A General Strategic Capacity Planning Model under Demand Uncertainty

Woonghee Tim Huh,[1] Robin O. Roundy,[2] Metin Çakanyildirim[3]

[1] *Department of Industrial Engineering and Operations Research, Columbia University, New York, New York 10027*

[2] *School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York 14853*

[3] *School of Management, University of Texas at Dallas, Richardson, Texas 75083*

Received 29 October 2003; revised 24 August 2005; accepted 30 September 2005
DOI 10.1002/nav.20128
Published online 12 December 2005 in Wiley InterScience (www.interscience.wiley.com).

Abstract: Capacity planning decisions affect a significant portion of future revenue. In equipment intensive industries, these decisions usually need to be made in the presence of both highly volatile demand and long capacity installation lead times. For a multiple product case, we present a continuous-time capacity planning model that addresses problems of realistic size and complexity found in current practice. Each product requires specific operations that can be performed by one or more tool groups. We consider a number of capacity allocation policies. We allow tool retirements in addition to purchases because the stochastic demand forecast for each product can be decreasing. We present a cluster-based heuristic algorithm that can incorporate both variance reduction techniques from the simulation literature and the principles of a generalized maximum flow algorithm from the network optimization literature. © 2005 Wiley Periodicals, Inc. Naval Research Logistics 53: 137–150, 2006

Keywords: capacity planning; stochastic demand; simulation; submodularity; semiconductor industry

1. INTRODUCTION

Because highly volatile demands and short product life cycles are commonplace in today's business environment, capacity investments are important strategic decisions for manufacturers. In the semiconductor industry, where the profit margins of products are steadily decreasing, manufacturers may spend up to 3.5 billion dollars for a state-of-the-art plant [3, 23]. The capacity decisions are complicated by volatile demands, rising costs, and evolving technologies, as well as long capacity procurement lead times. In this paper, we study the purchasing and retirement decisions of machines (or interchangeably, "tools"). The early purchase of tools often results in unnecessary capital spending, whereas tardy purchases lead to lost revenue, especially in the early stages of the product life cycle when profit margins are highest. The process of determining the sequence and timing of tool purchases and possibly retirements is referred to as *strategic capacity planning*.

Our strategic capacity planning model allows for multiple products under demand uncertainty. Demand evolves over time and is modeled by a set of scenarios with associated probabilities. We allow for the possibility of decreasing demand. Our model of capacity consumption is based on three layers: *tools* (i.e., machines), *operations*, and *products*. Each product requires a fixed, product-specific set of operations. Each operation can be performed on any tool. The time required depends on both the operation and the tool.

In our model time is a continuous variable, as opposed to the more traditional approach of using discrete time buckets. Our primary decision variables, one for each potential tool purchase or retirement, indicate the timing of the corresponding actions. In contrast, decision variables in typical discrete-time models are either binary or integer and are indexed by both tool groups and time periods. Our objective is to minimize the sum of the lost sales cost and the capital cost, each a function of tool purchase times and retirement times. Our continuous-time model has the advantage of having a smaller number of variables, although it may be difficult to find global optimal solutions for the resulting continuous optimization problem.

Many manufacturers, primarily those in high-tech industries, prefer to maintain a negligible amount of finished good inventory because technology products, especially highly profitable ones, face rapidly declining prices and a high risk of obsolescence. In particular, building up inventories ahead

Correspondence to: W.T. Huh (huh@ieor.columbia.edu)

of demand may not be economically sound for application-specific integrated circuits. Because high-tech products are in a sense "perishable," we assume no finished goods inventory. In addition, we assume that no back-ordering is permitted for the following reasons. First, unsatisfied demand frequently results in the loss of sales to a competitor. Second, delayed order fulfillment often results in either the decrease or the postponement of future demand. The end result approximates a lost sale. We remark that these assumptions of no-finished-goods and no back-ordering are also applicable to certain service industries and utility industries, in which systems do not have any buffer and require the co-presence of capacity and demand. These assumptions simplify the computation of instantaneous production and lost sales since they depend only on the current demand and capacity at a given moment of time.

In the case of multiple products, the aggregate capacity is divided among these products according to a particular policy. This tool-groups-to-products allocation is referred to as *tactical production planning*. While purchase and retirement decisions are made at the beginning of the planning horizon prior to the realization of stochastic demand, allocation decisions are recourse decisions made after demand uncertainty has been resolved. When demand exceeds supply, there are two plausible allocation policies for assigning the capacity to products: (i) the *Lost Sales Cost Minimization* policy minimizing instantaneous lost sales cost and (ii) the *Uniform Fill-Rate Production* policy equalizing the fill-rates of all products. Our model primarily uses the former, but can easily be extended to use the latter.

Our model is directly related to two threads of strategic capacity planning models, both of which address problems of realistic size and complexity arising in the semiconductor industry. The first thread is noted for the three-layer tool-operation-product model of capacity that we use, originating from IBM's discrete-time formulations. Bermon and Hood [6] assume deterministic demand, which is later extended by Barahona et al. [4] to model scenario-based demand uncertainty. Barahona et al. [4] have a large number of indicator variables for discrete expansion decisions, which results in a large mixed integer programming (MIP) formulation. Standard MIP computational methods such as branch-and-bound are used to solve this challenging problem. Our model differs from this work in the following ways: (i) using continuous variables, we use a descent-based heuristic algorithm as an alternative to the standard MIP techniques, (ii) we model tool retirement in addition to acquisition, and (iii) we consider the capital cost in the objective function instead of using the budget constraint. Other notable examples of scenario-based models with binary decisions variables include Escudero et al. [15], Chen, Li, and Tirupati [11], Swaminathan [27], and Ahmed and Sahinidis [1]; however, they do not model the operations layer explicitly.

The second thread of the relevant literature features continuous-time models. Çakanyildirim and Roundy [8] and Çakanyildirim, Roundy, and Wood [9] both study capacity planning for several tool groups for the stochastic demand of a single product. The former establishes the optimality of a bottleneck policy where tools from the bottleneck tool group are installed during expansions and retired during contractions in the reverse order. The latter uses this policy to jointly optimize tool expansions along with nested floor and space expansions. Huh and Roundu [18] extend these ideas to a multi-product case under the Uniform Fill-Rate Production policy and identify a set of sufficient conditions for the capacity planning problem to be reduced to a non-linear convex minimization program. This paper extends their model by introducing the layer of operations, the Lost Sales Cost Minimization allocation policy and tool retirement. This results in the non-convexity of the resulting formulation. Thus, our model marries the continuous-time paradigm with the complexity of real-world capacity planning.

We list a selection of recent papers on capacity planning. Davis et al. [12] and Anderson [2] take an optimal control theory approach, where the control decisions are expansion rate and workforce capacity, respectively. Ryan [24] incorporates autocorrelated product demands with drift into capacity expansion. Ryan [25] minimizes capacity expansion costs using option pricing formulas to estimate shortages. Also, Birge [7] uses option theory to study capacity shortages and risk. An extensive survey of capacity planning models is found in the article by Van Mieghem [28].

Our computational results suggest that the descent algorithm, with a proper initialization method, delivers good solutions and reasonable computation times. Furthermore, preliminary computational results indicate that capacity plans are not very sensitive to the choice of allocation policy, and both policies perform comparably. With the Uniform Fill-Rate Production policy, an instantaneous revenue calculation that is used repeatedly by the subroutines of the heuristic algorithm can be formulated as a generalized maximum flow problem; the solution of this problem can be obtained by a combinatorial polynomial-time approximation scheme that results in a potentially dramatic increase in the speed of our algorithm.

We assume that the stochastic demand is given as a finite set of scenarios. This demand model is consistent with current practice in the semiconductor industry. We also explore, in Section 5, the possibility that demand is instead given as a continuous distribution, e.g., the Semiconductor Demand Forecast Accuracy Model [10]. Borrowing results from the literature on Monte Carlo approximations of stochastic programs, we point out the existence of an inherent bias in the optimal cost of the approximation when the scenario sample size is small. We also describe applicable variance reduction techniques when samples are drawn on an *ad hoc* basis.

This paper is organized as follows. Section 2 lays out our strategic capacity formulation under two capacity allocation policies. Section 3 describes our heuristic algorithm, and its computational results are reported in Section 4. Section 5 presents how our software can be efficiently used when the demand is a set of continuous distributions that evolve over time. We briefly conclude with Section 6.

2. MODEL

2.1. Formulation

Let the continuous variable t represent a time between 0 and T, the end of the planning horizon. We use p, w, and m to index product families in $\mathcal{P}$, operations in $\mathcal{W}$, and tool groups in $\mathcal{M}$, respectively. All tools in a tool group are identical; this is how tool groups are actually defined. We denote by $M(w)$ the set of tools that can perform operation w and by $W(m)$ the set of operations that tool group m can perform. During the planning horizon, we purchase N_m^P (retire N_m^R) tools belonging to tool group m.[1] Purchases or retirements of tools in a tool group are indexed by n, $1 \le n \le N_m^P$, or $1 \le n \le N_m^R$. Random demand for product p is given by $D_p(t) = D_{s,p}(t)$, where s indexes a finite number of scenarios $\mathcal{S}$.

Our formulation uses input data and variables presented below. We reserve the usage of the word *time* for the calendar time t, as opposed to the processing duration of operations or productive tool capacities available. To avoid confusion, we refer to the duration of operations or tool capacities available at a given moment of time using the phrase *machine-hours*.

Input Data

- $b_{w,p}$ Number of operations of type w required to produce a unit of product p (typically integer, but fractional values are allowed).
- $h_{m,w}$ Amount of machine-hours required by a tool in group m to perform operation w.
- u_m^o Total capacity (productive machine-hours per month) of tool group m at the beginning of the time horizon.
- u_m^1 Capacity of each tool in group m (productive machine-hours per month).
- $P_m^P(t)$ Purchase price of a tool in group m at time t (a function of the continuous scalar t).
- $P_m^R(t)$ Sale price for retiring a tool in group m at time t. May be positive or negative.
- $c_p(t)$ Per-unit lost sales cost for product p at time t.
- $D_{s,p}(t)$ Instantaneous demand of product p in scenario s at time t.
- π_s Probability of scenario s.

[1] In our model, we fix *a priori* the number of tool purchases and retirements for each tool group. This is consistent with practice in the semiconductor industry. When these numbers are unknown (e.g., specified by the minimum and maximum quantities), our model can be a lower part of a two-level decision model, which determines the optimal choice of N_m^P and N_m^R either by exhaustion or heuristically.

We eliminate subscripts to construct vectors or matrices by listing the argument with different products p, operations w, and/or tool indices m. For example, $B := (b_{w,p})$ is the production-to-operation matrix and $H := (h_{m,w})$ is the machine-hours-per-operation matrix. Note that we concatenate only p, w, or m indices. Thus, $D_s(t) = (D_{s,p}(t))$ for demand in scenario s, and $c(t) = (c_p(t))$ for per-unit lost sales cost vectors at time t. We assume the continuity of c_p and $D_{s,p}$ and the continuous differentiability of P_m^P and P_m^R.

Primary Variables

- $\tau_{m,n}$ The time of the nth tool purchase within group m.
- $\gamma_{m,n}$ The time of the nth tool retirement within group m.

Auxiliary Variables

- $X_{s,w,m}(t)$ Number of products that pass through operation w on tool group m in scenario s at time t.
- $U_m(t)$ Capacity of tool group m at time t.
- $V_{s,p}(t)$ Unmet demand of product p in scenario s at time t.
- $\overline{V}_{s,p}(t)$ Satisfied demand of product p in scenario s at time t. Thus, $\overline{V}_{s,t}(t) = D_{s,p}(t) - V_{s,p}(t)$.

The decision variables of primary interest are the times $\tau_{m,n}$ and $\gamma_{m,n}$ of tool purchases and retirements. All the other variables are consequences of these two sets of variables.

Capacity Planning Formulation

$$\min \int_{t=0}^{T} \sum_s \pi_s \sum_p c_p(t) V_{s,p}(t)\, dt + \sum_m \left[\sum_{n=1}^{N_m^P} P_m^P(\tau_{m,n}) - \sum_{n=1}^{N_m^R} P_m^R(\gamma_{m,n}) \right] \quad (1)$$

$$\text{subject to } V_{s,p}(t) + \overline{V}_{s,p}(t) = D_{s,p}(t) \quad \forall s, p, t \quad (2)$$

$$\sum_p b_{w,p} \overline{V}_{s,p}(t) = \sum_{m \in M(w)} X_{s,w,m}(t) \quad \forall s, w, t \quad (3)$$

$$\sum_{w \in W(m)} h_{m,w} X_{s,w,m}(t) \le U_m(t) \forall s, m, t \quad (4)$$

$$u_m^o + u_m^1 [|\{n : \tau_{m,n} \le t\}| - |\{n : \gamma_{m,n} \le t\}|] = U_m(t) \quad \forall m, t \quad (5)$$

$$0 \le \tau_{m,1} \le \tau_{m,2} \le \cdots \le \tau_{m,N_m^P} \le T \quad \forall m \tag{6}$$

$$0 \le \gamma_{m,1} \le \gamma_{m,2} \le \cdots \le \gamma_{m,N_m^R} \le T \quad \forall m \tag{7}$$

$$U_m(t), V_{s,p}(t), \overline{V}_{s,p}(t), X_{s,w,m}(t) \ge 0 \quad \forall s, w, m, p, t. \tag{8}$$

The objective function (1) minimizes the sum of the expected lost sales cost and the capital cost, where capital cost is the tool purchase cost minus the tool sale price upon retirement. Note that the capital cost is a separable function of the $\tau_{m,n}$'s and $\gamma_{m,n}$'s. Constraint (2) defines unmet demand in terms of realized demand and satisfied demand. Constraint (3) forces enough tool capacity to be allocated to meet the satisfied demand for each product family. Constraint (4) indicates that the allocation of a tool's capacity across different product families is subject to the capacity (machine-hours) of the tool group. The capacity of a tool group is given in (5). Constraints (6) and (7) define the order of purchases and retirements within a tool group. We can easily accommodate additional timing constraints, such as $\tau_{1,3} \le \gamma_{1,3}$. Non-negativity constraints on $U_m(t)$ in (8) assure that only existing machines can be retired.

2.2. Remarks

We make several remarks about the formulation.

1. *The Lost Sales and No Inventory Holding*. To support strategic capacity planning decisions, these simplifying assumptions are made. These assumptions are justifiable in many industries that face a high risk of obsolescence, rapidly decreasing prices, and demand volatility. They are used, for example, in the semiconductor industry [6].
2. *Capacity Allocation Policy*. Tool purchase and retirement decisions are made in the presence of demand uncertainty at the beginning of the planning horizon. These decisions determine the set of tools available at any time t in the planning horizon. Tool capacity is allocated to operations after the demand has been observed, at each time t. Insufficient capacity to satisfy the realized demand results in lost sales. Because of the lost sales and no inventory assumptions, the capacity allocation depends only on realized demand and the set of available tools at a given time.

   Our capacity allocation policy, the *Lost Sales Cost Minimization* policy, minimizes the instantaneous lost sales cost at t. At time t, for a given per-unit lost sales cost vector $c := (c_p(t))$, tool capacity vector $U := (U_m(t))$, and realized demand vector $d := (d_p(t))$, the unmet demand $V := (V_p(t))$ is an optimal solution to the following linear program (LP).

Capacity Allocation LP with the Lost Sales Cost Minimization Policy

$$LS(c, U, D) := \min \sum_p c_p V_p \tag{9}$$

$$\text{s. t. } V_p + \overline{V}_p = d_p \quad \forall p \tag{10}$$

$$\sum_p b_{w,p} \overline{V}_p = \sum_{m \in M(w)} X_{w,m} \quad \forall w \tag{11}$$

$$\sum_{w \in W(m)} h_{m,w} X_{w,m} \le U_m \quad \forall m \tag{12}$$

$$V_p, \overline{V}_p, X_{w,m} \ge 0 \quad \forall w, m, p. \tag{13}$$

As time t changes, the Capacity Allocation LP problem with parameters $c(t)$, $U(t)$, and $D_s(t) := (D_{s,p}(t))$ becomes a parametric linear program in which several parameters change simultaneously. A closed form expression for the expected lost sales cost in terms of available capacities for all t cannot generally be obtained. Although the total expected lost sales cost over the planning horizon is difficult to obtain, the expected lost sales cost $\sum_s \pi_s LS(c(t), U(t), D_s(t))$ at time t can be easily computed by solving a linear program for each scenario s.

For fixed (τ, γ), define the lost sales cost and capital cost as

$$F^{LS}(\tau, \gamma) := \int_{t=0}^{T} \sum_s \pi_s LS(c(t), U(t), D_s(t))\, dt,$$

$$F^C(\tau, \gamma) := \sum_m \left[\sum_{n=1}^{N_m^P} P_m^P(\tau_{m,n}) - \sum_{n=1}^{N_m^R} P_m^R(\gamma_{m,n}) \right].$$

Now, the Capacity Planning Formulation in (1)–(8) is equivalent to minimizing

$$F(\tau, \gamma) := F^{LS}(\tau, \gamma) + F^C(\tau, \gamma)$$

over (τ, γ) subject to constraints (6) and (7). We will see that while it is difficult to obtain F^{LS}, we can easily compute its partial derivatives.

3. *Lost Sales Minimization and Revenue Maximization*. When capacities are fixed, the minimization of lost sales in the Capacity Allocation LP (9)–(13) is equivalent to the instantaneous maximization of revenue,

$$R(c, U, D) := \text{Maximize } \sum_p c_p \overline{V}_p \tag{14}$$

   subject to constraints (10)–(13). We name $R(c, U, D)$ as the instantaneous revenue.

4. *Non-Convexity Properties*. One of our main goals is to find the boundary between the versions of the capacity planning problem that are provably solvable and those that are not. The boundary is drawn by the Convexity Conditions summarized in Section 2.3. Huh and Roundy [18] show the convexity of the objective function $F(\tau, \gamma)$ under these conditions and present an efficient algorithm. In contrast, our capacity planning problem is not convex. For such a problem, we develop a heuristic solution technique.

2.3. Convexity Conditions

In this section, we list the *Convexity Conditions* and summarize their implications. These conditions are sufficient for the joint convexity of F.

A. Demand $D_{s,p}(t)$ is non-decreasing with respect to time t. Only tool acquisitions are considered, i.e., the number N_m^R of possible tool retirements is zero.
B. The cost $P_m^P(t)$ of purchasing tool m is convex in time t, and the per-unit lost sales cost $c_p(t)$ is non-decreasing in t.
C. The product mix remains constant in each scenario s; i.e., $D_{s,p}(t) = \Delta_s(t) \cdot \Phi_s$ where $\Delta_s(t)$ is a scalar and Φ_s is a deterministic product mix vector.
D. Each operation is performed by a unique tool group; i.e., the machine-hours-per-operation matrix $H = (h_{m,w})$ is a square matrix, and all non-diagonal entries are infinite.
E. For capacity allocation, use the *Uniform Fill-Rate Production* policy (see [18]). It ensures that the fill-rates of all products are the same. It can easily be accommodated in our formulation by adding the following constraints:

$$\overline{V}_{s,p}(t) = \sigma_s(t) \cdot D_{s,p}(t) \ \forall s, t, \tag{15}$$

$$0 \le \sigma_s(t) \le 1 \ \forall s, t, \tag{16}$$

where variable $\sigma_s(t)$ is the common fill-rate of all product families in scenario s at time t.

We can show that without any one of the above five Convexity Conditions, $F(\tau, \gamma)$ may not be convex. To appreciate the necessity of Condition A, suppose demand is deterministic and that there is only one product family, one tool group, and one operation type. If demand is non-decreasing over time, then the marginal increase in the lost sales cost associated with a tool is non-decreasing in the purchase time of the tool, i.e., the lost sales cost is convex. However, if demand decreases, a similar reasoning shows that the lost sales cost can be concave in the tool purchase times. Therefore, the violation of Condition A leads to non-convexity. An analogous argument with constant demand shows that Condition B is necessary for convexity. Examples illustrating the necessity of Conditions C and D can be constructed.

Now we turn our attention to Condition E. The example below shows that the convexity of the expected lost sales cost does not necessarily hold if the Lost Sales Cost Minimization allocation policy is used in place of Condition E.

EXAMPLE 1: Suppose that there are two product families, whose deterministic demand is 6 units each at time $t = 1$. Their per-unit lost sales cost is 1000 dollars each. The first product family requires $b_{11} = 20$ units of the first operation and $b_{31} = 20$ of the third operation. The second product family requires $b_{22} = 20$ units of the second operation and $b_{32} = 20$ units of the third operation. Each operation can be performed only by the corresponding dedicated tool group and requires 20 units of processing. Let $u^o = (1200, 1200, 3600)$ and $(u_1^1, u_2^1, u_3^1) = (1200, 1200, 3600)$. Suppose that the current solution is $\tau^o = (\tau_{11}^o, \tau_{21}^o, \tau_{31}^o) = (1, 1, 2)$. It can be verified that the directional derivative of the lost sales cost is $\nabla_{(-1,0,0)} F^{LS}(\tau^o) = \nabla_{(0,-1,0)} F^{LS}(\tau^o) = \nabla_{(-1,-1,0)} F^{LS}(\tau^o) = 6000 - 9000 = -3000$ dollars along $(-1, 0, 0)$, $(0, -1, 0)$, and $(-1, -1, 0)$. (Each of the directional derivatives corresponds to the change in production from point A in Fig. 1 to point B, C or D). It follows that the lost sales cost is not jointly convex in τ_{11} and τ_{21} at τ^o.

Implications of the Uniform Fill-Rate Production Policy. For capacity allocation, both lost sales minimization and uniform fill-rate policies exist in the capacity planning literature, and variants of both are used in practice by semiconductor companies. Under the Uniform Fill-Rate Production Policy, the instantaneous revenue maximization problem is to optimize objective (14) subject to (10)–(13) and (15) and (16). Under Convexity Conditions C and D, this computation

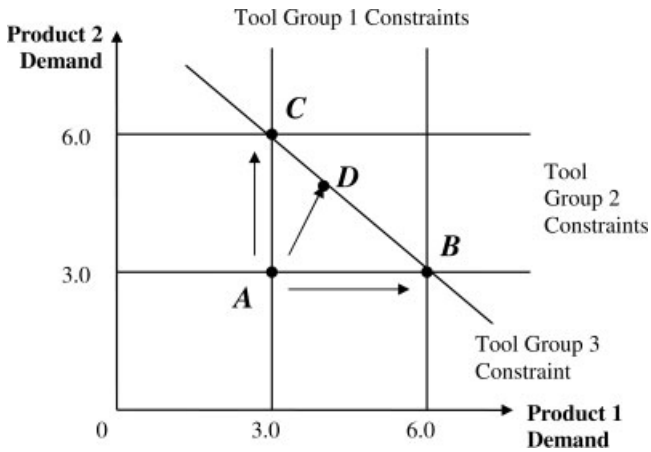

Figure 1. Instantaneous production at time $t = 1$ when τ_{11} and τ_{21} are perturbed as in Example 1.

is straightforward and is equivalent to finding a bottleneck tool constraint. Without the Convexity Conditions, the proposition below shows that the instant revenue maximization problem can be solved by a fully combinatorial-time algorithm. This is significant for efficiency because the bulk of the running time of our heuristic algorithm in Section 3 is spent on solving linear programs, in particular maximizing instantaneous revenue computation. The proof is based on the reduction of the instantaneous revenue maximization to a network problem called the generalized maximum concurrent flow problem.

PROPOSITION 1: Suppose that finite entries in the tool requirement matrix H form a sparse matrix, i.e., the number of finite entries is linear in ω, where $\omega = \max\{|\mathcal{W}|, |\mathcal{M}|\}$. Assume the Uniform Fill-Rate Production policy. For any $\epsilon > 0$, a fully polynomial-time algorithm finds a feasible solution of the instantaneous revenue maximization problem such that the objective value of the feasible solution is at least $(1 - \epsilon)$ times the optimal value. This algorithm runs in $O(\epsilon^{-2}\omega^2 \log \omega)$ time.

3. SOLUTION APPROACH

This section presents a heuristic algorithm to solve the Capacity Planning Formulation presented in Section 2. Our algorithm is a modification of the divide-and-conquer method by Huh and Roundy [18].

In our algorithm, all potential tool purchases and retirements are partitioned into clusters based upon when they occur; all purchases and retirements in a given cluster are made at the same time. At each iteration, we perform one of the three following subroutines: (i) *cluster descent*: find an optimal time for a cluster within a specific interval; (ii) *cluster splitting*: divide a cluster into two clusters; and (iii) *cluster merging*: combine two clusters with the same associated time into one cluster.

Cluster Structure. Our algorithm is based on the concept of clusters. Given the values of the primary decision variables $\Omega = \{\tau_{m,n} : m \in \mathcal{M}, 1 \le n \le N_m^P\} \cup \{\gamma_{m,n} : m \in \mathcal{M}, 1 \le n \le N_m^R\}$, the associated *cluster structure* $\mathcal{C}$ is an ordered partition of Ω such that two variables belong to the same cluster of the partition if they have the same values. Such a set is called a *cluster*. All variables in a cluster C assume the common *value* t_C. The order of the partition is consistent with the order of the values of its clusters, i.e., if $\mathcal{C} = (C_1, \ldots, C_k)$, then $t_{C_1} \le t_{C_2} \le \cdots \le t_{C_k}$. Note that not all primary variables with the same value need to belong to the same cluster, i.e., there can be more than one cluster with the same value.

Description of the Algorithm. Given an initial feasible cluster structure, the algorithm randomly selects one of the clusters and attempts to perform one of the three subroutines. It is a descent algorithm that repeatedly improves the value of the solution in each iteration. The algorithm terminates when no subroutine modifies the value or structure of any of the clusters.

We describe below each subroutine in detail. These subroutines take an advantage of the following observation by Huh and Roundy [18]. While the objective function F is not separable, it is *quasi-separable*, i.e., its partial derivative with respect to a variable $\tau_{m,n}$ depends on the value of other variables only through the set consisting of all variables that are less than $\tau_{m,n}$. The same can be said for a variable $\gamma_{m,n}$.

(i) *Cluster Descent.* This subroutine finds a good value of a given cluster $C \in \mathcal{C}$ in the interval defined by the values of the previous and subsequent clusters. While the objective function of this problem is not convex, it is a single-dimensional global optimization problem with bounds. Essentially, this subroutine looks for a better solution within a line segment. Although it is easier to solve the single-dimensional (non-convex) global optimization problem compared to the multiple-dimensional case, there is no straightforward algorithm to find the global minimizer. Consequently, the cluster descent subroutine finds a *local* minimizer in our heuristic algorithm.

In our problem, it is non-trivial to evaluate $F(\tau, \gamma)$ as discussed in Section 2, but we can obtain the directional derivative of $F(\tau, \gamma)$ in the interior of the interval, where the derivative is taken along the direction of the characteristic vector of χ_C. (The entry of this binary vector χ_C corresponding to a decision variable is 1 if and only if this variable belongs to C.) Such a direction corresponds to simultaneously moving all the variables in the cluster. The subroutine uses the first-order condition of local optimality and seeks to find a time t in the interval such that the directional derivative changes its sign from non-positive to non-negative at time t. The computational experiment in Section 4 uses MATLAB's `fzero` function. If no such t exists, then it reports an appropriate end point.

(ii) *Cluster Splitting.* This subroutine subdivides a given cluster $C \in \mathcal{C}$ into two clusters $C^- \cup C^+$, where C^- precedes C^+. We say C^- is sent to the left of the current value and C^+ is sent to the right. Intuitively, this subroutine looks for a descent direction within a subset of all possible descent directions. Let t_C be the value of cluster C. The cluster splitting $C^- \cup C^+$ of C is optimal if the directional derivative along the direction $-\chi_{C^-} + \chi_{C^+}$ at t_C is minimized. It can be shown that it is equivalent to minimizing the directional derivative along $-\chi_{C^-}$ instead of along $-\chi_{C^-} + \chi_{C^+}$. We describe an integer

programming formulation for the optimal cluster splitting problem. Let $C_L \subseteq \Omega$ be the union of all clusters preceding C in the cluster structure. Let $\eta(C_L)$ denote a vector that is indexed by tool groups and represents the net number of tool purchases corresponding to C_L, i.e., the mth entry of $\eta(C_L)$ is $|\{n : \tau_{m,n} \in C_L\}| - |\{n : \gamma_{m,n} \in C_L\}|$. Recall that $LS(c(t), U(t), D_s(t))$ represents the instantaneous lost sales cost at time t for a given set of parameters under demand scenario s. We also use the notation $LS(c(t), \eta(C_L), D_s(t))$ when there is no ambiguity. Let y^P and y^R denote the number of purchases and the number retirements in the cluster C that should be perturbed to the left from t_C. Then, the optimal splitting is the solution of the following problem.

Cluster Splitting IP

$$\begin{aligned}
\text{Min} \quad & \sum_s \pi_s LS(c(t), \bar{n}(C_L) + y^P - y^R, t_C) \\
& - \sum_m \left[y_m^P \frac{d}{dt} P_m^P(t_C) - y_m^R \frac{d}{dt} P_m^R(t_C) \right] \\
\text{s. t.} \quad & 0 \le y_m^P \le |\{n : \tau_{m,n} \in C\}| \ \ \forall m \\
& 0 \le y_m^R \le |\{n : \gamma_{m,n} \in C\}| \ \ \forall m \\
& y_m^P, y_m^R \quad \text{integer } \forall m.
\end{aligned}$$

We note that the Cluster Splitting IP is a single-period capacity planning problem. When the size of the cluster C is big, there is no efficient method known to solve the above IP in general. In the paper by Barahona et al. [4], a mixed integer capacity planning problem is approached using a truncated branch-and-bound procedure. In this paper, we propose and use a *repeated probabilistic rounding procedure*. We solve the linear programming relaxation of the IP. Each fractional variable of the LP relaxation solution is probabilistically rounded up or down based on the fractional value. We then compute the value of the objective function of the IP corresponding to the rounded solution. We repeat this process several times and select the best rounded solution.

Both the truncated branch-and-bound procedure and the repeated probabilistic rounding procedure typically fail to efficiently produce the optimal solution to cluster splitting. However, both deliver a reasonably good solution, with the quality of solution being dependent upon computational effort. The branch-and-bound procedure also gives bounds on the optimal value, but the repeated probabilistic rounding procedure often produces a reasonable solution faster. For our application, obtaining the best possible splitting is not always crucial. In the heuristic method we present below, a suboptimal decision in placing certain purchases or retirements in the wrong cluster can later be detected and corrected. Our heuristic method uses the repeated probabilistic rounding procedure but can easily incorporate a branch-and-bound procedure.

(iii) *Cluster Merging.* When two adjacent clusters have the same value, we can replace these two clusters with a combined cluster. Combining two clusters, followed by a cluster split, allows for the possibility that the variables in those clusters will be reordered.

Comparison to Huh and Roundy [18]. The capacity planning model by Huh and Roundy [18] assumes the five Convexity Conditions, which ensure the joint convexity of F. Thus, there is a globally optimal solution, which is found by their algorithm. Their algorithm differs from ours in the following ways. *Cluster Descent*: The convexity of F implies that the single-dimensional global optimization problem in this subroutine is a convex program. Therefore, the first-order condition is sufficient for optimality. *Cluster Splitting*: The cluster splitting problem is a submodular function minimization problem, which is solvable in strongly polynomial time. *Cluster Merging*: There is no cluster merging subroutine. Their algorithm starts with one big cluster and progressively splits it into smaller clusters.

Our heuristic algorithm is an adaptation of their algorithm to a more general problem. The following theorem shows that this algorithm terminates with a locally optimal solution assuming the optimality of the cluster split subroutine. The proof is omitted since it takes a similar approach to the convexity case proof by Huh and Roundy [18].

THEOREM 2: If the cluster structure cannot be modified by any of the three subroutines, then the corresponding capacity planning solution (τ, γ) is a locally optimal solution for the capacity planning problem, for some neighborhood defined by any norm in the $\Re^\Omega$ space.

4. COMPUTATION

This section presents computational results. We describe the implementation environment, data set, and initialization methods, followed by a discussion on the outcome of experiments.

Implementation Environment. We implement our algorithm using the MATLAB 6.5 language. As seen in Section 3, we frequently need to solve linear programs as part of both

the cluster descent and the cluster split subroutines. We do not use MATLAB's own function to solve linear programs; instead, we use CPLEX 6.6. Experience indicates that for large linear programming problems, CPLEX runs significantly faster than MATLAB. Warm-start strategies, starting a linear program with a good initial solution, perform better in CPLEX than in MATLAB. We use a MATLAB interface to the CPLEX Callable Library called MatCPX, developed by Nathan Edwards [14]. Our experiments were conducted on a Sun 420R Server, running the Solaris 8 operating system.

Data. The data set is obtained from two sources. The tool data comes from the Sematech (Semiconductor Manufacturing Technology) database, similarly used by Çakanyildirim and Roundy [8]. We consider 155 tool groups, totaling 3100 potential purchases and the same number of potential retirements. The number of operations is 714. The demand forecast data is modeled using SeDFAM [10] and is based on an industrial data set provided by a large U.S. semiconductor manufacturer. For the experiments in this section, we use a fixed set of four scenarios with equal probability. The planning horizon is 16 quarters, during which the total demand is expected to grow from 16 thousand wafer-starts to 45 thousand wafer-starts per quarter. Ten product families go through various stages of the product life cycle.

Parameters. Two parameters define the precision attained in subroutines. The first is the *minimum number of splits*. It is used in the cluster split subroutine and is the minimum number of probabilistic roundings used to select the best subdivision of a cluster. If the subroutine fails to split a cluster, it will continue with more roundings, up to five times this number or until a split is found. The second parameter is the *descent tolerance*, which is the termination tolerance on the magnitude of the directional derivative in the cluster descent subroutine. We choose the minimum number of splits from $\{25, 50, 100\}$ and the descent tolerance from $\{0.1, 0.01, 0.001\}$.

Initialization. There are several methods to initialize purchase times $\tau = (\tau_{m,n})$. *Serial intialization* is based on a heuristic used by Barahona et al [4]. We select a set of discrete times in the planning horizon, say $\{1.0, 2.0, \ldots, \lfloor T \rfloor\}$, and set all $\tau_{m,n}$'s to the least of this set. We choose t from the set in increasing order starting with $t = 1.0$ and split the cluster whose value is t, obtaining up to two subdivided clusters. The minimum number of splits during the initialization is fixed at 100. The left cluster, if it exists, remains at t, and the right cluster, if it exists, is moved to the next t in the set. We repeat this process with the next t in the set.

There are a couple of alternative ways to initialize. One method is to randomly generate N_m^P number of points in the planning horizon, sort them, and assign them to $\{\tau_{m,n} | n = 1, 2, \ldots, N_m^P\}$ for each tool group m. This is called a *random initialization*. Another method is to start with one cluster; i.e., for all pairs (m, n), set $\tau_{m,n} = t_o$ for some $t_o \in [0, T]$. We refer to this as a *big-cluster initialization*. This is the initialization used in the divide-and-conquer method of Huh and Roundy [18].

Cost Estimation. The evaluation of the solution produced by the algorithm requires some explanation. Section 2 shows the difficulty of precisely computing the expected lost sales cost $F^{LS}(\tau, \gamma)$ since its integrand is the optimal value of a linear program parameterized by the variable of integration. The objective function (1) of the Capacity Planning Formulation is the sum of the expected lost sales cost and the capital cost. For the remainder of this paper, the reported cost values are obtained using an approximation of the expected lost sales cost—replace integration with a summation having a fixed time increment of 0.2. As a result, the reported cost values are not exact.

Results. The experimental results are summarized in Tables 1, 2, and 3, each corresponding to a different initialization method. The objective values of the algorithm's outputs are shown under the heading "Cost." Under "CPU Hours" we report the average CPU running time per replication in hours. The CPLEX running time includes the time spent by both the linear programming solver of CPLEX and the interface between MATLAB and CPLEX. (There is no direct application program interface in the MATLAB or C language for retrieving the elapsed time by CPLEX only.) The initialization running time reported is for serial initialization. The running times for CPLEX and initialization are not mutually exclusive because serial initialization calls CPLEX.

Table 1 is the experimental outcome using serial initialization. For each parameter setting we test the algorithm three times, labeled "Best," "Median," and "Worst," each with a distinct randomized initial solution. Suboptimal solutions arise from either the existence of multiple local minima or the algorithm's inability to detect a descent direction during the cluster split subroutine. The best performance of 3 runs for each parameter setting is at most 0.021 billion more than 4.222 billion dollars, the minimum objective value known for this problem. We also note that initialization is fast, and the quality of the initial solution is consistent and good, about 0.2 billion dollars greater than the best known solution. The majority of the running time is spent by the linear program solver and the MATLAB–CPLEX interface.

Tables 2 and 3 are based on random initialization and big-cluster initialization, respectively. The best performance for each parameter setting is often within 0.1 billion of 4.222 billion dollars. Increasing the minimum number of splits and decreasing the descent tolerance produces a better solution at the expense of increased computation. The improvement in the quality of the solution is significant when the descent

Table 1. Experimental results: Serial initialization.

Min No. Split	Descent Tol.	Cost (10^9) Best Init.	Best Final	Median Init.	Median Final	Worst Init.	Worst Final	CPU Hours per Repl. CPLEX	CPU Hours per Repl. Init.	CPU Hours per Repl. Overall
25	0.1	4.343	4.238	4.341	4.243	4.342	4.249	0.250	0.101	0.268
25	0.01	4.342	4.223	4.341	4.225	4.352	4.226	0.675	0.100	0.715
25	0.001	4.344	4.223	4.341	4.224	4.343	4.225	1.067	0.102	1.258
50	0.1	4.347	4.243	4.340	4.244	4.343	4.247	0.281	0.101	0.312
50	0.01	4.344	4.222	4.342	4.224	4.343	4.226	0.833	0.102	0.978
50	0.001	4.349	4.224	4.340	4.224	4.352	4.225	1.516	0.101	1.721
100	0.1	4.339	4.237	4.344	4.239	4.339	4.244	0.423	0.100	0.463
100	0.01	4.339	4.224	4.338	4.225	4.341	4.225	1.114	0.100	1.372
100	0.001	4.342	4.222	4.338	4.225	4.339	4.225	1.555	0.101	1.853

Note. Each row corresponds to three replications.

tolerance is changed from 0.1 to 0.01, but marginal from 0.01 to 0.001. This can possibly be explained by the gap size of 0.2 in the evaluation of the objective function. Repeatedly running the algorithm with either of the initializations and reporting the best solution will yield reasonably good solutions. Both alternative initializations were unable to find solutions better than those found by the serial initialization, suggesting that the algorithm with serial initialization is unlikely to terminate with a bad solution.

The biggest limitation of our computational work is clearly the fact that it is based on a single industrial data set. Real-world forecast and demand data are politically very sensitive, and we have not been successful in obtaining another large data set. In the future we hope that computational results of our algorithms based on other industrial data sets will emerge.

Capacity Allocation Policy. The capacity allocation policy has a significant impact on running time. Does it also have a significant impact on performance? In particular, how good are the tool purchase and retirement times obtained using the Uniform Fill-Rate Production Policy when evaluated using Lost Sales Cost Minimization? In Table 4 we study this question, using serial initialization. We note that increasing computational effort, as controlled by parameter settings, does not improve solution quality. Solution costs are slightly higher than those reported in Table 1, but the difference is only a few percentage points. However, CPU time is significantly lower. This experiment suggests that when we are interested in the Lost Sales Cost Minimization, the Uniform Fill-Rate Production policy yields reasonably good solutions and reduces computational effort. A practical application of the capacity planning model includes order acceptance and lead-time quotation [13]. For such purposes, short computational running times are preferred, and one may prefer the Uniform Fill-Rate Production policy in cost-minimization settings.

Table 2. Experimental results: Random initialization.

Min No. Split	Descent Tol.	Cost (10^9) Best	Cost (10^9) Median	Cost (10^9) Worst	CPU Hours per Repl. CPLEX	CPU Hours per Repl. Overall
25	0.1	4.286	4.613	4.753	1.039	1.516
25	0.01	4.261	4.281	4.513	1.459	2.034
25	0.001	4.254	4.263	4.266	1.707	2.348
50	0.1	4.335	4.489	4.551	0.984	1.469
50	0.01	4.265	4.271	4.282	1.395	1.979
50	0.001	4.296	4.441	4.750	1.732	2.384
100	0.1	4.298	4.687	4.693	1.061	1.566
100	0.01	4.250	4.253	4.270	1.876	2.606
100	0.001	4.266	4.528	5.009	2.347	3.229

Note. Each row corresponds to three replications.

Table 3. Experimental results: Big-cluster initialization.

Min No. Split	Descent Tol.	Cost ($10^9)			CPU Hours per Repl.	
		Best	Median	Worst	CPLEX	Overall
25	0.1	4.356	4.382	4.398	0.147	0.154
25	0.01	4.254	4.255	4.255	0.542	0.631
25	0.001	4.252	4.256	4.259	0.981	1.165
50	0.1	4.298	4.363	4.365	0.208	0.226
50	0.01	4.251	4.253	4.256	0.697	0.825
50	0.001	4.252	4.258	4.258	1.351	1.639
100	0.1	4.270	4.351	4.364	0.296	0.334
100	0.01	4.248	4.252	4.259	1.136	1.399
100	0.001	4.251	4.251	4.255	1.738	2.219

Note. Each row corresponds to three replications.

5. REFINEMENTS AND EXTENSIONS

5.1. Extensions to Continuous Demand Forecast Distribution

Continuous Demand Forecast Distribution. The demand model $D(t) = (D_p(t))$ presented in Section 2 consists of a fixed number of scenarios, each of which evolves continuously over time. The instantaneous demand forecast is specified by a discrete set of point forecasts with associated probability weights. Based on this demand model, the algorithm in Section 3 makes tool purchase and retirement decisions that minimize the weighted sum of costs over various scenarios. An alternative approach is that the demand *forecast* at a given time has a continuous distribution in the space of product families. (Here, terminologies such as "discrete" and "continuous" refer to the support of the instantaneous demand $D(t)$, not how time t itself is modeled.) In this section, we discuss how our model and algorithm, originally designed to work with a set of demand scenarios, can be employed with a continuous demand distribution.

The following reasons suggest that in modeling instantaneous demand forecasts, a continuous distribution is preferable to a set of scenarios. The first two points are related to the generation of demand forecasts while the third point is specifically related to capacity planning. First, in many cases, demand is inherently continuous. Unless demand arises from a few buyers asking for fixed quantities, the possible set of demand realizations is practically infinite. Second, the variance is often high. In practice, the point forecast for a fixed time in the future evolves as the forecast is updated over time. Forecasts are subject to error and revision. Çakanyildirim and Roundy [10] present a scheme for modeling the evolution of forecasts and estimating the variance of forecast errors. In the semiconductor industry this variance is significant, advocating the use of a distribution for demand forecasts. Third, a continuous demand distribution may generate a more robust capacity plan than a finite number of discrete scenarios. If a small number of demand scenarios are used in the scenario-based model, the optimization may find a solution that caters to those demand scenarios, but performs inconsistently for

Table 4. Experimental results: Serial initialization when the Uniform Fill-Rate Production allocation policy is used for the cluster split subroutine.

Min No. Split	Descent Tol.	Cost ($10^9)						CPU Hours per Repl.		
		Best		Median		Worst				
		Init.	Final	Init.	Final	Init.	Final	CPLEX	Init.	Overall
25	0.1	4.410	4.287	4.448	4.346	4.449	4.347	0.165	0.086	0.177
25	0.01	4.413	4.286	4.441	4.339	4.457	4.349	0.291	0.085	0.355
25	0.001	4.415	4.284	4.446	4.337	4.443	4.338	0.504	0.085	0.597
50	0.1	4.401	4.283	4.418	4.291	4.443	4.338	0.206	0.082	0.218
50	0.01	4.421	4.280	4.401	4.286	4.444	4.340	0.393	0.083	0.490
50	0.001	4.398	4.289	4.411	4.290	4.444	4.339	0.825	0.083	0.981
100	0.1	4.407	4.278	4.405	4.292	4.450	4.343	0.205	0.082	0.236
100	0.01	4.402	4.286	4.415	4.289	4.447	4.335	0.605	0.082	0.740
100	0.001	4.397	4.289	4.402	4.293	4.453	4.341	1.054	0.082	1.343

Note. Lost sales costs are computed using the Lost Sale Cost Minimization Policy. Each row corresponds to three replications.

other possible demand outcomes. In the rest of Section 4, we assume that demand is given as a continuous distribution and that we can generate as many discrete samples from it as we need.

The major drawback of the continuous distribution is computational complexity. This is one of the reasons why nearly all capacity planning practices in the semiconductor industry use scenario-based models of demand. The instantaneous lost sales cost needs to be evaluated repeatedly in the cluster descent and cluster splitting subroutines. This evaluation involves finding the expectation of the optimal value (9) of a linear program, where the bounds on variables are stochastic. It can be approximated using the Monte Carlo method by taking samples of demand from the distribution. For example, we assume that the instantaneous demand is a correlated multivariate log-normal distribution, in which parameters depend on time t. Generating a $|\mathcal{P}|$-dimensional multivariate log-normal distribution with the parameter $(\mu(t), \Sigma(t))$ involves generating a vector as a multivariate normal distribution with mean $\mu(t)$ and covariance matrix $\Sigma(t)$ and then exponentiating it component-wise (see, for example, [21] for details).

The difference between the scenario-based model and the continuous distribution model with the Monte Carlo method is conceptual and subtle. Computationally, both models evaluate the expected lost sales using sets of discrete demand points. The scenario-based model is an optimization problem that takes the set of input scenarios as given. This is the approach taken in Section 4. In comparison, the continuous distribution model is a simulation-optimization problem that also needs to decide how to choose a good, finite set of discrete forecasts.

Scenario Sampling. When the demand forecast is given by a continuous distribution, one can use the following simple idea: generate a set of scenarios based on the continuous distribution and then use it as an input to the scenario-based model, which is an approximation to the continuous distribution problem. This is called *scenario sampling*. The scenario-based demand model that we use in Sections 2–4 can be viewed as an implementation of a scenario sampling technique. This method contrasts with *ad hoc sampling* techniques in which independent sample points are picked for each evaluation of the objective function. These techniques potentially cause the optimization to be driven by random noise, but scenario sampling provides stability to an optimization algorithm.

If scenarios are chosen independently from a common distribution, then the following result implies a negative bias of the optimal cost in the scenario-based model, and the bias is reduced as the number of scenarios increases.

PROPOSITION 3: [22]. Let z^* be the optimal value of a stochastic minimization problem $f(x, \xi)$ with decision variables x and stochastic parameters ξ. Let $\xi^1, \xi^2, \ldots,$ be an i.i.d. sample from the distribution of ξ. Then, $Ez_n^* = E\min_x[\frac{1}{n}\sum_{i=1}^n f(x, \xi^i)] \leq z^*$. Furthermore, if $\xi'^1, \xi'^2, \ldots, \xi'^{n+1}$ is also i.i.d. from the distribution of ξ used to define z_{n+1}^*, then $Ez_n^* \leq Ez_{n+1}^*$.

The sequence $\xi^1, \xi^2, \ldots, \xi^n$ and the sequence $\xi'^1, \xi'^2, \ldots, \xi'^{n+1}$ can be independent or otherwise. The intuition behind the above proposition is that if a small sample of scenarios is used, then the scenario-based model yields a solution that caters to the specified sample only. To our knowledge, no paper in the capacity planning literature has pointed out the relationship between the sample size and the bias of the optimal value. The rate of convergence is exponential in the size of the sample (see [26], and [20] and the references therein).

Table 5. The impact of the number of scenarios. Serial initialization.

No. of Scenarios	Cost ($\$10^9$)		CPU Hours			
	Mean	Std	CPLEX	Init.	Overall	Std
2	3.942	0.272	0.130	0.055	0.150	0.021
4	4.088	0.245	0.288	0.101	0.327	0.047
8	4.099	0.149	0.661	0.199	0.763	0.150
16	4.115	0.135	1.392	0.412	1.658	0.276

Note. The minimum number of splits is 25, and the descent tolerance is 0.05. Each row corresponds to 20 replications. Demand scenarios are randomly generated for each replication.

In Table 5 we report a result of an experiment in which the number of demand scenarios varies. We use a data set similar the one used in Section 4. As the number of demand scenarios increases, the objective value increases as predicted by Proposition 3. The sample standard deviation is statistically significant, but it decreases with sample size.

5.2. Variance Reduction Techniques

Ad Hoc Sampling. One of the advantages of *ad hoc* techniques is the freedom to select sample points for each lost sales cost evaluation dynamically. For example, the choice may depend on the current capacity $U(t) = (U_m(t))$ of tool groups.

We are interested in efficiently computing the expected value of the instantaneous lost sales cost $E_D LS(c, U, D)$, or equivalently, the expected revenue $E_D R(c, U, D)$. When there is no ambiguity, we write $R(D)$ in place of $R(c, U, D)$. We now study two variance reduction techniques for the Monte Carlo method.

Importance Sampling. We choose $h_1(d)$ to be an approximation of $R(d)$. Importance sampling works well when this approximation is close to $R(d)$ (see [19]). We use $h_1(d) =$

Table 6. Comparison of instantaneous revenue obtained using naive sampling and importance sampling.

		Naive Sampling		Importance Samling	
Case	No. Tools	Mean Rev ($\$10^6$)	Std	Mean Rev ($\$10^6$)	Std
1	5	129.49	12.020	129.61	9.484
2	25	136.57	12.854	136.51	9.821
3	203	162.81	20.382	162.76	13.774
4	485	177.81	29.066	177.59	19.441
5	545	183.79	31.834	184.21	20.424
6	583	187.11	34.354	187.39	19.489

Note. Each row corresponds to 10,000 replications. We use 4 demand scenarios fixed for all cases; 10 product families.

$\min\{c \cdot d, R(\infty)\}$ where $c \cdot d$ is the revenue under infinite capacity and $R(\infty)$ is revenue under infinite demand. Let $f(d)$ be the probability density function of D. We generate demands d using the density $h(d) = h_1(d)f(d)/\int h_1(y)f(y)\,dy$ via the acceptance–rejection method (see [21]). We take the average of $R(d)/h_1(d)$ and multiply it by $\int h_1(y)f(y)dy$. Its expectation is

$$\int_y h_1(y)f(y)\,dy \cdot E_h\left[\frac{R(d)}{h_1(d)}\right] = \int_y h_1(y)f(y)\,dy \cdot \int_x \frac{R(x)}{h_1(x)}h(x)\,dx = E_D R(D).$$

There is no general analytic expression for $\int h_1(x)f(x)dx$, but we obtain it using another simulation with a larger sample size. In Table 6, experimental results using importance sampling are reported in comparison to the results of a naive sampling. We see the similarity of mean revenues obtained by the two methods and a significant reduction of variance using importance sampling.

Variance Reduction by Conditioning: Let $\Phi(d) = d/|d|$ be the unit-length directional vector of d. We let $g(\cdot)$ be the probability density of $\Phi(D)$ and condition on $\Phi(D)$ to obtain

$$E_D R(D) = E_{\Phi(D)}[E[R(D)|\Phi(D)]] = \int_\phi E[R(D)|\Phi(D) = \phi]g(\phi)\,d\phi.$$

If it is easy to evaluate or closely approximate $E[R(D)|\Phi(D) = \phi]$ for a given ϕ; then we approximate $E_D R(D)$ by generating ϕ from distribution $g(\cdot)$ and taking an average of $E[R(D)|\Phi(D) = \phi]$. Conditioning provably reduces the sampling variance (see [21]). Conditioning leads to a demand model in which samples are taken as demand rays instead of points. For a fixed ϕ, let L_ϕ be a random scalar corresponding to the magnitude of D along direction ϕ, and let $R_\phi(l) = R(l\phi)$. How difficult then is it to evaluate $E_{L_\phi}[R_\phi(L_\phi)]$? The scalar function $R_\phi(\cdot)$ is the optimal value of a parametric linear program, and it follows that $R_\phi(\cdot)$ is piece-wise linear, concave, and increasing (see, e.g., [17]). Within each piece, we can analytically compute $E_{L_\phi}[R_\phi(L_\phi)]$. For the Lost Sales Cost Minimization allocation policy, there may potentially be many break-points. However, for the Uniform Fill-Rate Production policy, it can be shown that $R_\phi(\cdot)$ has at most one break point, and thus $E_{L_\phi}[R_\phi(L_\phi)]$ may be computed analytically.

Using the Uniform Fill-Rate Production Policy, Table 7 reports the computational impact of conditioning. Conditioning reduces variance, but the magnitude of this reduction is shown to be small. This is likely attributed to the fact that ray-based

Table 7. Comparison of instantaneous Lost Sales Cost and Revenue obtained without conditioning (point-based) and with conditioning (ray-based) when Uniform Fill-Rate Production allocation policy is used.

			Point-based sampling		Ray-based sampling	
	Case	No. Tools	Mean ($\$10^6$)	Std	Mean ($\$10^6$)	Std
Lost Sales Cost	1	5	106.64	59.382	107.65	58.389
	2	25	100.20	61.198	101.21	59.153
	3	203	72.80	62.705	73.67	57.627
	4	485	48.12	60.625	49.30	57.776
	5	545	35.04	56.270	36.60	54.716
	6	583	28.04	49.982	30.01	48.770
Revenue	1	5	96.79	16.539	96.79	16.538
	2	25	103.24	17.770	103.24	17.764
	3	203	130.63	28.584	130.77	28.429
	4	485	155.32	29.572	155.14	28.304
	5	545	168.40	28.728	167.85	26.613
	6	583	175.40	29.437	174.43	27.412
Potential revenue			203.44	56.964	204.44	56.709

Note. Each row corresponds to 10,000 replications. Cases and demand scenarios are as in Table 6.

Table 8. Comparison of instantaneous Lost Sales Cost and Revenue obtained without conditioning (point-based) and with conditioning (ray-based) when the Uniform Fill-Rate Production allocation policy is used.

	No. Tools	Point-based sampling		Ray-based sampling	
		Mean ($\$10^6$)	Std	Mean ($\$10^6$)	Std
Lost Sales Cost	5	17.282	30.159	17.210	22.811
	9	15.469	29.196	15.391	22.052
	18	13.479	27.857	13.416	20.916
	35	10.657	25.499	10.633	18.879
	48	10.152	25.074	10.138	18.563
	88	7.079	22.027	7.077	16.184
	116	6.728	21.732	6.729	16.041
	148	6.567	21.586	6.575	15.967
	199	3.671	16.943	3.718	12.205
	245	3.552	16.645	3.594	12.029
	252	3.552	16.645	3.594	12.029
Revenue	5	44.962	10.347	44.992	6.963
	9	46.775	11.811	46.810	8.086
	18	48.766	13.614	48.786	9.540
	35	51.587	16.582	51.569	12.007
	48	52.092	17.203	52.064	12.497
	88	55.165	20.453	55.125	14.847
	116	55.517	20.877	55.473	15.073
	148	55.678	21.074	55.627	15.181
	199	58.573	25.486	58.484	18.812
	245	58.692	25.697	58.607	18.967
	252	58.692	25.697	58.607	18.967
Potential revenue		62.244	35.441	62.202	27.287

Note. Each row corresponds to 10,000 replications; 4 product families.

conditioning reduces the dimensionality of the demand of product families only by 1, from 10 to 9. It is supported by another test, reported in Table 8, in which conditioning demonstrates a greater reduction in variance when the number of product families is 4 instead of 10.

6. CONCLUSION

In this paper, we presented a continuous-time strategic capacity planning model that features multiple product families, flexible capacity allocation policy choices, the layering of operations, and tool retirement. It is a generalization of models by Çakanyildirim and Roundy [8], Çakanyildirim, Roundy, and Wood [9], and Hun and Roundy [18]. We devised a cluster-based heuristic algorithm, which performed extremely well in our computational experiments. We presented variance reduction techniques that are useful when the stochastic demand is specified as an evolution of a distribution. In addition, we also presented a reduction of the instantaneous lost sales cost computation under the Uniform Fill-Rate production policy to a generalized maximum concurrent flow problem. Due to the difficulty of obtaining real-world demand and forecast data from leading semiconductor companies, our computational results are unfortunately based on the single industrial data set described above. Our algorithms should be evaluated empirically using more more industrial data sets to demonstrate the robustness that we believe our solution approach to have.

APPENDIX

PROOF OF PROPOSITION 1: We describe the generalized maximum concurrent flow problem. A network $G = (N, A)$ consists of a set N of nodes and a set A of directed arcs. Each arc $a = (n_1, n_2) \in A$ has capacity $\upsilon(a)$ and a gain factor $\rho(a)$. When a volume $g(a)$ of flow enters the arc $a = (n_1, n_2)$, a volume $\rho(a) \cdot g(a)$ of flow leaves the arc. The capacity $\upsilon(a)$ limits the amount of flow entering the arc a. Given k source-sink pairs (s^j, t^j) and demands d^j at sinks, the generalized maximum concurrent flow problem is to maximize λ such that each sink receives λd^j amount of flow sent from s^j, for all j. The following result shows the existence of a polynomial-time approximation scheme for this problem.

PROPOSITION 4: [16]. For any error parameter $\epsilon > 0$, a fully polynomial-time algorithm finds a feasible solution of the generalized maximum concurrent flow problem such that the objective value of the feasible solution is at least $(1 - \epsilon)$ times the optimal value. This algorithm runs in $O(\epsilon^{-2}(|A| + |N| \log |N|)(|A| + k))$ time, where k is the number of source-sink pairs.

Under the Uniform Fill-Rate Production policy, the proportion of satisfied demand is equal across all products. Similarly, the proportion of finished work by operations is also equal across tools. Operation w faces $\sum_p b_{w,p} D_p$

amount of work, but satisfies only $\sigma \sum_p b_{w,p} D_p$. The instantaneous revenue computation problem can be recast as

$$\begin{aligned} \max_{X,\sigma} \quad & \sigma \\ \text{s. t.} \quad & \sum_{w \in W(m)} h_{m,w} X_{w,m} \le U_m \ \forall m \\ & \left(\sum_p b_{w,p} D_p \right) \sigma - \sum_{m \in M(w)} X_{w,m} \le 0 \ \forall w \\ & 0 \le \sigma \le 1 \\ & X_{w,m} \ge 0 \ \forall w, m. \end{aligned}$$

It can be shown that the above linear program is a generalized maximum concurrent flow problem, in which the underlying network is a bipartite graph. The node partition consists of the set $\mathcal{M}$ of tool groups and the set $\mathcal{W}$ of operations. The arcs correspond to finite entries of the tool requirement matrix H. The gain factor for arc (m, w) is $1/h_{m,w}$ so that flow $h_{m,w} X_{w,m}$ enters the arc and $X_{w,m}$ leaves it. Typically, the number of tool groups is bounded above by the number of operations. From $\omega = \max\{|\mathcal{W}|, |\mathcal{M}|\}$, the running time in Proposition 4 becomes $O(\epsilon^{-2} \omega^2 \log \omega)$.

ACKNOWLEDGMENTS

The authors thank Shane Henderson and Michael Todd for their assistance and insightful comments in preparing this document. They are also grateful to the anonymous referees and the associate editor for valuable suggestions. Research support was given by the Semiconductor Research Corporation Task ID: 490.001 and Graduate Fellowship Program, as well as the Natural Science and Engineering Council of Canada Postgraduate Scholarship. Computational work was conducted with the Cornell Computational Optimization Project, with support from the National Science Foundation and the Office of Naval Research.

REFERENCES

[1] S. Ahmed and N.V. Sahinidis, An approximation scheme for stochastic integer programs arising in capacity expansion, Oper Res 51 (2003), 461–471.
[2] E.G. Anderson, The nonstationary staff-planning problem with business cycle and learning effects, Manage Sci 47 (2001), 817–832.
[3] S.E. Ante, Global chip sales up 1.3% in recovery year, BusinessWeek. www.businessweek.org. March 7, 2003.
[4] F. Barahona, S. Bermon, O. Gunluk, and S. Hood, Robust Capacity Planning in Semiconductor Manufacturing, Technical Report RC22196, IBM Research Division, Yorktown Heights, NY, 2001,
[5] D.L. Benavides, J.R. Duley, and B.E. Johnson, As good as it gets: Optimal fab design and deployment, IEEE Trans Semiconduct Manufact 12 (1999), 281–287.
[6] S. Bermon and S.J. Hood, Capacity optimization planning system (CAPS), Interfaces 29 (1999), 31–50.
[7] J.R. Birge, Option methods for incorporating risk into linear capacity planning models, Manufact Serv Oper Manage 2 (2000), 19–31.
[8] M. Çakanyildirim and R. Roundy, Optimal Capacity Expansion and Contraction under Demand Uncertainty, Technical report, School of Management, University of Texas at Dallas, Richardson, TX, 2001.
[9] M. Çakanyildirim, R. Roundy, and S. Wood, Optimal machine capacity expansions with nested limitations stochastic demand, Nav Res Logis 51 (2004), 217–241.
[10] M. Çakanyildirim and R.O. Roundy, SeDFAM: Semiconductor demand forecast accuracy model, IIE Trans 34 (2002), 449–465.
[11] Z.-L. Chen, S. Li, and D. Tirupati, A scenario based stochastic programming approach for technology and capacity planning, Comp Oper Res 29 (1998), 781–806.
[12] M.H.A. Davis, M.A.H. Dempster, S.P. Sethi, and D. Vermes, Optimal capacity expansion under uncertainty, Adv Appl Probab 19 (1997), 156–176.
[13] B. Dietrich, Use of Optimization with IBM's Supply Chain. Keynote presentation at the Second Meeting of the Value Chain Academic-Industry Consortium, Lucent Technologies, Bell Labs, Murray Hill, NJ, May 22, 2003.
[14] N.J. Edwards, Approximation algorithms for the multi-level facility locatino problem, Ph.D. thesis, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, 2001.
[15] L.F. Escudero, P.V. Kamesam, A.J. King, and R.J.B. Wets, Production planning with scenario modelling, Ann Oper Res 43 (1993), 311–335.
[16] L. Fleischer and K. Wayne, Faster approximation algorithms for generalized network flow, Math Program 91 (2002), 215–238.
[17] F.S. Hillier and G.J. Lieberman, Intoduction to mathematical programming, second ed., McGraw–Hill, New York, New York, 1995.
[18] W.T. Huh and R.O. Roundy, A continuous-time strategic capacity planning model, Nav Res Logis 52 (2005), 329–343.
[19] J.P.C. Kelijnen, Statistical techniques in simulation: Part 1, Dekker, New York, New York, 1974.
[20] A.J. Kleywegt, A. Shapiro, and T.H. de Mello, The sample average approximation method for stochastic discrete optimization, SIAM J Optimiz 12 (2001), 479–592.
[21] A.M. Law and W.D. Kelton, Simulation modeling and analysis, third ed., McGraw–Hill, New York, New York, 2000.
[22] W.-K. Mak, D.P. Morton, and R.K. Wood, Monte Carlo bounding techniques for determinig solution quality in stochstic programs, Oper Res Lett 24 (1999), 47–56.
[23] O. Port, Chips on monster wafers. BusinessWeek. www.businessweek.org. November 4, 2002.
[24] S.M. Ryan, Capacity expansion with lead times and autocorrelated random demand, Nav Res Logis 50 (2003), 167–183.
[25] S.M. Ryan, Capacity expansion for random exponential demand growth with lead times, Manage Sci 50 (2004), 740–748.
[26] A. Shapiro and T.H. de Mello, On the rate of convergence of optimal solutions of Monte Carlo approximations of stochastic programs, SIAM J Optimiz 11 (2000), 70–86.
[27] J.M. Swaminathan, Tool capacity planning for semiconductor fabrication facilities under demand uncertainty, Eur J Oper Res 120 (2002), 545–558.
[28] J.A. Van Mieghem, Managing capacity as a portfolio: Review of capacity investment and hedging, Manufact Serv Oper Manage 5 (2002), 269–302.