Nils Löhndorf and Stefan Minner

Simulation Optimization for the Stochastic Economic Lot Scheduling Problem

Simulation Optimization for the Stochastic Economic Lot Scheduling Problem

Nils Löhndorf, Stefan Minner
Department of Business Administration, University of Vienna, 1210 Vienna, Austria
{nils.loehndorf@univie.ac.at, stefan.minner@univie.ac.at}

We study simulation optimization methods for the stochastic economic lot scheduling problem. In contrast to prior research, we focus on methods that treat this problem as a black box. Based on a large-scale numerical study, we compare approximate dynamic programming with a global search for parameters of simple control policies. We propose two value function approximation schemes based on linear combinations of piecewise-constant functions as well as control policies that can be described by a small set of parameters. While approximate value iteration worked well for small problems with three products, it was clearly outperformed by the global policy search as soon as problem size increased. The most reliable choice in our study was a globally optimized fixed-cycle policy. An additional analysis of the response surface of model parameters on optimal average cost revealed that the cost effect of product diversity was negligible.

Keywords: Inventory, Multi-Product, Lot-Sizing and Scheduling, Stochastic Demand, Simulation Optimization, Approximate Dynamic Programming

1. Introduction

Lot-sizing and scheduling are classical problems of production planning, with particularly many applications in the process industry. Most researchers treat this problem as a deterministic optimization problem, since this task is usually seen as short term and operational. Although this assumption is reasonable in some production environments, there are many applications where demand uncertainty requires integrating lot-sizing and scheduling with safety stock planning. Besides finding the optimal production sequence and respective lot sizes, production planning needs to provide the right amount of flexibility in response to uncertainty.

We address the problem of scheduling production of multiple products on a single machine with significant setup times under uncertain demand in continuous time. In the literature, this problem

is known as the *stochastic economic lot scheduling problem* (SELSP). The SELSP is a computationally complex problem, where the deterministic counterpart, the economic lot scheduling problem (ELSP), is already NP hard (Hsu 1983). For literature reviews on the ELSP, we refer to Elmaghraby (1978) and Davis (1990). A comprehensive literature review on stochastic lot scheduling problems with a focus on modeling and solution methods is provided by Sox et al. (1999). Winands et al. (2011) review and classify the literature on the SELSP according to sequencing and lot-sizing decisions and include several practical applications.

In this work, we focus on a problem that arises in make-to-stock environments so that we restrict our brief literature review to these environments. In general, the SELSP can be formulated as a semi-Markov decision process (SMDP) (Graves 1980, Qiu and Loulou 1995), but since this formulation suffers from the curse of dimensionality only small problem instances can be solved to optimality. Most research is therefore dedicated towards simpler policies. Gallego (1990) and Bourland and Yano (1994) both propose procedures where a production plan is set in advance and then a policy is used that restores the plan in response to uncertain demand. For Poisson demands, Federgruen and Katalan (1996) propose analytical solutions to find optimal base-stock levels and idle times for a given sequence. In Federgruen and Katalan (1998), the authors derive the optimal relative frequencies for each product, from which a fixed sequence can be constructed. For more general renewal processes, Anupindi and Tayur (1998) use infinitesimal perturbation analysis to find optimal base-stock levels for a given production sequence, and Markowitz et al. (2000) propose optimal control policies for pure rotation cycles using heavy-traffic approximation. Other approaches are in Krieg and Kuhn (2002), Wagner and Smits (2004), and Brander and Forsberg (2006). Although a fixed sequence is often a good choice, the optimal sequence is likely to be dynamic and has to take the entire vector of inventory states into account. For products with identical parameters, Vaughan (2007) finds that a dynamic sequence resulting from order-point methods outperforms a fixed cyclical schedule in systems with a large number of products and low utilization. Graves (1980) and Gascon et al. (1994) compare several heuristic scheduling rules where the sequencing decision is determined by a product's number of periods of supply. Altiok

and Shiue (1994, 1995) derive optimal (s,S) policies for dynamic sequences and Poisson demands by analyzing the underlying Markov chain. Paternina-Arboleda and Das (2005) use a two-level approach of first searching for optimal base-stock levels and then using reinforcement learning to optimize the sequencing decisions.

Although much progress has been seen in simulation optimization, only few authors discuss approximations to optimize the SMDP (Paternina-Arboleda and Das 2005) or propose black box algorithms to optimize control policies (Anupindi and Tayur 1998). Our contribution is to close this gap by proposing two different simulation optimization approaches. First, as a methodology to address the curse of dimensionality, approximate dynamic programming (ADP) has received considerable attention (Powell 2007). ADP uses Monte Carlo simulation to approximate the state-dependent value function of a dynamic program, avoiding a complete enumeration of the state space. We propose two approximate value functions based on linear combinations of piecewise-constant functions and then use an ADP algorithm to find the weights of these functions. Second, even for simple control policies closed-form solutions are complex, so that finding the right parameters is computationally challenging. Global optimizers therefore present a promising alternative. We propose representations of simple base-stock policies amenable to unconstrained global optimization for cyclic as well as dynamic production sequences. To search for the optimal parameters of these policies, we resort to the *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) algorithm (Hansen and Ostermeier 2001).

To study the SELSP as well as the various solution methods, we performed a large-scale simulation study to answer the following questions. Which simulation optimization method produces the best policy on average and how often? How much worse is a policy if it is not best? Under which circumstances is a particular policy better than another? How far from optimal is the best policy found by simulation optimization? What is the influence of model parameters on average cost?

The paper is organized as follows. In Section 2, we state model assumptions and formulate the problem as a semi-Markov decision process. In Section 3, we briefly review (exact) value iteration, which enables us to optimally solve small problem instances for benchmark purposes. Then, we

present our approximate dynamic programming solution as well as the policy search approach applied to four simple control policies. In Section 4, we present the results of an extensive simulation study.

2. Model Formulation

2.1. Assumptions and Notation

We consider the continuous time stochastic economic lot scheduling problem with $n \in \{1, 2, ..., N\}$ products. There is a single machine that can only manufacture one product at a time. If the machine state changes from idle to busy or from one product to another, the machine has to be set up. This requires a deterministic, sequence-independent setup time W_n and incurs a setup cost of A_n. We further assume that the setup status cannot be preserved over an idle period. The production for one unit of product n requires a deterministic production time p_n. During a setup or production of a single item, interruption is not permitted. Inventories are subject to holding cost h_n per item and unit of time and cannot exceed a maximum inventory level $\bar{y}_n$. Demand for each product n follows a compound renewal process with interarrival distribution F_n^A and demand size distribution F_n^D. Interarrival times and demand size are independent for each product and across products. As in Altiok and Shiue (1995) and Krieg and Kuhn (2002), we assume that unsatisfied customer demand is lost at cost v_n per item, but we allow for partial fulfillment of an order.

We model the compound renewal process as *stuttering* Poisson process, where arrivals follow a Poisson process and the demand size per arrival follows a geometric distribution. In contrast to a pure Poisson process, this enables us to model a stochastic demand process where the variance is different from the demand rate. Denote λ_n as the arrival rate of demand for product n and $C_n(t)$ as the number of arrivals of product n during a time interval of length t. Then, the probability for the number of arrivals being equal to k is Poisson distributed and given by

$$P_n^A(C_n(t) = k) = \frac{(\lambda_n t)^k}{k!} \exp(-\lambda_n t), \ k \in \mathbb{N}_+, \tag{1}$$

with mean number of arrivals $\lambda_n t$. Denote $0 < q_n \leq 1$ as the probability of the demand size D_n per arrival being equal to 1. The probability of the demand size being equal to d is given by

$$P_n^D(D_n = d) = q_n(1-q_n)^{d-1},\ d \in \mathbb{N}_+, \tag{2}$$

with mean demand size $1/q_n$. In general, compound Poisson demand over a time interval of length t is defined as

$$P(D_n(t) = d) = \sum_{i=0}^{d} \frac{(\lambda t)^i}{i!} \exp(-\lambda t) f^d(i), \tag{3}$$

where $f^d(i)$ denotes the probability that d demands occur on i demand occasions. Recursive computation of stuttering Poisson demands (Feeney and Sherbrooke 1966), where $C_n(t) = k$ customers with total demand $D_n(t) = d$ arrive during time period t, is given by

$$P(D_n(t) = d) = \sum_{k=0}^{d} f_n(d,k), \tag{4}$$

with $f_n(d,k) = (1-q_n) f_n(d-1,k) + q_n \frac{\lambda_n t}{k} f_n(d-1,k-1)$, $f_n(0,0) = \exp(-\lambda_n t)$, $f_n(d,k) = 0$ if $d < k$, and $f_n(d,0) = 0$ if $d > 0$.

For a given period demand with mean μ_n and standard deviation σ_n, the required stuttering Poisson parameters to obtain the same first two moments are $q_n = 2\mu_n(\mu_n + \sigma_n^2)^{-1}$ and $\lambda_n = \mu_n q_n$. Note that P_n^D is only defined if $q_n \leq 1$ so that feasible mean-variance combinations are limited to $\mu_n \leq \sigma_n^2$.

2.2. Semi-Markov Decision Process

We model the SELSP under compound Poisson demand as an infinite horizon, average cost SMDP. In an SMDP, decisions are only made after a change of the system state which is relevant for the decision-making process. During such a decision epoch, the system state may change several times due to multiple demand arrivals, but no decision can be made.

We describe the state of the production system by the vector $S = (m, y) \in \mathcal{S}$, where y denotes the inventory state $y = (y_1, ..., y_N)$, with $0 \leq y_n \leq \bar{y}_n$ and m the machine state, with $m \in \{0, ..., N\}$. Machine status $m = 0$ indicates that the machine is idle and $m = n > 0$ that the machine is currently set up for product n. We assume that sojourn times are finite and a finite number of transitions takes place during a (finite) decision epoch. Denote $u \in \mathcal{U} = \{0, ..., N\}$ as a production decision. A

new decision epoch begins after production or setup of an item has been completed, or, in case the machine has been idle, upon arrival of new customer demand.

To solve the infinite horizon SMDP, we have to find an optimal policy π^* such that the average cost per unit of time g is minimized. Let g^* denote the minimal expected average cost and $V^*(S)$ the minimal average cost (or value) when being in state S. Denote $P(S'|S,u)$ as the probability of a transition from state S to successor state S' when decision u is taken; and denote $\bar{\tau}(S,u)$ as the average time and $c(S,u)$ as the total cost associated with state S and decision u prior to transition to S'. Then, the optimization problem is to find g^* and $V^*(S)$, such that

$$V^*(S) = \min_{u \in \mathcal{U}} \left\{ c(S,u) - g^* \bar{\tau}(S,u) + \sum_{S' \in \mathcal{S}} P(S'|S,u) V^*(S') \right\} \quad \forall\ S \in \mathcal{S}. \tag{5}$$

For the problem described, the Markov chain of every stationary policy has a unichain transition matrix, so that the expected average cost do not vary with the initial state.

The probability of a transition from state S to S' after decision u is given by

$$P(S'|S,u) = \prod_{n=1}^{N} P_n(S'|S,u). \tag{6}$$

The product-specific transition probabilities $P_n(S'|S,u)$ are defined as

$$P_n(S'|S,u) = \begin{cases} \lambda_n P\big(D_n = y_n - y'_n\big) \big(\sum_{m=1}^{N} \lambda_m\big)^{-1} & \text{if } u = 0 \text{ and } y'_n > 0, \\ \lambda_n P\big(D_n \geq y_n\big) \big(\sum_{m=1}^{N} \lambda_m\big)^{-1} & \text{if } u = 0 \text{ and } y'_n = 0, \\ P(D_n(p_n) = y_u - y'_u + 1) & \text{if } n = u,\ u = m \text{ and } y'_n > 1, \\ P(D_n(p_n) \geq y_u) & \text{if } n = u,\ u = m \text{ and } y'_n = 1, \\ P(D_n(p_u) = y_n - y'_n) & \text{if } n \neq u,\ u = m \text{ and } y'_n > 0, \\ P(D_n(p_u) \geq y_n) & \text{if } n \neq u,\ u = m \text{ and } y'_n = 0, \\ P(D_n(W_u) = y_n - y'_n) & \text{if } u \neq m \text{ and } y'_n > 0, \\ P(D_n(W_u) \geq y_n) & \text{if } u \neq m \text{ and } y'_n = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

If the machine goes idle ($u = 0$), we multiply the probability of the next event being demand for product n with the probability of demand being either of size equal to $y_n - y'_n$ for $y'_n > 0$ or of size greater than y_n for $y'_n = 0$. If the machine is set up to produce an item other than n ($u = m = n$), demand over p_n periods either depletes inventory from y_n to $y'_n > 1$ or to $y'_n = 1$, so that $y'_n \geq 1$ always includes the previously produced item. If the machine is already set up but produces another

product ($u = m \neq n$), demand over p_u periods either depletes inventory from y_n to $y'_n > 0$ or to zero. If the machine has to be set up ($u \neq m$), demand over W_u periods either depletes inventory from y_n to $y'_n > 0$ or to zero.

The average sojourn time $\bar{\tau}(S,u)$ of a decision epoch is given by

$$\bar{\tau}(m,y,u) = \begin{cases} p_u & \text{if } u = m, \\ W_u & \text{if } u \neq m, \\ \left(\sum_{n=1}^{N} \lambda_n\right)^{-1} & \text{if } u = 0. \end{cases} \tag{8}$$

Note that in case the machine goes idle, the average sojourn time until the next demand event is a Poisson process with rate $\sum_{n=1}^{N} \lambda_n$.

The cost $c(S,u)$ of being in state S and taking decision u consists of setup or variable manufacturing costs, inventory holding costs, and lost sales penalty costs.

$$c(S,u) = \begin{cases} c_u + \sum_{n=1}^{N} \left(H_n(y_n, p_u) + v_n \sum_{d=y_n+1}^{\infty} (d-y_n) P(D_n(p_u) = d)\right) & \text{if } u = m \\ A_u + \sum_{n=1}^{N} \left(H_n(y_n, W_u) + v_n \sum_{d=y_n+1}^{\infty} ((d-y_n) P(D_n(W_u) = d)\right) & \text{if } u \neq m, \\ \sum_{n=1}^{N} \left(h_n y_n + v_n \lambda_n \sum_{d=y_n+1}^{\infty} (d-y_n) P_n^D(d)\right) \left(\sum_{i=1}^{N} \lambda_i\right)^{-1} & \text{if } u = 0. \end{cases} \tag{9}$$

The determination of product-specific and time-dependent expected holding cost $H_n(y_n,t)$ requires to track each of the y_n items. Demand for item k occurs on the l-th demand event if $D_n^{(l-1)} < k$ and $D_n^{(l)} \geq k$ where $D_n^{(l)}$ is the cumulative demand for product n over l transactions and $F_n^{(l)}$ denotes the l-fold convolution of the demand size distribution

$$\begin{aligned} \theta_n(k,l) &= P(D_n^{(l-1)} < k \wedge D_n^{(l)} \geq k) = \sum_{x=1}^{k-1} P(D_n^{(l-1)} = x) P(D_n \geq k - x) \\ &= \sum_{x=1}^{k-1} f_n^{(l-1)}(x)(1 - F_n^D(k-x-1)) = F_n^{(l-1)}(k-1) - F_n^{(l)}(k-1), \end{aligned}$$

with $F_n^{(l)}(1) = 0$, $F_n^{(0)} = 1$. Note that $k \geq 1$ and for $l = 1$, $P(D_n \geq k) = 1 - F_n^{(l)}(k-1)$. The convolution is given by

$$F_n^{(l)}(k) = \sum_{h=1}^{k-1} F_n^{(l-1)}(h) f_n^D(k-h). \tag{10}$$

The time until the l-th demand is Erlang distributed with parameters λ_n and l. Then, for a given inventory level y_n, the expected holding costs over a time interval of length t are given by

$$H_n(y_n,t) = h_n \sum_{k=1}^{y_n} \sum_{l=1}^{k} P(D_n(l-1) < k \wedge D_n(l) \geq k) \left(\int_0^t x E_{n,l}(x) dx + t \int_t^{\infty} E_{n,l}(x) dx\right)$$

$$= h_n \sum_{k=1}^{y_n} \sum_{l=1}^{k} (F_n^{(l-1)}(k-1) - F_n^{(l)}(k-1)) \times \left(\frac{l}{\lambda_n} (1 - P_n^A(C_n(t) \leq l)) + t \cdot P_n^A(C_n(t) \leq l-1) \right). \quad (11)$$

3. Solution Methods

3.1. Relative Value Iteration

The solution to the system of equations given in (5) is unique w.r.t. g^* and unique up to an additive constant for all $V(S)$. Therefore, we can set the average cost of an arbitrary state S_0 to zero and express the average cost of being in a state relative to this value. The optimal average costs and the relative average costs of being in a state can then be successively approximated by the following recursion scheme, where g^j and $V^j(S)$ denote the respective values obtained in iteration j,

$$g^j = \min_{u \in \mathcal{U}} \left\{ \frac{c(S_0, u) + \sum_{S \in \mathcal{S} \setminus S_0} P(S|S_0, u) V^j(S)}{\bar{\tau}(S_0, u)} \right\}. \quad (12)$$

The average cost is the minimal cost over all possible decisions u (in the arbitrarily chosen state S_0) – expressed by the direct cost of taking decision u in state S_0, $c(S_0, u)$ – plus the optimal future costs over all transitions to other states S with their respective values $V^j(S)$, normalized by the average time of being in state S_0 under decision u.

The relative values are updated such that the new value is the optimal direct cost of being in a state plus the costs of all transitions to other states, estimated through the values from the previous iteration. These values are then normalized on a per time unit basis by subtracting the current estimate of the optimal average cost.

$$V^{j+1}(S) = V^j(S) - g^j + \beta \cdot \min_{u \in \mathcal{U}} \left\{ \frac{c(S, u) + \sum_{S' \in \mathcal{S} \setminus S_0} P(S'|S, u) V^j(S') - V^j(S)}{\bar{\tau}(S, u)} \right\}, \quad (13)$$

for $S \in \mathcal{S} \setminus S_0$ and with $\beta < \bar{\tau}(S, u) \ \forall \ S \in \mathcal{S}, \ u \in \mathcal{U}$. Note that β is an algorithmic parameter to ensure convergence of the iteration scheme through weighting the previous future cost estimate and the current estimate. In addition, this successive improvement scheme provides a lower and an upper bound on the optimal gain in each iteration (Schweitzer 1971).

(1) Input arguments: value function $\bar{V}(\,\cdot\,; w_0)$, starting state S

(2) Do for $t = 1, 2, \ldots, T$

(2.1) Solve $u^* \leftarrow \arg\min_{u \in \mathcal{U}} \bar{V}\big(S, u; w_{t-1}\big)$

(2.2) Compute $\big(c_t, \tau_t, S'\big) \leftarrow S^M\big(S, u^*\big)$

$$r \leftarrow c + \exp(-\gamma\tau) \min_{u \in \mathcal{U}} \bar{V}\big(S', u; w_{t-1}\big)$$

(2.3) Update $w_t = U^V\big(w_{t-1}, S, u^*, r\big)$, $S \leftarrow S'$

(3) Return value function $\bar{V}(\,\cdot\,; w_T)$

Figure 1 Approximate value iteration for SMDPs

3.2. Approximate Value Iteration

Since relative value iteration is subject to the curse of dimensionality for larger problems, we propose to use approximate value iteration instead (see Figure 1). For a given approximate value function $\bar{V}(\,\cdot\,; w_0)$ with initial parameters w_0 and a starting state S, the algorithm simulates a sample path of T decision epochs while updating the control policy *online*. The main loop consists of three steps: (2.1) a (greedy) control policy selects the best decision based on the value estimate of the previous iteration for the given state of the system; (2.2) a simulation model S^M samples the state transition function and returns a realization of the immediate cost c, the sojourn time τ and the successor state S', which gives the discounted reward,

$$r = c + \exp(-\gamma\tau) \min_{u \in \mathcal{U}} \bar{V}\big(S', u; w_{t-1}\big); \tag{14}$$

(2.3) a function U^V updates the value estimate of making the greedy decision u^* in state S. After T decision epochs, the algorithm returns the final estimate of the value function approximation.

For approximate value iteration, we use discounted reward as a proxy for average reward. In early experiments, we found this formulation to be more stable than approximating the average reward directly. The discount factor γ can therefore be regarded as a purely algorithmic parameter which has to be set sufficiently large to obtain a nearly average cost optimal policy without risking numerical stability.

3.2.1. Value function approximation by stochastic gradients. The updating function U^V is based on stochastic gradient algorithms, a popular class of methods for function approximation which are particularly well-suited for approximate value iteration (Bertsekas 2007, Powell 2007). In contrast to (non-)linear regression methods, stochastic gradient algorithms have only $\mathcal{O}(n)$ time complexity and are able to estimate the mean of a random variable *online* while new samples are being collected.

A stochastic gradient algorithm changes the parameters of a function approximator to fit the observations from a data set. Let the approximate value function $\bar{V}$ be a linear combination of basis functions ϕ_i with real-valued weights w_i and $i \in \{1, 2, ..., D\}$. A basis function ϕ_i may be any non-linear function of S and u. Denote w and Φ as the corresponding vectors of length D, with $w^\top$ as transpose. Then, the approximate value function is given by

$$\bar{V}(S, u; w) = w^\top \Phi(S, u) = \sum_{i=1}^{D} w_i \phi_i(S, u). \tag{15}$$

A stochastic gradient algorithm adjusts the weight vector w after each observation in the direction that minimizes the *mean squared error*, $\min_w \frac{1}{2}\big(\bar{V}(S, u; w) - r\big)^2$. For a linear function, the *stochastic* sample gradient with respect to w is given by $\Phi(S, u)$. Since the true gradient is unknown, we adjust the weight vector in the direction of the gradient only by a small amount $\alpha_t \in (0, 1]$, referred to as stepsize. The function U^V that updates the weight vector is then given by

$$w_t = U^V\big(w_{t-1}, S, u, r\big) = w_{t-1} + \alpha_t\big(r - \bar{V}(S, u; w_{t-1})\big)\Phi(S, u). \tag{16}$$

For a stationary policy, the weight vector w is guaranteed to converge to a local optimum as long as we gradually reduce the stepsize to zero (Bertsekas 2007, p. 333). Practical convergence is thereby largely affected by choosing the right stepsize for each updating step. Although the optimal stepsize schedule is unknown, experimental work has shown that there exist simple stepsize rules that work well in practice (Powell 2007, Ch. 6). One of these rules is the *generalized harmonic* stepsize, which is given by $\alpha_t = ab(a + t - 1)^{-1}$, with $a \in \mathbb{R}^+$ and $b \in (0, 1]$ as scaling parameters.

3.2.2. Piecewise-constant value functions. We use a linear combination of piecewise-constant functions as approximation scheme and apply the stochastic gradient algorithm to update the weights of the constant function segments. We assign two separate functions to each production decision: one function for the case when the machine state is changed after a decision and one function for the case when the machine state remains the same. This separation takes the different sojourn times during setup or production of an item into account. We then model each of these functions as a linear combination of piecewise-constant basis functions of the inventory states.

Denote $\bar{v}_j$ as the *partial* value function, with $j \in \{0, ..., 2N\}$. The piecewise-constant function that assigns two partial value functions to each decision is given by

$$\bar{V}(S, u; w) = \begin{cases} \bar{v}_u(y; w) & \text{if } u \neq m \text{ or } u = 0, \\ \bar{v}_{N+u}(y; w) & \text{otherwise.} \end{cases} \tag{17}$$

Note that for the decision to go idle, $u = 0$, we use only one function, since the sojourn time for an idle epoch is independent of the machine state.

For the partial value functions $\bar{v}_j$, we test two different approximation schemes. The first approximation is the sum over N piecewise-constant functions of each inventory state variable. The intuition is that the expected immediate cost can be computed separately for each product, since expected holding and lost sales cost of one product are independent of other products. This makes the immediate cost function separable in the inventory state variables y_n. The separable (first-order) partial value function is then given by

$$\bar{v}_j(y; w) = \sum_{n=1}^{N} \bar{v}_{jn}^{(1)}\big(y_n; w\big). \tag{18}$$

Each function $\bar{v}_{jn}^{(1)}$ is a piecewise-constant function with K disjoint intervals of width $d_n^{(1)} = \frac{\bar{y}_n + 1}{K}$: $n \in \{1, ..., N\}$,

$$\bar{v}_{jn}^{(1)}(y_n; w) = \sum_{k=1}^{K} w_{I(j,n,k)} \cdot \phi_{I(j,n,k)}(y_n) = \sum_{k=1}^{K} w_{I(j,n,k)} \cdot \mathbb{1}[(k-1)d_n^{(1)}, k d_n^{(1)})(y_n), \tag{19}$$

where $I(\cdot)$ maps the multi-dimensional index to a unique index of an element of the weight vector. Note that by setting $d_n^{(1)} = \frac{\bar{y}_n + 1}{K}$, the right-open interval $[(K-1)d_n^{(1)}, K d_n^{(1)})$ contains the maximum inventory level for $K \leq \bar{y}_n$.

If we take into account that the approximate value function is not only an estimate of the immediate cost, but also of the discounted value of future states and decisions, then we cannot assume separability any more. To consider dependencies among inventory state variables, we propose a second approximation scheme, where we add the sum of piecewise-constant functions over all $\binom{N}{2}$ combinations of inventory state variables to the first approximation. The (second-order) partial value function is then given by

$$\bar{v}_j(y;w) = \sum_{n=1}^{N} \Big(\bar{v}_{jn}^{(1)}\big(y_n;w\big) + \sum_{m=n+1}^{N} \bar{v}_{jnm}^{(2)}(y_n, y_m; w) \Big). \tag{20}$$

Each function $\bar{v}_{jnm}^{(2)}$ is a piecewise-constant function with two arguments and $L \times L$ disjoint segments with edge lengths of $d_n^{(2)} = \frac{\bar{y}_n+1}{L}$ and $d_m^{(2)} = \frac{\bar{y}_m+1}{L} : n, m \in \{1, ..., N\}$, so that

$$\begin{aligned} \bar{v}_{jnm}^{(2)}(y_n, y_m; w) &= \sum_{k=1}^{L} \sum_{l=1}^{L} w_{I'(j,n,m,k,l)} \cdot \phi_{I'(j,n,m,k,l)}(y_n, y_m), \\ &= \sum_{k=1}^{L} \sum_{l=1}^{L} w_{I'(j,n,m,k,l)} \cdot \mathbb{1}[(k-1)d_n^{(2)}, kd_n^{(2)})(y_n) \cdot \mathbb{1}[(l-1)d_m^{(2)}, ld_m^{(2)})(y_m), \end{aligned} \tag{21}$$

with $I'(\cdot)$ as another index function.

Let us briefly review the space complexity of the approximate value functions. The first approximation with a weight vector of length $D = \big(2N+1\big)KN$ has a worst-case space complexity of $\mathcal{O}(KN^2)$, while the second approximation with $D = \big(2N+1\big)\big(KN + L^2\frac{N(N-1)}{2}\big)$ has $\mathcal{O}(N^3L^2)$. Both schemes therefore have only polynomial space complexity, which is low compared to laying a coarse grid over the state space. A grid where each inventory state is aggregated into C intervals would produce a weight vector of length $D = \big(2N+1\big)C^N$ which has an exponential worst-case complexity of $\mathcal{O}(NC^N)$ and would thus be itself subject to the curse of dimensionality.

3.3. Direct Policy Search

An alternative to approximating the value function and using this function to control the decision process is to directly search for optimal parameters of simpler control policies. To guide the search for the optimal parameter vector, we resort to the CMA-ES algorithm (Hansen and Ostermeier 2001). CMA-ES generates new candidate vectors from a multivariate normal distribution, i.e.,

(1) Input arguments: initial guess μ^x, trust region σ^x

(2) Do for $i = 1, 2, \ldots, I$

(2.1) Get $(x^1, ..., x^K) \leftarrow G^M(K)$ from internal model

(2.2) Do for $k = 1, 2, \ldots, K$

(2.2.1) Do for $t = 1, 2, \ldots, T$

(2.2.1.1) Compute $\big(c_t, \tau_t, S\big) \leftarrow S^M\big(S, \pi(S; x^k)\big)$

(2.2.2) Compute $r^k \leftarrow \sum_{t=e+1}^{T} c_t \left(\sum_{t=1}^{T} \tau_t\right)^{-1}$

(2.3) Update internal model $U^M\big((x^1, ..., x^K), (r^1, ..., r^K)\big)$

(3) Return best solution x^*

Figure 2 Generic policy search for production control

$\mathcal{N}\big(\mu^x, \mathrm{diag}(\sigma^x)\big)$, which serves as an internal model of promising search steps, where dependencies among parameters are described by the covariance matrix. Throughout the search process, the algorithm updates the distribution's mean vector and its covariance matrix to increase the likelihood of previously successful search steps.

Figure 2 outlines a generic formulation of the CMA-ES algorithm. Denote $\pi(\,\cdot\,; x)$ as control policy which is characterized by a (continuous) parameter vector x. The objective is to search for an x that minimizes the expected average cost. The algorithm is initialized with a *guess* of the best solution, μ^x, as well as a *trust* region, σ^x, in which the solution is likely to be found. The main loop consists of three steps: (2.1) the algorithm generates a set of K candidate solutions $(x^1, ..., x^K)$ from the internal model G^M which controls the search process; (2.2) for each of the resulting control policies, the algorithm simulates the transition process and records the average cost; (2.3) the algorithm updates the internal model using the sampled information about the mapping of parameters $(x^1, ..., x^K)$ to average rewards $(r^1, ..., r^K)$. Evidently, the algorithm searches for the global optimum of a noisy, non-convex objective function, without guarantee of finding the best solution after I iterations. Let us now introduce four control policies which can be fully described by a parameter vector x.

(1) Input arguments: integer frequencies R

(2) Group identical frequencies $L_i = \{n : R_n = i\}$

(3) Do for $i = 1, 2, \ldots, \max_{n \in N} R_n$

(3.1) $d \leftarrow \dim(Q) R_i^{-1}$

(3.2) Do for $j = 1, 2, \ldots, i$

(3.2.1) $z \leftarrow \lfloor jd + 0.5 \rfloor + j \dim(L_i) + 1$

(3.2.2) $Q \leftarrow Q' : Q'_k = \begin{cases} Q_k & \forall\ k < z, \\ L_{i,k-z} & \forall\ z \le k < z + \dim(L_i), \\ Q_{k-\dim(L_i)} & \forall\ k \ge z + \dim(L_i) \end{cases}$

(4) Return sequence Q

Figure 3 Heuristic method to generate an evenly spaced production sequence from given integer frequencies

3.3.1. Fixed-cycle policy. An intuitive solution to the problem of producing multiple products on a single machine is to fix a sequence in which these products are being produced in addition to quantities and (possibly) idle times. The fixed-cycle policy follows an idea originally proposed by Dobson (1987) for the ELSP and adapted by Federgruen and Katalan (1998) for the SELSP. The authors find the optimal production frequency for each product and then use this information to construct a fixed production sequence.

Denote $R \in \mathbb{N}^N$ as the set of integer frequencies and $Y \in \mathbb{N}^N$ as the set of order-up-to levels. We map the corresponding continuous parameter vector $x \in \mathbb{R}^{2N}$ to these two sets by decoding the vector into $R_n = \lfloor |x_{N+n}| \rfloor + 1$ and $Y_n = \lfloor |x_{N+n}| \rfloor + 1$. Since the product with the highest frequency can be scheduled at most every other time, we restrict the maximum frequency to be less than or equal to the sum of all other frequencies.

Figure 3 outlines a simple heuristic method to generate a production sequence $Q = \{Q_1, ..., Q_J\}$ of length J from a given set of integer frequencies. The heuristic assigns each product to a set of products with identical frequencies, $L_i = \{n : R_n = R_i\}$. Each set L_i is inserted into the sequence multiple times according to its frequency. In step (3.2.2) the algorithm inserts L_i at position z and shifts the element currently at that position to the right. Suppose we have $R = \{2, 4, 1, 4, 2\}$, which

gives us $L_1 = \{3\}, L_2 = \{1,5\}, L_3 = \{\}, L_4 = \{2,4\}$. Then, the heuristic would insert the L_i's into Q in the following way:

$$Q^1 = \{3\} \rightarrow Q^2 = \{1,5,3,1,5\} \rightarrow Q^3 = \{1,5,3,1,5\} \rightarrow Q^4 = \{2,4,1,2,4,5,3,2,4,1,2,4,5\}$$

The resulting production sequence features products being roughly evenly spaced over the entire cycle according to their integer frequencies.

For a given position j in sequence Q and order-up-to levels Y, the fixed-cycle policy is now given by

$$\pi(S;x) = \begin{cases} 0 & \text{if } \forall\ n : y_n = Y_n, \\ Q_{z(j)} & \text{otherwise,} \end{cases} \tag{22}$$

where the recursive function z is defined as

$$z(j) = \begin{cases} j & \text{if } y_k < Y_k : k = Q_j, \\ z\big(j \bmod J + 1\big) & \text{otherwise.} \end{cases} \tag{23}$$

For a given position j, the function returns the next position in the sequence for which $y_n < Y_n$, where the modulus ensures that the production cycle is repeated as soon as $j = J$. Note that this approach allows for simultaneous optimization of production sequence and base-stock levels, in contrast to Anupindi and Tayur (1998) who propose a two-stage approach of fixing a schedule and then searching for base-stock levels.

As initial guess for the policy search, we propose to use a heuristic solution that is based on the *common cycle solution* to the ELSP. Denote k as a safety factor and $\hat{T}$ as the common cycle time. Then, we obtain a policy, where for each product n we set

$$R_n = 1, \quad Y_n = \max\left\{ \left\lfloor \mu_n \hat{T} + k\sigma_n \sqrt{\hat{T}} \right\rceil, 1 \right\}. \tag{24}$$

Note that $Y_n \geq 1$ is a lower bound on the order-up-to level, since production would be zero otherwise. The common cycle time $\hat{T}$ and the safety factor k are given by

$$\hat{T} = \max\left\{ \sqrt{\frac{2\sum_{n=1}^{N} A_n}{\sum_{n=1}^{N} h_n \mu_n (1-\mu_n p_n)}}, \frac{\sum_{n=1}^{N} W_n}{1 - \sum_{n=1}^{N} \mu_n p_n} \right\}, \quad k = \Phi^{-1}\left(\frac{v_n}{v_n + h_n \hat{T}} \right), \tag{25}$$

with Φ^{-1} as inverse normal distribution. Using the quantile of the compound Poisson distribution instead would add little additional value, since k is merely a parameter of the initial guess. As trust region we use $\sigma_i^x = \max\left\{ \frac{1}{2}\mu_i^x, \delta \right\}$ for all policies, with $\delta \geq 1$ to ensure exploration in case $\mu_i^x = 0$.

3.3.2. Fixed-cycle policy with preemption. A major drawback of using a fixed cycle in a stochastic production environment is its lack of flexibility. For example, assume that product i is next but still close to its order-up-to-level while product j, which is in line after i, is already out of stock. Then, it could be better to preempt production of j, instead of setting up for i and risking lost sales of j. Moreover, under the fixed-cycle policy the machine only goes idle when all products are at their order-up-to levels, which may not be the best choice when utilization is low, but inventory holding cost high.

To overcome these drawbacks, we suggest to add two additional control parameters, a *preemption* point and a *can-order* level. Denote $f \in \mathbb{R}_+^N$ as before, $Y^{(1)} \in \mathbb{N}^N$ as the set of preemption points, $Y^{(2)} \in \mathbb{N}^N$ as the set of can-order levels, and $Y^{(3)} \in \mathbb{N}^N$ as the set of order-up-to-levels. The corresponding parameter vector $x \in \mathbb{R}^{4N}$ can then be decoded into $f_n = |x_n| + 1$, $Y_n^{(2)} = \lfloor |x_{N+n}| \rfloor$, $Y_n^{(1)} = \max\{Y_n^{(2)} - 1 - \lfloor |x_{2N+n}| \rfloor, -1\}$ and $Y_n^{(3)} = Y_n^{(2)} + 1 + \lfloor |x_{3N+n}| \rfloor$.

Instead of rotating the entire sequence, as in Gallego (1990), we preempt the critical product, thereby turning the fixed production cycle into a dynamic one. When one or more products are at their preemption points or below, we update the production sequence and move the next product with $y_n \leq Y_n^{(1)}$ from its original position to the position that comes next. On the other side, when all products are still above their can-order level, the machine goes idle.

Given the current position j in sequence Q, the fixed-cycle policy with preemption is given by

$$\pi(S;x) = \begin{cases} 0 & \text{if } \forall\ n : y_n > Y_n^{(2)}, \\ Q'_{j+1} & \text{if } \exists\ n : y_n \leq Y_n^{(1)}, \\ Q_{z'(j+1)} & \text{otherwise,} \end{cases} \tag{26}$$

where $Q' = Z(Q, z''(j+1), j+1)$. Z is defined as

$$Z(Q,i,j) = \begin{cases} Q' \in \mathbb{Q} : Q'_j = Q_i, Q'_{k+1} = Q_k \forall\ j \leq k < i, Q'_k = Q_k \forall\ k < j \wedge k > i & \text{if } j \leq i, \\ Q' \in \mathbb{Q} : Q'_j = Q_i, Q'_{k-1} = Q_k \forall\ i < k \leq j, , Q'_k = Q_k \forall\ k < j \wedge k > i & \text{otherwise,} \end{cases} \tag{27}$$

with $\mathbb{Q}$ as the power set of Q. The function Z returns a new sequence which is identical to Q, except that the product at position i is moved to position j. The recursive functions z' and z'' are given by

$$z'(j) = \begin{cases} j & \text{if } y_k < Y_k^{(3)} : k = Q_j, \\ z'\big(j \bmod J + 1\big) & \text{otherwise,} \end{cases} \tag{28}$$

$$z''(j) = \begin{cases} j & \text{if } y_k \le Y_k^{(1)} : k = Q_j, \\ z''\big(j \bmod J+1\big) & \text{otherwise.} \end{cases} \tag{29}$$

Note that the preemption persists in the next decision epoch, so that Q' becomes Q after transition from S to S'.

As initial guess for the policy search, we propose to use the fixed-cycle policy and set $Y_n^{(3)} = Y_n$ and $Y_n^{(1)} = -1, Y_n^{(2)} = Y_n^{(3)} - 1$.

3.3.3. Base-stock policy. An alternative to following a fixed production sequence is to trigger new production orders based entirely on current inventory levels (Graves 1980). In addition to an order-up-to level which determines production quantities, we define a reorder point that initiates new production orders. In case two or more products reach their reorder points after a production run has been completed, we use the earliest *run-out time* as priority rule (Gascon et al. 1994). The run-out time is defined as the average time until product n is out-of-stock minus its setup time.

Denote $Y_n^{(1)}$ as reorder point and $Y_n^{(2)}$ as order-up-to level, with $Y_n^{(1)} = |x_n|$ and $Y_n^{(2)} = Y_n^{(1)} + 1 + |x_{N+n}|$ for a given vector $x \in \mathbb{R}^{2N}$. Let $I = \{n : y_n \le Y_n^{(1)}\}$ define the set of products with inventories below their reorder points. The control policy is then given by

$$\pi(S;x) = \begin{cases} m & \text{if } m > 0 \text{ and } y_m < Y_m^{(2)}, \\ \arg\min\limits_{i \in I} \left\{ \frac{y_i}{\mu_i} - W_i \right\} & \text{else if } \exists\ n : y_n \le Y_n^{(1)}, \\ 0 & \text{else if } \forall\ n : y_n > Y_n^{(1)}. \end{cases} \tag{30}$$

As initial guess for the policy search, we propose to use a heuristic solution that is based on the Doll and Whybark heuristic adapted by Gascon et al. (1994). Denote k as a safety factor and $\hat{T}$ as the common cycle time as before. Then, we obtain an (s,S) policy, where for each product n we set

$$Y_n^{(1)} = \max\left\{ \lfloor \mu_n W_n + k \cdot \sqrt{\sigma_n^2 \hat{T}} \rfloor, 0 \right\}, \quad Y_n^{(2)} = Y_n^{(1)} + \max\left\{ \lfloor \mu_n (1 - \mu_n p_n) \hat{T} \rfloor, 1 \right\}. \tag{31}$$

Note that $Y_n^{(1)} \ge 0$ in order to trigger a production order, and $Y_n^{(2)} - Y_n^{(1)} \ge 1$ to avoid that production is zero.

3.3.4. Can-order base-stock policy. A major drawback of using a simple base-stock policy is its inability to respond to critical inventory levels during a production run. For example, assume that product i is set up and below its order-up-to level but far above its reorder point while product j is already out-of-stock. Then, it could be better to interrupt production of i and change over to j.

For the can-order base-stock policy, we suggest to use a *can-order* point as well as a *can-order-up-to* level in addition to reorder point and order-up-to-level. Denote $Y_n^{(1)}$, $Y_n^{(2)}$ as before and $Y_n^{(3)}$ as can-order point and $Y_n^{(4)}$ as can-order-up-to level, with $Y_n^{(3)} = Y_n^{(1)} + |x_{2N+n}|$ and $Y_n^{(4)} = Y_n^{(3)} + 1 + |x_{3N+n}|$ for a given $x \in \mathbb{R}^{4N}$. Let $I = \{n : y_n \leq Y_n^{(1)}\}$ as before and $I' = \{n : y_n \leq Y_n^{(3)}\}$ define the set of products with inventories below their can-order points. The control policy is then given by

$$\pi(S;x) = \begin{cases} m & \text{if } m > 0 \text{ and } y_m < Y_m^{(2)} \\ \arg\min\limits_{i \in I} \left\{ \frac{y_i}{\mu_i} - W_i \right\} & \text{else if } \exists\ n : y_n \leq Y_n^{(1)}, \\ m & \text{else if } \forall\ n : y_n > Y_n^{(1)} \text{ and } m > 0 \text{ and } y_m < Y_m^{(4)} \\ \arg\min\limits_{i \in I'} \left\{ \frac{y_i}{\mu_i} - W_i \right\} & \text{else if } \forall\ n : y_n > Y_n^{(1)} \text{ and } \exists\ n : y_n \leq Y_n^{(3)}, \\ 0 & \text{else if } \forall\ n : y_n > Y_n^{(3)} \end{cases} \tag{32}$$

The machine continues production as long as there exists a product with an inventory level below its can-order point. When a product is above its can order-up-to level but another drops below its reorder point, production can be interrupted to set up the critical product.

As initial guess for the policy search, we propose to use a simple base-stock policy and set $Y_n^{(1)} = Y_n^{(2)}, Y_n^{(3)} = Y_n^{(4)}\ \forall\ n$.

4. Numerical Study

4.1. Experimental Design

To answer our research questions, we carried out an extensive numerical study, for which we generated instances of model parameters which are maximally different and cover a large range of parameters.

For each instance of the problem, we choose seven values of model parameters for each of the N products: mean demand per period, its variance, lost sales cost, holding cost, setup cost, setup

Design Parameter	Min. Value	Max. Value
Avg Mean Demand	5	5
Div Mean Demand	0	0.5
Avg Lost Sales Cost	100	100
Div Lost Sales Cost	0	0.5
Avg Hold/LS Cost ($\bar{\kappa}$)	0.001	0.01
Div Hold/LS Cost	0	0.5
Avg Setup/Prod Time ($\bar{\eta}$)	1	100
Div Setup/Prod Time	0	0.5
Avg CV Demand ($\bar{c}_D$)	0.5	1.5
Div CV Demand	0	0.5
Avg Load Factor ($\bar{\rho}$)	0.3	0.9
Div Load Factor	0	0.5

Table 1 Intervals of the design parameters that span the experimental area

time, and production time. For our study, we first fixed mean demand and defined the variance through the coefficient of variation (CV Demand). Second, we expressed production time through the workload that would result from producing N products with identical demand and production rates, $\rho_n = N\mu_n p_n$ (Load Factor). Third, with the production time given by mean demand and load factor, we derived the setup time from the ratio of setup to production time (Setup/Prod Time). Fourth, we fixed the lost sales cost and expressed the holding cost through the ratio of holding to lost sales cost (Holding/LS Cost). Finally, we set all setup costs to zero, since setup cost often represent no more than the cost of working time during a setup, which is already covered by the setup time.

Since the number of parameters increases proportionally in the number of products N, we defined experimental design variables which summarize an entire set of N parameter values. If we view each value of a set as a realization of a uniform random variable, we can describe the set by an average and a coefficient of variation, with the latter serving as a measure of parameter diversity, e.g., diversity in mean demand or lost sales cost. For our study, we fixed the averages, Avg Mean Demand and Avg Lost Sales Cost, to 5 and 100, respectively, and then sampled the *diversity factors*, Div Mean Demand and Div Lost Sales Cost, from the interval $[0.0, 0.5]$. Averages (Avg) and diversity factors (Div) of CV Demand, Load Factor, Setup/Prod Time, and Holding/LS Cost were sampled accordingly. The ranges of these design parameters are given in Table 1.

With Avg Mean Demand and Avg Lost Sales Cost fixed, a problem instance can now be described

by a ten-dimensional design point. For our numerical study, we generated 1000 design points for problems with $N \in \{3, 5, 10\}$ products by sampling a ten-dimensional Faure sequence. This gives us a so-called space-filling design which has the useful property that design points lie not only at the edges of the hypercube that spans the experimental area but also at its interior (Chen et al. 2006).

To decode a design point into a model configuration, we used a variant of *descriptive sampling* (Saliby 1990). Denote X as a uniform random variable, with average $\bar{X}$ and coefficient of variation (diversity factor) c_X, to describe a set of N parameters. Using descriptive sampling, a realization of a parameter x_n associated with the n-th product is then given by

$$x_n = x'_{\Theta(n)},\ x'_j = \bar{X} + \sqrt{6}\left(\tfrac{j-1}{N-1} - \tfrac{1}{2}\right)\bar{X}c_X,\ n, j \in \{1, \ldots, N\}, \tag{33}$$

with Θ serving as a random mapping from n to j, i.e., we shuffle. For example, for a Load Factor with average $\bar{\rho} = 0.5$ and coefficient of variation $c_\rho = 0.2$, one possible permutation of the set of parameters for a three-product problem would be $\{0.5, 0.378, 0.622\}$.

4.2. Implementation

We implemented the solution algorithms, the simulation model and our experiments in Java, and used SPSS 17 for our statistical analyses. As implementation of the CMA-ES algorithm, we used the Java source code provided by Hansen (2007). For all solution algorithms, we carried out pretests to optimize their algorithmic parameters which are summarized in Table 2. Note that for approximate value iteration (AVI), we have to define maximum inventory levels. We therefore first optimized the base-stock policy over an unbounded state space using direct policy search (DPS) and then set $\bar{y}_n = 1.2Y_n^{(2)}$. For policy evaluation, we generated a single sample path using common random numbers over 1,000,000 decision epochs, after an initial transient phase of 10,000 decision epochs. We then evaluated all policies that were optimized by AVI or DPS using this sample path.

		Num. of Products (N)		
		3	5	10
AVI	T	100,000,000	200,000,000	500,000,000
	γ	0.01	0.01	0.01
	a	1,000,000	2,000,000	5,000,000
	b	0.1	0.05	0.02
	K	20	20	20
	L	20	20	20
DPS	IK	900	2,500	10,000
	T	100,000	100,000	100,000
	e	1000	1000	1000
	δ	5	5	5

Table 2 Algorithmic parameters

4.3. Results

4.3.1. Influence of model parameters on average cost. We studied the influence of the design parameters on the expected average cost by conducting an analysis of variance (ANOVA). Since each additional product increases the total mean demand, the expected average cost increases proportionally in the number of products. To remove this effect, we refer to *cost per product* as the expected average cost divided by the number of products.

We ran separate ANOVAs for different policy groups: FCP, BSP, AVI and BEST. We refer to the group of fixed-cycle policies as FCP, base-stock policies as BSP and approximate value iteration with one of the two approximation schemes as AVI. For each simulated problem instance, we selected the lowest cost within a policy group for the analysis. Additionally, we created an auxiliary group, BEST, which contained the lowest known cost for each instance. The percentage of variance in cost per product that can be explained by a design parameter is measured by the coefficient of determination (r^2).

The results of the ANOVA are shown in Table 3. As can be seen from the last row (Linear Model), all factors together explain 75 to 82 percent of the variance in cost per product, irrespective of the policy group. Regardless of the solution method, the r^2 of all diversity factors is close to zero. This indicates that all policies were capable of compensating diversity in model parameters, so that the cost effect of diversity becomes negligible. The most important factors are Avg Load Factor, Avg Setup/Prod Time, and Avg Holding/LS Cost. Since Avg Load Factor and Avg Setup/Prod

	BEST		FCP		BSP		AVI		
Factor	r^2	F	r^2	F	r^2	F	r^2	F	df
Num. of Products	0.00	4.4	0.00	20.5	0.00	32.9	0.03	391.6	1
Div Mean Demand	0.00	9.3	0.00	41.0	0.01	150.3	0.00	1.0	1
Div Lost Sales Cost	0.00	41.1	0.00	21.8	0.00	0.2	0.00	51.5	1
Avg Hold/LS Cost	0.28	4720.6	0.29	4504.5	0.23	2728.0	0.22	3231.6	1
Div Hold/LS Cost	0.00	7.1	0.00	4.2	0.00	4.7	0.00	4.9	1
Avg Setup/Prod Time	0.22	3603.7	0.21	3341.5	0.19	2264.2	0.19	2907.4	1
Div Setup/Prod Time	0.00	5.0	0.00	4.3	0.00	1.9	0.00	1.1	1
Avg CV Demand	0.03	533.6	0.03	514.5	0.02	260.5	0.02	296.9	1
Div CV Demand	0.00	13.0	0.00	8.5	0.00	7.0	0.00	11.5	1
Avg Load Factor	0.28	4710.7	0.27	4268.7	0.29	3449.0	0.34	5157.2	1
Div Load Factor	0.00	0.4	0.00	0.7	0.00	7.1	0.00	9.2	1
Linear Model	0.82	1245.4	0.81	1161.3	0.75	812.1	0.81	1101.8	11

Sample Size = 3000, r^2 = coefficient of determination, F = F test statistic, df = degrees of freedom.

Table 3 ANOVA of the influence of design parameters on cost per product for different policies

Time largely affect system utilization, this indicates that, within the given range of parameters, utilization is a more important cost driver than variability. However, variability has a much larger influence on FCP (F=514.5) than on BSP (F=260.6) or AVI (F=296.9), which indicates that dynamic-cycle policies are better able to deal with demand uncertainty than fixed-cycle policies. Also, except for AVI, the number of products does not have a noteworthy impact on cost per product, which can be seen from the comparatively low F-values. We therefore expect the solution quality of FCP and BSP to remain fairly stable even for larger problems.

4.3.2. Policy evaluation. In our second analysis, we wanted to find out which solution method performs best. As before, we refer to the fixed-cycle policy as FCP (FCP_0 = common cycle solution, FCP_1 = fixed-cycle policy, FCP_2 = fixed-cycle policy with preemption) and to the base-stock policy as BSP (BSP_0 = Doll & Whybark heuristic, BSP_1 = base-stock policy, BSP_2 = can-order base-stock policy). For both types of policies, we used the less sophisticated policy as initial guess during direct policy search. We refer to AVI with the first approximation scheme as AVI_1 and with the second scheme as AVI_2. For all policies, we recorded the mean cost per product for 3, 5, and 10 products, as well as for low and high levels of Avg Load Factor (low: $\bar{\rho} < 0.6$, high: $\bar{\rho} \geq 0.6$) and Avg Setup/Prod Time (low: $\bar{\eta} < 51$, high: $\bar{\eta} \geq 51$). (For a justification, see Section

N	$\bar{\rho}$	$\bar{\eta}$	Size	*BEST*	FCP_0	FCP_1	FCP_2	BSP_0	BSP_1	BSP_2	AVI_1	AVI_2
3	low	low	255	*0.57*	0.74	0.61	0.61	0.91	0.64	0.62	0.90	**0.59**
		high	247	*0.88*	1.13	0.97	0.96	1.48	1.03	1.00	1.38	**0.90**
	high	low	249	*0.93*	1.51	1.02	1.02	1.59	1.15	1.10	1.17	**0.98**
		high	249	*1.65*	3.68	1.83	**1.80**	3.12	2.17	2.08	2.02	**1.81**
5	low	low	251	*0.51*	0.65	0.55	**0.53**	0.66	**0.53**	**0.53**	1.14	0.55
		high	250	*0.87*	1.05	0.93	**0.90**	1.21	0.93	0.92	1.31	0.92
	high	low	255	*0.96*	1.43	1.02	**0.98**	1.47	1.13	1.10	1.44	1.14
		high	244	*1.72*	2.98	1.81	**1.73**	2.79	2.06	2.02	2.44	2.02
10	low	low	254	*0.47*	0.63	0.56	0.50	0.54	0.47	**0.47**	1.17	0.56
		high	246	*0.82*	1.03	0.96	0.87	0.98	**0.83**	**0.83**	2.20	1.05
	high	low	251	*0.92*	1.27	1.04	**0.95**	1.23	1.00	0.99	2.06	1.41
		high	249	*1.75*	2.92	1.91	**1.77**	2.83	2.12	2.05	3.85	2.57
	Mean		3000	*1.00*	1.58	1.10	**1.05**	1.56	1.17	1.14	1.75	1.20

Table 4 Comparison of standardized mean cost per product for different policies and problem categories

4.3.3.) The mean costs per product in each problem category were standardized by the mean cost per product of BEST across all problem instances.

The results of the analysis are shown in Table 4. The lowest mean in each category is highlighted in bold. (Note that when the difference to the second lowest average is not significant, i.e., $p \geq 0.01$, both values are highlighted.) Overall, FCP_2 yields the lowest mean cost per product. For small problems with three products, AVI_2 returns the lowest mean, but looses its competitiveness as the problem size increases. Since AVI_1 yields a much higher mean than AVI_2 across all categories, we conclude that an approximate value function that is completely separable in the inventory state variables is insufficient to approximate the true value function of the SMDP. The heuristics, FCP_0 and BSP_0, are also not competitive. Moreover, we observe that the cost induced by FCP_2 are considerably lower than those of BSP_2 when both, Avg Load Factor and Avg Setup/Prod Time, are high. While the difference between FCP_1 and FCP_2 gets larger as the number of products increases, the improvement of using BSP_2 over BSP_1 is small.

Again, like in the ANOVA, we grouped the policies into FCP, BSP and AVI. For each group and each category, we then recorded the standardized means of the lowest in-group cost, as before (Mean). Additionally, we recorded the mean absolute deviation (MAD) of the lowest in-group cost

				Mean			MAD			Frequency		
N	$\bar{\rho}$	$\bar{\eta}$	Size	FCP	BSP	AVI	FCP	BSP	AVI	FCP	BSP	AVI
3	low	low	255	0.61	0.62	**0.59**	0.05	0.10	**0.03**	16%	42%	**42%**
		high	247	0.95	1.00	**0.89**	0.11	0.14	**0.03**	30%	14%	**57%**
	high	low	249	1.01	1.09	**0.97**	0.10	0.20	**0.07**	25%	21%	**54%**
		high	249	1.79	2.07	**1.73**	0.25	0.44	**0.14**	46%	6%	**48%**
5	low	low	251	**0.53**	**0.52**	0.55	**0.02**	0.04	0.04	20%	**64%**	16%
		high	250	**0.89**	0.92	0.91	**0.04**	0.08	0.06	**40%**	37%	23%
	high	low	255	**0.98**	1.10	1.13	**0.03**	0.18	0.18	**62%**	27%	11%
		high	244	**1.72**	2.01	1.98	**0.04**	0.34	0.28	**82%**	12%	6%
10	low	low	254	0.50	**0.47**	0.56	0.03	**0.01**	0.09	3%	**97%**	0%
		high	246	0.87	**0.82**	1.05	**0.05**	0.07	0.23	9%	**91%**	0%
	high	low	251	**0.95**	0.98	1.39	**0.04**	0.18	0.47	34%	**66%**	0%
		high	249	**1.76**	2.04	2.50	**0.05**	0.45	0.75	**65%**	35%	0%
	Mean		3000	**1.04**	1.13	1.18	**0.07**	0.23	0.23	36%	**43%**	21%

Table 5 Comparison of different groups of policies

from the lowest cost across all groups (BEST). For a parameter instance i, the absolute deviation of FCP is given by

$$|\text{Cost}_i(\text{FCP}) - \min\{\text{Cost}_i(\text{FCP}), \text{Cost}_i(\text{BSP}), \text{Cost}_i(\text{AVI})\}|.$$

We then computed the MAD across all instances within a category, omitting cases where the policy yields the lowest known cost, which gives us the MAD over those instances where the policy was not best. Also, for each category, we recorded the relative frequency of how often a group provided the lowest cost (Frequency).

The results of the analysis are shown in Table 5. FCP yields the overall lowest mean across all instances. With the exception of the three product case, it also returns the lowest mean in categories with a high Avg Load Factor. On the other hand, BSP has the highest number of instances where it is the best policy, in particular for problem instances with 10 products. Considering that FCP yields the lowest MAD overall, this implies that whenever BSP is worse its difference to the best policy must be larger on average than whenever FCP is worse.

Table 6 summarizes the average computational time of each policy measured on an Intel

	Total time (in sec.)						Time per state transition (in 10^{-6} sec.)					
N	FCP_1	FCP_2	BSP_1	BSP_2	AVI_1	AVI_2	FCP_1	FCP_2	BSP_1	BSP_2	AVI_1	AVI_2
3	19.8	20.1	19.3	49.5	43.7	70.2	0.22	0.22	0.21	0.55	0.44	0.70
5	58.4	60.1	57.7	144.5	135.0	368.3	0.23	0.24	0.23	0.58	0.68	1.84
10	271.8	284.4	271.0	650.3	728.0	5046.9	0.27	0.28	0.27	0.65	1.46	10.09

Table 6 Average computational times

Core2Duo with 3.0 GHz and 4 GB memory using the 64 bit versions of Java 6 and Windows 7. Columns 2 to 7 show the average total time, and additionally, since we use more state transitions for larger problems (see Table 2), columns 8 to 13 show the average time per state transition. We can see that AVI is in general more expensive than FCP or BSP, which is particularly true for AVI_2 with the highest computational cost. Also, while problem size has only a small effect on the time per state transition when using direct policy search, the cost increase for AVI is substantial. Nevertheless, computational times are still well within practical limits.

We conclude that, although a base-stock policy is more frequently better, a fixed-cycle policy is the more robust choice. Moreover, the fixed-cycle policies are generally better than the base-stock policies in highly utilized systems. In contrast, approximate value iteration only works well in systems with a small number of products, and only with a value function approximation that considers dependencies among inventory state variables, which is computationally much more expensive.

4.3.3. A discrete choice model for production policies. The previous analysis has shown that there are some model configurations where it is better to use a base-stock policy and others where it is better to use a fixed-cycle policy. To analyze the drivers that make one policy perform better than the other, we developed a discrete choice model, which can be described by the *logit* function

$$\text{logit}^{-1}(Y_i) = \frac{\exp(Y_i)}{1+\exp(Y_i)},\ Y_i = b_0 + b_1 X_{1i} + ... + b_k X_{ki}.$$

We used this model to specify the probability that BSP performs better than FCP. To estimate the coefficients of the logit function, we ran a binomial logistic regression. We used backwards stepwise regression with the design parameters as independent variables and 'BSP is best' as dependent

Predictor	b	Std Error	Wald χ^2	df	p-Value	Odds Ratio
Num. of Products	0.390	0.019	443.113	1	0.000	1.477
Avg Setup/Prod Time	-0.031	0.002	292.071	1	0.000	0.969
Avg Load Factor	-7.282	0.329	489.731	1	0.000	0.001
Avg Hold/LS Cost	-112.512	18.464	37.130	1	0.000	0.000
Div Mean Demand	-3.084	0.341	81.658	1	0.000	0.046
Constant	4.825	0.273	312.507	1	0.000	n/a
Test			χ^2	df	p-Value	
Log-Likelihood Ratio			1422.000	1	0.000	
Hosmer-Lemeshow			20.275	8	0.009	

Cox and Snell R^2=0.377, Nagelkerke R^2=0.504, b = beta coefficient, df = degrees of freedom, n/a = not applicable.

Table 7 Logistic regression analysis of the frequency for BSP having lower average cost than FCP

	Predicted		
Observed	FCP is best	BSP is best	% Correct
FCP is best	1277	308	80.6
BSP is best	328	1087	76.8
Overall			78.8

Table 8 Observed and predicted frequency for BSP having lower cost per product than FCP

variable. Starting with the variable with the lowest Wald statistic, we iteratively removed variables as long as the change in the log-likelihood ratio was not significant.

Table 7 summarizes the findings of the logistic regression analysis. The odds ratio of 1.477 indicates that a larger number of products makes it more likely for BSP to perform better than FCP. We can also see this tendency in Table 5, where the frequency of BSP as the best policy increases in the number of products. Avg Load Factor and Avg Setup/Prod Time, on the other hand, decrease the likelihood of BSP being the best policy (odds ratios < 1). Although Avg Holding/LS Cost and Div Mean Demand are significant, they are less important, which can be seen by their lower Wald statistics.

Table 8 compares the observed frequency for BSP having lower average cost than FCP to the frequency predicted by the discrete choice model. The prediction is accurate in about 80% of all cases, compared to 50% if we would randomly choose a policy. The standardized MAD between cost of BSP versus FCP in case of choosing the wrong policy is 0.033, compared to 0.07 when

we always choose FCP. This implies that even if the wrong policy was chosen the error would be smaller than if the most robust policy was used.

The results highlight the strengths and weaknesses of each policy. Systems with a large number of products, for instance, require flexible production policies, which increases the odds that a base-stock policy performs better. In highly utilized systems, however, the base-stock policy discriminates products with low setup times, as it always skips to the product with the lowest runout time. This effect increases the odds that a fixed-cycle policy performs better, as this policy strictly follows a predefined production sequence. This also confirms the finding of Vaughan (2007) that order-point policies outperform cyclical policies in cases where the number of products is large and system utilization low.

4.3.4. Comparison with optimal policy. In our final analysis, we take a look at the difference between policies obtained through simulation optimization and the optimal policy derived from relative value iteration (RVI). To keep problems computationally tractable for RVI, we only study problems with three products and fixed the maximum inventory level at $\bar{y}_n = 25$. We stopped RVI when the gap between lower and upper bound was less than one percent. Since the resulting inventory state space is too restrictive for most problems in the previous sample, we created a new sample, where we reset the intervals of the design parameters Avg Hold/LS Cost and Avg Setup/Prod Time to $[0.01, 0.1]$ and $[1, 20]$, respectively.

Table 9 shows the mean cost per product over these 250 instances, categorized and standardized as in Tables 4 and 5, with a new definition of low and high levels of Avg Setup/Prod Time (low: $\bar{\eta} < 10.5$, high: $\bar{\eta} \geq 10.5$). As in Table 4, the best heuristic policy over all is obtained by AVI_2, with cost per product less than 5 percent above the upper bound of RVI on average. For FCP_2, cost are less than 6 percent above, for BSP_2, less than 10 percent above the upper bound on average. Since the number of products only has a small influence on cost per product (see Table 3), we conjecture that the performance of BSP and FCP will be similar for larger problems. Also note that results for BSP_2 have improved, since the mean Avg Setup/Prod Time in the new sample is now lower than in the old one, which supports the findings of the logistic regression analysis.

N	$\bar{\rho}$	$\bar{\eta}$	Size	*RVI*	FCP_0	FCP_1	FCP_2	BSP_0	BSP_1	BSP_2	AVI_1	AVI_2
3	low	low	64	*1.92*	2.25	2.02	2.03	2.24	2.03	**2.00**	2.13	2.04
		high	63	*2.68*	2.95	2.80	2.77	2.93	2.81	2.78	2.95	**2.74**
	high	low	61	*2.67*	3.25	2.92	2.90	3.25	3.00	2.91	3.04	**2.84**
		high	62	*3.62*	4.68	3.88	3.75	4.69	4.25	4.21	3.94	**3.69**
Mean			250	*2.72*	3.30	2.92	2.87	3.29	3.04	2.99	3.03	**2.84**

Table 9 Comparison of standardized mean average cost for tractable model instances with 3 products

5. Conclusion

We have studied simulation optimization methods for the stochastic economic lot scheduling problem. Based on a large-scale numerical study, we found that the classic ADP approach of approximate value iteration and stochastic gradient updates is not competitive for larger problems. Initial tests with alternative ADP approaches did not perform better. Using (recursive) least squares in place of stochastic gradients to update the value function had been considered but was computationally too expensive. Replacing the piecewise-constant approximation with a linear interpolation also did not significantly improve approximation quality. Applying soft-max or epsilon-greedy exploration during sampling provided no advantage over pure exploitation. These results turn the SELSP into a good benchmark for future research in ADP.

Simple production policies optimized by a global search algorithm provide the most comprehensible and robust solution to this problem. In our numerical study, base-stock policies were more often the better choice, but were outperformed by fixed-cycle policies in highly utilized systems. The most reliable choice overall was a fixed-cycle policy which preempts production of a product as soon as the inventory level of an upcoming product drops below a certain level. When comparing this policy with the optimal policy for small, tractable problem instances, we observe that costs are less than six percent above the minimal costs on average.

An interesting extension of the current model would be to consider sequence-dependent setup times, which would require a reformulation of the value function approximation as well as the development of new control policies. Other extensions, such as backlogs instead of lost sales or uncertain process times could be motivated by specific applications but do not significantly change

problem complexity with respect to simulation optimization.

References

Altiok, T., G.A. Shiue. 1994. Single-stage, multi-product production/inventory systems with backorders. *IIE Transactions* **26**(2) 52–61.

Altiok, T., G.A. Shiue. 1995. Single-stage, multi-product production/inventory systems with lost sales. *Naval Research Logistics* **42**(6) 889–913.

Anupindi, R., S. Tayur. 1998. Managing stochastic multiproduct systems: model, measures, and analysis. *Operations Research* **46**(3) 98–111.

Bertsekas, D.P. 2007. *Dynamic Programming and Optimal Control*, vol. 2. 3rd ed. Athena Scientific, Belmont, Massachusetts.

Bourland, K.E., C.A. Yano. 1994. The strategic use of capacity slack in the economic lot scheduling problem with random demand. *Management Science* **40** 1690–1704.

Brander, P., R. Forsberg. 2006. Determination of safety stocks for cyclic schedules with stochastic demands. *International Journal of Production Economics* **104**(2) 271–295.

Chen, C.C.P., K.-L. Tsui, R.R. Barton, M. Meckesheimer. 2006. A review on design, modeling and applications of computer experiments. *IIE Transactions* **38**(4) 273–291.

Davis, Samuel G. 1990. Scheduling economic lot size production runs. *Management Science* **36**(8) 985–998.

Dobson, G. 1987. The economic lot-scheduling problem: achieving feasibility using time-varying lot sizes. *Operations Research* **35**(5) 764–771.

Elmaghraby, S.E. 1978. The economic lot scheduling problem (ELSP): review and extensions. *Management Science* **24**(6) 587–598.

Federgruen, A., Z. Katalan. 1996. The stochastic economic lot scheduling problem: cyclical base-stock policies with idle times. *Management Science* **42**(6) 783–796.

Federgruen, A., Z. Katalan. 1998. Determining production schedules under base-stock policies in single facility multi-item production systems. *Operations Research* **46**(6) 883–898.

Feeney, G.J., C.C. Sherbrooke. 1966. The (s-1,s) inventory policy under compound Poisson demand. *Management Science* **12**(5) 391–411.

Gallego, G. 1990. Scheduling the production of several items with random demands in a single facility. *Management Science* **36**(12) 1579–1592.

Gascon, A., R.C. Leachman, P. Lefrancois. 1994. Multi-item, single-machine scheduling problem with stochastic demands: a comparison of heuristics. *International Journal of Production Research* **32**(3) 583–596.

Graves, S.C. 1980. The multi-product production cycling problem. *AIIE Transactions* **12**(3) 233–240.

Hansen, N. 2007. CMA-ES source code in Java. URL `http://www.lri.fr/~hansen/cmaes_java.jar`.

Hansen, N., A. Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* **9**(2) 159–195.

Hsu, W.-L. 1983. On the general feasibility test of scheduling lot sizes for several products on one machine. *Management Science* **29**(1) 93–105.

Krieg, G.N., H. Kuhn. 2002. A decomposition method for multi-product kanban systems with setup times and lost sales. *IIE Transactions* **34** 613–625.

Markowitz, D.M., M.I. Reiman, L.M. Wein. 2000. The stochastic economic lot scheduling problem: heavy traffic analysis of dynamic cyclic policies. *Operations Research* **48**(1) 136–154.

Paternina-Arboleda, C.D., T.K. Das. 2005. A multi-agent reinforcement learning approach to obtaining dynamic control policies for stochastic lot scheduling problem. *Simulation Modelling Practice and Theory* **13**(5) 389–406.

Powell, W.B. 2007. *Approximate Dynamic Programming*. Wiley series in probability and statistics, Wiley-Interscience.

Qiu, J., R. Loulou. 1995. Multiproduct production/inventory control under random demands. *IEEE Transactions on Automatic Control* **40**(2) 350–356.

Saliby, E. 1990. Descriptive sampling: a better approach to Monte Carlo simulation. *The Journal of the Operational Research Society* **41**(12) 1133–1142.

Schweitzer, P.J. 1971. Iterative solution of the functional equations of undiscounted Markov renewal programming. *Journal of Mathematical Analysis and Applications* **34**(3) 495–501.

Sox, C.R., P.L. Jackson, A. Bowman, J.A. Muckstadt. 1999. A review of the stochastic lot scheduling problem. *International Journal of Production Economics* **62**(3) 181 – 200.

Vaughan, T.S. 2007. Cyclical schedules vs. dynamic sequencing: Replenishment dynamics and inventory efficiency. *International Journal of Production Economics* **107**(2) 518–527.

Wagner, M., S.R. Smits. 2004. A local search algorithm for the optimization of the stochastic economic lot scheduling problem. *International Journal of Production Economics* **90**(3) 391–402.

Winands, E.M.M., I.J.B.F. Adan, G.J. van Houtum. 2011. The stochastic economic lot scheduling problem: a survey. *European Journal of Operational Research* **210**(1) 1–9.