An Approximate Dynamic Programming Approach for a Product Distribution Problem

Abstract

This paper proposes an approximate dynamic programming-based method for optimizing the distribution operations of a company manufacturing a certain product in numerous plants and distributing it to different regional markets for sale. The production at each plant and in each time period follows a nonstationary stochastic process. The product can be stored at the plants or shipped to a regional market to serve the random demand. The proposed solution method formulates the problem as a dynamic program and uses approximations of the value function. We develop a tractable procedure to compute the marginal worth of a unit of inventory at a production plant. This quantity is used to update the value function approximations in an iterative improvement scheme. We numerically show that our method yields high quality solutions.

Keywords: Inventory, distribution, approximate dynamic programming.

Managing inventories in supply chain systems with geographically distributed manufacturing facilities requires careful coordination. While planning the delivery of products to the customers, one should consider many factors, such as the current inventory levels, the forecasts of customer demands, the production capacities and the forecasts of future production quantities. The decisions for different manufacturing facilities and for different time periods display complex interactions, and the models attempting to capture these interactions can easily get intractable.

In this paper, we consider the distribution operations of a company producing a product in numerous production plants and shipping it to different customer locations for sale. In each time period, a certain amount of product becomes available at each production plant. Before observing the realization of random customer demands, the company has to decide what proportion of this amount should be stored at the production plants and what proportion should be shipped to the customer locations. Once a certain amount of product is shipped, a newsvendor-type profit is obtained at each customer location: Revenue is gained on the amount sold, shortage cost is incurred on the unsatisfied demand. The left over product at the customer locations cannot be stored and has to be disposed at a salvage value.

Our work is motivated by the distribution problem faced by a company processing fresh produce that will eventually be sold at local markets. These markets are setup outdoors for short periods of time, and hence, the perishable product cannot be stored at these locations.

However, the processing plants are equipped with storage facilities that can store the product for relatively longer periods of time. Depending on the supply of fresh produce, the production quantities fluctuate over time and are not necessarily deterministic. This creates a need to look ahead and plan to store the product at the processing plants in order to avoid situations where the company will not be able to serve a profitable market in a future time period.

The solution method we propose formulates the problem as a dynamic program and replaces the value functions with tractable approximations. We develop a new method to compute the marginal worth of an incremental unit of product at a production plant in a certain time period. This quantity, referred to as the "policy gradient," is used to iteratively improve the value function approximations. An important strength of our strategy is that it is purely sampling-based and does not require the knowledge of the transition probabilities. Furthermore, under our value function approximation scheme, the problem reduces to solving sequences of min-cost network flow problems, which can be done very efficiently.

Our basic approach builds on previous research. Godfrey & Powell (2001) present an iterative method to approximate the value functions arising from two-time-period resource allocation problems. Later, in Godfrey & Powell (2002), they extend this idea to multi-time-period dynamic fleet management problems. (Also, see Powell & Topaloglu (2003) and references therein for an overview of the research that precedes and follows this work.) This paper extends this body of research in three ways: 1) Powell, Ruszczynski & Topaloglu (to appear) propose a new method to approximate the value functions arising from two-time-period resource allocation problems. Their method possesses convergence properties, and is, arguably, easier to implement than the one used by Godfrey & Powell (2002). This paper investigates the effectiveness of this new method in a multi-time-period product distribution setting. 2) We introduce the notion of policy gradients that measure the worth of an incremental unit of product at a certain location and in a certain time period. We provide a tractable algorithm to compute the policy gradients and show how they can be used to improve the value function approximations. 3) When it comes to multi-time-period problems, our theoretical understanding of value function approximation techniques is limited, and what works for one problem class does not necessarily work for another. Previous research shows that these techniques are quite powerful in the dynamic fleet management context. In this paper, we point out another problem class for which dynamic programming approximation

is of value.

The literature dealing with the spatial allocation of inventories is quite rich, and there exist a variety of approaches including inventory-theoretic analyses and stochastic dynamic programming-based bounding and approximation procedures. The reader is referred to Karmarkar (1981), Federgruen & Zipkin (1984), Karmarkar (1987), Jackson (1988), Fumero & Vercellis (1994), Cheung & Powell (1996) and Rappold & Muckstadt (2000) for representative applications of these approaches. Of particular interest is the work by Cheung & Powell (1996), where the authors formulate a problem similar to ours as a dynamic program and use approximations of the value function. Their strategy constructs the value function approximations by working backwards through time. When constructing the approximation for a certain time period, they simultaneously consider all possible realizations of the random variables in that time period. To contrast their approach with ours, we work forwards through time, and in a certain time period, we use only one sample realization of the random variables at that time period. However, we make multiple passes over the whole planning horizon, adjusting and improving the value function approximations after each pass.

The approximate dynamic programming field has been active within the past two decades. A majority of the recent work seeks to approximate the value function $V(\cdot)$ as $V(z) \approx \sum_{i \in \mathcal{K}} \gamma_i \hat{V}_i(z)$, where $\{\hat{V}_i(\cdot) : i \in \mathcal{K}\}$ are fixed basis functions and $\{\gamma_i : i \in \mathcal{K}\}$ are scalar multipliers. The challenge is to find a set of values for $\{\gamma_i : i \in \mathcal{K}\}$ such that $\sum_{i \in \mathcal{K}} \gamma_i \hat{V}_i(z)$ is a "good" approximator of $V(z)$. For example, temporal difference and Q-learning methods use sampled trajectories to iteratively update the multipliers. (See Bertsekas & Tsitsiklis (1996) for a general overview and Watkins & Dayan (1992), Tsitsiklis & Van Roy (1997) for convergence results.) On the other hand, the approximate linear programming approach finds the values of $\{\gamma_i : i \in \mathcal{K}\}$ by solving a linear program. These linear programs can be very large since they contain one constraint for every state-action pair, and hence, are usually solved only approximately (de Farias & Van Roy (2003)). In addition to advances in theory, numerous successful applications appeared in inventory control (Van Roy, Bertsekas, Lee & Tsitsiklis (1997)), inventory routing (Kleywegt, Nori & Savelsbergh (2002), Adelman (2004)) and dynamic fleet management (Godfrey & Powell (2002)).

In this paper we make the following contributions: 1) We propose a tractable, approximate

dynamic programming-based solution algorithm for a stochastic, nonstationary, multiple-plant, multiple-customer inventory allocation problem. 2) In order to update the parameters of the value function approximations, we develop a policy gradient approach that computes the worth of an incremental unit of product at a certain location in a certain time period. 3) Numerical experiments show that our method yields high quality solutions. They also provide insight into the conditions that render stochastic models more effective than simpler deterministic ones.

In Section 1, we describe the problem and formulate it as a dynamic program. Section 2 shows how to approximate the value function in a tractable manner, and Section 3 introduces the idea of updating and improving these approximations using samples of the random quantities. The updating procedure in Section 3 is not practical since it assumes the knowledge of the exact value function. Section 4 develops a tractable procedure by introducing the concept of policy gradients. Section 5 presents our numerical results showing that the proposed method yields high quality solutions and characterizing the situations where it pays off to use a stochastic model.

1 Problem formulation

We have a set of production plants producing a certain product to satisfy the demand occurring at a set of local markets, which we refer to as customer locations. At the beginning of a time period, a random amount of product is produced at each plant. (The production decisions at the plants are outside the scope of the problem and are simply modeled as random processes, possibly with some correlation among the plants.) Before observing the demand at the customer locations, we have to decide how much product to ship from each plant to each customer location and how much product to hold at each plant. The portion of the demand that is not met is lost and we pay a shortage cost. The product can be stored at the plants and the left over product at the customer locations is disposed at a salvage value. The demand and the production occur in discrete quantities and also we are allowed to ship in discrete quantities only. The objective is to maximize the expected profit over a finite horizon. Basic elements of our problem are

$\mathcal{T} =$ Set of time periods in the planning horizon, $\{1, \ldots, T\}$ for some finite T.

$\mathcal{P} =$ Set of production plants.

$\mathcal{C} =$ Set of customer locations.

$c_{ijt} =$ Cost of shipping one unit of product from production plant i to customer location j in time period t.

$\rho_{jt} =$ Revenue per unit of product sold at customer location j in time period t.

$\sigma_{jt} =$ Salvage value per unit of unsold product at customer location j in time period t.

$\pi_{jt} =$ Shortage cost of not being able to satisfy a unit of demand at customer location j in time period t.

$h_{it} =$ Holding cost per unit of product held at production plant i in time period t.

$P_{it} =$ Random variable for the production quantity at production plant i in time period t.

$D_{jt} =$ Random variable for the demand at customer location j in time period t.

$x_{ijt} =$ Amount of product shipped from production plant i to customer location j in time period t.

$y_{jt} =$ Total amount of product shipped to customer location j in time period t.

$z_{it} =$ Amount of product held at production plant i in time period t.

$I_{it} =$ Beginning inventory at production plant i in time period t right after the realization of the random production P_{it}.

By suppressing one or more of the indices in the variables defined above, we denote a vector composed of the elements ranging over the suppressed indices. For example, $x_t = \{x_{ijt} : i \in \mathcal{P}, j \in \mathcal{C}\}$ and $z_t = \{z_{it} : i \in \mathcal{P}\}$. In the remainder of this section, we define the set of feasible decisions and the one-period profit function, and formulate the problem as a dynamic program.

Set of feasible decisions – Decisions are made after the production for the current time period is realized. Given the beginning inventory I_t in time period t, the set of feasible decisions is

$$\begin{aligned} \mathcal{Y}(I_t) = \{(x_t, y_t, z_t) : \sum_{j \in \mathcal{C}} x_{ijt} + z_{it} &= I_{it} && \text{for all } i \in \mathcal{P} \\ \sum_{i \in \mathcal{P}} x_{ijt} - y_{jt} &= 0 && \text{for all } j \in \mathcal{C} \\ x_{ijt}, y_{jt}, z_{it} &\in \mathbb{Z}_+ && \text{for all } i \in \mathcal{P}, j \in \mathcal{C}\}. \end{aligned}$$

Note that $I_t = z_{t-1} + P_t$ and we define $\mathcal{X}(z_{t-1}, P_t) = \mathcal{Y}(z_{t-1} + P_t)$, which will be useful shortly.

One-period expected profit – If we ship y_{jt} units of product to customer location j and a total demand of D_{jt} units is realized at this location, then the profit we obtain is

$$F_{jt}(y_{jt}, D_{jt}) = \rho_{jt} \min\{y_{jt}, D_{jt}\} + \sigma_{jt} \max\{y_{jt} - D_{jt}, 0\} - \pi_{jt} \max\{D_{jt} - y_{jt}, 0\}.$$

Letting $F_{jt}(y_{jt}) = \mathbb{E}\{F_{jt}(y_{jt}, D_{jt})\}$, $F_{jt}(y_{jt})$ becomes the expected profit that can be obtained at customer location j by shipping y_{jt} units of product in time period t. If the random variable D_{jt} takes integer values and $\rho_{jt} + \pi_{jt} \geq \sigma_{jt}$, $F_{jt}(\cdot)$ can be shown to be piecewise-linear and concave with the points of nondifferentiability forming a subset of positive integers. In this case, $F_{jt}(\cdot)$ is completely described by its right-hand slopes at integers and $F_{jt}(0) = -\pi_{jt}\mathbb{E}\{D_{jt}\}$. Letting $f_{jt}(y_{jt})$ be the right-hand slope of $F_{jt}(\cdot)$ at an integer point y_{jt}, $f_{jt}(y_{jt})$ can be computed by

$$\begin{aligned}
F_{jt}(y_{jt}+1, D_{jt}) - F_{jt}(y_{jt}, D_{jt}) &= \begin{cases} \sigma_{jt} & \text{if } D_{jt} \leq y_{jt} \\ \rho_{jt} + \pi_{jt} & \text{if } y_{jt} + 1 \leq D_{jt}, \end{cases} \\
f_{jt}(y_{jt}) = F_{jt}(y_{jt}+1) - F_{jt}(y_{jt}) &= \mathbb{E}\left\{ F_{jt}(y_{jt}+1, D_{jt}) - F_{jt}(y_{jt}, D_{jt}) \right\} \qquad (1) \\
&= \sum_{s=0}^{y_{jt}} \sigma_{jt}\ \mathbb{P}\{D_{jt} = s\} + \sum_{s=y_{jt}+1}^{\infty} (\rho_{jt} + \pi_{jt})\ \mathbb{P}\{D_{jt} = s\} \\
&= \sigma_{jt}\ \mathbb{P}\{D_{jt} \leq y_{jt}\} + (\rho_{jt} + \pi_{jt})\ \mathbb{P}\{D_{jt} \geq y_{jt} + 1\}.
\end{aligned}$$

Shortly, we will use $F_{jt}(\cdot)$ in the objective function of a min-cost network flow problem and Figure 1 shows how to do this. Between nodes a and b, we define one arc with cost $f_{jt}(s)$ and upper bound 1 for all $s = 0, 1, \ldots$. Since $\sum_{s=0}^{y_{jt}-1} f_{jt}(s) + F_{jt}(0) = F_{jt}(y_{jt})$, if the total flow coming into node a is y_{jt}, then the costs incurred by the flows on all of these arcs is equal to $F_{jt}(y_{jt}) - F_{jt}(0)$. From (1), it is easy to see that if D_{jt} is bounded by d_{jt}, then $F_{jt}(\cdot)$ is a linear function with slope σ_{jt} over the interval $[d_{jt}, \infty)$. Therefore, the maximum number of arcs required is $d_{jt} + 1$. (See Nemhauser & Wolsey (1988) for more on using piecewise-linear functions in linear programs.)

Given this characterization of $F_{jt}(\cdot)$, we define the one-period expected profit function as

$$p_t(x_t, y_t, z_t) = \sum_{j \in \mathcal{C}} F_{jt}(y_{jt}) - \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{C}} c_{ijt}\ x_{ijt} - \sum_{i \in \mathcal{P}} h_{it}\ z_{it}.$$

Dynamic programming formulation – Assuming the random variables $\{P_t : t \in \mathcal{T}\}$ are independent, the problem can be formulated as a dynamic program by using I_t as the state

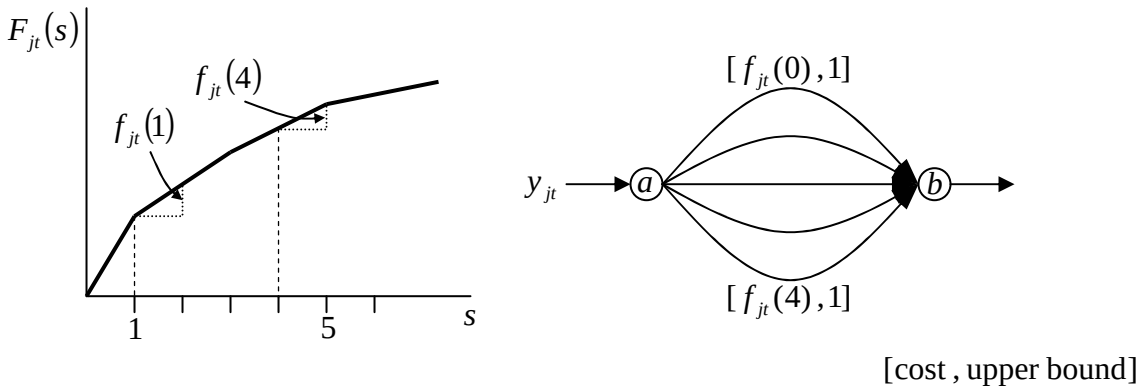

Figure 1: Representing $F_{jt}(\cdot)$ in min-cost network flow problems.

variable in time period t. The value functions $\{G_t(\cdot) : t \in \mathcal{T}\}$ have to satisfy the optimality equation

$$G_t(I_t) = \max_{(x_t,y_t,z_t)\in\mathcal{Y}(I_t)} p_t(x_t, y_t, z_t) + \mathbb{E}\{G_{t+1}(z_t + P_{t+1})\}.$$

However, solving these types of optimality equations through classical backward recursion techniques (see, for example, Puterman (1994)) is usually intractable due to the well-know "curse of dimensionality." Instead, we propose using approximations of the value functions. Denoting the approximation of the value function $G_{t+1}(\cdot)$ by $\hat{G}_{t+1}(\cdot)$, one might consider solving

$$\max_{(x_t,y_t,z_t)\in\mathcal{Y}(I_t)} p_t(x_t, y_t, z_t) + \mathbb{E}\{\hat{G}_{t+1}(z_t + P_{t+1})\} \tag{2}$$

to make the decisions for time period t for any given value of initial inventory I_t. If $\hat{G}_{t+1}(\cdot)$ is a "good approximation" to $G_{t+1}(\cdot)$, we may hope that solving (2) will yield good solutions.

However, computing the expectation in (2) can be problematic in practical applications due to dependencies between the random variables involved or the structure of the approximation. In order to alleviate this complication, we use an alternative dynamic programming formulation that uses (z_{t-1}, P_t) as the state variable in time period t. Seemingly, this state variable complicates the problem since the value function clearly depends on $I_t = z_{t-1} + P_t$ rather than on z_{t-1} and P_t separately, but it will be useful in dealing with the expectation. Now the value functions have to satisfy the optimality equation

$$V_t(z_{t-1}, P_t) = \max_{(x_t,y_t,z_t)\in\mathcal{X}(z_{t-1},P_t)} p_t(x_t, y_t, z_t) + \mathbb{E}\{V_{t+1}(z_t, P_{t+1})\}.$$

$\mathbb{E}\{V_t(z_{t-1}, P_t)\}$ is a function of z_{t-1} only and we set $V_t(z_{t-1}) = \mathbb{E}\{V_t(z_{t-1}, P_t)\}$. The optimal decisions for any value of z_{t-1} and realization of P_t can be found by solving

$$V_t(z_{t-1}, P_t) = \max_{(x_t, y_t, z_t) \in \mathcal{X}(z_{t-1}, P_t)} p_t(x_t, y_t, z_t) + V_{t+1}(z_t). \tag{3}$$

(Note that $V_{t+1}(\cdot)$ is a function of the decisions made at time period t, but we still use the time index $t+1$ for notational consistency.) We now replace the value function $V_{t+1}(\cdot)$ by an approximation $\hat{V}_{t+1}(\cdot)$, and solve the following problem to make the decisions for time period t:

$$\max_{(x_t, y_t, z_t) \in \mathcal{X}(z_{t-1}, P_t)} p_t(x_t, y_t, z_t) + \hat{V}_{t+1}(z_t). \tag{4}$$

2 Dynamic programming approximations

In order to select the functional forms for the value function approximations, we start with a well-known result that can be shown by backwards induction over time (see Karmarkar (1981)):

Property 1 *$V_t(z_{t-1})$ is a concave function of z_{t-1} in the sense that for any $z^0, z^1, z^2 \in \mathbb{Z}_+^{|\mathcal{P}|}$, $0 \leq \alpha \leq 1$, such that $z^0 = \alpha z^1 + (1-\alpha) z^2$, we have $V_t(z^0) \geq \alpha V_t(z^1) + (1-\alpha) V_t(z^2)$.*

An immediate implication of this property is that an incremental unit of product stored at a production plant yields decreasing marginal profit, that is $V_t(z_{t-1}) - V_t(z_{t-1} - e_i) \geq V_t(z_{t-1} + e_i) - V_t(z_{t-1})$, where we use e_i to denote the $|\mathcal{P}|$-dimensional unit vector with all 0 elements except for a 1 in the element corresponding to $i \in \mathcal{P}$. To mimic this property, we use concave approximations. Furthermore, for computational tractability, we use separable approximations. Therefore, our value function approximations take the form

$$\hat{V}_t(z_{t-1}) = \sum_{i \in \mathcal{P}} \hat{V}_{it}(z_{i,t-1}),$$

where each $\hat{V}_{it}(z_{i,t-1})$ is a piecewise-linear, concave function of $z_{i,t-1}$ with points of nondifferentiability being a subset of positive integers. If the cumulative production at plant i up to time period t is bounded by b_{it}, then the relevant domain for $\hat{V}_{it}(\cdot)$ is $[0, b_{it}]$. Letting $\hat{v}_{it}(s) = \hat{V}_{it}(s+1) - \hat{V}_{it}(s)$, $\hat{V}_{it}(\cdot)$ can be represented in a min-cost network flow problem using an argument similar to the one in Figure 1. We also note that the concavity of $\hat{V}_{it}(\cdot)$ implies that $\hat{v}_{it}(s) \geq \hat{v}_{it}(s+1)$ for all $s = 0, 1, \ldots, b_{it} - 1$. Letting $\{u_{jt}(s) : s = 0, \ldots, d_{jt}\}$ and $\{w_{i,t+1}(s) : s = 0, \ldots, b_{i,t+1}\}$ be the flow

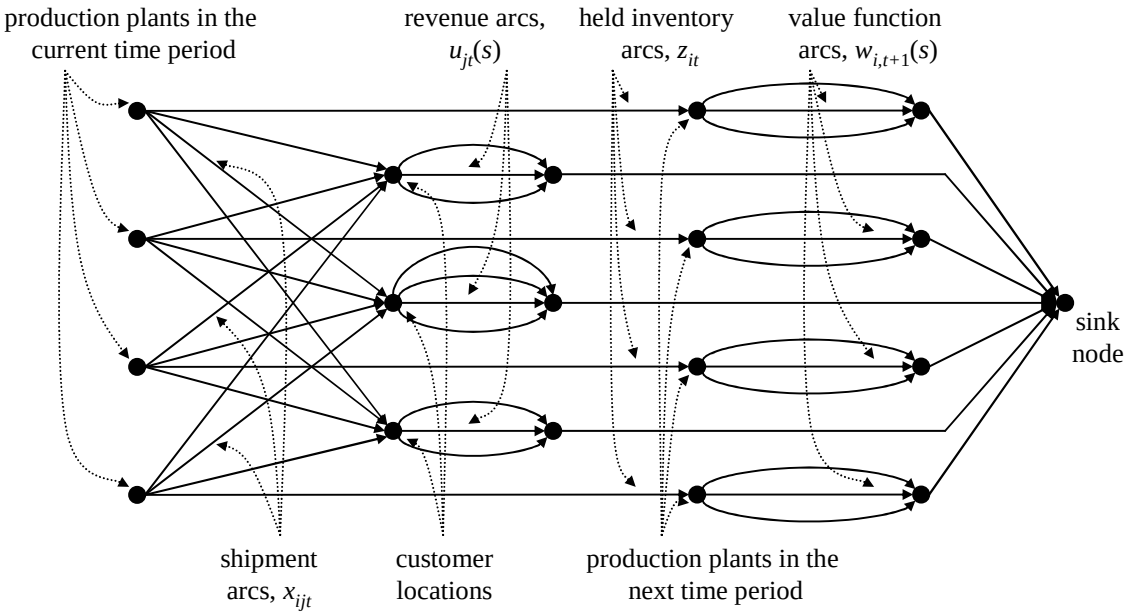

Figure 2: Problem (5)-(12) is a min-cost network flow problem.

variables associated with the set of arcs characterizing $F_{jt}(\cdot)$ and $\hat{V}_{i,t+1}(\cdot)$, problem (4) becomes

$$\max \sum_{j\in\mathcal{C}} \sum_{s=0}^{d_{jt}} f_{jt}(s)\ u_{jt}(s) - \sum_{i\in\mathcal{P}} \sum_{j\in\mathcal{C}} c_{ijt}\ x_{ijt} - \sum_{i\in\mathcal{P}} h_{it}\ z_{it} + \sum_{i\in\mathcal{P}} \sum_{s=0}^{b_{i,t+1}} \hat{v}_{i,t+1}(s)\ w_{i,t+1}(s) \tag{5}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{C}} x_{ijt} + z_{it} = z_{i,t-1} + P_{it} \qquad \text{for all } i \in \mathcal{P} \tag{6}$$

$$\sum_{i\in\mathcal{P}} x_{ijt} - y_{jt} = 0 \qquad \text{for all } j \in \mathcal{C} \tag{7}$$

$$y_{jt} - \sum_{s=0}^{d_{jt}} u_{jt}(s) = 0 \qquad \text{for all } j \in \mathcal{C} \tag{8}$$

$$z_{it} - \sum_{s=0}^{b_{i,t+1}} w_{i,t+1}(s) = 0 \qquad \text{for all } i \in \mathcal{P} \tag{9}$$

$$0 \le u_{jt}(s) \le 1 \qquad \text{for all } j \in \mathcal{C}, s = 0, \ldots, d_{jt} \tag{10}$$

$$0 \le w_{i,t+1}(s) \le 1 \qquad \text{for all } i \in \mathcal{P}, s = 0, \ldots, b_{i,t+1} \tag{11}$$

$$x_{ijt}, y_{jt}, z_{it} \in \mathbb{Z}_+ \qquad \text{for all } i \in \mathcal{P}, j \in \mathcal{C}. \tag{12}$$

Note that constraints (7) and (8) can be combined to get $\sum_{i\in\mathcal{P}} x_{ijt} = \sum_{s=0}^{d_{jt}} u_{jt}(s)$ for all $j \in \mathcal{C}$. Problem (5)-(12) is the min-cost network flow problem shown in Figure 2. Constraints (6) are the flow balance constraints for the nodes on the left side of the figure. The combined form

Step 1 Initialize iteration counter $n = 1$. Initialize $\hat{V}_t^n(\cdot)$ to 0 for all $t \in \mathcal{T}$.

Step 2 Initialize time counter $t = 1$ and $z_{t-1}^n = \mathbf{0}$.

Step 3 Sample a realization of P_t, say $\hat{P}_t^n$.

Step 4 Solve problem (4) corresponding to z_{t-1}^n and sampled $\hat{P}_t^n$ to set

$$(x_t^n, y_t^n, z_t^n) = \underset{(x_t,y_t,z_t)\in\mathcal{X}(z_{t-1}^n,\hat{P}_t^n)}{\arg\max} p_t(x_t, y_t, z_t) + \hat{V}_{t+1}^n(z_t).$$

Step 5 Set $t = t + 1$. If $t \leq T$, go to Step 3.

Step 6 Update the value function approximations by using the samples of the random quantities. For the moment, we denote this by $(\hat{V}_1^{n+1}(\cdot), \ldots, \hat{V}_T^{n+1}(\cdot)) = \mathcal{U}(\hat{V}_1^n(\cdot), \ldots, \hat{V}_T^n(\cdot), \hat{P}_1^n, \ldots, \hat{P}_T^n)$.

Step 7 Set $n = n + 1$. If another iteration is needed, go to Step 2.

Figure 3: The general solution methodology.

of constraints (7) and (8) are the flow balance constraints for the nodes labeled as "customer locations." Each set of parallel arcs leaving one of these nodes represent the profit function $F_{jt}(\cdot)$ for customer location j. Constraints (9) are the flow balance constraints for the nodes labeled as "production plants in the next time period." The parallel arcs leaving these nodes represent the value function approximations. Finally, the node labeled as "sink node" has supply $-\sum_{i\in\mathcal{P}}(z_{i,t-1} + P_{it})$. Its flow balance constraint is redundant and not shown in problem (5)-(12).

The solution methodology we propose starts with a set of value function approximations and iteratively tries to improve these approximations by using samples of the random quantities. This idea is summarized in Figure 3. In this figure, the function $\mathcal{U}(\cdot)$ is a high-level operator that takes the approximations and the realizations of the random quantities for the current iteration and updates the approximations. While updating, it is important to maintain the concavity of the successive approximations so that problem (4) can be solved as a min-cost network flow problem in the next iteration. The next two sections describe the nature of $\mathcal{U}(\cdot)$ in detail.

3 Updating the value function approximations

At iteration n, we denote the value function approximations by $\{\hat{V}_1^n(\cdot), \ldots, \hat{V}_T^n(\cdot)\}$ and the samples of the production quantities by $\{\hat{P}_1^n, \ldots, \hat{P}_T^n\}$. We also use $\{(x_1^n, y_1^n, z_1^n), \ldots, (x_T^n, y_T^n, z_T^n)\}$ to

denote the decisions made in each time period by using the approximations $\{\hat{V}_1^n(\cdot), \ldots, \hat{V}_T^n(\cdot)\}$ and the samples $\{\hat{P}_1^n, \ldots, \hat{P}_T^n\}$ in problem (4). The idea behind the updating procedure is to use $\{\hat{V}_1^n(\cdot), \ldots, \hat{V}_T^n(\cdot)\}$ and $\{\hat{P}_1^n, \ldots, \hat{P}_T^n\}$ to obtain a set of value function approximations $\{\hat{V}_1^{n+1}(\cdot), \ldots, \hat{V}_T^{n+1}(\cdot)\}$ that are better approximators of the exact value functions $\{V_1(\cdot), \ldots, V_T(\cdot)\}$.

Each of our value function approximations is characterized by a sequence of slopes and we would like to use gradient information regarding the value functions to update the approximations. Assume that for all $i \in \mathcal{P}$ and $t \in \mathcal{T}$, we are able to obtain

$$\phi_t^n(e_i) = V_t(z_{t-1}^n + e_i, \hat{P}_t^n) - V_t(z_{t-1}^n, \hat{P}_t^n),$$

where $V_t(z_{t-1}^n, \hat{P}^n)$ is as defined in (3). Note that $\phi_t^n(e_i)$ gives the incremental worth of a product at production plant i at the beginning of time period t under the optimal policy.

Our approximation strategy approximates $V_t(z_{t-1}^n + e_i) - V_t(z_{t-1}^n)$ by $\hat{V}_t^n(z_{t-1}^n + e_i) - \hat{V}_t^n(z_{t-1}^n)$. The former can be written as

$$V_t(z_{t-1}^n + e_i) - V_t(z_{t-1}^n) = \mathbb{E}\left\{V_t(z_{t-1}^n + e_i, \hat{P}_t^n) - V_t(z_{t-1}^n, \hat{P}_t^n)\right\} = \mathbb{E}\{\phi_t^n(e_i)\},$$

whereas the separability of $\hat{V}_t^n(\cdot)$ implies

$$\hat{V}_t^n(z_{t-1}^n + e_i) - \hat{V}_t^n(z_{t-1}^n) = \hat{V}_{it}^n(z_{i,t-1}^n + 1) - \hat{V}_{it}^n(z_{i,t-1}^n) = \hat{v}_{it}^n(z_{i,t-1}^n).$$

Therefore, $\hat{v}_{it}^n(z_{i,t-1}^n)$ is an approximator of $\mathbb{E}\{\phi_t^n(e_i)\}$ and we want to use $\phi_t^n(e_i)$ to adjust the slope of $\hat{V}_{it}^n(\cdot)$ at $z_{i,t-1}^n$. The following procedure is proposed by Powell et al. (to appear) to adjust the slopes $\{\hat{v}_{it}^n(0), \ldots, \hat{v}_{it}^n(b_{it})\}$ in order to obtain the slopes $\{\hat{v}_{it}^{n+1}(0), \ldots, \hat{v}_{it}^{n+1}(b_{it})\}$, which characterize the value function approximation $\hat{V}_{it}^{n+1}(\cdot)$ in the next iteration:

1. Set the vector $\{q_{it}^n(s) : s = 0, \ldots, b_{it}\}$ to

$$q_{it}^n(s) = \begin{cases} (1-\alpha^n)\hat{v}_{it}^n(s) + \alpha^n \phi_t^n(e_i) & \text{for } s = z_{i,t-1}^n \\ \hat{v}_{it}^n(s) & \text{otherwise,} \end{cases} \tag{13}$$

   where α^n is the step size parameter at iteration n with $0 \leq \alpha^n \leq 1$.

2. Set the vector $\hat{v}_{it}^{n+1} = \{\hat{v}_{it}^{n+1}(s) : s = 0, \ldots, b_{it}\}$ to

$$\hat{v}_{it}^{n+1} = \arg\min \sum_{s=0}^{b_{it}} (r(s) - q_{it}^n(s))^2 \tag{14}$$

$$\text{s.t.} \quad r(s) \geq r(s+1) \quad \text{for all } s = 0, \ldots, b_{it} - 1. \tag{15}$$

Step 1 above updates the slope of $\hat{V}_{it}^n(\cdot)$ around $z_{i,t-1}^n$ by $\phi_t^n(e_i)$. However, after this updating procedure, the piecewise-linear function $Q_{it}^n(\cdot)$ characterized by the slopes $\{q_{it}^n(0), \ldots, q_{it}^n(b_{it})\}$ is not necessarily concave. In Step 2, we find the concave function that is "closest" to $Q_{it}^n(\cdot)$ in the sense of the objective function (14). (Powell et al. (to appear) show that there is a closed form solution to problem (14)-(15).) We note that computing $\phi_t^n(e_i)$ requires knowing the exact value function $V_t(\cdot)$. In the next section, we derive a procedure to get a good approximation to $\phi_t^n(e_i)$.

4 Obtaining the policy gradients

We begin by defining the decision and state transfer function at time period t for iteration n:

$$X_t^n(z_{t-1}, P_t) = \underset{(x_t,y_t,z_t)\in\mathcal{X}(z_{t-1},P_t)}{\arg\max} p_t(x_t, y_t, z_t) + \hat{V}_{t+1}^n(z_t), \tag{16}$$

$$Z_t^n(z_{t-1}, P_t) = z_t \text{ if and only if } X_t^n(z_{t-1}, P_t) = (\cdot, \cdot, z_t). \tag{17}$$

Therefore, the decision function $X_t^n(z_{t-1}, P_t)$ takes the inventory held at each production plant in time period $t-1$ and the production quantities in time period t, and returns the decisions under the policy characterized by the value function approximations $\{\hat{V}_1^n(\cdot), \ldots, \hat{V}_T^n(\cdot)\}$. $Z_t^n(z_{t-1}, P_t)$ is a linear transformation that returns the last $|\mathcal{P}|$ elements (i.e. the inventory holding decisions) of $X_t^n(z_{t-1}, P_t)$. (To properly define the decision function in (16), we assume an arbitrary ordering among the elements of $\mathbb{Z}_+^{|\mathcal{P}||\mathcal{C}|+|\mathcal{C}|+|\mathcal{P}|}$, and when multiple optimal solutions exist, the arg max operator returns the optimal solution with the lowest order.)

We recursively define the cumulative profit function at time period t for iteration n as

$$\Pi_t^n(z_{t-1}, P_t, \ldots, P_T) = p_t(X_t^n(z_{t-1}, P_t)) + \Pi_{t+1}^n(Z_t^n(z_{t-1}, P_t), P_{t+1}, \ldots, P_T), \tag{18}$$

with $\Pi_{T+1}^n(\cdot) = 0$. As before, at iteration n, we denote the value function approximations by $\{\hat{V}_1^n(\cdot), \ldots, \hat{V}_T^n(\cdot)\}$, the samples of the production quantities by $\{\hat{P}_1^n, \ldots, \hat{P}_T^n\}$ and the decisions by $\{(x_1^n, y_1^n, z_1^n), \ldots, (x_T^n, y_T^n, z_T^n)\}$. Then by the definition of the decision and state transfer function,

$$(x_t^n, y_t^n, z_t^n) = X_t^n(z_{t-1}^n, \hat{P}_t^n), \tag{19}$$

$$z_t^n = Z_t^n(z_{t-1}^n, \hat{P}_t^n), \tag{20}$$

$$\Pi_t^n(z_{t-1}^n, \hat{P}_t^n, \ldots, \hat{P}_T^n) = p_t(x_t^n, y_t^n, z_t^n) + \ldots + p_T(x_T^n, y_T^n, z_T^n). \tag{21}$$

(21) can be verified by carrying out a backward induction on (18), and using (19) and (20). Naturally, $\Pi_1^n(\mathbf{0}, \hat{P}_1^n, \ldots, \hat{P}_T^n)$ is the value of the objective function at iteration n.

In order to update the value function approximation $\hat{V}_{it}^n(\cdot)$, we propose using

$$\hat{\phi}_t^n(e_i) = \Pi_t^n(z_{t-1}^n + e_i, \hat{P}_t^n, \ldots, \hat{P}_T^n) - \Pi_t^n(z_{t-1}^n, \hat{P}_t^n, \ldots, \hat{P}_T^n),$$

as opposed to $\phi_t^n(e_i) = V_t(z_{t-1}^n + e_i, \hat{P}_t^n) - V_t(z_{t-1}^n, \hat{P}_t^n)$. Note that $\hat{\phi}_t^n(e_i)$ gives the incremental worth of a product at production plant i at the beginning of time period t under the policy characterized by the value function approximations $\{\hat{V}_1^n(\cdot), \ldots, \hat{V}_T^n(\cdot)\}$. An important point is that $\hat{\phi}_t^n(e_i)$ assesses the impact of an incremental inventory not only at time period t, but also at time periods $t+1, \ldots, T$.

Clearly, one can compute $\hat{\phi}_t^n(e_i)$ by physically incrementing the inventory at production plant i and rerunning the current policy starting from time period t. However, doing this for all $i \in \mathcal{P}$ and $t \in \mathcal{T}$ would be very time consuming. Here, our objective is to develop a procedure that computes $\hat{\phi}_t^n(e_i)$ for all $i \in \mathcal{P}$ and $t \in \mathcal{T}$ from a single run. Using (18), we have

$$\begin{aligned}\hat{\phi}_t^n(e_i) = &\left\{p_t(X_t^n(z_{t-1}^n + e_i, \hat{P}_t^n)) - p_t(X_t^n(z_{t-1}^n, \hat{P}_t^n))\right\} \\ &+ \left\{\Pi_{t+1}^n(Z_t^n(z_{t-1}^n + e_i, \hat{P}_t^n), \hat{P}_{t+1}^n, \ldots, \hat{P}_T^n) - \Pi_{t+1}^n(Z_t^n(z_{t-1}^n, \hat{P}_t^n), \hat{P}_{t+1}^n, \ldots, \hat{P}_T^n)\right\}.\end{aligned} \tag{22}$$

We show separately how to compute the two terms in the curly brackets in (22).

1. Note that problem (16) is the min-cost network flow problem shown in Figure 2. Then, the change in the optimal solution resulting from incrementing the supply of a node is given by a min-cost flow-augmenting path into the sink node on the right side of this figure (see, for example, Powell (1989)). Hence, in Figure 4, the cost of the partial flow-augmenting path from node i into the sink node (excluding the costs on the value function approximation arcs) gives $p_t(X_t^n(z_{t-1} + e_i, P_t)) - p_t(X_t^n(z_{t-1}, P_t))$. We define the function $\Delta_t^n(\cdot)$ that computes this difference for $z_{t-1} = z_{t-1}^n$ and $P_t = \hat{P}_t^n$:

$$\Delta_t^n(e_i) = p_t(X_t^n(z_{t-1}^n + e_i, \hat{P}_t^n)) - p_t(X_t^n(z_{t-1}^n, \hat{P}_t^n)). \tag{23}$$

As a side remark, we emphasize that $\Delta_t^n(e_i)$ for all $i \in \mathcal{P}$ can efficiently be computed by a single flow-augmenting tree calculation that computes the flow-augmenting paths from every node into the sink node. (Powell (1989) gives an algorithm that computes the min-cost flow-augmenting paths from every node into the sink node.)

2. Furthermore, since the change in the optimal solution resulting from incrementing the supply

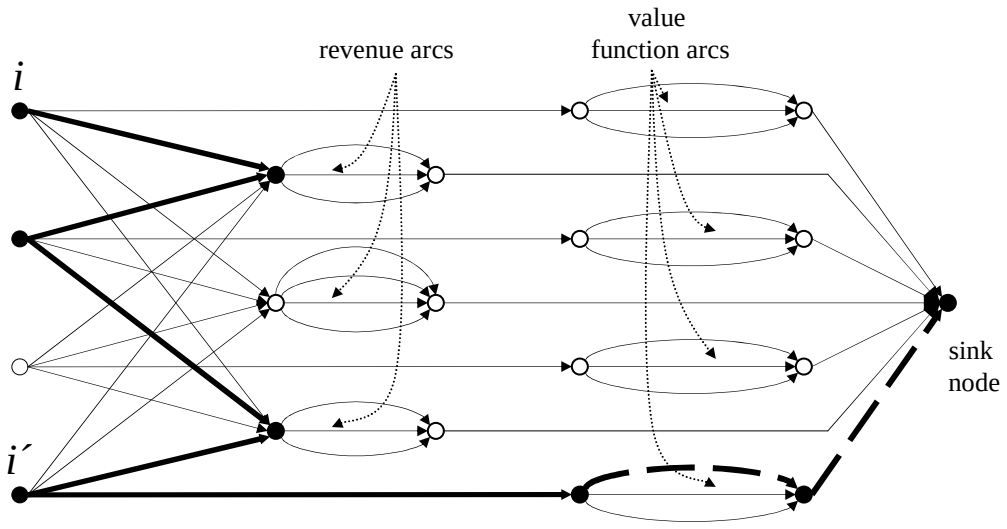

Figure 4: The change in the optimal solution resulting from incrementing the supply of node i is given by a min-cost flow-augmenting path into the sink node.

of a node is given by a min-cost flow-augmenting path into the sink node in Figure 4, the following property holds:

Property 2 *For all $z_{t-1} \in \mathbb{Z}_+^{|\mathcal{P}|}$ and all realizations of P_t, the vector $Z_t^n(z_{t-1}+e_i, P_t) - Z_t^n(z_{t-1}, P_t)$ is either zero or a positive, integer, unit vector.*

Proof Fix z_{t-1} and P_t. The decisions $X_t^n(z_{t-1} + e_i, P_t)$ differs from $X_t^n(z_{t-1}, P_t)$ by a flow-augmenting path from node i into the sink node in Figure 4. If the last arc in this flow-augmenting path is one of the value function approximation arcs, then $Z_t^n(z_{t-1} + e_i, P_t) - Z_t^n(z_{t-1}, P_t)$ is a unit vector. If the last arc in this flow augmenting path is one of the revenue arcs, then $Z_t^n(z_{t-1} + e_i, P_t) - Z_t^n(z_{t-1}, P_t)$ is zero. □

We define the function $\delta_t^n(\cdot)$ that computes the difference in Property 2 for $z_{t-1} = z_{t-1}^n$ and $P_t = \hat{P}_t^n$:

$$\delta_t^n(e_i) = Z_t^n(z_{t-1}^n + e_i, \hat{P}_t^n) - Z_t^n(z_{t-1}^n, \hat{P}_t^n).$$

$\delta_t^n(e_i)$ for all $i \in \mathcal{P}$ can also be computed by a single flow-augmenting tree calculation. Then, $Z_t^n(z_{t-1}^n + e_i, \hat{P}_t^n) = Z_t^n(z_{t-1}^n, \hat{P}_t^n) + \delta_t^n(e_i) = z_t^n + \delta_t^n(e_i)$ and

$$\Pi_{t+1}^n(Z_t^n(z_{t-1}^n + e_i, \hat{P}_t^n), \hat{P}_{t+1}^n, \ldots, \hat{P}_T^n) - \Pi_{t+1}^n(Z_t^n(z_{t-1}^n, \hat{P}_t^n), \hat{P}_{t+1}^n, \ldots, \hat{P}_T^n) = \tag{24}$$
$$\Pi_{t+1}^n(z_t^n + \delta_t^n(e_i), \hat{P}_{t+1}^n, \ldots, \hat{P}_T^n) - \Pi_{t+1}^n(z_t^n, \hat{P}_{t+1}^n, \ldots, \hat{P}_T^n) = \hat{\phi}_{t+1}^n(\delta_t^n(e_i)).$$

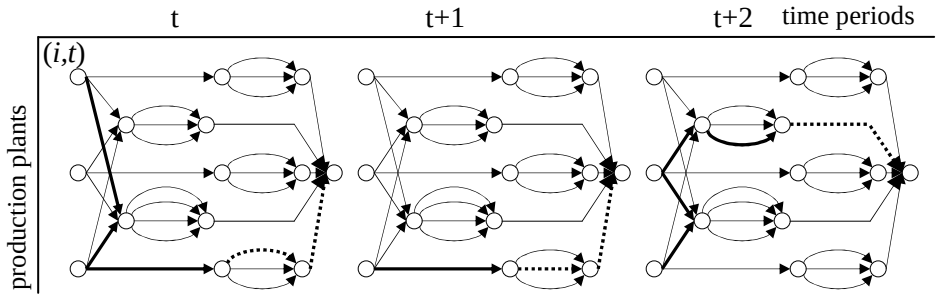

Figure 5: In order to compute $\hat{\phi}_t^n(e_i)$, we essentially connect the partial flow-augmenting paths starting from node (i,t).

Note that when $\delta_t^n(e_i) = \mathbf{0}$, $Z_t^n(z_{t-1}^n + e_i, \hat{P}_t^n) = Z_t^n(z_{t-1}^n, \hat{P}_t^n)$ and the difference in (24) is zero. Therefore, we define $\hat{\phi}_t^n(\mathbf{0}) = 0$ for all $t \in \mathcal{T}$ to conveniently cover this case.

Bringing these two steps together using (22), (23) and (24), we get

$$\hat{\phi}_t^n(e_i) = \Delta_t^n(e_i) + \hat{\phi}_{t+1}^n(\delta_t^n(e_i)).$$

Thus, the idea is to start with the last time period T and compute $\hat{\phi}_T^n(e_i) = \Delta_T^n(e_i)$ for all $i \in \mathcal{P}$. We then move back to time period $T-1$ and compute $\Delta_{T-1}^n(e_i)$ and $\delta_{T-1}^n(e_i)$ for all $i \in \mathcal{P}$. $\hat{\phi}_{T-1}^n(e_i)$ can now be computed as $\hat{\phi}_{T-1}^n(e_i) = \Delta_{T-1}^n(e_i) + \hat{\phi}_T^n(\delta_{T-1}^n(e_i))$ for all $i \in \mathcal{P}$. A visual representation of the method is presented in Figure 5. In order to compute $\hat{\phi}_t^n(e_i)$, we essentially add the costs of the partial flow-augmenting paths starting from node (i,t).

Finally, we note that the computational requirement of this procedure is one flow-augmenting tree calculation for each time period and the storage requirement is storing $|\mathcal{P}||\mathcal{T}|$ of the $\Delta_t^n(e_i)$ and $\delta_t^n(e_i)$ values for each iteration.

Figure 6 presents the complete solution algorithm. As a stopping criterion, we use a bound on the total number of iterations. Letting $v^n = \{v_{it}^n(s) : i \in \mathcal{P}, t \in \mathcal{T}, s = 0, \ldots, b_{it}\}$, an alternative stopping criterion could be to stop when $||v^{n+1} - v^n||$ becomes small.

5 Computational results

In this section, our primary objective is to show that the proposed solution method yields high quality solutions reasonably fast. As a benchmark strategy, we use a deterministic model that uses point forecasts of the future. In this way, we also quantify what can be gained from a

Step 1 Initialize iteration counter $n = 1$. Initialize $\hat{V}_t^n(\cdot)$ to 0 for all $t \in \mathcal{T}$.

Step 2 Simulate the system for all $t \in \mathcal{T}$: Initialize time counter $t = 1$ and $z_{t-1}^n = \mathbf{0}$.

Step 2.1 Sample a realization of P_t, say $\hat{P}_t^n$.

Step 2.2 Solve problem (4) corresponding to z_{t-1}^n and sampled $\hat{P}_t^n$ to set

$$(x_t^n, y_t^n, z_t^n) = \underset{(x_t,y_t,z_t)\in\mathcal{X}(z_{t-1}^n,\hat{P}_t^n)}{\arg\max} p_t(x_t, y_t, z_t) + \hat{V}_{t+1}^n(z_t).$$

Step 2.3 For each $i \in \mathcal{P}$ compute and store $\Delta_t^n(e_i)$ and $\delta_t^n(e_i)$.

Step 2.4 Set $t = t + 1$. If $t \leq T$, go to Step 2.1.

Step 3 Compute the policy gradients for all $t \in \mathcal{T}$: Initialize time counter $t = T$ and $\hat{\phi}_{T+1}^n(\cdot) = 0$.

Step 3.1 Compute $\hat{\phi}_t^n(e_i) = \Delta_t^n(e_i) + \hat{\phi}_{t+1}^n(\delta_t^n(e_i))$ for all $i \in \mathcal{P}$.

Step 3.2 Set $t = t - 1$. If $t \geq 1$, go to Step 3.1.

Step 4 Update the value function approximations for all $i \in \mathcal{P}$ and $t \in \mathcal{T}$: Set

$$\{q_{it}^n(s) : s = 0, \ldots, b_{it}\} = \begin{cases} (1-\alpha^n)\hat{v}_{it}^n(s) + \alpha^n \hat{\phi}_t^n(e_i) & \text{for } s = z_{i,t-1}^n \\ \hat{v}_{it}^n(s) & \text{otherwise,} \end{cases}$$

$$\{\hat{v}_{it}^{n+1}(s) : s = 0, \ldots, b_{it}\} = \arg\min \sum_{s=0}^{b_{it}} \left(r(s) - q_{it}^n(s)\right)^2$$
$$\text{s.t.} \quad r(s) \geq r(s+1) \quad \text{for all } s = 0, \ldots, b_{it} - 1.$$

Step 5 Set $n = n + 1$. If one more iteration is needed, go to Step 2.

Figure 6: The solution algorithm.

stochastic model instead of a deterministic one, and characterize the important problem parameters that can potentially affect the choice between a stochastic and a deterministic model. Finally, under special cases, the problem we are interested in can be solved optimally. In these cases, we compare the performance of the proposed solution method with the optimal solution.

Benchmark strategy – The deterministic model we use is the so-called "rolling horizon strategy" and is parameterized by the rolling horizon length R. Under this strategy, in order to make the decisions in time period t for any realization of the production quantities P_t and the held

inventory z_{t-1}, we solve an R-time period problem:

$$\max \sum_{s=t}^{(t+R-1)\wedge T} p_s(x_s, y_s, z_s) \tag{25}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{C}} x_{ijt} + z_{it} = z_{i,t-1} + P_{it} \quad \text{for all } i \in \mathcal{P} \tag{26}$$

$$\sum_{j\in\mathcal{C}} x_{ijs} + z_{is} - z_{i,s-1} = \mathbb{E}\{P_{is}\} \quad \text{for all } i \in \mathcal{P}, s = t+1, \ldots, (t+R-1)\wedge T \tag{27}$$

$$\sum_{i\in\mathcal{P}} x_{ijs} - y_{js} = 0 \quad \text{for all } j \in \mathcal{C}, s = t, \ldots, (t+R-1)\wedge T \tag{28}$$

$$x_{ijs}, y_{js}, z_{is} \in \mathbb{Z}_+ \quad \text{for all } i \in \mathcal{P}, j \in \mathcal{C}, s = t, \ldots, (t+R-1)\wedge T, \tag{29}$$

where $a \wedge b = \min\{a, b\}$. (Note that this model is not fully deterministic, in the sense that we still use the distributions of demand random variables in the objective function.) After solving this problem, we implement the decisions for only time period t. As R increases, the rolling horizon strategy is expected to yield better solutions, but in general this is not guaranteed. We also note that even if the production quantities take integer values, their expectations may not and it may be hard to obtain integer solutions to problem (25)-(29). In our numerical work we round $\mathbb{E}\{P_{is}\}$, so that problem (25)-(29) becomes a min-cost network flow problem with integer data. A number of setup runs showed that this does not deteriorate the solution quality. However, this becomes a concern when dealing with problems where the production variables can only take small values such as 0, 1 or 2, and the validity of rounding is dubious.

Testing the performance of our solution method is composed of two sets of iterations: training and testing. In the training iterations, we apply the complete algorithm in Figure 6, and the aim of the training stage is to construct a good approximation to the value function. After the training stage, in the testing iterations, we apply the algorithm in Figure 6 without Steps 3 and 4. Therefore, the testing iterations do not attempt to improve the value function approximations and their purpose is to evaluate the quality of the approximations obtained in the training stage. We test the approximations after 50, 250 and 500 training iterations using 200 testing iterations.

Data sets and experimental setup – Our data sets involve 9 production plants and 41 customer locations spread over a 1000×1000 region. The planning horizon is 28 time periods.

We assume that each customer location can be served by the closest N production plants. We set $c_{ijt} = \bar{c}\ \varepsilon_{ij}$, where ε_{ij} is the Euclidean distance between i and j and $\bar{c} = 1.6$. The expected

Problem no.	N	VM	$\bar{\sigma}$	$\bar{P}$
Base	4	8	100	4000
N 1	1	8	100	4000
N 2	2	8	100	4000
N 8	8	8	100	4000
VM 4	4	4	100	4000
VM 1	4	1	100	4000
VM 0	4	0	100	4000
$\bar{\sigma}$ 800	4	8	800	4000
$\bar{\sigma}$ 400	4	8	400	4000
$\bar{\sigma}$ -100	4	8	-100	4000
$\bar{\sigma}$ -400	4	8	-400	4000
$\bar{\sigma}$ -800	4	8	-800	4000
$\bar{P}$ 1000	4	8	100	1000
$\bar{P}$ 2000	4	8	100	2000
$\bar{P}$ 6000	4	8	100	6000
$\bar{P}$ 8000	4	8	100	8000
$\bar{P}$ 16000	4	8	100	16000

Table 1: Characteristics of the test problems.

profit function for customer location j in time period t depends on $\rho_{jt} + \pi_{jt}$ and σ_{jt} (see the expected profit function in (1)). Therefore, without loss of generality, we set $\pi_{jt} = 0$. We set $\rho_{jt} = \rho_j$ and $\sigma_{jt} = \sigma_j$, where ρ_j and σ_j are drawn from the uniform distribution with mean $\bar{\rho}$ and $\bar{\sigma}$ respectively. Similarly, the holding cost at production plant i in time period t is $h_{it} = h_i$, where h_i is drawn from the uniform distribution with mean $\bar{h}$. We set $\bar{\rho} = 1000$ and $\bar{h} = 20$.

The production and demand random variables are taken to be Gamma-distributed with variance to mean ratio VM, and after sampling, we round the realizations of these random variables. In order to generate the distributions for the production random variables, we use three functions: $f_x, f_y : 1000 \times 1000 \rightarrow [0, 1]$, $f_{\mathcal{T}} : \mathcal{T} \rightarrow [0, 1]$. For production plant i with geographical coordinates (a, b), the mean of the production random variable in time period t is proportional to $f_x(a)f_y(b)f_{\mathcal{T}}(t)$. By changing these three functions we can generate different spatial and temporal fluctuations in production. The same methodology is used to generate the distributions for the demand random variables. We fix the expected number of demands over the planning horizon at 4000.

In our experimental setup, we take a base problem and vary its parameters to obtain problems with different characteristics. Our test problems are listed in Table 1. All column headings of this table are described above except for $\bar{P}$, which is the expected production at all the production

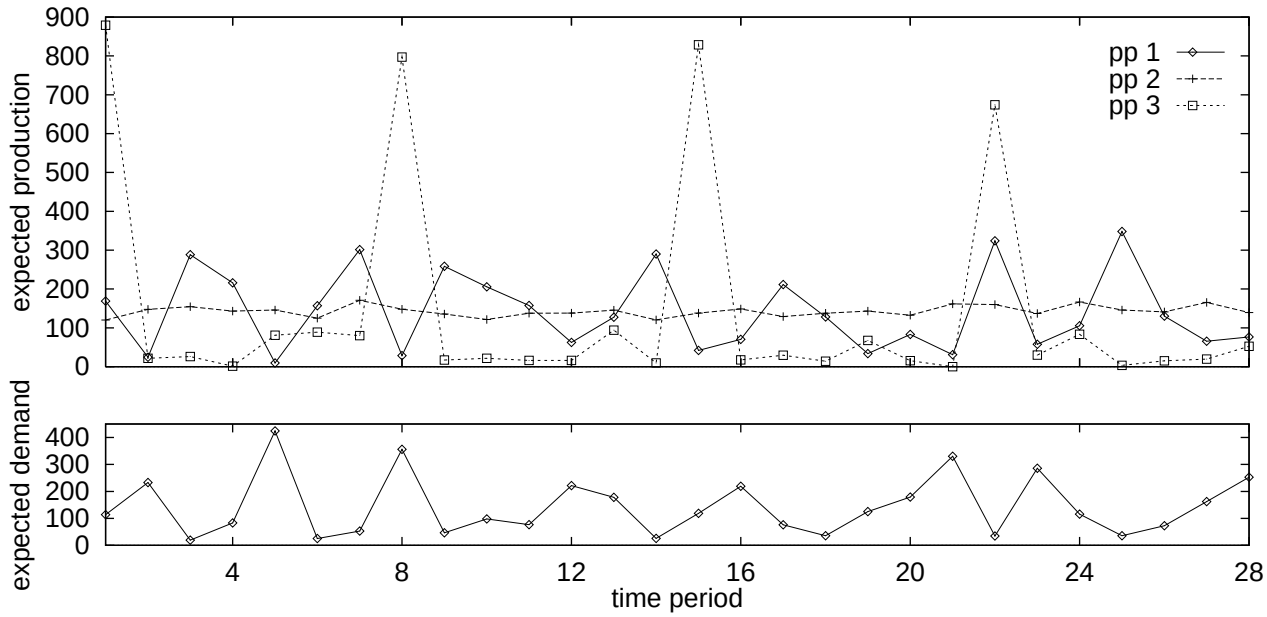

Figure 7: Production and demand patterns used in the computational experiments.

plants over the planning horizon (i.e. $\bar{P} = \mathbb{E}\left\{\sum_{t\in\mathcal{T}, i\in\mathcal{P}} P_{it}\right\}$).

We run each problem using three different production patterns characterizing the expected production quantities in different time periods. These patterns, along with the average demand at each time period, are shown in Figure 7. In all our experiments, we use the same demand pattern. (We note that these patterns are characterized by the specific function chosen for $f_{\mathcal{T}}(\cdot)$.) Using different production patterns is important in establishing the validity of our solution method. For example, inventory holding decisions are more crucial for a production pattern in which a large production quantity is followed by many small production quantities. In such a situation, a unit of product manufactured in a certain time period may have to be held for a long time in order to be used most profitably. Throughout, we use step size parameter $\alpha^n = 20/(40+n)$ in Step 4 of the algorithm in Figure 6.

In order to find the best rolling horizon length, we applied the rolling horizon strategy on problem "Base" with different values of R. Table 2 shows the average objective value over 200 samples, along with the CPU time per sample. Increasing the length of the rolling horizon beyond 8 time periods contributes to the objective value only marginally. Considering how sensitive the run times are to R, we fix the rolling horizon length at 8 in our subsequent experiments.

R	1	2	4	6	8	12	16
Avg. obj.	1,617,367	1,664,677	1,761,115	1,803,260	1,818,436	1,822,860	1,823,018
CPU (sec.)	0.11	0.30	1.15	2.61	5.03	14.47	25.87

Table 2: Performance of the rolling horizon strategy with changing R.

Computational results – We build the value function approximations using 50, 250 and 500 training iterations, and test the quality of these approximations using 200 samples. As a benchmark strategy, we use the 8-period rolling horizon method. To illustrate the importance of making decisions by considering the future random quantities, we also present the performance of the myopic strategy that completely ignores the future. (Using the dynamic programming approximation method by setting the value function approximations to zero achieves this).

We summarize our findings in Tables 3, 5, 7 and 8. In these tables, the first three sets of columns show the objective values obtained by the myopic, rolling horizon (RH) and dynamic programming approximation (DPA) strategies (after 50, 250 and 500 training iterations). The next column shows the percent difference between the performances of RH and DPA after 500 training iterations. The column labeled "DPA$\succ$RH" gives the percentage of the testing samples for which DPA yields better solution than RH. The last two columns give the CPU time per iteration for DPA and RH. CPU time per iteration includes solving 28 problems of the form (5)-(12) or (25)-(29) (respectively for DPA and RH), implementing the decisions and updating the value function approximations. (Using the reported CPU times, one can deduce the CPU time required to construct the value function approximations through a certain number of training iterations.) A common observation from Tables 3, 5, 7, 8 is that DPA yields better objective values than RH, and since problems (5)-(12) and (25)-(29) respectively "span" 1 and 8 time periods, the CPU times for DPA are much faster than those for RH. We now look at each table in detail.

Table 3 shows the results for problems with changing number of production plants that can serve a customer location. First, the difference between the performances of DPA and RH diminishes as N increases. This is due to the fact that as the number of plants serving a customer increases, it becomes easier to make up for an inventory shortage in one plant by using the inventory in another plant. Second, increasing N from 1 to 2 results in a large jump in the objective value, whereas increasing N further yields only marginal improvements (see Figure

Prod. pat.	Prob. no.	Myopic Objective	RH Objective	DPA Objective 50 itns.	250 itns.	500 itns.	Perc. imp.	DPA $\succ$ RH	CPU (sec.) DPA	RH
pp 1	N 1	1,585,162	1,716,800	1,714,453	1,755,949	1,762,748	2.61	100	0.32	2.90
	N 2	1,612,020	1,807,575	1,789,222	1,824,760	1,830,293	1.24	100	0.36	3.93
	Base	1,617,367	1,818,436	1,797,715	1,832,679	1,837,988	1.06	100	0.38	5.03
	N 8	1,614,699	1,818,316	1,795,887	1,832,516	1,837,911	1.07	100	0.43	6.89
pp 2	N 1	1,727,589	1,839,281	1,829,371	1,857,990	1,862,051	1.22	100	0.3	3.02
	N 2	1,727,589	1,903,384	1,884,445	1,909,594	1,912,712	0.49	96	0.33	4.08
	Base	1,721,163	1,909,916	1,889,120	1,913,786	1,917,089	0.37	93	0.36	5.21
	N 8	1,718,900	1,909,949	1,889,061	1,914,095	1,917,092	0.37	95	0.40	6.88
pp 3	N 1	1,195,717	1,429,955	1,377,960	1,475,694	1,497,096	4.48	100	0.33	2.79
	N 2	1,300,106	1,535,753	1,479,601	1,570,177	1,584,507	3.08	100	0.36	3.81
	Base	1,316,885	1,550,000	1,503,892	1,586,714	1,599,036	3.07	100	0.39	4.86
	N 8	1,317,749	1,555,709	1,504,648	1,587,660	1,599,771	2.75	100	0.43	6.34

Table 3: Results for problems with changing N.

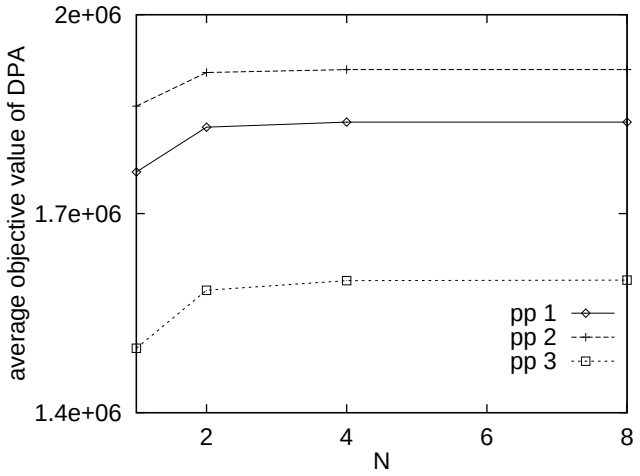

Figure 8: Increasing N beyond 3-4 yields only marginal improvements in performance.

8). This shows that introducing "redundancy" into the supply chain by connecting a customer location to more than one production plant improves the performance, but the improvement quickly diminishes. (See Jordan & Graves (1995) for similar supply chain configuration issues.) Third, when $N = 1$, each plant serves disjoint sets of customers and the problem decomposes into $|\mathcal{P}|$ problems, each having a one-dimensional state variable. In this case, the problem can be solved to optimality by using classical backward dynamic programming. For problems with $N = 1$, Table 4 shows that our approach yields objective values that are very close to the ones obtained by the optimal policy. Fourth, the number of variables in problems (5)-(12) and (25)-(29) increase with N. However, the CPU time for DPA is affected less by this increase.

Prod. pat.	Optimal	DPA
pp 1	1,763,091	1,762,748
pp 2	1,862,724	1,862,051
pp 3	1,499,143	1,497,096

Table 4: Comparison of DPA with the optimal policy when $N = 1$.

Prod. pat.	Prob. no.	Myopic Objective	RH Objective	DPA Objective 50 itns.	DPA Objective 250 itns.	DPA Objective 500 itns.	Perc. imp.	DPA $\succ$ RH	CPU (sec.) DPA	CPU (sec.) RH
pp 1	Base	1,617,367	1,818,436	1,797,715	1,832,679	1,837,988	1.06	100	0.38	5.03
	VM 4	1,678,393	1,892,370	1,883,323	1,907,399	1,909,269	0.89	100	0.35	5.05
	VM 1	1,687,486	1,907,269	1,904,680	1,923,061	1,923,699	0.85	100	0.34	4.99
	VM 0	1,694,528	1,922,889	1,923,224	1,932,682	1,932,688	0.51	100	0.29	7.03
pp 2	Base	1,721,163	1,909,916	1,889,120	1,913,786	1,917,089	0.37	93	0.36	5.21
	VM 4	1,771,752	1,970,944	1,958,116	1,974,820	1,975,664	0.24	100	0.33	6.18
	VM 1	1,786,735	1,987,089	1,979,759	1,990,412	1,990,675	0.18	100	0.32	5.22
	VM 0	1,802,422	2,000,472	1,998,811	2,002,459	2,002,461	0.10	100	0.29	5.19
pp 3	Base	1,316,885	1,550,000	1,503,892	1,586,714	1,599,036	3.07	100	0.39	4.86
	VM 4	1,379,169	1,639,521	1,625,832	1,678,560	1,684,041	2.64	100	0.39	4.96
	VM 1	1,392,954	1,658,276	1,662,473	1,699,830	1,701,885	2.56	100	0.37	4.99
	VM 0	1,400,954	1,672,104	1,688,030	1,712,249	1,712,297	2.35	100	0.32	4.84

Table 5: Results for problems with changing VM for production random variables.

Table 5 presents the results for problems with changing variance-to-mean-ratio for the production random variables. As the variance of the random quantities increases, using a stochastic model pays off, and the gap between RH and DPA becomes more noticeable. When $VM = 0$, the problem is deterministic, and 28-period RH yields the optimal solution. Table 6 shows that DPA gives results that are very close to the optimal objective value for these deterministic problems.

Table 7, shows the results for problems with changing average salvage value. As the average salvage value approaches to the average revenue (which is fixed at 1000), all the available inventory in a time period can be pushed to the customer locations to exploit the high salvage value, and the incentive to store inventory decreases. This diminishes the value of a stochastic model. Conversely, when the salvage value is very low, a unit of product shipped to a "wrong" customer location is penalized heavily, and this increases the value of a stochastic model.

Prod. pat.	Optimal	DPA
pp 1	1,932,691	1,932,688
pp 2	2,002,463	2,002,461
pp 3	1,712,300	1,712,297

Table 6: Comparison of DPA with the optimal objective value when $VM = 0$.

Prod. pat.	Prob. no.	Myopic Objective	RH Objective	DPA Objective 50 itns.	DPA Objective 250 itns.	DPA Objective 500 itns.	Perc. imp.	DPA $\succ$ RH	CPU (sec.) DPA	CPU (sec.) RH
pp 1	$\bar{\sigma}$ 800	3,516,386	3,578,538	3,574,803	3,578,896	3,578,858	0.01	55	0.19	2.88
	$\bar{\sigma}$ 400	1,791,559	2,322,222	2,249,862	2,329,010	2,334,063	0.51	99	0.28	3.93
	Base	1,617,367	1,818,436	1,797,715	1,832,679	1,837,988	1.06	100	0.38	5.03
	$\bar{\sigma}$ -100	1,454,244	1,566,843	1,560,837	1,578,897	1,580,496	0.86	99	0.41	4.41
	$\bar{\sigma}$ -400	1,183,002	1,230,159	1,230,338	1,235,350	1,234,903	0.38	81	0.49	3.44
	$\bar{\sigma}$ -800	719,738	799,816	800,602	831,386	831,399	3.8	96	0.67	2.61
pp 2	$\bar{\sigma}$ 800	3,524,981	3,557,892	3,557,043	3,558,738	3,559,044	0.03	78	0.19	3.13
	$\bar{\sigma}$ 400	2,022,110	2,377,977	2,326,191	2,379,896	2,383,362	0.23	93	0.27	4.21
	Base	1,721,163	1,909,916	1,889,120	1,913,786	1,917,089	0.37	93	0.36	5.21
	$\bar{\sigma}$ -100	1,559,156	1,659,685	1,654,353	1,665,200	1,666,044	0.38	92	0.36	4.75
	$\bar{\sigma}$ -400	1,286,708	1,322,181	1,324,656	1,326,043	1,326,035	0.29	72	0.41	3.57
	$\bar{\sigma}$ -800	846,507	909,131	910,665	929,332	931,317	2.38	98	0.57	2.61
pp 3	$\bar{\sigma}$ 800	3,470,782	3,522,768	3,521,413	3,524,811	3,524,931	0.06	94	0.18	2.82
	$\bar{\sigma}$ 400	1,740,405	2,173,023	2,080,946	2,184,274	2,193,274	0.92	100	0.26	3.95
	Base	1,316,885	1,550,000	1,503,892	1,586,714	1,599,036	3.07	100	0.39	4.86
	$\bar{\sigma}$ -100	1,118,719	1,294,213	1,259,982	1,313,700	1,322,671	2.15	98	0.53	4.10
	$\bar{\sigma}$ -400	796,517	922,331	908,495	962,370	969,419	4.86	97	0.66	3.32
	$\bar{\sigma}$ -800	419,194	511,709	504,820	557,585	568,468	9.98	98	0.77	2.75

Table 7: Results for problems with changing $\bar{\sigma}$.

Table 8 presents the results with changing average total production quantity. This set of experiments confirm the intuition that when there is simply not enough product in the system, all the available inventory in a certain time period has to be shipped to the customer locations in the optimal solution and the inventory holding decisions are of second nature. Therefore, for tightly constrained systems, RH performs closely to DPA.

6 Concluding remarks

In this paper, we presented a distribution model under stochastic production quantities. The challenge was to setup a "look ahead" mechanism in order to predict possible shortages in a plant and to store inventory accordingly. Our approach formulated the problem as a dynamic program, replaced the value functions with tractable approximations and improved these approximations using samples of the random quantities. We empirically showed that our model performed better than the rolling horizon strategy and myopic solutions can be arbitrarily bad. The results in Tables 3, 5, 7 and 8 indicated that using a stochastic model becomes especially important when a certain customer location can be served by relatively few production plants, or the variability of the production random variables is high, or the salvage value of the product is low, or the production quantities exceed the demand by large margins.

Prod. pat.	Prob. no.	Myopic Objective	RH Objective	DPA Objective 50 itns.	250 itns.	500 itns.	Perc. imp.	DPA $\succ$ RH	CPU (sec.) DPA	RH
pp 1	$\bar{P}$ 1000	525,351	756,124	751,791	758,427	758,570	0.32	99	0.21	3.22
	$\bar{P}$ 2000	962,059	1,252,186	1,233,735	1,256,064	1,258,064	0.47	99	0.29	4.39
	Base	1,617,367	1,818,436	1,797,715	1,832,679	1,837,988	1.06	100	0.38	5.03
	$\bar{P}$ 6000	1,934,931	1,995,449	2,003,612	2,024,379	2,026,743	1.54	100	0.47	4.59
	$\bar{P}$ 8000	1,667,113	2,006,927	2,016,057	2,034,482	2,034,145	1.34	99	0.50	4.26
	$\bar{P}$ 12000	599,131	1,860,707	1,816,824	1,909,531	1,908,773	2.52	100	0.68	3.98
pp 2	$\bar{P}$ 1000	586,554	764,014	761,782	765,470	765,587	0.21	89	0.19	3.07
	$\bar{P}$ 2000	1,038,862	1,288,068	1,274,571	1,289,991	1,291,279	0.25	95	0.26	4.09
	Base	1,721,163	1,909,916	1,889,120	1,913,786	1,917,089	0.37	93	0.36	5.21
	$\bar{P}$ 6000	2,096,060	2,133,378	2,137,393	2,147,796	2,148,943	0.72	93	0.43	4.69
	$\bar{P}$ 8000	1,932,892	2,143,980	2,154,547	2,160,581	2,162,860	0.87	97	0.49	4.25
	$\bar{P}$ 12000	950,613	1,975,492	1,955,358	2,023,046	2,024,805	2.44	100	0.65	3.95
pp 3	$\bar{P}$ 1000	525,259	690,785	671,016	695,004	696,911	0.88	100	0.21	3.53
	$\bar{P}$ 2000	850,651	1,117,300	1,073,812	1,131,540	1,138,654	1.88	100	0.30	4.42
	Base	1,316,885	1,550,000	1,503,892	1,586,714	1,599,036	3.07	100	0.39	4.86
	$\bar{P}$ 6000	1,379,924	1,717,417	1,654,506	1,753,936	1,770,177	2.98	100	0.46	4.97
	$\bar{P}$ 8000	1,063,598	1,750,704	1,680,496	1,792,603	1,806,673	3.10	100	0.53	5.00
	$\bar{P}$ 12000	-2,372	1,647,813	1,490,896	1,719,360	1,726,535	4.56	100	0.61	4.39

Table 8: Results for problems with changing $\bar{P}$.

References

Adelman, D. (2004), 'A price-directed approach to stochastic inventory routing', *Operations Research* **52**(4), 499–514.

Bertsekas, D. & Tsitsiklis, J. (1996), *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA.

Cheung, R. K.-M. & Powell, W. B. (1996), 'Models and algorithms for distribution problems with uncertain demands', *Transportation Science* **30**(1), 43–59.

de Farias, D. P. & Van Roy, B. (2003), 'The linear programming approach to approximate dynamic programming', *Operations Research* **51**(6), 850–865.

Federgruen, A. & Zipkin, P. (1984), 'Approximations of dynamic, multilocation production and inventory problems', *Management Science* **30**(1), 69–84.

Fumero, F. & Vercellis, C. (1994), 'Capacity analysis in repetitive assemble-to-order manufacturing systems', *European Journal of Operational Research* **78**(204–215).

Godfrey, G. A. & Powell, W. B. (2001), 'An adaptive, distribution-free approximation for the newsvendor problem with censored demands, with applications to inventory and distribution problems', *Management Science* **47**(8), 1101–1112.

Godfrey, G. A. & Powell, W. B. (2002), 'An adaptive, dynamic programming algorithm for stochastic resource allocation problems I: Single period travel times', *Transportation Science* **36**(1), 21–39.

Jackson, P. L. (1988), 'Stock allocation in a two-echelon distribution system or "what to do until your ship comes in"', *Management Science* **34**(7), 1988.

Jordan, W. C. & Graves, S. C. (1995), 'Principles on the benefits of manufacturing process flexibility', *Management Science* **41**(4), 577–594.

Karmarkar, U. S. (1981), 'The multiperiod multilocation inventory problems', *Operations Research* **29**, 215–228.

Karmarkar, U. S. (1987), 'The multilocation multiperiod inventory problem: Bounds and approximations', *Management Science* **33**(1), 86–94.

Kleywegt, A. J., Nori, V. S. & Savelsbergh, M. W. P. (2002), 'The stochastic inventory routing problem with direct deliveries', *Transportation Science* **36**(1), 94–118.

Nemhauser, G. & Wolsey, L. (1988), *Integer and Combinatorial Optimization*, John Wiley & Sons, Inc., Chichester.

Powell, W. B. (1989), 'A review of sensitivity results for linear networks and a new approximation to reduce the effects of degeneracy', *Transportation Science* **23**(4), 231–243.

Powell, W. B., Ruszczynski, A. & Topaloglu, H. (to appear), 'Learning algorithms for separable approximations of stochastic optimization problems', *Mathematics of Operations Research* .

Powell, W. B. & Topaloglu, H. (2003), Stochastic programming in transportation and logistics, *in* A. Ruszczynski & A. Shapiro, eds, '*Handbook in Operations Research and Management Science*, Volume on *Stochastic Programming*', North Holland, Amsterdam.

Puterman, M. L. (1994), *Markov Decision Processes*, John Wiley and Sons, Inc., New York.

Rappold, J. A. & Muckstadt, J. A. (2000), 'A computationally efficient approach for determining inventory levels in a capacitated multiechelon production-distribution system', *Naval Research Logistics* **47**, 377–398.

Tsitsiklis, J. & Van Roy, B. (1997), 'An analysis of temporal-difference learning with function approximation', *IEEE Transactions on Automatic Control* **42**, 674–690.

Van Roy, B., Bertsekas, D. P., Lee, Y. & Tsitsiklis, J. N. (1997), A neuro dynamic programming approach to retailer inventory management, *in* 'Proceedings of the IEEE Conference on Decision and Control'.

Watkins, C. J. C. H. & Dayan, P. (1992), 'Q-Learning', *Machine Learning* **8**, 279–292.