DEPARTMENT OF OPERATIONS RESEARCH
UNIVERSITY OF AARHUS

Publication no. 2000/2

On Using Stochastic Programming to Plan the Multiperiod Capacity expansion of One Connection in Telecommunications

Morten Riis
Kim Allan Andersen

ISSN 1398-8964

Department of Mathematical Sciences
Telephone: +45 8942 3188
E-mail: institut@imf.au.dk

Building 530, Ny Munkegade
DK-8000 Aarhus C, Denmark
URL: www.imf.au.dk

ON USING STOCHASTIC PROGRAMMING TO PLAN THE MULTIPERIOD CAPACITY EXPANSION OF ONE CONNECTION IN TELECOMMUNICATIONS

MORTEN RIIS AND KIM ALLAN ANDERSEN

ABSTRACT. We consider the problem of expanding the capacity of a single connection in a telecommunications network with uncertain demand. The problem can be formulated as a two-stage stochastic integer program. Using this formulation the problem can be solved in two phases as proposed by Laguna [11]. We show how the second phase which is a shortest path procedure can be made more efficient by using a simple preprocessing rule. Next we discuss the alternative of formulating the problem as a multi-stage stochastic integer program and give a recursive solution procedure for this formulation as well. Strengths and weaknesses of the two-stage versus the multi-stage formulations are discussed.

1. INTRODUCTION

Capacity expansion problems have been studied extensively in the literature. Examples taken from telecommunications include Balakrishnan et al. [1],[2], Bienstock and Gunluk [3] and Chang and Gavish [7],[8], all of which consider the capacity expansion problem for the entire network. In order to make such an approach practicable it is customary to make the simplifiyng assumption that the cost of expanding the capacity of a connection is (piecewise) linear. This will generally not be the case, though, since capacity is typically installed in lumps. Bienstock and Gunluk studies the polyhedral structure of the problem when capacity is installed in multiples of fixed batch sizes but they do not include a multiperiod formulation. Instead, we will follow the approach taken by Saniee [17] and Laguna [11]. We consider the capacity expansion problem of a single connection in a telecommunications network over a given time horizon. We assume that we have a number of different technologies available for installation in order to meet the demand in each time period. The capacity supplied by one component of technology i $(i = 1, \ldots, I)$ is denoted by c_i and the price of the component is p_i. The problem is to decide the number of components of each technology to install in each time period t $(t = 1, ..., T)$ in order to meet demand. Assuming that demand D_t in each period is known, this problem can be formulated as a multiconstraint knapsack problem:

$$\min \sum_{t=1}^{T} \sum_{i=1}^{I} \gamma^{t-1} p_i x_{it} \tag{1.1}$$

$$\text{s.t.} \sum_{r=1}^{t} \sum_{i=1}^{I} c_i x_{ir} \geq D_t \quad \forall t \tag{1.2}$$

$$x_{it} \in \mathbb{N} \cup \{0\} \qquad \forall i, t \tag{1.3}$$

Key words and phrases. Capacity expansion, Telecommunication, Stochastic Programming.

where γ is a discount factor and x_{it} is the number of components of technology i to be installed in period t. This is the formulation proposed by Saniee [17] who also gives an efficient solution procedure for the problem.

In real life, though, the assumption that demand in each period is known at the time the decision is made will not be satisfied. Instead, the demand in period t should be thought of as depending on the outcome of some random variable ξ_t. This means that the decision on capacity expansion cannot be based on actual demand. The only information that is available is the partial knowledge of demand conveyed through the distribution of the random vector $\xi = (\xi_1, \ldots, \xi_T)$. (This may be thought of as some kind of forecast of demand.) To be specific we let the random vector $\xi = (\xi_1, \ldots, \xi_T)$ be defined on some probability space $(\Xi, \mathcal{F}, P)$. We shall assume that ξ has finite support:

$$\Xi = \{\xi^1, \ldots, \xi^S\} \tag{1.4}$$

This allows us to model uncertainty in terms of *scenarios*, a scenario $s(\xi^j)$ being a sequence of outcomes of random demand:

$$s(\xi^j) = (D_1(\xi_1^j), \ldots, D_T(\xi_T^j)) \qquad j = 1, \ldots, S \tag{1.5}$$

with corresponding probabilities

$$\pi(s(\xi^j)) = P(\xi^j) \qquad j = 1, \ldots, S \tag{1.6}$$

In most of the following we will suppress the dependency on the random vector ξ. Thus we simply say that we have a finite set $\mathcal{S}$ of scenarios with probabilities $\pi(s)$, $s \in \mathcal{S}$. Also, we will refer to the t'th coordinate of s, i.e. the demand in period t under scenario s, simply as $D_t(s)$.

It is well known (see e.g. Birge and Loveaux [4]) that the expected value problem, obtained by replacing $D_t(\xi_t)$ by its expected value in the above formulation, may not produce very good solutions to the problem. In this article we will discuss different ways of incorporating uncertainty in the problem formulation using stochastic programming. Capacity expansion under uncertainty has been approached by several authors under the more general framework of robust optimization. Robust optimization was introduced by Mulvey et al. [15] who give an example taken from capacity expansion in power generation. Other examples include Malcolm and Zenios [13] who also consider capacity expansion in power generation and Laguna [11] who considers a problem similar to the one discussed in this article. Stochastic programming has also been used as a modelling tool in telecommunications. Dempster et al. [9] uses chance-constrained programming to solve the capacity expansion problem subject to certain grade of service constraints. Sen et al. [18] study a capacity expansion problem in which the expected number of unserved requests is minimized subject to limitations on the total capacity expansion. Both of these articles consider the capacity expansion problem for the entire network, but neither include multi-period formulations nor do they describe costs by the appropriate step function considered in this article.

In section 2 we formulate the problem as a two-stage stochastic integer program with simple recourse and describe the solution procedure proposed by Laguna [11]. We also present a new preprocessing rule which is later demonstrated to decrease computation time considerably, the effect being somewhat dependent on the input

data. In section 3 we discuss the alternative of using a multi-stage formulation instead of the two-stage formulation, and argue that this may be a more appropriate way to incorporate uncertainty in the model. Usually multi-stage stochastic integer programming problems are too hard to solve in practice. One of the only general purpose algorithms for such problems is a heuristic using tabu search presented in Løkketangen and Woodruff [12], but also the dual decomposition procedure developed by Carøe and Schultz [6] for two-stage problems may be adapted to handle multi-stage formulations. For the particular problem studied in this article it is possible to solve the multi-stage problem to optimality. We present a recursive formulation of the multi-stage problem which can be solved by direct calculation in section 4. All of the algorithms previously discussed have been implemented in C++ primarily to test the effect of the new preprocessing rule for the two-stage problem and to investigate whether or not the recursive multi-stage algorithm is practicable. In section 5 we report the results of some numerical experiments. Finally, in section 6 we give some conclusions and directions for future research.

2. A Two-Stage Formulation

Not knowing the demand at the time the decision on capacity expansion has to be made, means that we may have to accept that we cannot meet all demand in some periods. We let $z_t(s)$ denote the amount of demand exceeding capacity in period t under scenario s. Laguna [11] uses this approach to formulate the problem as a robust optimization problem, using a general penalty cost function ρ depending on $z_t(s)$ and the probability of occurrence $\pi(s)$. If the function ρ is linear and we assume that the cost $q(s)$ of not being able to meet one unit of demand under scenario s is known, this is essentially a two-stage stochastic integer programming problem with simple recourse. The two-stage capacity expansion problem ($TSCE$) is:

$$TSCE = \min \sum_{t=1}^{T} \sum_{i=1}^{I} \gamma^{t-1} p_i x_{it} + \sum_{s \in \mathcal{S}} \pi(s) \sum_{t=1}^{T} \gamma^{t-1} q(s) z_t(s) \tag{2.1}$$

$$\text{s.t.} \sum_{r=1}^{t} \sum_{i=1}^{I} c_i x_{ir} + z_t(s) \geq D_t(s) \qquad \forall t, s \tag{2.2}$$

$$x_{it}\,, z_t(s) \in \mathbb{N} \cup \{0\} \qquad \forall i, t, s \tag{2.3}$$

We assume that both demand $D_t(s)$ and capacity per unit of technology i, c_i are integers, so that the variables $z_t(s)$ too are integers. If the parameters are rational this can be achieved by scaling.

Note that we allow the cost of lost demand to differ among scenarios to provide maximum generality. This is not in compliance with the situation described by Laguna but does not affect the solution procedure given below. The question of actually finding/estimating the cost of lost demand $q(s)$ is a complicated matter, since it is seldom readily available. If the network provider has the possibility to rent capacity from competing network providers, $q(s)$ may be taken as the price of such rented capacity. Otherwise $q(s)$ may simply be thought of as lost revenue possibly with the addition of a penalty cost reflecting customer dissatisfaction.

It is important to note the distinction between periods and stages. Even though (2.1)-(2.3) is a multiperiod problem we only have two stages. The first stage includes the decisions x_{it}, which have to be taken without knowing the outcome of random

demand, and the second stage includes realizations of the variables $z_t(s)$ that occur after all uncertainty has been revealed - i.e. when we know which scenario has occurred. We will return to this issue in section 3.

Laguna solves (2.1)-(2.3) by a procedure which is similar in spirit to that proposed by Saniee [17]. The problem is solved in two phases consisting of a sequence of knapsack problems and a shortest path problem, respectively. Let D_{max} denote the maximum demand in any period under any scenario:

$$D_{max} = \max_{s \in \mathcal{S}} \max_{1 \leq t \leq T} \{D_t(s)\} \tag{2.4}$$

First, we find the cost of installing at least y ($y = 0, \ldots, D_{max}$) units of capacity by solving a sequence of knapsack problems:

$$M(y) = \min \sum_{i=1}^{I} p_i x_i \tag{2.5}$$

$$\text{s.t.} \sum_{i=1}^{I} c_i x_i \geq y \tag{2.6}$$

$$x_i \in \mathbb{N} \cup \{0\} \quad \forall i \tag{2.7}$$

Note that $M(y)$ gives the price of installing *exactly* y units of capacity if and only if $M(y) < M(y+1)$. (We assume that all p_i and c_i are strictly positive.)

The problem (2.1) - (2.3) can now be formulated in terms of the total capacity y_t to be installed in period t:

$$TSCE = \min \sum_{t=1}^{T} \gamma^{t-1} M(y_t) + \sum_{s \in \mathcal{S}} \pi(s) \sum_{t=1}^{T} \gamma^{t-1} q(s) z_t(s) \tag{2.8}$$

$$\text{s.t.} \sum_{r=1}^{t} y_r + z_t(s) \geq D_t(s) \qquad \forall t, s \tag{2.9}$$

$$y_t \in \{0, \ldots, D_{max}\} \qquad \forall t \tag{2.10}$$

$$z_t(s) \in \mathbb{N} \cup \{0\} \qquad \forall t, s \tag{2.11}$$

Laguna solves this problem as a shortest path problem in a directed graph constructed in the following way. The nodes of the graph are ordered in $T+2$ columns numbered from 0 to $T+1$. The columns $1, \ldots, T$ represent the time periods in our time horizon and each has $D_{max}+1$ nodes such that the node $v_{t,k}$ represents the situation that at least k units of capacity have been installed by period t. For $y = 0, \ldots, k$ all nodes $v_{t-1,k-y}$ are connected to node $v_{t,k}$ ($t = 2, \ldots, T$ and $k = 0, \ldots, D_{max}$) by edges with cost:

$$c(v_{t-1,k-y}\,, v_{t,k}) = \gamma^{t-1} M(y) + \gamma^{t-1} \sum_{s \in \mathcal{S}} \pi(s) q(s) |D_t(s) - k|^+ \tag{2.12}$$

The first and the last column each has one node, v_0 and v_{T+1}, respectively. v_0 is connected to all nodes $v_{1,k}$ ($k = 0, \ldots, D_{max}$) by edges with cost:

$$c(v_0\,, v_{1,k}) = M(k) + \sum_{s \in \mathcal{S}} \pi(s) q(s) |D_1(s) - k|^+ \tag{2.13}$$

Finally, all nodes $v_{T,k}$ ($k = 0, \ldots, D_{max}$) are connected to v_{T+1} by edges with zero cost.

$TSCE$ can now be found as the cost of the shortest (v_0-v_{T+1})-path in the graph by the following forward recursion:

$$F_1(k) = c(v_0\,, v_{1,k}) \quad k = 0, \ldots, D_{max} \tag{2.14}$$

$$F_t(k) = \min_{0 \le y \le k} \left[F_{t-1}(k-y) + c(v_{t-1,k-y}\,, v_{t,k})\right] \qquad k = 0, \ldots, D_{max} \quad t = 2, \ldots, T \tag{2.15}$$

$$TSCE = \min_{0 \le k \le D_{max}} F_T(k) \tag{2.16}$$

As the computational experiments of Laguna indicate, the time consuming part of this procedure is solving the shortest path problem. As each edge of the graph is considered exactly once in the above procedure, an efficient way to reduce the computation time is to reduce the number of edges in the graph. With this in mind we make the following observations:

Observation 1. *If $M(\bar{y}) = M(\bar{y}+1)$ we see that for $k = \bar{y}, \ldots, D_{max} - 1$:*

$$\begin{aligned} c(v_{t-1,k-\bar{y}}\,, v_{t,k}) &= \gamma^{t-1} M(\bar{y}) + \gamma^{t-1} \sum_{s \in \mathcal{S}} \pi(s) q(s) |D_t(s) - k|^+ \\ &\ge \gamma^{t-1} M(\bar{y}+1) + \gamma^{t-1} \sum_{s \in \mathcal{S}} \pi(s) q(s) |D_t(s) - k - 1)|^+ \\ &= c(v_{t-1,k-\bar{y}}\,, v_{t,k+1}) \end{aligned}$$

Observation 2. *Since we always have $M(y-1) \le M(y)$ we see that for $k < k'$:*

$$\begin{aligned} c(v_{t,k}\,, v_{t+1,k'}) &= \gamma^t M(k'-k) + \gamma^t \sum_{s \in \mathcal{S}} \pi(s) q(s) |D_{t+1}(s) - k'|^+ \\ &\ge \gamma^t M(k'-k-1) + \gamma^t \sum_{s \in \mathcal{S}} \pi(s) q(s) |D_{t+1}(s) - k'|^+ \\ &= c(v_{t,k+1}\,, v_{t+1,k'}) \end{aligned}$$

Thus if $M(\bar{y}) = M(\bar{y}+1)$, the subpath $(v_{t-1,k-\bar{y}}\,, v_{t,k+1}\,, v_{t+1,k'})$ will always be cheaper than the subpath $(v_{t-1,k-\bar{y}}\,, v_{t,k}\,, v_{t+1,k'})$ for $k = \bar{y}, \ldots, D_{max} - 1$ and $k' > k$. (The argument does not hold for $k = D_{max}$ because there is no node $v_{t,D_{max}+1}$). Hence, we can remove all edges $(v_{t-1,k-\bar{y}}, v_{t,k})$ $(k = \bar{y}, \ldots, D_{max} - 1)$ from the graph. That is, we remove all edges corresponding to the installment of $\bar{y}$ units of capacity in a period, since we are able to install $\bar{y}+1$ units of capacity at no additional cost. This changes the recursive shortest path algorithm for the intermediate nodes (2.15) as we now have:

$$F_t(k) = \min_{\substack{0 \le y \le k \\ y \in Y}} \left[F_{t-1}(k-y) + c(v_{t-1,k-y}\,, v_{t,k})\right] \qquad k = 0, \ldots, D_{max} - 1 \quad t = 2, \ldots, T \tag{2.17}$$

where

$$Y = \{y \in \{0, \ldots, D_{max} - 1\} \mid M(y) < M(y+1)\} \tag{2.18}$$

As before $F_1(k)$, $k = 0, \ldots, D_{max}$, $F_t(D_{max})$, $t = 2, \ldots, T$ and $TSCE$ are found using (2.14), (2.15) and (2.16) respectively. (We do not remove the edges $(v_0, v_{1,k})$ $k \notin Y$ in order to make sure that all values $F_1(k)$ are well-defined.)

Obviously, the effect of this simple preprocessing rule is highly dependent on the specific problem data, in particular p_i and c_i $(i = 1, \ldots, I)$. If the condition $M(y) = M(y+1)$ is satisfied for many $y \in \{0, \ldots, D_{max} - 1\}$, we would expect the preprocessing rule to decrease computation time considerably and this conjecture is confirmed by the numerical experiments presented in section 5.

The computational complexity of the algorithm is $O(\max\{ID_{max}, TD_{max}^2 S\})$, the first term corresponding to the solution of the series of knapsack problems and the second to the shortest path procedure. (Laguna [11] does not include the number of periods T in the complexity.) Two things are worth noting. The complexity will usually be dominated by the last term corresponding to the shortest path procedure, and as mentioned, the numerical experiments reported by Laguna and in section 5 seem to indicate that this is in fact the most time consuming part of the algorithm. As the shortest path procedure is the part of the algorithm, which is improved by the new preprocessing rule, we may hope for considerable time savings. Still the computational complexity of the algorithm is not changed by the preprocessing rule and thus the sensitivity of the algorithm with respect to T and S is still expected to be significant.

3. Two-Stage vs. Multi-Stage Formulations

Even though the formulation of the problem given in the previous section explicitly includes the uncertainty of demand, one might argue that it does not provide a very good description of the actual process of planning capacity expansion under uncertainty. The reason for this is the fact that we do not allow the decisions to depend on the outcomes of the stochastic variables as they are realized. The formulation in the previous section forces the decision maker to plan the capacity expansion for the entire time horizon before knowing any outcomes of the random demands, i.e. the capacity installed in each period will be the same no matter which scenario occurs. It does not seem reasonable, though, that the amount of capacity installed in period t should not depend on the actual demands realized up to time t - if a rapid increase in demand has been observed we would like to install more capacity and vice versa.

We denote the history of demands up to time t as realized in scenario s by s_t:

$$s_t = (D_1(s), \ldots, D_t(s)) \qquad t = 1, \ldots, T \quad s \in \mathcal{S} \tag{3.1}$$

We shall refer to this as a *subscenario*. Two scenarios s and s' are said to be *indistinguishable* at time t if $s_t = s'_t$. We denote the set of distinguishable subscenarios at time t by $\mathcal{S}_t$.

There is another way to think of the subscenarios, which we will briefly consider. At each time t we can partition the scenarios into bundles A_t of indistinguishable scenarios. We denote the set of scenario bundles at time t by $\mathcal{A}_t$. The partitions form a tree-like structure in the sense that each bundle $A_t \in \mathcal{A}_t$ is a union of bundles $A_{t+1} \in \mathcal{A}_{t+1}$ called the descendants of A_t. This simply means that given a history of demands up to time t (which is the same for all scenarios in a bundle A_t), we have a set of possible outcomes of the random demand in the following period corresponding to a branching of the scenarios in A_t. Clearly, there is a one to one correspondence between the subscenarios s_t and the scenario bundles A_t. Thus in the following we shall refer by s_t interchangeably to the subscenario as defined by (3.1) and to the

scenario bundle which it represents, the exact meaning hopefully being clear from the context.

Since the partitions form a tree-like structure we can think of the scenarios in terms of a *scenario tree*. The nodes of the tree are ordered in T columns corresponding to the time periods. The leaves of the tree (nodes in column T) represent the S different scenarios and each node in column t corresponds to a bundle of scenarios that are indistinguishable at time t. In other words each node in the tree corresponds to a unique subscenario s_t.

As before we denote the probability of a scenario s by $\pi(s)$. We will also refer to the probability of a subscenario s_t:

$$\pi(s_t) = \sum_{s':s'_t=s_t} \pi(s') \tag{3.2}$$

The set of descendants of s_t will be denoted by $\mathcal{D}(s_t)$ and finally, we denote by $\pi(s_{t+1}|s_t)$ the conditional probability of subscenario s_{t+1} given subscenario s_t for $s_{t+1} \in \mathcal{D}(s_t)$.

4. A Multi-Stage Formulation Solved Recursively

In this section we consider the situation where we have a time delay in the installment of capacity, so that the capacity that we plan to install now will not be available for customers before the beginning of the next period. Thus we are assuming that we have to decide on the capacity to install in the next period without knowing the actual demand in that period (but we are able to take the realized demands in previous periods into account). We let x_{i1} denote the number of components of technology i to be installed now. Likewise $x_{it}(s_{t-1})$ $(t = 2, \ldots, T)$ denotes the number of components of technology i to be installed in period t knowing the demand in period $t-1$ as realized in subscenario s_{t-1}. Since we have to plan the capacity expansion one period ahead, we may still have to accept that we cannot meet all demand in all periods. As before we let $z_t(s_t)$ denote the amount of demand exceeding capacity in period t under scenario s. The multi-stage capacity expansion problem may now be formulated as:

$$MSCE = \min \sum_{t=1}^{T} \gamma^{t-1} \left(\sum_{s_{t-1} \in \mathcal{S}_{t-1}} \pi(s_{t-1}) \sum_{i=1}^{I} p_i x_{it}(s_{t-1}) + \sum_{s_t \in \mathcal{S}_t} \pi(s_t) q(s_t) z_t(s_t) \right) \tag{4.1}$$

$$\text{s.t.} \sum_{r=1}^{t} \sum_{i=1}^{I} c_i x_{ir}(s_{r-1}) + z_t(s_t) \geq D_t(s) \qquad \forall t, s \tag{4.2}$$

$$x_{it}(s_{t-1}) \, , z_t(s_t) \in \mathbb{N} \cup \{0\} \qquad \forall i, t, s \tag{4.3}$$

(For notational convenience we refer to x_{i1} as $x_{i1}(s_0)$ in (4.1)-(4.3) and let $\pi(s_0) = 1$ and $\mathcal{S}_0 = \{s_0\}$.)

Once again we will solve the problem by initially solving the series of knapsack problems (2.5)-(2.7) in order to find the cost of installing y units of capacity ($y = 0, \ldots, D_{max}$). Now, we introduce the variable y_1 denoting the amount of capacity to be installed now and the variables $y_{t+1}(s_t)$ $(t = 1, \ldots, T-1)$ denoting the amount of capacity to be installed in period $t+1$ knowing the demand in period t as realised in subscenario s_t. Problem (4.1)-(4.3) may then be given an equivalent formulation similar to (2.8)-(2.11). Instead, we will give a recursive formulation of the problem

which allows us to solve it directly. We let $Q_t(k, s_t)$ denote the expected cost at time t of having installed a total of k units of capacity under subscenario s_t. For the final period we have:

$$Q_T(k, s_T) = \min q(s_T) z_T(s_T) \tag{4.4}$$
$$\text{s.t. } z_T(s_T) \geq D_T(s) - k \tag{4.5}$$
$$z_T(s_T) \in \mathbb{N} \cup \{0\} \tag{4.6}$$

and for $t = 1, \ldots, T-1$:

$$Q_t(k, s_t) = \min q(s_t) z_t(s_t) + \gamma M(y_{t+1}(s_t)) + \gamma \sum_{s_{t+1} \in \mathcal{D}(s_t)} \pi(s_{t+1}|s_t) Q_{t+1}(k + y_{t+1}(s_t), s_{t+1}) \tag{4.7}$$
$$\text{s.t. } z_t(s_t) \geq D_t(s) - k \tag{4.8}$$
$$y_{t+1}(s_t) \in Y_k \tag{4.9}$$
$$z_t(s_t) \in \mathbb{N} \cup \{0\} \tag{4.10}$$

where:

$$Y_k = Y \cup \{D_{max} - k\} \qquad k = 0, \ldots, D_{max} \tag{4.11}$$

and the set Y was defined in section 2. (By the same argument that we used in section 2 we only consider installing y units of capacity if $M(y) < M(y+1)$, the only exception being the situation where we choose to install enough capacity to reach a total capacity of D_{max}.) The multi-stage capacity expansion problem can now be stated as:

$$MSCE = \min M(y_1) + \sum_{s_1 \in \mathcal{S}_1} \pi(s_1) Q_1(y_1, s_1) \tag{4.12}$$
$$\text{s.t. } y_1 \in Y_0 \tag{4.13}$$

This formulation allows us to solve the problem directly using a backward recursion. For the final period we have:

$$Q_T(k, s_T) = q(s_T)|D_T(s) - k|^+ \qquad k = 0, \ldots, D_{max} \quad s_T \in \mathcal{S}_T \tag{4.14}$$

For all intermediate periods ($t = 1, \ldots, T-1$) we have:

$$Q_t(k, s_t) = q(s_t)\left|D_t(s) - k\right|^+ + \min_{\substack{0 \leq y \leq D_{max} - k \\ y \in Y_k}} \left[\gamma M(y) + \gamma \sum_{s_{t+1} \in \mathcal{D}(s_t)} \pi(s_{t+1}|s_t) Q_{t+1}(k + y, s_{t+1})\right] \qquad k = 0, \ldots, D_{max} \quad s_t \in \mathcal{S}_t \tag{4.15}$$

And finally the solution for the multi-stage problem is:

$$MSCE = \min_{y \in Y_0} \left[M(y) + \sum_{s_1 \in \mathcal{S}_1} \pi(s_1) Q_1(y, s_1)\right] \tag{4.16}$$

We will briefly consider the computational complexity of the recursive solution procedure for the multi-stage formulation given above. The series of knapsack problems (2.5)-(2.7) used to determine the values of $M(y)$ $(y = 0, \ldots, D_{max})$ may be solved in $O(ID_{max})$ time. (See e.g. Pisinger [16] or Martello and Toth [14].) For each node in the scenario tree we have to evaluate either (4.14),(4.15) or (4.16), the dominating time complexity obviously being that of (4.15). We now define:

$$\bar{S} = \max_{1 \le t < T} \max_{s_t \in \mathcal{S}_t} \{|\mathcal{D}(s_t)|\} \tag{4.17}$$

i.e. $\bar{S}$ denotes the maximum number of descendants of any subscenario. The sum in (4.15) is calculated in $O(\bar{S})$ time and this has to be done for $k = 0, \ldots, D_{max}$ and $y \in Y_k$. Since $|Y_k|$ is $O(D_{max})$, we see that the time complexity of evaluating (4.15) for fixed values of t and s_t is $O(D_{max}^2 \bar{S})$. As the number of nodes in the scenario tree is $O(TS)$, the overall time complexity for the recursive solution procedure is $O(\max\{ID_{max}, TD_{max}^2 S\bar{S}\})$, which only differs from that of the two-stage formulation by a factor of $\bar{S}$.

5. Computational Experiments

Laguna [11] implemented the algorithm described in section 2 and carried out a number of experiments to investigate the sensitivity of the procedure with respect to the parameters I, T, and S. His experiments indicate that the CPU-time used by the procedure is highly sensitive to an increase in the number of periods T which can be ascribed to the fact that an increase in T implies an increase in the maximum demand D_{max}. His experiments also exhibits an increase in CPU-time for increasing numbers of scenarios S, whereas the algorithm seems to be quite insensitive to the number of different technologies I. This indicates that the CPU-time needed to solve the series of knapsack problems (2.5)-(2.7) is negligible compared to the CPU-time needed for the shortest path procedure (2.14)-(2.16).

We implemented the algorithm described in section 2 in C++ with and without the new preprocessing rule primarily to test whether the preprocessing rule in fact speeds up the algorithm and secondly to investigate whether the sensitivity of the procedure with respect to T, I and S is changed when the preprocessing rule is applied. The experiments were carried out on a HP-UX 9000 workstation and were performed on instances of the problem generated randomly according to the precepts used by Laguna:

$$D_t(s) = \sum_{r=1}^{t} d_r(s) \qquad \forall t, s \tag{5.1}$$

$$d_t(s) \sim N(\mu, \sigma^2) \qquad \forall t, s \tag{5.2}$$

$$c_i \sim U(1, \mu) \qquad \forall i \tag{5.3}$$

$$p_i = \mu + c_i + \tilde{p}_i \qquad \forall i \tag{5.4}$$

$$\tilde{p}_i \sim U(-\sigma, \sigma) \qquad \forall i \tag{5.5}$$

$$\pi(s) = \frac{1}{S} \qquad \forall s \tag{5.6}$$

The input parameters $D_t(s)$, c_i and p_i were all subsequently rounded to integer values. Onless otherwise specified, the instances were generated using $\mu = 100$ and $\sigma = 10$ in compliance with Laguna [11]. Likewise the discount factor γ was set to 0.86 and the cost of lost demand was fixed to the value 5 used by Laguna.

The results of the first series of experiments are summarized in tables 1 to 3. $TSCE1$ refers to the CPU-time (in seconds) used by the procedure without the preprocessing rule and $TSCE2$ refers to the CPU-time (in seconds) when the rule is applied. All reported CPU-times are averages over 10 independently generated instances. We also report the maximum demand D_{max} and the number of different solutions from the series of knapsack problems denoted by $|Y|$. (The set Y was defined by (2.18).) These numbers are reported because the reason for us to believe that the preprocessing rule is efficient, is that it brings the number of installment levels considered in the minimization down from D_{max} to $|Y|$. Finally, we report the reduction in CPU-time and in the number of installment levels considered in the minimization when the new preprocessing rule is applied.

TABLE 1 $I = 10$ and $S = 100$

T	4	6	8	10	12
$TSCE1$	5.74	19.88	50.01	94.20	161.94
$TSCE2$	0.89	3.06	6.53	17.08	26.01
Reduction	84.4%	84.6%	87.0%	81.9%	83.9%
D_{max}	446.6	663.7	867.8	1081.4	1287.3
$\|Y\|$	72.2	105.4	124.2	215.5	218.2
Reduction	83.8%	84.1%	85.7%	80.1%	83.0%

TABLE 2 $I = 10$ and $T = 10$

S	10	50	100	500	1000
$TSCE1$	16.23	49.62	94.20	444.71	892.20
$TSCE2$	2.84	9.55	17.08	66.28	142.11
Reduction	82.5%	80.8%	81.9%	85.1%	84.1%
D_{max}	1034.8	1070.1	1081.4	1091.2	1106.0
$\|Y\|$	174,6	210.6	215.5	175.2	187.6
Reduction	83.1%	80.3%	80.1%	83.9%	83.0%

TABLE 3 $T = 10$ and $S = 100$

I	4	10	20	50	100
$TSCE1$	94.22	94.20	96.08	95.08	95.03
$TSCE2$	9.73	17.08	22.57	34.48	46.98
Reduction	89.7%	81.9%	76.5%	63.7%	50.6%
D_{max}	1082.8	1081.4	1084.7	1077.6	1078.3
$\|Y\|$	113.0	215.5	270.2	411.2	552.0
Reduction	89.6%	80.1%	75.1%	61.8%	48.8%

Let us first consider tables 1 and 2 where the number of technologies are fixed at 10. First of all we note that the algorithm provides consistently shorter CPU-times when the new preprocessing rule is applied with time savings between 80% and 85%. Secondly we note that the sensitivity of the algorithm with respect to the parameters T and S is similar to the one reported by Laguna, also when the preprocessing rule is applied. Finally, we note that the time savings closely correspond to the reduction in

the number of installment levels considered in the minimization, a fact that becomes even more obvious when we turn to table 3. In table 3 the number of time periods and the number of scenarios have been fixed at 10 and 100, respectively. As reported by Laguna the procedure without the preprocessing rule seems to be quite insensitive to the number of technologies, but when the rule is applied we note an increase in the CPU-time for increasing numbers of technologies. This can be ascribed to the fact that an increasing number of technologies implies an increasing number of different solutions from the series of knapsack problems as indicated by $|Y|$. Again we note the close relationship between the time saving and the reduction in the number of installment levels considered.

Finally we conducted a series of experiments to investigate the behavior of the procedure and the effect of the preprocessing rule when the size of demand is changed. Ten random instances were generated for 5 different values of μ and the same numbers as before were recorded. In all of the experiments we used $\sigma = \mu/10$ corresponding to simple scaling of the demand increments (5.2). Thus we have not investigated the effect of the variance of the demand increments on the procedure. The results of the experiments are reported in table 4.

TABLE 4 $T = 10$, $S = 100$ and $I = 10$

μ	10	20	50	100	200
$TSCE1$	1.00	3.83	23.54	94.20	375.28
$TSCE2$	0.72	2.06	7.08	17.08	41.26
Reduction	28.1%	46.1%	69.9%	81.9%	89.0%
D_{max}	108.3	215.3	540.5	1081.4	2159.2
$\|Y\|$	75.8	120.4	170.0	215.5	258.2
Reduction	30.0%	44.1%	68.5%	80.1%	88.0%

As expected the CPU-time is drastically reduced when μ is smaller as a result of the reduction in D_{max}. We also see that the effect of the preprocessing rule decreases for smaller values of μ, once again a fact that is explained by the smaller reductions in the number of installment levels considered.

The recursive solution procedure for the multi-stage problem was also implemented in C++ and a series of experiments was carried out to test if the method is practicable. In all experiments we used a fixed number $\bar{S}$ of descendants for all nodes in the scenario tree. The random instances were again generated using the precepts (5.1)-(5.5). Some care has to be taken, though, when generating demand by (5.1)-(5.2), as we now have to collect the scenarios in scenario bundles in each period. This was done by generating $\bar{S}$ demand values for period 1, for each of these values $\bar{S}$ independent demand increments (5.2) were generated and so forth. In all experiments we used:

$$\pi(s_{t+1}|s_t) = \frac{1}{\bar{S}} \qquad \forall s_{t+1} \in \mathcal{D}(s_t) \tag{5.7}$$

resulting in the same uniform scenario probabilities (5.6) as used in the previous experiments.

The results of the experiments are reported in tables 5 to 7. Once again we report the average CPU-time (in seconds) calculated from 10 independently generated instances. The CPU-time used by the recursive solution procedure for the multi-stage problem is refered to by $MSCE$. We also report the CPU-time used by the enhanced procedure for the two-stage problem when solving problems of comparable

size (with respect to T, I and S). This is done partly to further investigate the capabilities of the two-stage algorithm and partly to compare solution times for the two-stage and the multi-stage problems.

TABLE 5 $\bar{S} = 2$ and $I = 10$

T	4	6	8	10	12
S	16	64	256	1024	4096
$MSCE$	0.26	2.34	14.00	101.29	620.82
$TSCE2$	0.21	2.19	20.91	148.79	987.74

TABLE 6 $\bar{S} = 2$ and $T = 10$

I	4	10	20	50	100
S	1024	1024	1024	1024	1024
$MSCE$	39.35	101.29	174.74	262.26	329.01
$TSCE2$	70.09	148.79	234.44	329.32	436.05

TABLE 7 $T = 4$ and $I = 10$

$\bar{S}$	2	4	6	8	10
S	16	256	1296	4096	10000
$MSCE$	0.26	2.21	10.07	31.51	60.43
$TSCE2$	0.21	1.95	10.99	31.38	78.11

When comparing the two different formulations, one must be aware of the fact that they do not solve the exact same problem. As mentioned earlier, we believe that the multi-stage formulation provides a more accurate description of the actual process of planning capacity expansion under uncertainty. In section 4 we saw that the improved description provided by the multi-stage formulation comes at the cost of an increase in the computational complexity of the solution procedure. Still one should keep in mind, though, that the scenarios are handled much more efficiently by the solution procedure for the multi-stage problem than by that for the two-stage problem. When solving the multi-stage problem the scenarios are ordered in bundles of indistinguishable scenarios in each period which means that the number of distinct scenarios that are considered in early periods is very limited. When solving the two-stage problem on the other hand, the scenarios are considered independently, hence the actual number of distinct demand values in a given period is of no importance.

When studying tables 5 to 7 it seems that the efficiency obtained by collecting the scenarios into bundles when solving the multi-stage problem is more than enough to outweigh the increase in computational complexity. Thus we see that solution times are actually shorter for the multi-stage problem than for the two-stage problem, at least for the larger instances. In accordance with our previous discussion we see that the time savings become more distinct when the number of periods is increased. That is, when the number of periods is increased the difference between the number of scenarios S and the number of distinct subscenarios in early periods also increases and the effect of collecting the scenarios into bundles is intensified.

Apart from that, we note that the sensitivity of the solution procedure for the multi-stage problem with respect to the parameters I, T and $\bar{S}$ is generally as expected. As for the two-stage problem the algorithm is highly sensitive to the number of periods and somewhat sensitive to the number of scenarios. The number of different technologies once again has a modest effect on CPU-time, which is ascribed

to the increased number of different solutions from the series of knapsack problems when a larger number of technologies is available.

6. Conclusions

We have studied two different approaches for the multiperiod capacity expansion problem with uncertain demand. In section 2 we considered a two-stage formulation of the problem. An algorithm proposed by Laguna [11] was discussed and we presented a new preprocessing rule which was shown to be very effective in section 5. As argued in section 3, though, we think that a multi-stage formulation is the most appropriate way to include uncertainty in the model, as it allows the decision-maker to adapt his decisions to the development in demand. Usually multi-stage stochastic integer programs are too hard to solve in practice. In this case, however, the structure and simplicity of the problem allows us to solve it for the multi-stage formulation by the recursive algorithm given in section 4. In addition we saw that the time complexity of the recursive solution procedure for the multi-stage formulation differed only by a factor of $\bar{S}$ from that of the two-stage algorithm. In fact the numerical experiments reported in section 5 seem to indicate that the multi-stage algorithm is even faster than the two-stage algorithm due to the more efficient handling of the scenarios. Hence for this problem it may in fact be possible to use the more appropriate multi-stage formulation in practice. The finding that the multi-stage algorithm seemed to perform better than the two-stage algorithm was a very interesting one. Thus it may be the case that modelling multi-period problems as two-stage stochastic programs (as is often the practice, see e.g. Carøe et al. [5] or Dentcheva and Römisch [10]) is in fact in some cases inefficient as well as inappropriate.

As mentioned, the problem that we have discussed in this article is a very simple one. In practice we would like to solve the capacity expansion problem for an entire telecommunications network, taking into account the actual routing of calls between OD-pairs instead of treating the demand for bandwith on a single connection as exogenous. Still the problem of expanding the capacity of a single connection with uncertain demand may indeed have some relevance, as it may appear as a subproblem in such a capacity expansion problem for the entire network. Further research has to be done to explore this possibility.

References

[1] A. Balakrishnan, T. L. Magnanti, A. Shulman, and R. T. Wong. Models for planning capacity expansion in local access telecommunication networks. *Annals of Operations Research*, 33:239–284, 1991.

[2] A. Balakrishnan, T. L. Magnanti, and R. T. Wong. A decomposition algorithm for local access telecommunications network expansion planning. *Operations Research*, 43:58–76, 1995.

[3] D. Bienstock and O. Gunluk. Capacitated network design. Polyhedral structure and computation. *INFORMS Journal on Computing*, 8(3):243–259, 1996.

[4] J. R. Birge and F. V. Louveaux. *Introduction to Stochastic Programming*. Springer-Verlag, Berlin Heidelberg New York, 1997.

[5] C. C. Carøe, A Ruszczynski, and R. Schultz. Unit commitment under uncertainty via two-stage stochastic programming. In C. C. Carøe and D. Pisinger, editors, *Proceedings NOAS'97*, DIKU Rapport 97/23, pages 21–30. Department of Computer Science, University of Copenhagen, 1997.

[6] C. C. Carøe and R. Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24:37–45, 1999.

[7] S.-G. Chang and B. Gavish. Telecommunications network topological design and capacity expansion: Formulations and algorithms. *Telecommunication Systems*, 1:99–131, 1993.
[8] S.-G. Chang and B. Gavish. Lower bounding procedures for multiperiod telecommunications network expansion problems. *Operations Research*, 43:43–57, 1995.
[9] M. A. H. Dempster, E. A. Medova, and R. T. Thompson. A stochastic programming approach to network planning. *Teletraffic Contributions for the Information Age. Proceedings of the 15th International Teletraffic Congress - ITC 15*, 1:329–339, 1997.
[10] D. Dentcheva and W. Römisch. Optimal power generation under uncertainty via stochastic programming. In K. Marti and P. Kall, editors, *Stochastic programming: Numerical techniques and engineering applications*, number 458 in Lecture Notes in Economics and Mathematical Systems. Springer, Berlin, 1998.
[11] M. Laguna. Applying robust optimization to capacity expansion of one location in telecommunications with demand uncertainty. *Management Science*, 44(11):S101–S110, 1998.
[12] A. Løkketangen and D. L. Woodruff. Progressive hedging and tabu search applied to mixed integer (0,1) multistage stochastic programming. *Journal of Heuristics*, 2:111–128, 1996.
[13] S. A. Malcolm and S. A. Zenios. Robust optimization for power systems capacity expansion under uncertainty. *Journal of the Operations Research Society*, 45(9):1040–1049, 1994.
[14] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley Interscience, New York, 1990.
[15] J. M. Mulvey, R. J. Vanderbei, and S. J. Zenios. Robust optimization of large-scale systems. *Operations Research*, 43:264–281, 1995.
[16] D. Pisinger. *Algorithms for knapsack problems*. PhD thesis, University of Copenhagen, 1995.
[17] I. Saniee. An efficient algorithm for the multiperiod capacity expansion of one location in telecommunications. *Operations Research*, 43:187–190, 1995.
[18] S. Sen, R. D. Doverspike, and S. Cosares. Network planning with random demand. *Telecommunication Systems*, 3:11–30, 1994.

E-mail address: riis@imf.au.dk

Department of Operations Research, Ny Munkegade, 8000 Aarhus C, Denmark