chooses his demand from the set of N possible demands. Therefore there are N^C scenarios, and each scenario specifies the demand configuration of each customer. To remain tractable, the model allows only a few demand configuration per customer. This assumption, and the independence one, are not restrictive in the context of our study. Recall that we are interested in arbitrages that will serve as guidelines for later implementation decisions.

3 Using the Analytic Center Cutting Plane Method

Since the function $\mathcal{D}(., d_t)$ is piecewise linear convex, the problem (1) appears to be a nonsmooth optimization problem. Thus it can be tackled with the Analytic Center Cutting Plane Method introduced by Goffin, Haurie and Vial [11]. The basic idea of cutting plane methods is to construct a sequence $\{x^l\}$ of approximations to the solution of (3) as follows. Known pieces of the functions $\mathcal{D}(., d_t)$ are used to compute the current solution x^l at which subproblems (2) provide new information that allows to redefine the epigraph of $\mathcal{D}(., d_t)$ and to compute the next iterate x^{l+1}. For some capacity link expansion vector x^l, the dual formulation $SP(x^l)$ of the second stage problem, is given by

$$\begin{aligned}
\max \quad & (\mathrm{b} + x^l)^\top u_t + d_t^\top v_t \\
\text{s.t.} \quad & v_{t_{ki}} + \sum_{j \in p} u_{t_j} \le 0, \quad k \in \mathcal{K}, i \in Q_k,\ p \in P_k(i), \\
& v_{t_{ki}} \le \pi_k, \quad k \in \mathcal{K}, i \in Q_k \\
& u_t \le 0
\end{aligned} \tag{5}$$

From linear programming duality, we have

$$\sum_{k\in\mathcal{K}} \pi_k(\sum_{i\in Q_k} s^l_{tki}) = (u^l_t)^\top(\mathrm{b}+x^l) + (v^l_t)^\top d_t \triangleq \alpha^l_t$$

where s^l_t and (u^l_t, v^l_t) are respectively the primal and dual optimal solutions. Note that the constraints in (5) are independent of x. Therefore, for a general x and its corresponding optimal vector $(u_t(x), v_t(x))$, we have

$$\mathcal{D}(x, d_t) = (u_t(x))^\top(\mathrm{b}+x) + (v_t(x))^\top d_t \geq (u^l_t)^\top(\mathrm{b}+x) + (v^l_t)^\top d_t$$

since (u^l_t, v^l_t) is feasible for $SP(x)$. We then obtain the *optimality cut* which is a linear support of $\mathcal{D}(., d_t)$ at x^l:

$$\theta_t \geq (u^l_t)^\top(\mathrm{b}+x) + (v^l_t)^\top d_t = \alpha^l_t + (u^l_t)^\top(x-x^l)$$

The subproblem (2) is always feasible and X is defined by simple bound constraints: we do not have to generate *feasibility cuts*. Let $\{x^l\}_{l=1}^L$ be a sequence of points generated at some step. The cuts constructed at those points are used to define the *relaxed master problem* of (1):

$$\begin{aligned} \min \quad & c^\top x + p^\top\theta \\ \text{s.t.} \quad & \theta_t \geq \alpha^l_t + (u^l_t)^\top(x-x^l), \quad l=1,\dots,L, \quad t=1,\dots,T \\ & x \in X \end{aligned} \tag{6}$$

where p and θ are the T-vectors with coordinates p_t and θ_t respectively. We have an equivalent relaxed master problem

$$\begin{aligned} \min \quad & c^\top x + \tilde{\theta} \\ \text{s.t.} \quad & \tilde{\theta} \geq \gamma_l + \chi_l^\top(x-x^l), \quad l=1,\dots,L, \\ & x \in X, \end{aligned} \tag{7}$$

where $\gamma_l = \sum_{t=1}^T p_t\alpha^l_t$ and $\chi_l = \sum_{t=1}^T p_t u^l_t$. This means constructing cuts for the expectation function by averaging with the weights p_t the cuts for $\mathcal{D}(., d_t)$. As observed in [12], averaging the subgradients of all subproblems induces a loss of information and considerably slows down the algorithm. It is much preferable to introduce one cut (supporting hyperplane) for each recourse function. The way to proceed is to introduce epigraph variables θ_i, one per subproblem (scenario), and express the objective as the sum $\sum_i p_i \theta_i$. For the sake of a simpler presentation, we describe the aggregate version of the problem. For more details on the implementation of the disaggregate version, we refer to [12].

Let us rewrite the linear program (6) in a more compact form

$$\begin{aligned} \min \quad & c^\top x + p^\top \theta \\ \text{s.t.} \quad & \theta \geq \alpha^l + U^l(x - x^l), \quad l = 1, \dots, L, \\ & x \in X. \end{aligned} \tag{8}$$

The relaxed linear program (8) provides a lower bound $\underline{z}$ on the optimal solution of problem (1) while

$$\bar{z} = \min_{l=1,\dots,L} \left\{ c^\top x^l + p^\top \theta^l \right\}$$

provides an upper bound. At step l, the accuracy of the approximation to the optimal solution is given by the gap $\Delta = \bar{z} - \underline{z}$.

For a given upper bound $\bar{z}$, the set

$$\mathcal{LOC}_{\bar{z}} = \left\{ (x, \theta) : \; x \in X, \; \bar{z} \geq c^\top x + p^\top \theta, \; \theta \geq \alpha^l + U^l(x - x^l), \; l = 1, \dots, L \right\}$$

is called a *localization set*. It is the best outer approximation of the optimal set of (1). On the contrary of Kelley method, the solution proposed to the subproblems is not the minimizer of the relaxation, but the *analytic center* of the localization set. The analytic center is defined as the unique solution of the problem

$$\begin{aligned} \min \quad & \phi(x, \theta) \triangleq \ln(z - c^\top x - p^\top \theta) + \sum_{l=1}^{L} \sum_{t=1}^{T} \ln(\theta_t - \alpha_t^l - (u_t^l)^\top (x - x^l)) \\ \text{s.t.} \quad & (x, \theta) \in \mathcal{LOC}_{\bar{z}}. \end{aligned} \tag{9}$$

In other words, the analytic center is a point that maximizes the product of slacks in the localization set. It can also be viewed as a compromise between the proposals from the various subproblems and has a desirable stability property.

We are now ready to give the conceptual description of ACCPM. The method requires the solutions of the subproblems and adds cutting planes to the relaxed master problem. The main steps of its single iteration are the following:

One iteration of ACCPM

1. Compute the analytic center of $\mathcal{LOC}_{\bar{z}}$ and an associated lower bound $\underline{\tilde{z}}$.

2. Solve the subproblems (5) for each scenario $t = 1\ldots,T$, generate optimality cuts and an upper bound $\tilde{\bar{z}}$.

3. Update the bounds $\underline{z} := \max\{\underline{z}, \underline{\tilde{z}}\}$, $\quad \bar{z} := \min\{\bar{z}, \tilde{\bar{z}}\}$

4. Update the upper bound in the localization set and add the new cuts.

The method can be interpreted as a coordination scheme between the process we call *oracle* which solves the subproblems to provide the cuts and the upper bound, and the *generator* that updates the localization set, computes the analytic center and a new lower bound. The above steps are performed until a point is found such that $\Delta = (\bar{z} - \underline{z})/(1 + |\bar{z}|)$ falls below a given tolerance.

4 Implementation

As far as computations are concerned, the implementation mainly consists in putting together two main pieces of software: a query point generator and an LP solver to handle the subproblems.

4.1 Interface and problem generator

The capacity investment problem is generated from a relatively small set of parameters. The network data are the nodes, the arcs, the installed capacities and the expansion costs. The customer data concerns the list of O-D pairs per customer and the various demand configurations. Each demand configuration must be given a probability. Finally, there is a penalty cost per unit of unserved demand.

A scenario is a set of configurations, one per customer. The generator constructs the list of scenarios first, and then computes the related probabilities based on the assumption of independence among customers. The generator produces the subproblem matrix coefficients and provides all relevant information (number of subproblems, bounds on the investment for capacities, etc.) for the query point generator.

The interface links the query point generator and the subproblem solver, making it easy for the user to perform parametric analyses, e.g., on the penalty costs. The interface and the problem generator are both written in C++.

4.2 The query point generator

The query point generator is ACCPM, a general purpose code for convex nondifferentiable optimization [13]. The code is written in C++ and uses some Cholesky decomposition routines written in FORTRAN. This library has been compiled in Visual C++ under Windows NT environment. The implementation uses the standard settings of ACCPM. The code uses a priori box constraints on the variable x. There is no tuning for the user, though the relative precision parameter can be changed.

4.3 The linear multicommodity flow subproblems

In section 2 we described the multicommodity flow problem via the path-flow formulation. This is common practice in telecommunications environment, where path restrictions, on the length or on the number of arcs traversed, are often present. The path flow formulation is also relevant when the problem instance is huge and decomposition is the chosen solution method.

This is not the case for our problems of interest: the networks we consider here are small and bear no restriction on the paths. Since a decomposition approach is not appropriate, we resort to the alternative formulation based on flows on the arcs. This generates small to medium size LP's that are efficiently solved by commercial software, e.g., CPLEX. However, there may be many subproblems and each subproblem must be solved repeatedly. Therefore, great care must be taken in formulating those problems.

Let us revisit the formulation of section 2. Denote A the incidence matrix of the network G. To represent failure to serve demand d_i, $i \in Q_k$, we define the incidence vector $\gamma_i \in \mathbb{R}^m$, with $(\gamma_i)_u = 1$, if $u = O_i$, $(\gamma_i)_u = -1$, if $u = D_i$, and 0 otherwise. The arc-flow vector associated with the i-th demand is $f_i \in \mathbb{R}^m$, $i \in Q_k$, and $s_i \leq d_i$ denotes the portion of unserved demand d_i. The flow $(f_i)_{(u,v)}$ on arc $(u,v) \in E$ may be positive or negative, but it induces a load $|(f_i)_{(u,v)}|$ on the arc. To remain within a linear programming framework, we decompose the flow into a positive and negative part: $f_i = f_i^+ - f_i^-$, and have thus $|f_i| = f_i^+ + f_i^-$.

Subproblem 2 is now formulated as

$$
\begin{aligned}
\mathcal{D}(x,d) = \quad & \min \quad \sum_{k\in\mathcal{K}} \pi_k \Big(\sum_{i\in Q_k} s_i\Big) \\
& \text{s.t.} \quad \sum_{k\in\mathcal{K}} \sum_{i\in Q_k} (f_i^+ + f_i^-) \le \mathrm{b} + x, \\
& \qquad\quad A(f_i^+ - f_i^-) + \gamma_i s_i = \gamma_i d_i, \quad \forall i \in Q_k \text{ and } \forall k \in \mathcal{K}, \\
& \qquad\quad f_i^+ \ge 0,\ f_i^- \ge 0,\ 0 \le s_i \le d_i \quad \forall i \in Q_k \text{ and } \forall k \in \mathcal{K}.
\end{aligned} \tag{10}
$$

The unserved demand is treated as a flow on a virtual arc directly linking the source to the sink. This virtual arc has capacity d_i, but this value may be replaced by an arbitrary large one, since a feasible flow with $s_i > d_i$ is dominated by the feasible solution with zero flow and $s_i = d_i$. The latter has a lower cost and a zero load on the arcs.

The above formulation involves $Q = \sum_{k\in\mathcal{K}} |Q_k|$ blocks of flow constraints, each one of them associated with a customer and a pair of origin-destination nodes. We show now that it is possible to achieve a more compact formulation by aggregating all flows of O-D pairs sharing the same origin. We shall refer to it as the single-origin-multiple-destination formulation (SOMD), in contrast with the single-origin-single-destination (SOSD) formulation given in (10) [9].

Let us first give the explicit formulation of SOMD. To this end, we introduce the sets

$$S(u) = \{i \mid i \in Q_k,\ k \in \mathcal{K},\ \text{such that } O_i = u\},$$

one set for each $u \in V$. $S(u)$ collects all demands having origin in $u \in V$. We now define the matrices $\Gamma_u = \{\gamma_i\}_{i\in S(u)}$, the vector $\delta_u = \{d_i\}_{i\in S(u)}$ and the variables $\sigma_u = \{s_i\}_{i\in S(u)}$ and $\phi_u = \phi_u^+ - \phi_u^- \in \mathbb{R}^{|E|}$. The variables ϕ are the aggregate flows of all commodities sharing the same origin u. The SOMD formulation is

$$
\begin{aligned}
\mathcal{D}(x,d) = \quad & \min \quad \sum_{u\in V} \left(\sum_{k\in\mathcal{K},\, i\in Q_k \,:\, i\in S(u)} \pi_k s_i \right) \\
& \text{s.t.} \quad \sum_{u\in V} (\phi_u^+ + \phi_u^-) \le \mathrm{b} + x, \\
& \qquad\quad A(\phi_u^+ - \phi_u^-) + \Gamma_u \sigma_u = \Gamma_u \delta_u, \quad \forall u \in V, \\
& \qquad\quad \phi_u^+ \ge 0,\ \phi_u^- \ge 0,\ 0 \le \sigma_u \le \delta_u \quad \forall u \in V.
\end{aligned} \tag{11}
$$

Note that it would be possible to aggregate flows according to a common destination yielding a MOSD formulation. This choice is arbitrary, though it leads to different models. However, we have the folklore theorem:

Proposition 1 *The* SOSD *and* SOMD *formulations are equivalent.*

Proof. Any feasible solution of the SOSD formulation can be transformed into a feasible solution of the SOMD formulation by aggregating all flows emanating from a same origin. The costs are unchanged.

To prove the converse statement we first show that any optimal solution to the SOMD formulation can be retrieved by solving several independent simple flow problems, exactly one such problem per block (common origin) in the SOMD formulation. Let $\bar{\phi}$ be a globally optimal solution. The load vector induced by the flows in block u is $\beta_u = \bar{\phi}_u^+ - \bar{\phi}_u^-$. Consider the subproblem

$$\begin{aligned} \min \quad & \sum_{k\in\mathcal{K},\, i\in Q_k \,:\, i\in S(u)} \pi_k s_i \\ & A(\phi_u^+ - \phi_u^-) + \Gamma_u\sigma_u = \Gamma_u\delta_u, \quad \forall u \in V, \\ & \phi_u^+ + \phi_u^- \leq \beta_u, \\ & \phi_u^+ \geq 0,\, \phi_u^- \geq 0,\, 0 \leq \sigma_u \leq \delta_u. \end{aligned} \tag{12}$$

In this simple problem one cannot have simultaneously a direct flow $(\phi_u^+)_{(i,j)} > 0$ and an opposed flow $(\phi_u^-)_{(i,j)} > 0$ on any given arc $(i,j) \in E$. Therefore, the joint capacity constraint can be replaced by $(\phi_u^+)_{(i,j)} \leq \beta_u$ and $(\phi_u^-)_{(i,j)} \leq \beta_u$. We have thus a simple minimum cost flow problem on a capacitated network; clearly, the optimal value of this problem must be equal to the contribution of the optimal solution $\bar{\phi}$ in block u of the global formulation.

It is well-known that an optimal solution of a simple flow problem can be described in terms of flows on a set of directed paths from the source to the various sinks. By repeating the procedure on all blocks we obtain a decomposition that can be interpreted as a feasible solution to the SOSD problem. The latter must be optimal to SOSD, since any feasible solution of SOSD is feasible to SOMD and $\bar{\phi}$ is optimal for SOMD. ■

Some nodes may not be the origin of any demand, the total number of blocks in the SOMD formulation (11) is then at most m, presumably a much smaller number than Q.

Our test problems involve relatively few O-D pairs per client. So the size reduction factor is only a factor 2 or 3. Yet, it makes it worth using the SOMD formulation.

Again, failures are as special flows on virtual arcs from the source to the sinks, with bounded capacities. Let us show on an example that those constraints are necessary in SOMD contrary to SOSD. Consider the simple network of Figure 1 with four nodes and three links. The O-D pairs are $(1,3)$ and $(1,4)$ with demands $d_{13} = 2$ and $d_{14} = 1$. In the SOMD formulation node 1 is a supply node with total supply 3 and nodes 3 and 4 are demand nodes with demands 2 and 1. Arc $(1,2)$ has capacity one, and the failure costs at nodes 3 and 4 are 1 and 0 respectively. An optimal solution of the minimum cost flow problem with unbounded failures is given by the flows $f_{12} = 1$, $f_{23} = 2$ and $f_{24} = -1$ and the failures $s_3 = 0$ and $s_4 = 2$. This solution implies a direct transfer of flow from 4 to 3: it is not feasible to the original problem.

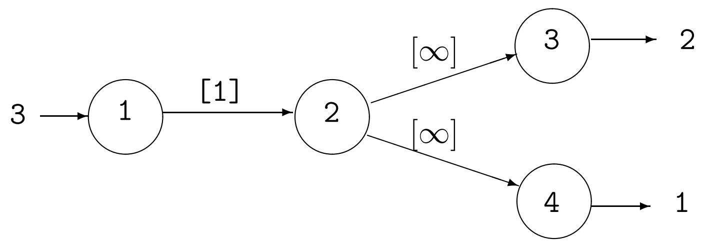


Figure 1: Academic 4 nodes-3 links network.

5 Computational Experiments

A code has been developed on the basis of section 3 to evaluate the capacity planning model. We use CPLEX 6.0 library routines for the purpose of solving the subproblems, setting the parameters of CPLEX to their default values, but we turned the option 'network' on. ACCPM allocates memory as needed: it is limited by the memory capacities of the machine at hand. First we use a PC with 400MHz and 384MB RAM running under Windows NT operating system and run a sequential implementation of our decomposition scheme. Next, we address a basic issue of a parallel implementation on a cluster of 5 Pentium processors with 300MHz and 384MB RAM running under Linux 5.1. Note that the PC's in the cluster are not as fast as the one used for the sequential implementation.

The subproblems have an important feature: they differ only in their right-hand sides. Therefore, they need not be loaded for each scenario separately. Only one linear subproblem needs to be in CPLEX memory, and we use the option change right-hand side to move from one scenario to the next. This situation is particularly favorable and makes it possible to handle the largest problems on a single PC.

Recall that at each iteration a lower and an upper bound are available. Iterations end when the relative difference between the best upper and lower bounds falls below a given tolerance. For all the numerical results that follow in this section, we set the stopping relative accuracy tolerance to 10^{-6}. The CPU time reported is always in seconds. In practice, lower tolerances might suffice to display a capacity investment able to meet satisfactorily the variety of scenarios.

5.1 Test Problems

We use a total of 16 test problems whose characteristics are given in Table 1 page 15 using four networks we wish to expand at least cost. These networks have respectively 12 nodes - 25 links, 26 nodes - 30 links, 26 nodes - 53 links and 19 nodes - 34 links. The numbers of Table 1 hide the large size of the capacity planning test problems. We give in Table 2 page 15 the corresponding dimensions[1] of the LP equivalent problems.

5.2 Algorithmic performance

To evaluate the performance of the method, we report the computing times (in the oracle and in total), and the number of outer iterations of the decomposition method. We give results for the sequential implementation on the PC 400 Mhz, and discuss the parallel implementation.

5.2.1 Ability to Handle Problem Size

We run ACCPM on all the test problems with $\kappa = 10$. Recall that κ is defined in (4) as a ratio between penalty and investment cost. Table 3, collects the results on these test problems. There are different factors that determine the problem size, the number of nodes and links of the network, the number of customers and the number of their different

[1]We consider SOMD formulation of the linear multicommodity flow problem (2).

Test problem	Network		# of customers (C)	# of demands per customer	total # of demands	# of outcomes per customer (N)	# of scenarios
	# nodes	# links					
1	12	25	5	5	25	3	3^5
2	12	25	6	5	30	3	3^6
3	12	25	7	5	35	3	3^7
4	26	30	7	5	35	3	3^7
5	26	30	7	6	42	3	3^7
6	26	30	7	7	49	3	3^7
7	26	53	8	5	40	2	2^8
8	26	53	9	5	45	2	2^9
9	26	53	10	5	50	2	2^{10}
10	19	34	8	5	40	2	2^8
11	19	34	8	6	48	2	2^8
12	19	34	8	7	56	2	2^8
13	19	34	8	8	64	2	2^8
14	19	34	8	10	80	2	2^8
15	19	34	10	5	50	2	2^{10}
16	19	34	7	10	70	3	3^7

Table 1: Characteristics of Test Problems

Test problem	First stage decision variables	Number of subproblems	Subproblem		Deterministic Equivalent LP	
			Columns	Rows	Columns	Rows
1	25	243	625	169	151900	41067
2	25	729	630	169	459295	123201
3	25	2187	635	169	1388745	369603
4	30	2187	1355	602	2963415	1316574
5	30	2187	1422	628	3109944	1373436
6	30	2187	1549	680	3387693	1487160
7	53	256	1948	521	498741	133376
8	53	512	1953	521	999989	266752
9	53	1024	1958	521	2005045	533504
10	34	256	1264	376	323618	70656
11	34	256	1340	395	343074	101120
12	34	256	1348	395	345122	101120
13	34	256	1356	395	347170	101120
14	34	256	1372	395	351266	101120
15	34	1024	1342	395	1374242	404480
16	34	2187	1362	395	2978728	863865

Table 2: Dimensions of Test Problems

configurations and finally the number of OD pairs. For instance, test problems 10-16 differ only with the number of OD pairs per customer. The number of scenarios is kept fixed. Each of the test problems 11-16 is built from the preceding by adding some OD pairs to each customer.

Some general observations can be made. First, the algorithm shows ability to solve large instances of the capacity planning problems in a relative small number of oracle calls. This number seems to be practically independent on the number of scenarios. Next, much of the time is consumed in the solutions of the subproblems, the time spent in computing analytic centers is relatively small. As mentioned before, the subproblems differ only in their right-hand sides. It is thus possible to let the problem data in the core memory, allowing an important saving in running time. Additional savings of computation time could be achieved through warm start. Presumably this could happen if consecutive scenarios are not too far from one another. We did observe a great discrepancy in the simplex iteration counts in CPLEX, from one scenario to the next, and from one oracle call to the next. However it does not seem possible to order scenarios in a favorable way, because it is not clear that there exists a quantified criterion of proximity between scenarios that would ensure efficient warm start. Our results correspond to a scenario order produced by our scenario generator.

5.2.2 Parallel Implementation

We address the parallel treatment of the subproblems using a cluster of five PC with 300MHz and 384MB of RAM and running under Linux 5.1. The parallel communication uses MPI [21]. As much of the CPU time is spent in solving the subproblems, only the subproblems are solved in parallel. We allocated one processor to the computation of the analytic center. The communication between the processors includes sending this query point to the other processors, and sending cuts from every processor to the one that handles the computation of the analytic center.

We solve the 16 test problems on 1 to 5 processors of the cluster machine. We report the results on Table 4. Clearly, the speed-ups are rather satisfactory. This nice behavior is a consequence of three favorable factors: a very small amount of time spent in computing analytic centers relative to the total time devoted in solving the subproblems, a

Test Problem	CPU Time		Capacity cost	# of oracle calls	# of cuts	Objective
	Oracle	Total				
1	96.46	97.29	1896.51	10	1119	2720.06
2	341.74	354.82	2435.90	11	4542	3273.45
3	1093.78	1163.65	3475.56	11	14006	4538.70
4	3072.63	3155.87	5791.80	16	25693	8902.14
5	3654.90	3746.30	8027.90	17	27469	11277.19
6	4392.66	4518.33	9382.55	17	30371	13320.43
7	780.07	784.58	11877.00	13	2049	14803.58
8	1772.18	1789.97	15194.57	14	4794	18234.90
9	4054.07	4116.45	17129.21	15	9285	20096.62
10	415.06	418.16	8448.90	13	1704	11291.07
11	470.41	474.29	12364.99	13	1609	15699.22
12	654.91	662.70	16097.46	16	2186	20545.79
13	670.92	679.08	20737.40	15	2203	26370.64
14	976.01	984.20	22951.70	14	2343	40795.42
15	2031.06	2065.63	13033.68	14	8464	16461.68
16	6218.43	6384.57	18904.22	14	22514	26842.07

Table 3: ACCPM on the test problems

limited amount of inter processor communication, and a good load balancing between the processors. The latter follows from the strong similarity between subproblems across the scenarios, making it likely that the solution time is roughly proportional to the number of subproblems. The simple strategy of splitting the subproblems in even numbers between the processors suffices to achieve a good load balance.

5.3 Impact of the stochastic programming approach

Here, we analyze the tradeoff between investment and failure costs as recommended by the stochastic programming approach. We also discuss on a simple example the nature of the stochastic programming solution, and show that it might be hard to predict it from a more traditional deterministic approach.

5.3.1 Investment vs Penalty Cost

First, we analyze the tradeoff between penalty costs π and capacity expansion cost. Figure 2 displays the different costs versus δ obtained with test problem 1 and Table 5, page 19, shows the detailed results for this test problem for different values of δ. As expected the total cost curve displays a concave shape. The investment cost and the failure cost have a

Test Problem	CPU Time		Speed up
	1 processor	5 processors	
1	130.08	39.34	3.31
2	388.33	110.66	3.51
3	1500.75	484.35	3.10
4	3897.86	1066.24	3.66
5	4322.29	1161.16	3.72
6	5188.03	1396.48	3.71
7	1059.90	286.22	3.70
8	2414.04	621.19	3.89
9	5020.08	1295.41	3.88
10	500.47	163.67	3.06
11	615.45	199.79	3.08
12	743.12	235.47	3.16
13	888.90	236.76	3.75
14	1234.36	324.99	3.80
15	2527.50	651.80	3.88
16	7996.66	2189.95	3.66

Table 4: Speed up on the parallel machine

less predictable shape, though it is clear that the failure cost should be zero at the extreme of the unit penalty cost range. Actually, the zero value holds for values larger than the maximum marginal cost of investment: a unit failure can always be met by an additional investment on an appropriate path from an origin-destination pair exhibiting unserved demand. (A bound for this marginal value is the sum of the unit investment costs on all arcs, because it allows the routing of at least one unit of flow from any O-D pair.) The above property does not hold if the invested capacities are bounded above, and if putting the capacities at their maximum value is not enough to meet the demand in a worse case scenario. This is precisely the case in our example, where the penalty cost is not zero at the end of the range.

κ	CPU Time	Capacity cost	Expected rejection penalty cost	Total cost	# of oracle calls	# of cuts
0.5	10.90	0.0	296.60	296.60	1	243
1.0	10.90	0.0	593.19	593.19	1	243
1.5	55.26	44.80	841.07	885.87	5	627
2.0	67.52	145.69	1005.40	1151.09	6	898
2.5	77.87	277.20	1110.38	1387.58	7	969
3.0	76.40	424.26	1171.92	1596.18	7	1013
3.5	89.30	655.77	1120.28	1776.05	8	1158
4.0	86.79	933.13	987.17	1920.30	8	1286
4.5	85.90	1171.86	854.28	2026.14	8	1286
5	93.65	1219.43	899.62	2119.05	9	1168
10	97.29	1896.51	824.55	2720.06	10	1119
15	93.13	2419.26	596.31	3015.57	10	1077
20	100.15	2665.93	507.13	3173.06	10	1128
25	97.90	2850.38	425.83	3276.21	11	1047

Table 5: Effect of Penalty Rejection Cost

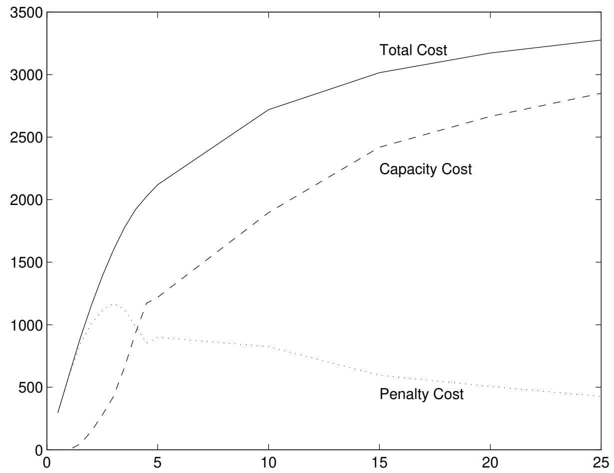


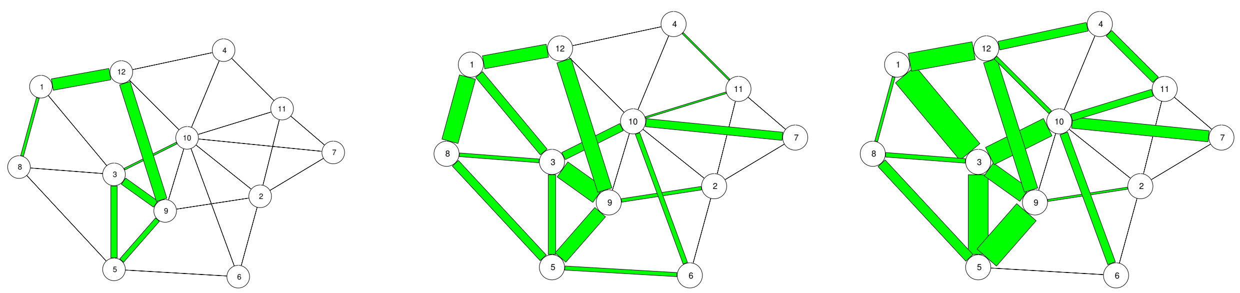
Figure 2: Tradeoff Between Capacity and Penalty Costs.

5.3.2 Stochastic programming versus deterministic approach

To illustrate the impact of a stochastic programming approach, we replaced the stochastic demand with a deterministic one. We considered two cases. We chose first the average demand $\bar{d}$ for each origin-destination pair. Then we used

$$d = \bar{d} + 0.5(d_{\max} - \bar{d}),$$

where $d_{\max}$ is the maximum demand over all demand configurations for the given-origin destination pair. We then solve the deterministic problem with a single scenario with demand configuration d. To evaluate the expected penalty cost associated with the recommended investment, we use the stochastic programming approach with fixed capacity investment. The results are displayed on Figure 3.



deterministic case 1: $d = \bar{d}$

Stochastic programming

deterministic case 2: $d = \bar{d} + 0.5(d_{\max} - \bar{d})$

Costs	Deterministic 1	Stochastic prg.	Deterministic 2
Investment	705	1896	2839
Expected penalty	3188	823	920
Total	3892	2720	3251

Figure 3: Stochastic vs. deterministic approach.

As expected, the figures reveal that the stochastic programming approach achieves the least total cost. It also appears that the investment cost in the stochastic programming solution takes an intermediary value between the two deterministic solutions. However, it is not possible to infer from the two deterministic solutions the structure of the stochastic one. To visualize this fact, the thickness of the links on Figure 3, page 20, is made proportional to the recommended capacity.

6 Conclusion

We have developed a model for network capacity planning problem under uncertainty. Uncertainty is treated in the framework of stochastic programming and the resulting non-differentiable problem is solved using the Analytic Center Cutting Plane Method.

Our preliminary computational experiments showed the ability of ACCPM to solve large instances of the problem on a PC with 400MHz and 384MB of RAM. Given the enormous size of the model, an interesting issue is using a parallelized version of ACCPM and make the model a closer match to the real world situations we are trying to describe. This is possible with less investment as we showed using a cluster of 5 PC's with 300MHz processors and 384MB of RAM.

Telecom markets have recently become so competitive that telecommunications companies advertise performance guaranties for their customers such as survivability (to ensure that the network has enough capacity to perform rerouting in case of link or node failure). Further work could be directed to the extension of the model presented here to meet survivability constraints.

References

[1] R.V. Ahuja, T.L. Magnanti, J.B. Orlin, *Network Flows : Theory, Algorithms and Applications*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[2] D. Alevras, M. Grötschel, R. Wessäly, "Capacity and survivability models for telecommunication networks", in Proceedings of *EURO/INFORMS* Meeting, Barcelona, pp. 187-199, 1997.

[3] A. Assad, Multicommodity network flows-a survey, *Networks* 8, pp. 37-91, 1978.

[4] O. Bahn, O. du Merle, J.-L. Goffin, J.-Ph. Vial, "A cutting plane method from analytic centers for stochastic programming", *Mathematical Programming*, 69, pp. 45-73, 1995.

[5] J.R. Birge, F.V. Louveaux, *Introduction to stochastic programming*, Springer, 1998.

[6] P. Chardaire, A. Lisser, "Simplex and interior point specialized algorithms for solving non-oriented multicommodity flow problems", *Operation Research*, submitted.

[7] G. Dahl, M. Stoer, "A cutting plane algorithm for multicommodity survivable network design problems", *INFORMS J. on Computing*, vol. 10, no 3, pp. 1-11, 1998.

[8] M.A.H. Dempster, E.A. Medova, R.T. Thompson, "A stochastic programming approach to network planning", *ITC* V. Ramaswami and P.E. Wirth Editors, pp. 329-339, 1997.

[9] J.M. Farvolden, W.B. Powell, I.J. Lustig, "A primal partitioning solution for the arc-chain formulation of a multicommodity network flow problem", *Operation Research*, vol 41, 4, pp. 669-693, 1993.

[10] E. Fragnière, J. Gondzio, J.-Ph. Vial, "Building and solving large-scale stochastic programs on an affordable distributed computing system", *Technical report* 98.11, Logilab/HEC, University of Geneva, June 1998.

[11] J.-L. Goffin, A. Haurie, J.-Ph. Vial, "Decomposition and nondifferentiable optimization with the projective algorithm", *Management Science*, 38(2), pp. 284-302, 1992.

[12] J.-L. Goffin, J. Gondzio, R. Sarkissian, J.-Ph. Vial, "Solving nonlinear multicommodity flow problems by the analytic center cutting plane method", *Mathematical Programming* 76 pp. 131-154, 1997.

[13] J. Gondzio, O. du Merle, R. R. Sarkissian, J.-Ph. Vial, "ACCPM - A library for convex optimization based on an analytic center cutting plane method", *EJOR*, 94, pp. 206-211, 1996.

[14] P. Kall, A. Ruszczyński, K. Frauendorfer, "Approximation techniques in stochastic programming", in Y. Ermoliev and R. J.-B. Wets eds. Numerical techniques for Stochastic Optimization, *Springer Series in Computational Mathematics 10*, pp. 33-64, 1988.

[15] J.F. Kennington, "A survey of linear cost multicommodity network flows", *Oper. Res.*, 2(26), pp. 209-236, 1978.

[16] A. Lisser, R. Sarkissian, J.-Ph. Vial, "Optimal joint synthesis of base and spare telecommunication networks", *Technical report*, Logilab/HEC, University of Geneva, November 1995.

[17] A. Lisser, R. Sarkissian, J.-Ph. Vial, "Mid-range planning of survivable telecommunications networks: Joint optimal synthesis of base and spare network capacities", *Technical report*, Logilab/HEC, University of Geneva, July 1998.

[18] A. Ouorou, P. Mahey, J.-Ph. Vial, "A survey of algorithms for convex multicommodity flow problems", *Management Science*, To appear.

[19] A. Ruszczyński, "Decomposition methods in stochastic programming", *Mathematical Programming* 79, pp. 333-353, 1997.

[20] S. Sen, R.D. Doverspike, S. Cosares, "Network planning with random demand", *Telecommunication Systems* 3, pp. 11-30, 1994.

[21] M. Snir, S.W. Otto, S. Huss-Lederman, D.W. Walker, J. Dongarra, *MPI: The complete reference*, The MIT Press.

[22] A. Tomasgard, J.A. Audestad, S. Dye, L. Stougie, M.H. Van Der Vlerk, S.W. Wallace, "Modelling aspects of distributed processing in telecommunication networks", *Annals of Operations Research*, To appear.

[23] A. Tomasgard, S. Dye, S.W. Wallace, J.A. Audestad, L. Stougie, M.H. Van Der Vlerk, "Stochastic optimization models for distributed communications networks", Technical report, Norwegian University of Science and Technology, 7034 Trondheim, 1997.