The value of multi-stage stochastic programming in capacity planning under uncertainty

Kai Huang† and Shabbir Ahmed‡∗

† School of Management, Binghamton University, State University of New York
‡ School of Industrial & Systems Engineering, Georgia Institute of Technology

April 26, 2005. Revised April 26, 2006; September 25, 2007; March 5, 2008.

Abstract

This paper addresses a general class of capacity planning problems under uncertainty, which arises, for example, in semiconductor tool purchase planning. Using a scenario tree to model the evolution of the uncertainties, we develop a multi-stage stochastic integer programming formulation for the problem. In contrast to earlier two-stage approaches, the multi-stage model allows for revision of the capacity expansion plan as more information regarding the uncertainties is revealed. We provide analytical bounds for the value of multi-stage stochastic programming (VMS) afforded over the two-stage approach. By exploiting a special substructure inherent in the problem, we develop an efficient approximation scheme for the difficult multi-stage stochastic integer program and prove that the proposed scheme is asymptotically optimal. Computational experiments with realistic-scale problem instances suggest that the VMS for this class of problems is quite high. Moreover the quality and performance of the approximation scheme is very satisfactory. Fortunately, this is more so for instances for which the VMS is high.

Key words: multi-stage stochastic programming, capacity planning, semiconductor tool planning, analysis of algorithms.


Area of review: Optimization.

∗Corresponding author. E-mail: sahmed@isye.gatech.edu
1 Introduction

Capacity planning, i.e., deciding the optimal timing and level of capacity acquisition and allocation, plays a crucial role in strategic level planning in a wide array of applications. This activity involves substantial commitment of capital resources and is significantly affected by uncertainties in the long-range forecasts, thereby making the associated decision problems very complex. For example, the initial investment in building a semiconductor wafer fab is close to two billion dollars, and every year the procurement of new tools to accommodate the high volatility in demand, product mix and technology could cost several million dollars (cf. Barahona et al. (2005), Hood et al. (2003), Swaminathan (2000)).

Owing to the inherent complexities, quantitative models for economic capacity planning under uncertainty have been the subject of intense research since the early 1960s (cf. Luss (1982)). Early approaches for solving stochastic capacity expansion problems are restricted to a single resource and based on simplifying assumptions on the underlying stochastic processes to render analytical tractability (cf. Bean et al. (1992), David et al. (1987), Freidenfelds (1980), Manne (1961)). More general stochastic programming based approaches that use scenarios to model the uncertain parameters within large-scale mathematical programs for multi-resource multi-item capacity planning have since been proposed (cf. Berman et al. (1994), Eppen et al. (1989), Fine and Freund (1990)). Most of these stochastic programming approaches are based on the two-stage paradigm, wherein the capacity acquisition schedule for the entire (multi-period) planning horizon is decided “here-and-now,” and capacity allocations are made on a period-by-period basis based on realized uncertainties and acquired capacities. In the context of semiconductor tool planning, such two-stage models are investigated in Barahona et al. (2005), Hood et al. (2003), Karabuk and Wu (2003), Swaminathan (2000) and Swaminathan (2002). Multi-stage stochastic programming models extend the two-stage paradigm by allowing revised decisions in each time stage based upon the uncertainty realized so far (cf. Birge (1985)). A multi-stage stochastic capacity planning model involving continuous capacity allocation decisions and fixed charge expansion costs is considered in Ahmed and Sahinidis (2003). The authors develop a LP-relaxation based heuristic for this problem and prove, via a probabilistic analysis, that the heuristic is asymptotically
optimal in the number of planning stages.

Motivated by applications in semiconductor tool planning, we address a general multi-stage stochastic capacity planning model involving discrete capacity acquisition decisions. Our model generalizes earlier two-stage approaches considered in Barahona et al. (2005), Hood et al. (2003), Swaminathan (2002, 2000), by allowing for revision of the capacity expansion plan as more information regarding the uncertainties is revealed. We provide analytical bounds for the value of multi-stage stochastic programming afforded over two-stage approaches. By exploiting a special substructure inherent in the problem, we develop an efficient approximation scheme for the multi-stage problem and prove that the proposed scheme is asymptotically optimal. Our asymptotic analysis is significantly different from that of Ahmed and Sahinidis (2003), since we consider discrete capacity acquisition levels and do not make any assumptions regarding the distributions of the underlying stochastic parameters. Finally, we present numerical results for a realistic-scale semiconductor tool planning problem to demonstrate the advantage of the proposed model and solution method.

2 Model development

In this section we present a mathematical formulation for the stochastic capacity planning problem under consideration. We first describe a specific deterministic model related to semiconductor tool planning, and then discuss deterministic and stochastic generalizations of this model.

2.1 A deterministic model for semiconductor tool planning

Consider a wafer fab consisting of a set of $\mathcal{M}$ tool types, that can process a set of $\mathcal{N}$ products (wafer types). Each product goes through a set $\mathcal{K}$ of processing steps, each of which can be performed on one or more tool types. Given product demands for each period of a planning horizon of $T$ periods, the tool planning problem is to decide how many of each tool type to acquire and how to allocate production to the tools in each period, so as to minimize the sum of tool acquisition costs, production costs, and costs for unmet demand.

We consider a deterministic version of a two-stage stochastic programming formulation
for the above problem presented in Swaminathan (2002). The following notation is used. Let $d_{jt}$ denote the demand (measured in units of “wafer starts”) of wafer type $j \in \mathcal{N}$ in period $t \in \{1, \ldots, T\}$ and $h_{ijk}$ denote the time (in hours per wafer start) required by processing step $k \in \mathcal{K}$ on wafer type $j \in \mathcal{N}$ on tool type $i \in \mathcal{M}$. We set $h_{ijk} = 0$ if step $k$ is not needed for wafer type $j$, and $h_{ijk} = \infty$ if step $k$ is required for wafer type $j$ but cannot be performed on tool type $i$. The per-period capacity (in hours) of tool type $i$ is denoted by $m_i$. In addition to the above data, we also have the costs parameters $a_{it}$, $b_{jt}$, $c_{ijk}$, and $g_{jt}$ denoting the (discounted) cost of tool type $i$ in period $t$, the cost per wafer start for releasing wafer type $j$ into production in period $t$, the cost per wafer start of processing step $k$ of product $j$ on tool type $i$, and the penalty cost of unit shortage of wafer type $j$ in period $t$, respectively.

The decision variables are as follows: $x_{it}$ denotes the number of tools of type $i$ acquired in period $t$; $u_{jt}$ denotes the number of wafer starts of type $j$ released into production in period $t$; $v_{ijkt}$ denotes the number of wafer starts of type $j$ whose $k$-th processing step is carried out in machine $i$ in period $t$; and $w_{jt}$ denotes the shortage of wafer type $j$ in period $t$. With this notation (summarized in Table 1), a mixed-integer linear programming model for the problem is:

$$\min \quad \sum_{t=1}^{T} \left( \sum_{i \in \mathcal{M}} a_{it} x_{it} + \sum_{j \in \mathcal{N}} b_{jt} u_{jt} + \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{N}} \sum_{k \in \mathcal{K}} c_{ijk} v_{ijkt} + \sum_{j \in \mathcal{N}} g_{jt} w_{jt} \right)$$  \hspace{1cm} (1)

s.t. \quad \sum_{j \in \mathcal{N}} \sum_{k \in \mathcal{K}} h_{ijk} v_{ijkt} \leq m_i \left( \sum_{\tau=1}^{t} x_{i\tau} \right) \quad \forall \ i \in \mathcal{M}, \ t = 1, \ldots, T \hspace{1cm} (2)

$$\sum_{i \in \mathcal{M}} v_{ijkt} \geq u_{jt} \quad \forall \ j \in \mathcal{N}, \ k \in \mathcal{K}, \ t = 1, \ldots, T \hspace{1cm} (3)$$

$$u_{jt} + w_{jt} \geq d_{jt} \quad \forall \ j \in \mathcal{N}, \ t = 1, \ldots, T \hspace{1cm} (4)$$

$$u_{jt}, \ v_{ijkt}, \ w_{jt} \in \mathbb{R}_+ \quad \forall \ i \in \mathcal{M}, \ j \in \mathcal{N}, \ k \in \mathcal{K}, \ t = 1, \ldots, T \hspace{1cm} (5)$$

$$x_{it} \in \mathbb{Z}_+ \quad \forall \ i \in \mathcal{M}, \ t = 1, \ldots, T. \hspace{1cm} (6)$$

The objective function (1) is the sum of tool acquisition costs, production costs and shortage penalties over the planning horizon. The first set of constraints (2) ensures that the total processing requirement (in hours) allocated to tool $i$ cannot exceed the installed
Indices and Sets:

$i$  Index for tool type  

$j$  Index for wafer type  

$k$  Index for processing step  

$\tau, t$  Indices for time periods  

$M$  Set of tool types $i \in M$  

$N$  Set of wafer types $j \in N$  

$K$  Set of processing steps $k \in N$  

$T$  Number of time periods $\tau, t \in \{1, \ldots, T\}$  

Parameters:

$a_{it}$  Cost of acquiring tool type $i$ in period $t$  

$b_{jt}$  Cost of releasing wafer type $j$ into production in period $t$  

$c_{ijk}$  Cost of processing step $k$ of wafer type $j$ on tool type $i$  

$d_{jt}$  Demand of wafer type $j$ in period $t$  

$h_{ijk}$  Time required by processing step $k$ on tool type $i$ for wafer type $j$  

$m_i$  Capacity of tool type $i$  

$g_{jt}$  Penalty cost of shortage of wafer type $j$ in period $t$  

Variables:

$u_{jt}$  Number of wafer starts of type $j$ released into production in period $t$  

$v_{ijkt}$  Number of wafer starts of type $j$ whose $k$-th processing step is carried out in machine $i$ in period $t$  

$w_{jt}$  Shortage of wafer type $j$ in period $t$  

$x_{it}$  Number of tools of type $i$ acquired in period $t$  

Table 1: Notation for tool planning model (1)-(6)
capacity; the second set of constraints (3) enforces that the actual production of wafer type $j$ is equal to the number of wafer starts that has completed all of its required processing steps; the third set of constraints (4) enforces that the production and shortage quantities together should exceed the demand; the fourth set of constraints (5) enforces non-negativity of the production-allocation-shortage variables; and the fifth set of constraints (6) enforces the integrality of the tool purchase decisions.

As mentioned in Swaminathan (2002) there are a number of assumptions in the formulation (1)-(6). The time periods in this strategic planning problem are considered to be quite long (between 4-6 months). As a result, it is assumed that inventory is not carried from period to period. Moreover, decisions pertaining to timing of processing steps and effects of congestion within a period are not modelled, as would be in a detailed scheduling model. Also, no degradation in tool capacity over the planning horizon is considered. Finally, even though lead time in tool acquisition is not explicitly modelled, deterministic lead times can be incorporated by adjusting the limits in the summation on the right hand sides of the constraints (2). These assumptions are similar to some of the other tool planning models in the literature (Barahona et al. (2005), Hood et al. (2003), Swaminathan (2000)).

2.2 A generic capacity planning model

We now present a general formulation of a capacity expansion model. Throughout the remainder of this paper we use boldface to denote matrices and vectors. Also, given a vector $a$, the notation $[a]_i$ denotes the $i$-th component of $a$. Similarly, $[A]_{ij}$ denotes the $ij$-th element of a matrix $A$.

The deterministic semiconductor tool planning model (1)-(6) is a special case of the following generic model:

$$\min \sum_{t=1}^{T} \left( a_t^\top x_t + b_t^\top y_t \right)$$ \hspace{1cm} (7)

s.t. \hspace{1cm} $A_t y_t \leq \sum_{\tau=1}^{t} x_\tau \hspace{1cm} \forall \hspace{0.1cm} t = 1, \ldots, T$ \hspace{1cm} (8)

$B_t y_t \geq d_t \hspace{1cm} \forall \hspace{0.1cm} t = 1, \ldots, T$ \hspace{1cm} (9)

$y_t \in \mathbb{R}_+^J, \ x_t \in \mathbb{Z}_+^I \hspace{1cm} \forall \hspace{0.1cm} t = 1, \ldots, T.$ \hspace{1cm} (10)
Formulation (7)-(10) considers capacity acquisition of a set of $I$ resources and allocation of their capacity to a set of $J$ tasks so as to satisfy demand of a set of $K$ items over a planning horizon of $T$ periods. The $I$-dimensional integer vector $x_t$ represents the capacity acquisition decisions (in number of units) for the resources, and the $J$-dimensional vector $y_t$ represents the operational level allocation of capacity to the tasks, in period $t$. The parameters $a_t \in \mathbb{R}^I$, $b_t \in \mathbb{R}^J$ and $d_t \in \mathbb{R}^K$ represent acquisition costs, allocation costs, and demands, respectively, in period $t$. The matrices $A_t \in \mathbb{R}^{I \times J}$ and $B_t \in \mathbb{R}^{K \times J}$ represent resource-task utilization coefficients in period $t$. The objective (7) is to minimize total costs. Constraints (8) ensure that, in each period, capacity allocated to the tasks does not exceed existing capacity, constraints (9) require that, in each period, the capacity allocated to the tasks satisfy required demand, and constraints (10) enforce non-negativity and integrality of the capacity allocation and capacity acquisition decisions, respectively.

To see the connection between the models (1)-(6) and (7)-(10), note that $x_t$ corresponds to the tool purchase decisions ($\{x_{it}\}_{i \in M}$) and $y_t$ corresponds to the production/shortage decisions ($\{u_{jt}\}_{j \in N}, \{v_{ijkt}\}_{i \in M, j \in N, k \in K}, \{w_{jt}\}_{j \in N}$), for period $t$. The parameters $a_t$ and $b_t$ correspond to the objective coefficients ($\{a_{it}\}_{i \in M}$) and ($\{b_{jt}\}_{j \in N}, \{c_{ijkt}\}_{i \in M, j \in N, k \in K}, \{g_{jt}\}_{j \in N}$), respectively. The matrices $A_t$ and $B_t$ correspond to the coefficients of the constraint sets (2) and (3)-(4), respectively. Finally, the vector $d_t$ corresponds to the right hand sides of the constraints (3)-(4), and is of the form $0, \{d_{jt}\}_{j \in N}$ where $0$ is a $|N||K|$-dimensional vector of zeroes. Thus $I = |M|, J = 2|N| + |M||N||K|$ and $K = |N||K| + |N|$.

Formulation (7)-(10) is fairly general. It considers multiple resources, tasks and products, and does not make any assumptions on time-trends in the cost or demand data. The deterministic capacity expansion models considered in Fong and Srinivasan (1981 a,b), Li and Tirupati (1994) and Rajagopalan (1994) are of the form of (7)-(10).

### 2.3 Stochastic programming extensions

Let us now extend the deterministic capacity planning model (7)-(10) to a stochastic setting. We assume that the problem parameters ($a_t, b_t, d_t, A_t, B_t$) are uncertain and evolve as a discrete time stochastic process with a finite support. This information structure can be interpreted as a scenario tree (cf. Ruszczyński and Shapiro (2003)) where a node $n$ in
level $t$ of the tree corresponds to a specific joint realization of the uncertain parameters
$\{(a_\tau, b_\tau, d_\tau, A_\tau, B_\tau)\}_1^T$. There are $T$ levels in the tree, one corresponding to each time period. Each node $n$ of the scenario tree, except the root ($n = 1$), has a unique parent $a(n)$, and each non-leaf node $n$ is the root of a sub-tree $T(n)$. The set $S_t$ denotes the nodes corresponding to time period $t$, and $t_n$ is the time period corresponding to node $n$. The probability of the realization associated with node $n$ is denoted by $p_n$. The probabilities of the nodes in a time period sum to one, i.e., $\sum_{n \in S_t} p_n = 1$ for all $t$, and the sum of probabilities of all child nodes of a parent node equals to the probability of the parent node, i.e., $\sum_{m \in T(n):a(m)=n} p_m = p_n$ for all $n \in T$. The path from the root node to a node $n$ is denoted by $P(n)$. If $n$ is a terminal (leaf) node, i.e., $n \in S_T$, then $P(n)$ corresponds to a scenario, and represents a joint realization of the problem parameters over all periods. There are $S$ leaf nodes corresponding to $S$ scenarios, i.e., $S = |S_T|$. We denote the whole tree $T(1)$ by $T$ and let $N_T$ be the number of nodes in this tree. For node $n$ the problem parameters in period $t_n$ are denoted by $(a_n, b_n, d_n, A_n, B_n)$. Some of the notation associated with a scenario tree is illustrated in Figure 1. We assume that the complete scenario tree describing all possible parameter realizations and associated probabilities is available.

Figure 1: Scenario tree notation
Let us first consider a two-stage model where the first stage involves deciding the capacity acquisition plan for all periods, regardless of the parameter realizations, and the second stage consists of deciding on the capacity allocation plan subject to available capacity and the parameter realizations. Thus the capacity acquisition variables are only indexed by time periods (since these do not change with different parameter realizations) while the allocation decisions are indexed by the nodes of the scenario tree. With an objective of minimizing the expected total costs, a two-stage stochastic programming extension of (7)-(10) is as follows:

\[
\min \sum_{t=1}^{T} \bar{a}_t^T x_t + \sum_{n \in T} b_n^T y_n \tag{11}
\]

\[
\text{s.t.} \quad A_n y_n \leq \sum_{\tau=1}^{t_n} x_\tau \quad \forall \ n \in T \tag{12}
\]

\[
B_n y_n \geq d_n \quad \forall \ n \in T \tag{13}
\]

\[
y_n \in \mathbb{R}_+^J \quad \forall \ n \in T \tag{14}
\]

\[
x_t \in \mathbb{Z}_+^I \quad \forall \ t = 1, \ldots, T. \tag{15}
\]

where \( \bar{a}_t = \sum_{n \in S_t} p_n a_n \), i.e., the expected capacity acquisition cost in period \( t \). The two-stage stochastic programming models considered in Swaminathan (2000) and Swaminathan (2002) are special cases of formulation (11)-(15). The models presented in Barahona et al. (2005) and Hood et al. (2003) are also similar to formulation (11)-(15). However, there, the uncertain parameters are defined over scenarios (paths in the scenario tree) rather than nodes of the scenario tree.

As mentioned earlier, the two-stage model formulation (11)-(15) does not allow any flexibility in the capacity acquisition plan with respect to the parameter realizations. To formulate a multi-stage stochastic programming model, we need to have the capacity acquisition decisions to be dependent on the parameter realizations, and hence the resulting model is as
The above multi-stage model involves both capacity expansion and allocation decisions corresponding to each node of the scenario tree. In addition to semiconductor tool planning, multi-stage stochastic capacity expansion models of the form (16)-(20) have been applied to capacity planning of chemical process networks (Ahmed and Sahinidis (2003)) and electricity distribution networks (Singh (2004), Singh et al. (2005)).

2.4 Single resource substructure

We conclude this section by discussing an important substructure of the multi-stage stochastic capacity planning model (16)-(20). Let us denote the $i$-th component of the vector $a_n$ by $a_{in}$ and that of $x_n$ by $x_{in}$, i.e., $[a_n]_i = a_{in}$ and $[x_n]_i = x_{in}$. Also, we use $\mathbf{x}$ and $\mathbf{y}$ to collectively denote the vectors $\{x_n\}_{n \in T}$ and $\{y_n\}_{n \in T}$, respectively.

The multi-stage capacity planning problem (16)-(20) can be restated as follows:

$$\begin{align*}
\min & \sum_{n \in T} p_n \left( a_n^\top x_n + b_n^\top y_n \right) \\
\text{s.t.} & A_n y_n \leq \sum_{m \in P(n)} x_m \quad \forall n \in T \\
& B_n y_n \geq d_n \quad \forall n \in T \\
& y_n \in \mathbb{R}^J_+ \quad \forall n \in T \\
& x_n \in \mathbb{Z}^I_+ \quad \forall n \in T.
\end{align*}$$

(16)

(17)

(18)

(19)

(20)

The above reformulation decomposes problem (16)-(20) into two separate problems, one (21) involving the capacity allocation decisions, and the other (22) involving the capacity acquisition decisions.
Observe that for a fixed sequence of capacity allocation decisions \( \{y_n\}_{n \in T} \), the optimal capacity acquisition decisions \( \{x_n\}_{n \in T} \) can be obtained via solving (22) independently for each resource \( i \). Suppressing the index \( i \) and using \( \delta_n \) to denote \([A_n y_n]_i\) problem (22) can be written as

\[
\min \sum_{n \in T} p_n a_n x_n \\
\text{s.t.} \quad \sum_{m \in \mathcal{P}(n)} x_m \geq \delta_n \quad \forall \ n \in T \\
x_n \in \mathbb{Z}_+ \quad \forall \ n \in T.
\]

Problem (23) can be interpreted as a single-resource stochastic capacity expansion problem. Here we are given a scenario tree \( T \) where a node \( n \) is associated with a realization of stochastic capacity acquisition cost \( a_n \) and demand \( \delta_n \). We need to decide capacity acquisitions \( x_n \) in each node \( n \) of the scenario tree \( T \) so that total acquired capacity exceeds current demand \( \delta_n \).

Key to the further developments in this paper is the study of the single resource problem (23). It can be shown (see Huang (2005)) that (23) is equivalent to a stochastic version of the classical dynamic deterministic lot-sizing problem which can be solved using a simple greedy algorithm (cf. Johnson (1957), Zipkin (2000)). Next we show that a similar result holds for the stochastic case. First observe that even though (23) is a multi-stage stochastic integer program, the following advantageous property holds.

**THEOREM 1** With integer demand parameters \( \{\delta_n\}_{n \in T} \), the LP relaxation of the single-resource stochastic capacity expansion problem (23) yields integral extreme point solutions.

**PROOF.** See the online companion. □

The above property leads to a strongly polynomial time algorithm for (23). Recall that \( N_T \) is the number of nodes in the scenario tree \( T \).

**THEOREM 2** With integer demand parameters \( \{\delta_n\}_{n \in T} \), the single-resource stochastic capacity expansion problem (23) can be solved in \( O(N_T \log N_T \log \log N_T) \) time.

**PROOF.** See the online companion. □
3 Value of multi-stage stochastic programming

Let the $v^{TS}$ and $v^{MS}$ denote the optimal objective function values of the two-stage model (11)-(15) and the multi-stage model (16)-(20), respectively. For a given set of problem parameters, it is easily verified that any solution to (11)-(15) is feasible to (16)-(20), and the objective function values corresponding to this solution are equal in both problems, thus

$$v^{TS} \geq v^{MS}.$$ 

That is, the overall cost of the multi-stage solution is no greater than that of the two-stage solution. This should come as no surprise, since, the multi-stage solution offers more flexibility in the capacity acquisition decisions with respect to the uncertain parameter realizations. We refer to the difference between the optimal objective values of the two-stage and multi-stage formulations as the value of multi-stage stochastic programming (VMS):

$$VMS = v^{TS} - v^{MS}.$$ 

Unfortunately, the value of multi-stage stochastic programming comes at the expense of solving a much larger and more difficult optimization model. Both (11)-(15) and (16)-(20) are stochastic integer programs, and in general, can be extremely difficult to solve. For our particular case, both models have the property that by fixing the capacity acquisition decisions (the $x$ variables), we can break the problem down to independent capacity allocation problems (in the $y$ variables) corresponding to each node of the scenario tree. Owing to this structure, Benders decomposition (cf. Benders (1962)) is particularly attractive for these problems. In case of (11)-(15) and (16)-(20), this would require us to solve master problems involving the integer variables $x$. While the two-stage model (11)-(15) involves $I \times T$ integer variables, the multi-stage model (16)-(20) involves $I \times N_T$ integer variables (recall that $N_T = |T|$), and for any non-trivial scenario tree $N_T >> T$. Consequently the computational difficulty of (16)-(20) is significantly more than that of (11)-(15). If the VMS is small, then this additional computational effort may not be worthwhile. However, we need a priori estimates of VMS to analyze this tradeoff. Next, we first describe simple bounds on VMS for the single-resource stochastic capacity expansion problem (23) and then use these to get bounds on the VMS for the more general problem (16)-(20).
3.1 VMS for the single-resource problem

Consider the linear relaxation of the single-resource stochastic capacity expansion problem (23) and let \( v^M \) denote its optimal objective function value, i.e.,

\[
v^M = \min_{n \in T} \sum_{n} p_{n} a_{n} x_{n} \\
\text{s.t. } \sum_{m \in P(n)} x_{m} \geq \delta_{n} \quad \forall \ n \in T \\
x_{n} \in \mathbb{R}^+ \quad \forall \ n \in T.
\] (24)

A two-stage model for this single-resource stochastic capacity expansion problem would require that the capacity decisions for each time period be the same irrespective of the parameter realization, i.e.,

\[
v^T = \min_{n \in T} \sum_{n} p_{n} a_{n} x_{n} \\
\text{s.t. } \sum_{m \in P(n)} x_{m} \geq \delta_{n} \quad \forall \ n \in T \\
x_{n} \in \mathbb{R}^+ \quad \forall \ n \in T \\
x_{m} = x_{n} \quad \forall \ m, n \in S_t, \forall \ t.
\] (25)

Next we attempt to bound the VMS = \( v^T - v^M \) using the problem parameters. Recall that the realizations of the stochastic cost and demand parameters are completely described by the scenario tree \( T \) whose node \( n \) corresponds to realization \((a_n, \delta_n)\). Let

\[
a^* = \max_{n \in T} \{a_n\}, \\
a_* = \min_{n \in T} \{a_n\}, \\
\delta^* = \max_{n \in T} \{\delta_m\}, \text{ and} \\
\overline{\delta} = \sum_{n \in S_t} p_n (\max_{m \in P(n)} \delta_m).
\]

That is, \( a^* \) is the maximum cost realized, \( a_* \) is the minimum cost realized, \( \delta^* \) is the maximum demand realized, and \( \overline{\delta} \) is the average (over the scenarios) of the maximum demand in a scenario. Thus \( \delta^* - \overline{\delta} \) gives an indication of the variability of the maximum demand across the scenarios.

**THEOREM 3** \( a_*\delta^* - a^*\overline{\delta} \leq \text{VMS} = v^T - v^M \leq a^*\delta^* - a_*\overline{\delta} \).
**Proof.** Note that any feasible solution $x$ for (24) has to satisfy

\[
\sum_{m \in \mathcal{P}(n)} x_m \geq \max_{m \in \mathcal{P}(n)} \delta_m \quad \forall \ n \in \mathcal{S}_T
\]

\[
\Rightarrow \sum_{n \in \mathcal{S}_T} p_n \left( \sum_{m \in \mathcal{P}(n)} x_m \right) \geq \sum_{n \in \mathcal{S}_T} p_n \left( \max_{m \in \mathcal{P}(n)} \delta_m \right)
\]

\[
\Leftrightarrow \sum_{t=1}^{T} \sum_{n \in \mathcal{S}_t} \left( \sum_{m \in \mathcal{S}_t \cap T(n)} p_m \right) x_n \geq \bar{\delta}
\]

\[
\Leftrightarrow \sum_{n \in T} p_n x_n \geq \bar{\delta},
\]

where the last step follows from the fact that

\[
\sum_{m \in \mathcal{S}_t \cap T(n)} p_m = p_n \quad \forall \ n \in T.
\]

Then if $x^*$ is an optimal solution for (24), we have

\[
v^{M} = \sum_{n \in T} p_n a_n x^*_n \geq a^*_n \sum_{n \in T} p_n x^*_n \geq a^*_n \bar{\delta}.
\]

(26)

Next, consider a feasible solution $\hat{x}$ to (24), such that $\hat{x}_n = \max_{m \in \mathcal{P}(n)} \delta_m - \max_{m \in \mathcal{P}(a(n))} \delta_m$ for all $n \in T$, and $\max_{m \in \mathcal{P}(a(1))} \delta_m = 0$. Then

\[
v^{M} \leq \sum_{n \in T} p_n a_n \hat{x}_n
\]

\[
\leq a^* \sum_{t=1}^{T} \sum_{n \in \mathcal{S}_t} p_n \left( \max_{m \in \mathcal{P}(n)} \delta_m - \max_{m \in \mathcal{P}(a(n))} \delta_m \right)
\]

\[
= a^* \sum_{n \in \mathcal{S}_t} p_n \left( \max_{m \in \mathcal{P}(n)} \delta_m \right)
\]

(27)

\[
= a^* \bar{\delta},
\]

where the third step follows the fact that

\[
\sum_{m \in \mathcal{S}_{t+1} \cap T(n)} p_m = p_n \quad \forall \ t, \ n \in \mathcal{S}_t.
\]

In the two-stage model (25), since the capacity decisions are identical for all nodes in any stage, these have to satisfy the largest possible cumulative demand in that stage, i.e., $\delta_n$ can be replaced with $\bar{\delta}_n = \max_{m \in \mathcal{S}_n} \delta_m$ in (25). Then, by applying the same analysis used for problem (24) to problem (25) with $\bar{\delta}_n$ replacing $\delta_n$, it can be shown that

\[
a^*_n \delta^* \leq v^T \leq a^* \delta^*.
\]

(28)

Combining (26), (27), and (28), the claim follows. \(\square\)
Suppose that the cost parameters $a_n$ are nearly constant, i.e., $a^* \approx a \approx a$, then Theorem 3 implies

$$VMS \approx a(\delta^* - \overline{\delta}).$$

Thus, VMS is directly related to the variability of the (maximum) demand across the scenarios. If demand variability is high then VMS is high, and the two-stage approach is likely to produce poor quality solutions. On the other hand, if there is little variability in the demand data, then as expected the multi-stage approach has little value.

### 3.2 VMS for the capacity planning problem

We shall now describe a lower bound on the VMS for the multi-stage capacity planning model (16)-(20) based on the analysis in the previous section and an optimal solution to the LP relaxation of the two-stage model (11)-(15). Since this LP relaxation can be solved fairly quickly, we can use this lower bound estimate to justify additional computational effort required to solve the difficult multi-stage model (16)-(20). Recall that $i$-th component of the vectors $a_n$ and $x_n$ in (16)-(20) are denoted by $a_{in}$ and $x_{in}$, respectively.

**Theorem 4** Let $\{y_{n}^{TLP}\}_{n \in T}$ be the capacity allocation decisions in an optimal solution to the linear relaxation of the two-stage model (11)-(15). For each resource $i = 1, \ldots, I$, let $\delta_{in} = [A_n y_n^{TLP}]_i$, $\delta_i^* = \max_{n \in T}\{\delta_{im}\}$, $\overline{\delta}_i = \sum_{n \in S_T} p_n (\max_{m \in P(n)}{\{\delta_{im}\}})$, $a_i^* = \max_{n \in T}\{a_{in}\}$, and $a_{i*} = \min_{n \in T}\{a_{in}\}$. Then

$$VMS \geq \sum_{i=1}^{I} \left( a_{i*} \delta_i^* - a_i^* \overline{\delta}_i \right) - \sum_{i=1}^{I} a_{i1}.$$ 

**Proof.** Note that

$$v_{TS}^T \geq \sum_{n \in T} p_n b_n^{TLP} y_n^{TLP} + \sum_{i=1}^{I} v_i^T$$

where

$$v_i^T = \min \sum_{n \in T} p_n a_{in} x_{in} \quad \text{s.t.} \quad \sum_{m \in P(n)}^{n} x_{im} \geq \delta_{in} \quad \forall \ n \in T$$

$$x_{in} \in \mathbb{R}_+ \quad \forall \ n \in T$$

$$x_{im} = x_{in} \quad \forall \ m, n \in S_t, \forall \ t.$$
Since \( \{y_n^{TLP}\}_{n \in \mathcal{T}} \) is a feasible capacity allocation for the multi-stage problem (16)-(20), we have

\[
v^{MS} \leq \sum_{n \in \mathcal{T}} p_n b_n^T y_n^{TLP} + \sum_{i=1}^I o^M_i
\]

where

\[
o^M_i = \min \sum_{n \in \mathcal{T}} p_n a_{in} x_{in} \\
\text{s.t.} \quad \sum_{m \in \mathcal{P}(n)} x_{im} \geq \delta_{in} \quad \forall \ n \in \mathcal{T} \\
x_{in} \in \mathbb{Z}_+ \quad \forall \ n \in \mathcal{T}
\]

\[
= \min \sum_{n \in \mathcal{T}} p_n a_{in} x_{in} \\
\text{s.t.} \quad \sum_{m \in \mathcal{P}(n)} x_{im} \geq \lceil \delta_{in} \rceil \quad \forall \ n \in \mathcal{T} \\
x_{in} \in \mathbb{R}_+ \quad \forall \ n \in \mathcal{T}
\]

\[
= \max \sum_{n \in \mathcal{T}} \lceil \delta_{in} + (\lceil \delta_{in} \rceil - \delta_{in}) \rceil \pi_{in} \\
\text{s.t.} \quad \sum_{m \in \mathcal{T}(n)} \pi_{im} \leq p_n a_{in} \quad \forall \ n \in \mathcal{T} \\
\pi_{in} \in \mathbb{R}_+ \quad \forall \ n \in \mathcal{T},
\]

in which the second equality comes from Theorem 1 and the third equality comes from linear program duality. We can further define:

\[
v^M_i = \min \sum_{n \in \mathcal{T}} p_n a_{in} x_{in} \\
\text{s.t.} \quad \sum_{m \in \mathcal{P}(n)} x_{im} \geq \delta_{in} \quad \forall \ n \in \mathcal{T} \\
x_{in} \in \mathbb{R}_+ \quad \forall \ n \in \mathcal{T}
\]

\[
= \max \sum_{n \in \mathcal{T}} \delta_{in} \pi_{in} \\
\text{s.t.} \quad \sum_{m \in \mathcal{T}(n)} \pi_{im} \leq p_n a_{in} \quad \forall \ n \in \mathcal{T} \\
\pi_{in} \in \mathbb{R}_+ \quad \forall \ n \in \mathcal{T}.
\]
Therefore,
\[
O_M^i \leq v_i^M + \max \sum_{n \in T} (\lceil \delta_{in} \rceil - \delta_{in}) \pi_{in}
\]
\[
\text{s.t. } \sum_{m \in T(n)} \pi_{im} \leq p_n a_{in} \quad \forall \ n \in T \]
\[
\pi_{in} \in \mathbb{R}_+ \quad \forall \ n \in T
\]
\[
\leq v_i^M + \max \sum_{n \in T} \pi_{in}
\]
\[
\text{s.t. } \sum_{m \in T(n)} \pi_{im} \leq p_n a_{in} \quad \forall \ n \in T \]
\[
\pi_{in} \in \mathbb{R}_+ \quad \forall \ n \in T
\]
\[
= v_i^M + \min \sum_{n \in T} p_n a_{in} x_{in}
\]
\[
\text{s.t. } \sum_{m \in P(n)} x_{im} \geq 1 \quad \forall \ n \in T
\]
\[
x_{in} \in \mathbb{R}_+ \quad \forall \ n \in T
\]
\[
= v_i^M + a_{i1},
\]
where the second inequality comes from \( [\delta_{in} ] - \delta_{in} \leq 1 \), the third equality comes from duality, the fourth equality comes from the fact that \( p_1 = 1 \) and an optimal solution to a stochastic single-resource capacity expansion problem with a demand of 1 unit in every node is to add 1 unit of capacity at the root node. Therefore, we have:
\[
V_{M} \geq \sum_{i=1}^{I} \left( v_i^T - v_i^M - a_{i1} \right),
\]
and the result follows from the bounds (27) and (28) derived in the proof of Theorem 3. □

4 An approximation algorithm

In this section we develop an approximation algorithm for the multi-stage capacity planning problem (16)-(20).

It can be easily shown that any instance of the NP-hard integer knapsack problem (cf. Garey and Johnson (1979)) with \( I \) items can be polynomially transformed to a single period instance of the deterministic capacity planning problem (7)-(10). Since (7)-(10) is just a single scenario instance of the stochastic models (11)-(15) and (16)-(20), we have the following result. The detailed proof is omitted in the interest of brevity.

**THEOREM 5** The deterministic capacity planning problem (7)-(10) and its stochastic counterparts (11)-(15) and (16)-(20) are NP-hard.
Motivated by this intractability, we propose the approximation scheme outlined in Figure 2. The algorithm exploits the decomposable structure revealed by the reformulation (21)-(22) of the problem.

Figure 2: An approximation scheme for the multi-stage capacity planning problem (16)-(20)

Algorithm 1

1: Solve the LP relaxation of (16)-(20). Let \((x^{LP}, y^{LP}) := \{(x_n^{LP}, y_n^{LP})\}_{n \in T}\) be an optimal solution and \(v^{LP}_{MS}\) be the optimal value. If \(x_n^{LP}\) is integral for all \(n\), stop and return \((x^{LP}, y^{LP})\).

2: For each resource \(i = 1, 2, ..., I\), solve independent single-resource capacity acquisition problems:

\[
\begin{align*}
\min & \quad \sum_{n \in T} p_n a_{in} x_{in} \\
\text{s.t.} & \quad \sum_{m \in P(n)} x_{im} \geq \left\lceil [A_n y_n^{LP}]_i \right\rceil \quad \forall \ n \in T \\
& \quad x_{in} \in \mathbb{Z}_+ \quad \forall \ n \in T,
\end{align*}
\]

and let \(x_{in}^H\) denote the corresponding solutions. Note that the integrality of the decision variables allows for the rounding up of \([A_n y_n^{LP}]_i\).

3: For each \(n \in T\), solve independent capacity allocation problems:

\[
\begin{align*}
\min & \quad b_n^\top y_n \\
\text{s.t.} & \quad A_n y_n \leq \sum_{m \in P(n)} x_m^H \\
& \quad B_n y_n \geq d_n, y_n \in \mathbb{R}^J_+, \\
\end{align*}
\]

where \(x_n^H = (x_{in}^H)_{i \in I}\). Let \(y_n^H\) denote the corresponding optimal solution.

4: Return \((x^H, y^H) := \{(x_n^H, y_n^H)\}_{n \in T}\).

Step 1 of Algorithm 1 requires the solution of the LP relaxation of (16)-(20). This problem is a multi-stage stochastic linear program which can, in general, be solved by the Nested L-Shaped Decomposition algorithm (cf. Birge (1985)). Step 2 requires the solution of \(I\) single-resource stochastic capacity expansion problems (23) which are multi-stage stochastic integer programs. Since the right-hand-sides are integral, as noted in Theorem 2, these problems can be solved very efficiently. Finally, Step 3 requires the solution of independent
simple linear capacity allocation problems for each node in the tree.

5 Analysis of the approximation algorithm

This section analyzes the optimality gap of the approximate solution produced by Algorithm 1. Given capacity acquisition-allocation solutions \((x, y)\), let us denote the corresponding objective function value as
\[
f(x, y) = \sum_{n \in T} p_n \left( a_n^\top x_n + b_n^\top y_n \right).
\]
Recall that \((x^{LP}, y^{LP})\) denotes the capacity acquisition-allocation solutions corresponding to the LP relaxation of (16)-(20), and \((x^H, y^H)\) denotes the capacity acquisition-allocation solutions returned by Algorithm 1. Let \((x^*, y^*)\) denote an optimal solution to (16)-(20). Then the optimality gap of \((x^H, y^H)\) is
\[
\text{GAP} = f(x^H, y^H) - f(x^*, y^*).
\]

**THEOREM 6** \(\text{GAP} \leq \sum_{i=1}^I \alpha_{i1}\), where 1 is the root node of the scenario tree.

**PROOF.** Note that
\[
\text{GAP} \leq f(x^H, y^H) - f(x^{LP}, y^{LP})
\]
\[
= f(x^H, y^H) - f(x^H, y^{LP}) + f(x^H, y^{LP}) - f(x^{LP}, y^{LP})
\]
\[
\leq f(x^H, y^{LP}) - f(x^{LP}, y^{LP}),
\]
where last inequality follows from the fact that \(f(x^H, y^H) \leq f(x^H, y^{LP})\) (recall that \(y^H\) is an optimal capacity allocation corresponding to \(x^H\), i.e., an optimal solution to capacity allocation problem solved in step 3 of Algorithm 1, whereas \(y^{LP}\) is just a feasible capacity allocation solution). Now
\[
f(x^H, y^{LP}) - f(x^{LP}, y^{LP}) = \sum_{i=1}^I \sum_{n \in T} p_n a_n \left( x^H_{in} - x^{LP}_{in} \right).
\] (29)
Recall that in Algorithm 1 the capacity acquisition subproblems separate over \( i \in I \). We can then analyze each subproblem independently. Note that, for any \( i \in I \),

\[
\sum_{n \in T} p_n a_{in} x_{in}^H = \min \sum_{n \in T} p_n a_{in} x_{in}
\]

\[
\text{s.t. } \sum_{m \in p(n)} x_{im} \geq \left\lfloor A_{n} y_{n}^{LP} \right\rfloor_i \quad \forall n \in T
\]

\[
x_{in} \in \mathbb{R}_+ \quad \forall n \in T
\]

\[
= \max \sum_{n \in T} \left\lfloor A_{n} y_{n}^{LP} \right\rfloor_i \pi_{in}
\]

\[
\text{s.t. } \sum_{m \in p(n)} \pi_{im} \leq p_n a_{in} \quad \forall n \in T
\]

\[
\pi_{in} \in \mathbb{R}_+ \quad \forall n \in T,
\]

(30)

and

\[
\sum_{n \in T} p_n a_{in} x_{in}^{LP} = \min \sum_{n \in T} p_n a_{in} x_{in}
\]

\[
\text{s.t. } \sum_{m \in p(n)} x_{im} \geq [A_{n} y_{n}^{LP}]_i \quad \forall n \in T
\]

\[
x_{in} \in \mathbb{R}_+ \quad \forall n \in T
\]

\[
= \max \sum_{n \in T} [A_{n} y_{n}^{LP}]_i \pi_{in}
\]

\[
\text{s.t. } \sum_{m \in p(n)} \pi_{im} \leq p_n a_{in} \quad \forall n \in T
\]

\[
\pi_{in} \in \mathbb{R}_+ \quad \forall n \in T,
\]

(31)

Thus

\[
\sum_{n \in T} p_n a_{in} (x_{in}^H - x_{in}^{LP}) \leq \max \sum_{n \in T} \left( \left\lfloor A_{n} y_{n}^{LP} \right\rfloor_i - [A_{n} y_{n}^{LP}]_i \right) \pi_{in}
\]

\[
\text{s.t. } \sum_{m \in p(n)} \pi_{im} \leq p_n a_{in} \quad \forall n \in T
\]

\[
\pi_{in} \in \mathbb{R}_+ \quad \forall n \in T
\]

\[
\leq \max \sum_{n \in T} \pi_{in}
\]

\[
\text{s.t. } \sum_{m \in p(n)} \pi_{im} \leq p_n a_{in} \quad \forall n \in T
\]

\[
\pi_{in} \in \mathbb{R}_+ \quad \forall n \in T,
\]

(32)

\[
\leq \min \sum_{n \in T} p_n a_{in} x_{in}
\]

\[
\text{s.t. } \sum_{m \in p(n)} x_{im} \geq 1 \quad \forall n \in T
\]

\[
x_{in} \in \mathbb{R}_+ \quad \forall n \in T
\]

\[
= a_{i1},
\]

where the first inequality follows from (30) and (31), the second inequality follows from the fact that \( \left\lfloor A_{n} y_{n}^{LP} \right\rfloor_i - [A_{n} y_{n}^{LP}]_i \leq 1 \), the third equality follows from duality, and the last equality follows from the fact that an optimal solution to a single resource capacity acquisition problem with a demand of 1 unit in every node is to add 1 unit of capacity at the root.
node. The result then follows from incorporating (32) in (29). □

Theorem 6 shows the surprising result that the optimality gap of Algorithm 1 is bounded above by a factor that is independent of the number of time stages, number of branches in the tree, number of tasks, or any problem data except for the sum of the capacity acquisition costs of the resources in the first stage. If we consider instances of (16)-(20) that have the same first-stage acquisition costs, but different topology of the scenario tree, then we have the following asymptotic quality guarantee for Algorithm 1.

**COROLLARY 1** \[
\lim_{T \to \infty} \frac{f(x^H, y^H) - f(x^*, y^*)}{T} = 0.
\]

**PROOF.** Immediate. □

**COROLLARY 2** Assume that,

(i) there exists \( \epsilon_1 > 0 \) such that, for each \( n \in \mathcal{T} \), there exists at least one product \( k_n \in \{1, \ldots, K\} \) whose demand is at least \( \epsilon_1 \), i.e., \( [d_n]_{k_n} \geq \epsilon_1 \); and

(ii) there exists \( \epsilon_2 > 0 \) such that, for each \( n \in \mathcal{T} \), and any task \( j \in \{1, \ldots, J\} \) and product \( k \in \{1, \ldots, K\} \) with a positive demand-task allocation ratio, i.e., \( [B_n]_{kj} > 0 \), the allocation cost \( [b_n]_{kj} \geq \epsilon_2[B_n]_{kj} \).

Then the following holds:

\[
\lim_{T \to \infty} \frac{f(x^H, y^H) - f(x^*, y^*)}{f(x^*, y^*)} = 0.
\]

**PROOF.** See the online companion. □

Note that the assumptions in Corollary 2 are not particularly restrictive. These only require that, for every node of the scenario tree, there always exists some positive demand, and that the unit allocation cost is never smaller than some positive level.
6 Computational results

In this section, we report on computational experiments with the proposed multi-stage stochastic programming approach for a realistic scale semiconductor tool planning problem. Our experiments focus on two objectives: (i) to investigate the value of multi-stage stochastic programming; and, (ii) to investigate the performance of the proposed approximation scheme. In the following, we first describe our experimental environment and then report on the experimental results in light of each of the above two objectives.

6.1 Experimental environment

Our test problem instances are derived from a realistic scale two-stage stochastic programming model for semiconductor tool planning from Barahona et al. (2005) and Hood et al. (2003). The formulation is very similar to the tool planning model (1)-(6) with an additional purchase budget constraint. Numerical data for instances of the model with two periods and 2, 3, and 4 scenarios are available in Ahmed (2004) (see the SEMI test set). The instances consists of 306 machine tools, 40 wafer types (products) and 2575 processing steps. The only uncertain parameters are demands of 7 of the 40 products. The demand data for the uncertain products for each scenario varies around that of a “base” scenario (having the highest probability).

We generate our test problem instances from the above data set as follows. We ignore the budget constraint since our approach is not designed to handle such a constraint. The original cost and demand data corresponding to the first period base scenario is used for the root node (node 1) of our scenario tree. The demand data (for the 7 products with uncertain demand) for each subsequent node is independently generated by multiplying the root node data with a random number generated from a lognormal distribution $\lambda(\mu, \sigma)$, where $\mu$ is the expectation and $\sigma$ is the standard deviation. We considered four trends of the demand with respect to the time period. These demand patterns are indicated in Table 2. In Table 2, $z_n$ is the demand of a product in node $n$, and $z_1$ is the demand of the product in the root node. Recall that $t_n$ is the stage number of node $n$ (if $n \in \mathcal{S}_t$, then $t_n = t$). So for all nodes in the same stage, we have the same demand distribution. The cost data is discounted at the rate
of 5% for each stage of the scenario tree.

Table 2: Demand patterns

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 constant mean, constant standard deviation.</td>
<td>( z_n ) generated from ( z_1 \lambda(1, 0.5) )</td>
</tr>
<tr>
<td>2 constant mean, increasing standard deviation.</td>
<td>( z_n ) generated from ( z_1 \lambda(1, 0.5 + 0.1t_n) )</td>
</tr>
<tr>
<td>3 increasing mean, constant standard deviation.</td>
<td>( z_n ) generated from ( z_1 \lambda(1 + 0.5t_n, 0.5) )</td>
</tr>
<tr>
<td>4 increasing mean, increasing standard deviation.</td>
<td>( z_n ) generated from ( z_1 \lambda(1 + 0.5t_n, 0.5 + 0.1t_n) )</td>
</tr>
</tbody>
</table>

We consider scenario trees with the number of stages (\( T \)) varying from 2 to 5, and the number of branches (\( B \)) for each non-leaf node varying from 2 to 5. Thus, there are, in total, 7 scenario tree structures. The nodes in these trees vary from 3 to 31. For each tree structure and demand pattern combination, we generate 5 problem instances, and report statistics averaged over these 5 instances. A total of \((7 \times 4 \times 5 =) 140\) problem instances are considered. To get a sense of the sizes of these instances, note that the smallest multi-stage instance with \( T = 2 \) and \( B = 2 \) consists of 8,763 constraints and 15,621 variables of which 918 are integers, and an instance with \( T = 5 \) and \( B = 2 \) consists 90,551 constraints and 161,417 variables of which 9,486 are integers.

Our experiments utilize C/C++ implementations of Algorithms 1 and 2. CPLEX 9.0 is used to solve the linear programs in Steps 1 and 3 of Algorithm 1. All numerical experiments are conducted on an IBM PC with 1024 MB RAM and a PENTIUM4 1.6GHz processor.

6.2 Value of multi-stage stochastic programming

To compare two-stage and multi-stage models, we define the Relative Value of Multi-stage Stochastic Programming as:

\[
RVMS = \frac{v^{TS} - v^{MS}}{v^{TS}},
\]

where \( v^{TS}, v^{MS} \) are the optimal values of two-stage model and multi-stage model, respectively. However, since it is hard to solve the two-stage and multi-stage models to optimality,
we consider the following lower bound:

$$\text{RVMS} \geq \frac{v_{LP}^{TS} - v_{HR}^{MS}}{v_{LP}^{TS}},$$

where $v_{LP}^{TS}$ is the optimal value of the linear relaxation of two-stage model and $v_{HR}^{MS}$ is the objective function value of an approximate solution (obtained using Algorithm 1) for the multi-stage model.

In Figure 3, we observe the behavior of the lower bound on RVMS (averaged over 5 instances) with respect to the number of stages $T$ for each of the 4 demand patterns. The number of branches $B$ is fixed at 2. Our first observation is that for all the four demand patterns, the RVMS lower bound increases as the number of stages increases. This implies that the value of multi-stage stochastic programming increases with the planning horizon. Our second observation is that, consistent with the theoretical analysis of Section 3, the value of multi-stage stochastic programming increases with the variability of demand (the RVMS lower bound is larger for demand patterns 2 and 4 that have increasing variability). Moreover, the rate at which the value increases with the planning horizon length also increases with demand variability.

In Figure 4, we observe the behavior of the lower bound on RVMS (averaged over 5
Figure 4: The value of multi-stage stochastic programming with increasing number of branches

instances) with respect to the number of branches $B$ for each of the 4 demand patterns. The number of stages $T$ is fixed at 3. We observe that in most cases, the value of multi-stage stochastic programming increases with the number of branches since the variability of the demand data increases. Also, as before, the rate at which the value increases with the number of branches also increases with demand variability.

6.3 Performance of the approximation scheme

In this section, we report on the solution quality and computational efficiency of Algorithm 1.

A measure of the quality of an approximate solution to the multi-stage model is the relative gap defined as:

$$\text{RGAP} = \frac{v_{HR}^{MS} - v^{MS}}{v^{MS}},$$

where $v_{HR}^{MS}$ and $v^{MS}$ denote the objective value of the approximate solution and that of an optimal solution of the multi-stage model, respectively. To avoid solving the multi-stage
model to optimality, we consider an upper bound on RGAP:

$$\text{RGAP} \leq \frac{v_{MS}^{HR} - v_{LP}^{MS}}{v_{LP}^{MS}},$$

where $v_{LP}^{MS}$ is the optimal value of the linear programming relaxation of the multi-stage model.

In Figure 5, we observe the behavior of the upper bound on RGAP (averaged over 5 instances) with respect to the number of stages $T$ for each of the 4 demand patterns. The number of branches $B$ is fixed at 2. Our first observation is that typically the upper bound on RGAP decreases, hence the approximate solution quality increases, with the increase in the number of stages. This is consistent with the theoretical analysis in Corollary 2. Our second observation is that the upper bound on RGAP decreases, hence the approximate solution quality increases, with increase in the demand variability. Comparing Figures 3 and 5, we find that, fortunately, the instance with high VMS are precisely the ones for which the approximation schemes provide good quality solutions.

In Figure 5, we observe the behavior of the upper bound on RGAP (averaged over 5 instances) with respect to the number of branches $B$ for each of the 4 demand patterns. The number of stages $T$ is fixed at 2. In this case, we observe that the upper bound on RGAP
is quite independent on the number of branches. This can be explained by the fact that the
optimal value of the multi-stage model is little affected by the number of branches. On the
other hand the effect of demand variability is again clear. Higher demand variability leads
to smaller upper bounds on RGAP, i.e., better quality approximate solutions.

Overall, the relative optimality gap of the approximation scheme is at most 15% and
could be as small as 1%. Moreover, as mentioned earlier, it is consistently observed that, for
instances with high VMS the relative optimality gap is small.

Finally, to appreciate the computational efficiency of the proposed approximation scheme,
note that for all of the instances considered, the approximation scheme never requires more
than 2 CPU minutes. By contrast, exact optimization of a multi-stage instance, with just 3
nodes in the scenario tree \((T = 2 \text{ and } B = 2)\), using the MIP solver of CPLEX 9.0 requires
over an hour.

7 Summary of contributions

In this paper, we propose a generic multi-period capacity planning problem under uncertainty
involving multiple resources, tasks and products.
First, we compare two-stage and multi-stage stochastic integer programming approaches for this problem. The concept of value of multi-stage stochastic programming (VMS) is discussed and informative analytical bounds are developed.

Second, by identifying and exploiting a key single resource substructure in the problem, we propose an efficient approximation scheme for the difficult multi-stage model. We show that the absolute optimality gap of the approximation scheme is bounded above by a factor that is independent of the number of time stages, number of branches in the scenario tree, number of tasks, or any problem data except for the sum of the capacity acquisition costs of the resources in the first stage. This leads to an asymptotic optimality guarantee of the approximation scheme with respect to the size of the scenario tree.

Finally, we present numerical results using realistic-scale problem instances corresponding to semiconductor tool planning. Our numerical results indicate that a lower bound on the relative VMS can be as high as 70%. Recall that this lower bound is obtained by comparing the cost of an approximate solution to the multi-stage model to that of a lower bound on the cost of an optimal solution of the two-stage model. Therefore, this suggests that even an approximate solution to the multi-stage model may be far superior to any optimal solution to the two-stage model. These results confirm that the VMS for these problems is quite high. Moreover the quality and performance of the approximation scheme is very satisfactory, more so, for cases where the VMS is high.

Acknowledgments

A preliminary version of this work was presented at the International Conference on Modelling and Analysis in Semiconductor Manufacturing in Phoenix, AZ in 2002, and has appeared in the unrefereed proceedings of this conference. This research has been supported by the National Science Foundation under grants DMI-0099726 and DMI-0133943. We thank two anonymous reviewers and the Associate Editor for very helpful comments.
References


Online companion to:

“The value of multi-stage stochastic programming in capacity planning under uncertainty”
by Kai Huang and Shabbir Ahmed

This online companion provides proofs of some results omitted from the main body of the paper. We also report some additional computational results. The numbered references and citations correspond to those in the main paper, and all new expressions, results, figures and tables are numbered contiguously following those in the main paper.

Proof of Theorem 1

THEOREM 1 With integer demand parameters \( \{\delta_n\}_{n \in T} \), the LP relaxation of the single-resource stochastic capacity expansion problem (23) yields integral extreme point solutions.

PROOF. Note that the constraint matrix of (23) is a 0-1 matrix with a 1 in the row (corresponding to node) \( i \) and the column (corresponding to node) \( j \) if \( i \in T(j) \). Let us order the rows of the constraint matrix such that for each column \( j \in T \), the rows corresponding to \( T(j) \) are consecutive, with row \( i' \in T(j) \) appearing before row \( i'' \in T(j) \) if \( i'' \in T(i') \). With such an ordering it is easy to see that the 1’s in each column appear in consecutive rows. Thus the constraint matrix is an interval matrix and is totally unimodular (cf. Nemhauser and Wolsey (1988)), and the result follows. \( \square \)
An efficient algorithm for the single-resource problem

By virtue of Theorem 1 and the fact that the right-hand-sides of the single-resource problem (23) are integral, we only need to find an efficient scheme for the linear program

\[
\begin{align*}
\min \quad & \sum_{n \in \mathcal{T}} c_n x_n \\
\text{s.t.} \quad & \sum_{m \in \mathcal{P}(n)} x_m \geq \delta_n \quad \forall n \in \mathcal{T} \\
& x_n \in \mathbb{R}_+ \quad \forall n \in \mathcal{T},
\end{align*}
\]

where \( \delta_n \in \mathbb{Z} \) and we have used \( c_n \) to succinctly denote \( p_n a_{n_l} \). The dual of (33) is

\[
\begin{align*}
\max \quad & \sum_{n \in \mathcal{T}} \delta_n \pi_n \\
\text{s.t.} \quad & \sum_{m \in \mathcal{T}(n)} \pi_m \leq c_n \quad \forall n \in \mathcal{T} \\
& \pi_n \in \mathbb{R}_+ \quad \forall n \in \mathcal{T},
\end{align*}
\]

Our proposed algorithmic scheme is based on a greedy approach for solving the dual problem (34). The scheme takes advantage of complementary slackness conditions to recover primal optimal solutions. Figure 8 summarizes the proposed strategy. Here, we assume that the parameters \( c_n \) and \( \delta_n \) are strictly positive for all \( n \). The scheme uses two different indexing schemes for the nodes in the tree \( \mathcal{T} \):

**Indexing scheme 1.** The nodes in \( \mathcal{T} \) are indexed \( 1, 2, \ldots, N_T \) in increasing order of their time stage, i.e., \( t_1 \leq t_2 \leq \cdots \leq t_{N_T} \). No particular ordering is imposed on the indices of the nodes in the same time stage. Thus the root node has an index of 1.

**Indexing scheme 2.** The nodes in \( \mathcal{T} \) are indexed \( 1, 2, \ldots, N_T \) in decreasing order of the corresponding cumulative demand, i.e., \( \delta_1 \geq \delta_2 \geq \cdots \geq \delta_{N_T} \). If \( \delta_m = \delta_n \), then \( m < n \) if \( t_m < t_n \).

The two indexing schemes corresponding to an exemplary scenario tree are illustrated in Figure 7.

As mentioned earlier, the greedy dual step first assigns the largest dual value (as permitted by the constraints) to the node with the largest demand, and then considers the node with the next largest demand, and so on. The marker \( m_k \) is assigned the index of the node (closest on the path \( \mathcal{P}(k) \)) to \( k \) whose corresponding dual constraint becomes tight when
the dual value for node $k$ is set. Note that once $k$ has a positive dual value, no other nodes in $T(m_k)$ will be further considered (any other node in $T(m_k)$ will satisfy the condition in line 4 of the dual step). Thus all nodes in $l \in T(m_k)$ except for the ones with $m_l > 0$, will have $\pi^*_l = 0$. The marker $m_k$ is used in the primal step to set the primal variables such that complementary slackness conditions are satisfied. Note that according to the algorithm, only a node $k$ with $m_k > 0$ could have a positive dual value, and a node $n$ could have a positive primal value only if $n = m_l$ for some node $l \in T(n)$.

The following results establish the validity of Algorithm 2. For the remainder of this section we use indexing scheme 2 for the node labels.

**Lemma 1** In each iteration $k \in \{1, \ldots, N_T\}$ of the dual step of Algorithm 2, the dual solution $\pi^*$ satisfies

$$\sum_{m \in T^k(n)} \pi^*_m \leq c_n \quad \text{for all } n \in \mathcal{T},$$

where $T^k(n) = T(n) \cap \{1, 2, \ldots, k\}$.

**Proof.** By induction on $k$, it can be seen that $c^k_n = c_n - \sum_{m \in T^k(n)} \pi^*_m$. Also, $n \in \mathcal{P}(k)$ if and only if $k \in T(n)$ and $\pi^*_k \leq c_n^{k-1}$ for all $n \in \mathcal{P}(k)$. Therefore, we always have $c^k_n \geq 0$ for any $n \in \mathcal{T}$. $\square$

Lemma 1 guarantees the feasibility of the dual solution $\pi^*$ (let $k = N_T$ in (35)). Furthermore, we also have $\sum_{l \in T^k(n)} \pi^*_l = c_n$ for all $n \in \operatorname{argmin}_{m \in \mathcal{P}(k)} \{c^k_m\}$. Since dual feasibility of $\pi^*$
Algorithm 2

The dual step:
1: label the nodes in $\mathcal{T}$ according to indexing scheme 2
2: initialize $\pi^*_n = 0$, $c^0_n = c_n$ and $m_n = 0$ for all $n \in \mathcal{T}$
3: for $k = 1, \ldots, N_T$ do
4: if there exists $n \in \mathcal{P}(k)$ such that $n = m_l$ for some $l \in \mathcal{T}(n)$ then
5: break
6: else
7: set $\pi^*_k = \min_{n \in \mathcal{P}(k)} \{c^{k-1}_n\}$
8: set $m_k = \arg\min_{n \in \mathcal{P}(k)} \{c^{k-1}_n\}$ such that $c^{k-1}_l > c^{k-1}_{m_k}$ for all $l \in \mathcal{P}(k) \setminus \mathcal{P}(m_k)$
9: set $c^k_n = c^{k-1}_n - \pi^*_k$ if $n \in \mathcal{P}(k)$ and $c^k_n = c^{k-1}_n$ otherwise
10: end if
11: end for

The primal step:
1: transform the node indices as well as $m_n$ to indexing scheme 1
2: initialize $x^*_n = 0$ for all $n \in \mathcal{T}$
3: for $n = 1, \ldots, N_T$ do
4: if there exists $l \in \mathcal{T}(n)$ such that $n = m_l$ then
5: set $x^*_n = \delta_l - \sum_{k \in \mathcal{P}(n) \setminus \{n\}} x^*_k$
6: end if
7: end for
implies $\sum_{l \in T(n)} \pi^*_l \leq c_n$ and $\pi^*_n \geq 0$, we also have:

$$\sum_{l \in T^s(n)} \pi^*_l = \sum_{l \in T(n)} \pi^*_l = c_n \quad \forall \ n \in \text{argmin}_{m \in P(k)}\{c_k^{k-1}\}. \tag{36}$$

That is, for any $l \in T(n)$ such that $l \notin \{1, \ldots, k\}$, $\pi^*_l=0$.

**Lemma 2** The primal solution $x^* = (x^*_1, x^*_2, \ldots, x^*_{N_f})$ produced by the primal step of Algorithm 2 is feasible.

**Proof.** By construction, if a node $n$ is such that $m_n > 0$, then the following equalities hold:

$$\sum_{m \in P(n)} x^*_m = \sum_{m \in P(m_n)} x^*_m = \delta_n. \tag{37}$$

Now consider a node $n$ such that $m_n = 0$. Let $l \in P(n)$ be such that $l = m_k$ for some node $k \in T(m_k)$. Note that such a node $l$ must always exists, since the root node is one such node.

Suppose $l$ be the closest (on the path $P(n)$) such node to $n$. Note that $n, k \in T(l)$ while $m_n = 0$ and $m_k > 0$, thus in the dual step node $k$ must have been considered before node $n$, i.e., $\delta_k \geq \delta_n$. According to (37), $\sum_{m \in P(n)} x^*_m = \sum_{m \in P(m_k)} x^*_m = \sum_{m \in P(k)} x^*_m = \delta_k \geq \delta_n$ (the first equality holds since $x^*_m = 0$ for $m \in P(n)\setminus P(m_k)$). \hfill \square

**Theorem 7** The solutions $x^*$ and $\pi^*$ returned by Algorithm 2 are optimal solutions of (33) and (34), respectively.

**Proof.** Lemmas 1 and 2 have proven the feasibility of $\pi^*$ and $x^*$. Here we show that $\pi^*$ and $x^*$ satisfies the complementary slackness conditions:

$$\pi^*_n > 0 \implies \sum_{m \in P(n)} x^*_m = \delta_n \tag{38}$$

$$\sum_{m \in T(n)} \pi^*_m < c_n \implies x^*_n = 0. \tag{39}$$

We prove by induction on the nodes indexed according to scheme 2.

The base case: Consider node 1. Note that $\pi^*_1 = c_{m_1} > 0$, then (38) follows from (37). For all other nodes $n \in T(m_1)$, $\pi^*_n = 0$. On the other hand, $\sum_{m \in T(m_1)} \pi^*_m = c_{m_1}$, thus
\{n \in T(m_1) : \sum_{m \in T(n)} \pi_m^* < c_n \} \subseteq T(m_1) \setminus \{m_1\}. Then (39) holds, since \( x_n^* = 0 \) for all \( n \in T(m_1) \setminus \{m_1\} \). Note that we have verified the complementary slackness conditions for all nodes in \( T(m_1) \), and not just node 1.

The induction step: Assume that we have checked nodes 1, ..., \( k \), and now consider node \( k+1 \). If \( m_{k+1} = 0 \), then this node has already been checked since then \( k+1 \in T(m_j) \) for some \( j < k+1 \). So we assume that \( m_{k+1} > 0 \). Denote \( \{1, 2, ..., k\} \) by \( \mathcal{H}(k) \) and \( \{m_1, m_2, ..., m_k\} \) by \( \mathcal{R}(k) \). Also define \( \mathcal{F}(k) = \cup \{T(m_n) : n \in \mathcal{H}(k), m_n > 0\} \). We now examine the nodes in \( T(m_{k+1}) \setminus \mathcal{F}(k) \). Notice for all nodes \( n \) in \( T(m_{k+1}) \setminus \mathcal{F}(k) \setminus \{k+1\} \), \( \pi_n^* = 0 \) since \( m_n = 0 \). Amongst the nodes in \( T(m_{k+1}) \setminus \mathcal{F}(k) \), only node \( k+1 \) could have a positive dual value. For node \( k+1 \), (38) then holds from (37). On the other hand, \( \sum_{m \in T(m_{k+1})} \pi_m^* = c_{m_{k+1}} \) from (36), so \( \{l \in T(m_{k+1}) \setminus \mathcal{F}(k) : \sum_{m \in T(l)} \pi_m^* < c_l \} \subseteq T(m_{k+1}) \setminus \mathcal{F}(k) \setminus \{m_{k+1}\} \). The conclusion then holds since \( x_m^* = 0 \) for all \( m \in T(m_{k+1}) \setminus \mathcal{F}(k) \setminus \{m_{k+1}\} \). \( \square \)

It can be shown (Huang (2005)) that by adopting an appropriate data structure, Algorithm 2 can be executed in no more than \( O(N_T \log N_T \log \log N_T) \) operations. We omit details of this complexity calculation here.

**Proof of Corollary 2**

**COROLLARY 2** Assume that,

(i) there exists \( \epsilon_1 > 0 \) such that, for each \( n \in \mathcal{T} \), there exists at least one product \( k_n \in \{1, \ldots, K\} \) whose demand is at least \( \epsilon_1 \), i.e., \( [d_n]_{k_n} \geq \epsilon_1 \); and

(ii) there exists \( \epsilon_2 > 0 \) such that, for each \( n \in \mathcal{T} \), and any task \( j \in \{1, \ldots, J\} \) and product \( k \in \{1, \ldots, K\} \) with a positive demand-task allocation ratio, i.e., \( [B_n]_{kj} > 0 \), the allocation cost \( [b_n]_j \geq \epsilon_2 [B_n]_{kj} \).

Then the following holds:

\[
\lim_{T \to \infty} \frac{f(x^H, y^H) - f(x^*, y^*)}{f(x^*, y^*)} = 0.
\]
PROOF. From Theorem 6 we know that
\[
\frac{f(x^H, y^H) - f(x^*, y^*)}{f(x^*, y^*)} \leq \sum_{i=1}^I \alpha_i \frac{f(x^*, y^*)}{f(x^*, y^*)}.
\]
Thus we only need to show that \( f(x^*, y^*) \to \infty \) as \( T \to \infty \). Since \( f(x^*, y^*) \geq f(x^{LP}, y^{LP}) \), we only need to show \( f(x^{LP}, y^{LP}) \to \infty \) as \( T \to \infty \). For this purpose we consider a finite \( T \) and we rewrite the linear relaxation of (16)-(20) as follows:
\[
\begin{align*}
\min & \quad \sum_{n \in T} p_n \left( a_n^\top x_n + b_n^\top y_n \right) \\
\text{s.t.} & \quad \sum_{m \in P(n)} x_m - A_n y_n \geq 0 \quad \forall n \in T \\
& \quad B_n y_n \geq d_n \quad \forall n \in T \\
& \quad y_n \in \mathbb{R}_+^J \quad \forall n \in T \\
& \quad x_n \in \mathbb{R}_+^I \quad \forall n \in T.
\end{align*}
\]
The dual of the above problem is
\[
\begin{align*}
\max & \quad \sum_{n \in T} d_n^\top v_n \\
\text{s.t.} & \quad \sum_{m \in T(n)} u_n \leq p_n a_n \quad \forall n \in T \\
& \quad -A_n^\top u_n + B_n^\top v_n \leq p_n b_n \quad \forall n \in T \\
& \quad u_n \in \mathbb{R}_+^I \quad \forall n \in T \\
& \quad v_n \in \mathbb{R}_+^K \quad \forall n \in T.
\end{align*}
\]
Denote the objective function of the dual program by \( g(u, v) \) and so by weak duality \( f(x^{LP}, y^{LP}) \geq g(u, v) \) for any feasible dual solution \((u, v)\). We will try to find a feasible solution \((\tilde{u}, \tilde{v})\) such that \( g(\tilde{u}, \tilde{v}) \geq T\epsilon \) for some \( \epsilon > 0 \). First, we assign \( \tilde{u}_n = 0 \) for all \( n \in T \). From assumption (i) that \([d_n]_{k_n} \geq \epsilon_1\), and the constraint \( B_n y_n \geq d_n \), it follows that there exists at least one \( j_{kn} \) such that \([B_n]_{kn,j_{kn}} > 0\), otherwise the problem is not feasible (we can always make the problem feasible by adding a very expensive artificial resource that can satisfy all demand). Now assign \([\tilde{v}_n]_{k_n} = \frac{p_n [b_n]_{j_{kn}}}{[B_n]_{kn,j_{kn}}} \), and \([\tilde{v}_n]_k = 0 \) for all \( k \neq k_n \). The constructed solution \((\tilde{u}, \tilde{v})\) is clearly dual feasible. Furthermore, from assumption (ii), \([\tilde{v}_n]_{k_n} \geq p_n \epsilon_2\). Thus, \( f(x^{LP}, y^{LP}) \geq g(\tilde{u}, \tilde{v}) = \sum_{n \in T} d_n^\top \tilde{v}_n \geq \sum_{n \in T} p_n \epsilon_1 \epsilon_2 = T\epsilon_1 \epsilon_2 \), where the first inequality follows from weak duality and the last equality follows from the fact that \( \sum_{n \in S_t} p_n = 1 \) for all \( 1 \leq t \leq T \). Thus for any \( T \) we have \( f(x^{LP}, y^{LP}) \geq T\epsilon_1 \epsilon_2 \), and so \( f(x^{LP}, y^{LP}) \to \infty \) as \( T \to \infty \). This completes the proof. \( \square \)
Additional Computational Results

We report here some additional experiments to test the solution times of the proposed approximation algorithm (Algorithm 1) under the experimental set-up described in Section 6.

The results are summarized in Table 3. In this table, the first nine columns are number of stages \((T)\), number of branches \((B)\), number of nodes in the scenario tree \((N)\), numbers of columns, rows and integer variables for the multi-stage stochastic program, solution times for the linear relaxation of the multi-stage stochastic program \((M-LP)\), heuristic \((M-HR)\), and linear relaxation of the two-stage stochastic program \((T-LP)\), respectively. The last two columns are the RVMS lower bound and the RGAP upper bound. The linear programs are solved by CPLEX solver. In the experiment, all the instances are generated under demand pattern 4 (increasing mean, increasing standard deviation).

To see how large instances can be solved by the proposed approximation algorithm, we increase the size of the scenario tree in two ways. First, we fix the number of branches of each non-leaf node to be 2, and increase the number of stages. The largest instance we can solve has 6 stages, and 63 nodes in the scenario tree. There are totally 328,041 columns, 184,023 rows and 19,278 integer variables. The solution times of the linear relaxation of the multi-stage stochastic program, heuristic, linear relaxation of the two-stage stochastic program are 67.54, 420.29, and 34.12 seconds, respectively. When we increase the stage to 7, there are 661,289 columns and 370,967 rows. The heuristic fails at its first step to solve the linear relaxation of the multi-stage stochastic program. Then we fix the number of stages to be 3, and increase the number of branches of non-leaf nodes. In the largest instance we can solve, the number of branches of each non-leaf node is 11, and the scenario tree has 133 nodes. There are totally 692,531 columns, 388,493 rows and 40,698 integer variables. The solution times of the linear relaxation of the multi-stage stochastic program, heuristic, linear relaxation of the two-stage stochastic program are 160.36, 973.88 and 61.7 seconds, respectively. When the number of branches of each non-leaf node is 12, the heuristic fails at its first step to solve the linear relaxation of the multi-stage stochastic program. The failure of the heuristic comes from the difficulty of solving the linear relaxation problem. We also
include some graphs (see Figure 9). The first two graphs in Figure 9 depict heuristic solution times, and the second two graphs depict heuristic solution quality and value of multi-stage stochastic programming.

Table 3: Comparison of CPU times (in seconds)

<table>
<thead>
<tr>
<th>T</th>
<th>B</th>
<th>N</th>
<th>Col.</th>
<th>Row.</th>
<th>Int.</th>
<th>M-LP</th>
<th>M-HR</th>
<th>T-LP</th>
<th>RVMS(%)</th>
<th>RGAP(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>7</td>
<td>36,449</td>
<td>20,447</td>
<td>2,142</td>
<td>3.63</td>
<td>9.25</td>
<td>2.48</td>
<td>21.68</td>
<td>1.92</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>15</td>
<td>78,105</td>
<td>43,815</td>
<td>4,590</td>
<td>12.20</td>
<td>33.47</td>
<td>7.06</td>
<td>41.20</td>
<td>0.99</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>31</td>
<td>161,417</td>
<td>90,551</td>
<td>9,486</td>
<td>33.85</td>
<td>123.22</td>
<td>18.85</td>
<td>44.02</td>
<td>0.61</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>63</td>
<td>328,041</td>
<td>184,023</td>
<td>19,278</td>
<td>67.54</td>
<td>420.29</td>
<td>34.12</td>
<td>69.58</td>
<td>0.44</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>13</td>
<td>67,691</td>
<td>37,973</td>
<td>3,978</td>
<td>10.98</td>
<td>27.17</td>
<td>8.07</td>
<td>31.44</td>
<td>2.16</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>21</td>
<td>109,347</td>
<td>61,341</td>
<td>6,426</td>
<td>23.68</td>
<td>65.74</td>
<td>13.41</td>
<td>46.62</td>
<td>2.25</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>31</td>
<td>161,417</td>
<td>90,551</td>
<td>9,486</td>
<td>36.67</td>
<td>121.70</td>
<td>18.56</td>
<td>49.00</td>
<td>2.18</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>43</td>
<td>223,901</td>
<td>125,603</td>
<td>13,158</td>
<td>53.91</td>
<td>196.89</td>
<td>24.99</td>
<td>46.61</td>
<td>2.09</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>57</td>
<td>296,799</td>
<td>166,497</td>
<td>17,442</td>
<td>73.19</td>
<td>359.88</td>
<td>28.43</td>
<td>56.96</td>
<td>2.18</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>73</td>
<td>380,111</td>
<td>213,233</td>
<td>22,338</td>
<td>121.47</td>
<td>598.91</td>
<td>45.05</td>
<td>68.01</td>
<td>2.14</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>91</td>
<td>473,837</td>
<td>265,811</td>
<td>27,846</td>
<td>142.79</td>
<td>778.94</td>
<td>61.34</td>
<td>62.75</td>
<td>2.11</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>111</td>
<td>577,977</td>
<td>324,231</td>
<td>33,966</td>
<td>176.55</td>
<td>1087.94</td>
<td>74.02</td>
<td>54.90</td>
<td>2.06</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>133</td>
<td>692,531</td>
<td>388,493</td>
<td>40,698</td>
<td>160.36</td>
<td>973.88</td>
<td>61.7</td>
<td>65.39</td>
<td>2.21</td>
</tr>
</tbody>
</table>

The experimental results reported in Section 6 are based on the average of five instances. To test the variability of the results, we report here some additional experimental results. We generate instances under the demand pattern 4 (increasing mean, increasing standard deviation). We fix the number of branches to 2, and increase the number of stages from 2 to 5. The results are shown in Figure 10, where for each scenario tree size, the interval between the minimum and the maximum RGAP upper bound and RVMS lower bound over five instances is depicted. From these experimental results, we find that the standard deviation of five instances is quite high (not negligible), however, that does not influence our conclusions about RVMS and RGAP. Our conclusion that when the scenario tree size grows,
the value of multi-stage stochastic program becomes larger, and the solution quality of the approximation scheme becomes better, still stands.

Figure 9: Solution times and quality

Figure 10: Variability