Approximate Dynamic Programming: Application to Process Supply Chain Management

Jaein Choi
Process Systems Team, Process Technology R&D, LG Chem. Ltd. Research Park, Daejon, 305-380, Korea

Matthew J. Realff and Jay H. Lee
School of Chemical and Biomolecular Engineering, Georgia Institute of Technology, Atlanta, GA 30332

DOI 10.1002/aic.10840
Published online April 10, 2006 in Wiley InterScience (www.interscience.wiley.com).

Many continuous process industries are required to operate multiproduct supply chains under significant uncertainty in market demands and prices, and mathematical programming formulations have been developed for the deterministic version of the problem. However, available extensions to the stochastic case cannot be readily used to obtain an optimal operating policy in practice because the uncertainty results in an explosion of the scenarios to be considered over an extended time horizon. In this article we represent the uncertainty through Markov chains and use an approach based on stochastic dynamic programming (DP), which can generate a dynamic operating policy that incorporates information about the uncertainty in the problem at each time step. DP is computationally infeasible because of the size of the state and action space for realistic problems. We propose a novel stochastic optimization algorithmic framework called "DP in a heuristically restricted state space." The confined state space is generated by simulating various potential scenarios under centralized dynamic inventory and production policy generated by combining static local inventory policy heuristics. The resulting DP policy responds to the time-varying demand for products by appropriately stitching together decisions made by the local static policies. This is a new formulation for effectively combining heuristic policies. © 2006 American Institute of Chemical Engineers *AIChE J,* 52: 2473–2485, 2006

Introduction

A significant problem for complex supply chain management (SCM) is the effective handling of uncertainty in the system. The resulting SC operating policy has to be flexible enough to deal effectively with uncertain parameter variations, such as the volume and timing of market demands. Failure to account for significant product demand fluctuations by deterministic planning models may lead to either excessively high production costs (translating to high inventory charges) or unsatisfied customer demand and loss of market share. Recognition of this fact has motivated recent work aimed at studying process planning and scheduling under demand uncertainty. Most of the research on this problem has largely focused on mathematical programming approach.[1-7] The stochastic attributes of the problem are translated into an equivalent deterministic form with certain types of uncertain parameters, often normally or exponentially distributed.

Mathematical programming approaches are not efficient methods for handling significant uncertainty in the system because of the limited capability of representing uncertainty in the equivalent deterministic form. The size of the equivalent deterministic form increases exponentially with the number of scenarios because it is a superstructure of a sufficient sample of the uncertain parameters to capture the problem variability. In a stochastic programming approach, the most practical way to

Correspondence concerning this article should be addressed to J. H. Lee at jay.lee@chbe.gatech.edu.

reduce the size of the superstructure is to reduce the number of stages as in two-stage stochastic programming. This requires shortening the prediction horizon of the uncertain parameter realizations, or making an unrealistic aggregation of future periods, leading to an unnatural, restrictive problem representation.

On the other hand, a general supply chain can be viewed as serial and distributed inventory systems, referred to as *multiechelon* inventory systems, if production facilities involved in the supply chain are simplified (that is, assumed to be without large lead time in production). Previous research[8-11] on "multiechelon" inventory systems has focused on finding analytical optimal "order-up-to-policies" for variants of the system under the "balanced assumption," where negative stock allocations to the retailers are possible. Although the analytical optimal policies for the multiechelon inventory systems are not directly applicable to the supply chain system addressed in this study, for which the balanced assumption is not valid, the "order-up-to-policies"—termed (s, S) policies—are adopted to generate heuristics for control (see the Heuristics: Combination of Static Inventory Control Policies section and *Simulation results for the heuristics* subsection).

This article develops a novel solution method that expands the representation of uncertainty to include a class of problems that is wider than that addressed in the literature to date. The solution of the SCM problem is a policy that can be interpreted as a series of decisions at each time unit. Thus, it is a *multistage* decision-making problem with a significant number of uncertain parameter realizations. Stochastic dynamic programming (DP)[12] can be used to solve this type of problem. However, stochastic DP is faced with the "curse of dimensionality"—an exponential increase in the state space as the problem size increases. Thus, most of the research on the DP approach for SCM is limited to small sizes of the problem.[13-15] The size of the state space is coupled not only to the state of the supply chain but also to factors in the "information states," which represent observed information regarding uncertain parameter variations. To overcome the computational intractability of the conventional DP approach, herein we adopt an algorithmic framework that uses information obtained from stochastic simulation of the heuristics, which we call *DP in a heuristically restricted state space*.[16-18]

This approach was applied to a stochastic resource-constrained project scheduling problem (RCPSP) in Choi et al.,[17] where the ability to address a large state space was verified by confining the original state space (with 230 billion states) to obtain a reasonably sized confined state space (with 371,168 states). The development required to apply this to the SCM problem is to tame the complexity of the action space. The decisions in the RCPSP and corresponding combination of the actions are much simpler than those of the SCM problem. In the SCM problem, the action space is continuous and, even though the actions can be aggregated and represented using a discrete action space, there are a large number of actions for a supply chain involving multiple material flows between manufacturing sites. For example, if 10 material flows are involved in a supply chain and each material flow is discretized to three discrete values, the total number of possible actions at each time point is 59,049 ($=3^{10}$). Thus, we cannot avoid large numbers of actions in the DP formulation of the problem. In conventional DP, a large number of actions makes the Bellman iteration and on-line decision-making procedures computationally intractable as a result of the increased search space for the optimal action for each decision stage. One contribution of the article is to develop a general method for combining heuristics for local supply chain management decision making to find systemwide operating policies. The combination occurs (1) because the decisions at any given time are regarded holistically by the dynamic program, and thus good sets of decisions will be captured, and (2) because the cost-to-go function of the dynamic program will enable correlations between decisions across time to be propagated back to earlier decision points. Finally, the dynamic program evaluates the use of different heuristics in the same state of the supply chain and can create dynamic inventory policies from static ones by switching between heuristics that are fundamentally different or differ only by the way they are parameterized.

In summary, the key contributions of this article are (1) developing an appropriate DP formulation for the SCM problem and (2) enhancing the methodology of "DP in a heuristically confined state space" to handle the inevitable action space complexity. The article is organized as follows. A representative SCM problem will be formulated with a Markov chain model to represent uncertain demand. An appropriate heuristic method for the given problem will be presented. Then, the conventional stochastic DP formulation will be developed as a basis for the "DP in a heuristically confined state space." A summary and further discussion will be given in the last part of the article.

Problem Description: SCM with Multiple Products under Uncertain Product Demands and Prices

The prototypical process industry SCM problem addressed herein has most of the essential components of a supply chain, including production and inventory control decisions and intermediate product lines. It suppresses the details on logistics such as various transportation options and multiple customer (market) locations because these are often not as significant for the business-to-business component of supply chains. We consider the SCM problem with M products. The products are manufactured at p_i plants and stored in q_i inventories, for $i = 1, \ldots, M$. The plants that produce the product i have their own raw material inventory linked to r_i suppliers, for $i = 1, \ldots, M$. Thus, the supply chain is a partially connected network involved with $\sum_i p_i$ plants, $\sum_i q_i$ inventories, and $\sum_i r_i$ raw material suppliers. Connections between supply chains involved with different production/distribution lines arise as a result of product recipes that use raw materials to produce intermediates for other products. An illustrative example of such a supply chain is depicted in Figure 1. The focus herein is on the uncertainty in demand for products and that in raw material prices. This makes an effective allocation of intermediates to final production steps very important. A detailed description of the uncertainty model is given in the next section. As stated earlier, logistic elements of the problem are kept simple by considering only one transportation option and unit transportation time for every material flow. Manufacturing times for the products are given as a multiple of time units and appear as manufacturing time delays in the plant. Other than the demands and prices of the products, all the problem parameters (inventory costs, manufacturing costs, manufacturing time, and de-

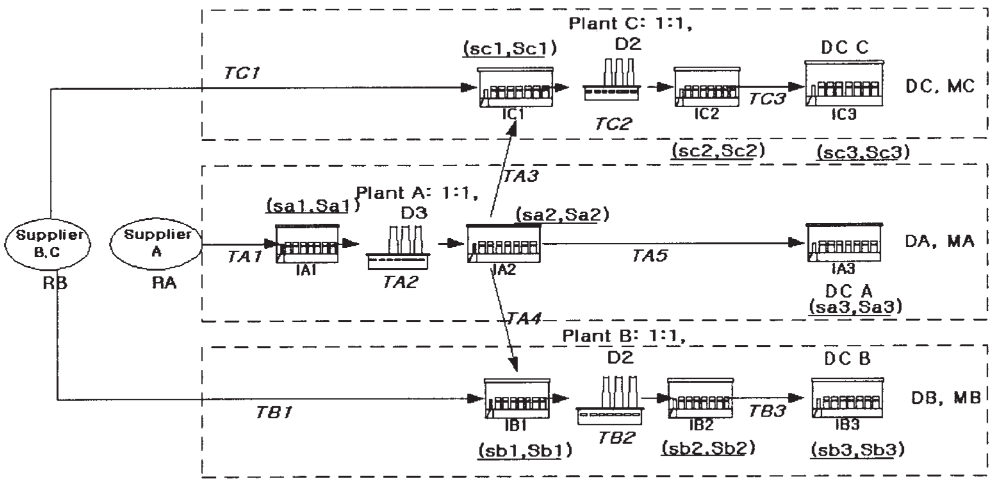

Figure 1. Illustrative example: SCM with three products under uncertainty.

fault setup cost for one batch of production) take known deterministic values. This is not an inherent limitation of the formulation but reflects our desire to keep the problem of manageable size for illustration of the key concepts of the article.

Markovian model of the uncertain parameters

Demand and price variations encountered by retailers or manufacturers have many different sources. They are often correlated, both among themselves (as a result of an underlying cause such as oil prices) and in time, such as in a seasonal variation. In previous supply chain literature, demand and price uncertainties in the SCM problem have been represented as Gaussian (normal) or exponential random variables[4,7,19,20] or uniformly distributed random variables.[21] However, it appears that auto- and cross-correlations among the uncertain parameters in this context have not previously been addressed in the context of process supply chains, with the exception of the contribution of the aforementioned parameters,[22] in which uncertain product demands are represented by a normal multivariate probability distribution. In our problem formulation, the demand and price of each product are modeled with a Markov chain. This mirrors our previous work using Markov chains for applications in stochastic resource-constrained project scheduling.[17] The use of Markov chain in modeling of market demand and price offers some advantages over previous approaches. For example, if the demand of a certain product is very high, the probability of a sharp demand decrease at the next time unit (a week or a month) would be very small. Thus, the current demand can be an important indication of the future demand realization. Furthermore, demand and price may not vary independently in the market.

In our SC model, we assume the uncertain demand and price of a product are realized as a pair and there are several different pairs of demand and price evolving according to a given Markov chain. The possible discrete values for the uncertain demand and price may represent the actual values or the mean values of the parameters to allow for more marginal variations to be captured. In summary, random (but correlated) uncertain demand and price for a product in the market are represented with n sets of demand and price and $n \times n$ state (in the Markov chain) transition probability matrix as depicted in Figure 2. In the illustrative example (Figure 1), five Markov chains are given to represent uncertain demands and prices of three products and uncertain price of raw materials from two external suppliers. This form of the Markov model does not capture any correlation across the products, but that could be incorporated by coupling together the states of the individual products, at the expense of a more complex transition matrix representation.

Heuristics: Combination of Static Inventory Control Policies

A crucial step in applying our framework is to develop heuristics, which give a reasonable solution (policy) for the given problem and can be simulated without a significant computational burden. In this section, we propose to develop such heuristic policies by combining available static inventory control policies. The given SCM problem has many material flows that have to be decided at each time unit. Each of those material flows is linked to two inventories in the supply chain. Thus, the decision of each material flow can be made by a single inventory control policy. One of the simplest inventory control policies is a static inventory control policy[23]—also known as an (s, S) policy—in which s is given as a reorder point to replenish the inventory level up to S. Even though the static inventory control policy is simple, with appropriate

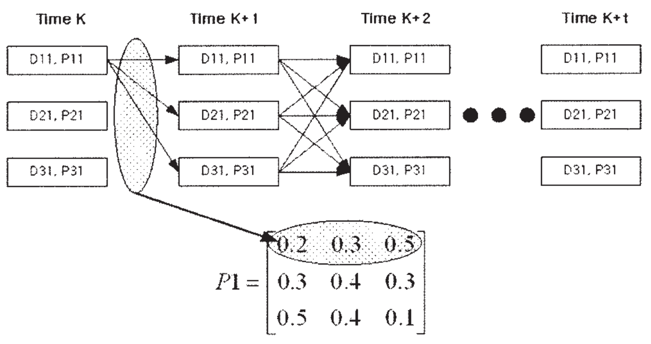

Figure 2. Representation of an uncertain demand and price with a Markov chain.

choice of s and S parameters, it is applicable to a wide range of operating conditions. Furthermore, under certain assumptions, the (s, S) policies can be shown to be optimal inventory control policies for classes of supply chain systems similar to those addressed in this study.[8-11] However, for the given SCM problem, the static inventory control policy is not sufficient to achieve good overall performance policy because of the need to vary the replenishment levels under the uncertain demand. It would be better to use a dynamic (s, S) policy that could respond to specific realizations of the uncertainty. As mentioned earlier, we adopt as starting policies a set of static inventory control policies that work reasonably and in a complementary manner. Each of the heuristics will be used for stochastic simulation in the next step of the solution framework as illustrated in the *Learning stage: simulation of the heuristic policies* subsection.

Conventional Stochastic DP Formulation

In this section, we develop a DP formulation, defining the state, action (decisions), the state transition rules, and the objective function (profit-to-go). Appropriate Bellman (value) iteration and on-line decision-making equations will be presented in the last part of this section.

State definition and aggregation

For the given SCM problem, three types of information are necessary to describe the status of the system.

(1) *Inventory Level*: All inventory levels in the system have to be included in the state. Thus, for the Q inventories in the system, the inventory level at time k, $I_{j1}(k)$, for $j1 = 1, \ldots, Q$, is defined as a state variable.

(2) *Ongoing Production*: For P plants in the system, the amount of ongoing production started at time $k - \ell$ at plant $j2$ is devoted by $O_{j2}(k - \ell)$, for $j2 = 1, \ldots, P$ and $\ell = 1, \ldots, \tau_{j2} - 1$, where τ_{j2} is the production time delay (time delay from the raw material inventory to the product inventory of the plant) for plant $j2$.

(3) *Information State*: In addition to the physical state variables defined above, the information state, which represents the current status of the uncertain parameters, should also be included in the state. The information state variable, which is represented as an integer, is a realized "state" of the Markov chain for the corresponding uncertain parameter or set of uncertain parameters. For L Markov chains in the system, the information state $z_{j3}(k)$, for $j3 = 1, \ldots, L$ is included in the state vector. $z_{j3}(k)$ can be one of the integers ranging from 1 to S_{j3} when the Markov chain $j3$ has S_{j3} possible states.

In summary, the state of the given problem is defined as follows:

$$X(k) = [I_1(k), \ldots, I_Q(k), O_1(k-1), \ldots, O_1(k-\tau_1 + 1), \ldots, O_P(k-1), \ldots, O_P(k-\tau_P + 1), z_1(k), \ldots, z_L(k)] \quad (1)$$

All the state variables in Eq. 1 are integers, given that the SCM problem is assumed to manipulate discrete lots of material. The explosion of the state space is governed by the number and range of the inventory levels. To avoid a state-space explosion, the state variables have to be aggregated and represented with discrete integer values. Besides the information state variables, z_{j3}, for $j3 = 1, \ldots, L$, which are naturally discrete, all other state variables (for inventory and ongoing production) are aggregated. The state aggregation is defined with three elements: (1) aggregation ranges, (2) representative index, and (3) representative value. The aggregated range defines the range of the state values to be treated as belonging to the same state. Once a state variable is aggregated, a representative index is assigned to the state variable and represented in the state space with the index. The representative value of an aggregated state is required to disaggregate the state to an actual value (such as an actual inventory level) and usually chosen as the median of the aggregation range.

Action definition

An action u of the problem is defined as a decision on all the material flows in the system:

$$u = [T_1, T_2, \ldots, T_R]^T \quad (2)$$

T_{j4} is an integer variable that represents a material flow $j4$, for $j4 = 1, 2, \ldots, R$. Some of the state variables, such as ongoing production, defined in Eq. 1, are included in previous actions as production decisions. According to the definition of actions in Eq. 2, the size of the action space, composed of all possible actions, is too large to be computed through the Bellman iteration, and in the real-time decision-making step of the DP. Thus, both the action space and the state space must be aggregated. The simple aggregation rules suggested for the state in the previous subsection are not appropriate for the action space. In general, a supply chain network includes many different material flows and the ranges of the material flows are diverse and large. Thus, the number of all possible actions will be astronomical even after aggregation. For example, if a supply chain network consists of 10 material flows to be decided at every unit time and each of the material flows is aggregated to just three representative values, the total number of actions in the action space will be $3^{10} = 59{,}049$. Therefore, the computational infeasibility of DP in a supply chain application is partly the result of the large number of possible actions and not just the state space explosion. Note that, in a discrete system, the computational load of the Bellman iteration in Eq. 6 is proportional to multiplication of the number of states and number of actions to be investigated. The computational problem created by large action (decision) spaces is circumvented by introducing an "implicit subaction space," which will be explained in detail in the subsection *Implicit subaction space for a state*.

State transition rules

In a discrete time system, the state at time $k + 1$ [that is, $X(k + 1)$] can be derived from the state at time k [that is, $X(k)$] and the control action (decision) [$u(k)$]. For the given problem, the state transition rules are linear material balance equations of all inventories as generalized in the following equation[†]:

[†] Detailed state transition rules are illustrated with an example from Eqs. 15–26 in the State Transition Rules subsection.

$$\text{Inventory Level at Time } k+1 = \text{Inventory Level at Time } k + \text{Input at Time } k - \text{Output at Time } k \quad (3)$$

The state variables can be classified into two types of variables in state transition.

(1) *Controllable state variables*: Those state variables that represent physical properties of the system such as the inventory levels and ongoing productions are evolved with the current and previous actions. The inventory levels are partially controllable because the uncertain demands also affect transitions of those state variables. Ongoing production results in fully controlled state variables because they are actual actions taken in the past. However, if random yields were included in the problem, then this assumption would not hold, which would make the state transitions more complex but not qualitatively change the problem structure.

(2) *Uncontrollable state variables*: Transitions of the information state variables are not dependent on the actions because the transitions are governed by the underlying Markov chains. The information state variables at time k depend only on the realized uncertain parameters at time k.

Because of the uncontrollable state variables, the state at time $k + 1$ is not unique, even though it evolves from the same state at time k and the same action by the state transition (Eq. 3). According to the underlying Markov chains, all possible next states $X(k + 1)$ are calculated with their realization probabilities for given $X(k)$ and $u(k)$. The realization probabilities of the possible next states are used in the calculation of the expected objective function value (profit-to-go) corresponding to the $X(k + 1)$ states.

Objective function: profit-to-go

The objective function to be maximized is the overall profit of the system. Because the operation of the supply chain is not limited to a specific finite time horizon, the problem is considered as an optimal control problem over an infinite time horizon. The overall profit is defined as the summation of the one-stage profit at each unit time over an infinite time horizon. To maximize the overall profit of the system over an infinite horizon, an approximate "profit-to-go" function, $J[X(k)]$, is defined as

$$J[X(k)] = E\{\text{Sum of all future profit}\} \simeq \hat{J}^0[X(k)] = E\left\{\sum_{i=0}^{H} \alpha^i \phi[X(k+i), u(k+i)]\right\} \quad (4)$$

with one-stage profit at time k, $\phi[X(k), u(k)]$ is defined by the following equation:

$$\phi[X(k), u(k)] = \text{Revenue}(k) - \text{Inventory Cost}(k) - \text{Manufacturing Cost}(k) - \text{Raw Material Cost}(k) \quad (5)$$

where $\phi(X, u)$ is a current profit function of the given state X and action u. The finite horizon H is taken to be long enough such that the discount factor α reduces future contributions to a level where they will not influence current decisions. The initial value of the approximated profit-to-go $\hat{J}^0(X)$ is calculated from appropriate suboptimal simulation data and the $\hat{J}^0(X)$ is used as an initial profit-to-go in the Bellman iteration step of DP, which refines it to the optimal profit-to-go $J^*(X)$.

Bellman iteration and real-time decision making

In DP, one calculates numerically the optimal profit-to-go function J^* by the Bellman iteration step. This computation can be done off-line, that is, before the policy is applied to the actual system, or on-line in a real-time dynamic programming setting.[24] In this article we take an off-line approach; for the SCM problem, the Bellman iteration equation is given as follows:

$$J^{i+1}[X(k)] = \max_{u(k)\in U} E\{\phi[X(k), u(k)] + \alpha J^i[X(k+1)|X(k), u(k)]\} \quad (6)$$

In the above, U is a discrete action space of all actions defined in Eq. 2. In finding the optimal $u(k)$, infeasible actions that make an inventory negative or violate the maximum production capacities of the plants in the system have to be excluded. The Bellman iteration is continued until J^i meets a certain convergence criterion, such as $\|(J^{i+1} - J^i)/J^i\|_\infty < 0.01$. If the J^i meets the convergence criterion, it is considered as the optimal profit-to-go J^* and is used for on-line decision making as follows. According to the "Principle of Optimality" of DP, the following decision $u^*(k)$ is the optimal action for the state $X(k)$, given at any time k:

$$u^*(k) = \arg\max_{u(k)\in U} E\{\phi[X(k), u(k)] + \alpha J^*[X(k+1)|X(k), u(k)]\} \quad (7)$$

The Algorithmic Framework: DP in a Heuristically Restricted State Space

All the necessary elements of the DP are defined in the previous section. Thus, the problem can be solved by using the appropriate Bellman equation shown in Eq. 6. However, the computational load of the full DP for a realistically sized example will exceed current computational resources. For example, the illustrative example shown in Figure 1 has a state space composed of 5.832×10^{10} discrete states even with state aggregation. The calculation of the number of states will be explained later in the Illustrative Example section. Furthermore, the number of discrete actions defined in Eq. 2 will be very large, even with the action space aggregation as discussed earlier. The number of states tends to increase exponentially with the problem size, number of inventories, number of products, and number of possible realizations. In our approximate DP framework, the part of the state space within which the cost-to-go is evaluated through the Bellman iteration is restricted to those visited during the simulation of the heuristic policies. The effectiveness of the algorithmic framework in handling a large state space was previously tested in Choi et al.[17] In this article, however, the algorithmic framework is modified to handle a large number of actions by introducing implicit subaction space for each state in the restricted state space. The implicit subaction space is defined with actions generated by a set of heuristics that have been applied during the heuristic simulation step of the framework. Once the action space complexity has been tamed, the general steps of applying the algorithmic framework are similar to those shown in Choi et al.[17] The main difference is the Bellman iteration and on-line decision making is performed not just over the confined state space but also over the implicit subactions spaces of the states:

(1) Stochastic simulation with the heuristic policies

(2) Identification of the restricted state space, which is composed of the states visited in the simulation and the first estimation of the profit-to-go values for the restricted state space using the simulation data

(3) Bellman iteration in the heuristically restricted state space and corresponding implicit subaction space

(4) Evaluation of the policy performance when applied to real-time decision making

Detailed descriptions of the steps are given in the next sections.

Learning stage: simulation of the heuristic policies

The purpose of the simulation is threefold. First, the simulation is performed to obtain a meaningful, manageably sized set of the states within which the DP is to be performed. Obtaining a reasonably sized subset containing trajectories of good policies is critical for solving the problem because DP over the entire state space is computationally infeasible for the given problem. For the simulation, a large number of uncertain parameter realization sets are generated by the underlying Markov chains. Each realization set represents one scenario along a certain time horizon out of the large number of possible scenarios. In the simulation, several different heuristics are applied for each realization and a set (trajectory) of visited states (as defined in the previous section) is obtained from each heuristic. Because each heuristic works in a different way, as they are designed to do, there can be several different state trajectories even for the same scenario. Those different state trajectories will be combined in the state space as profit-to-go in the later step (that is, Bellman iteration) of the algorithmic framework. The heuristic policies applied for the given problem will be described in a later section.

Second, the simulation provides initial “profit-to-go” values, which can be used to start up the Bellman iteration. There can be different “profit-to-go” values for a same state evolving into different trajectories (corresponding to different heuristics or realization scenarios). The initial “profit-to-go” for the state is an averaged “profit-to-go” value considering the number of times the state was visited.

Third, the initial “profit-to-go” values can be directly used for the *rollout* approach to find quick solution (policy) of the problem without the Bellman iteration. The rollout approach is described in the *Rollout approach: on-line decision making with initial profit-to-go* subsection for the illustrative example.

Implicit subaction space for a state

As stated earlier, the problem has a large action space resulting from the ability to control material flows involved in the supply chain. To reduce the computational load of the Bellman iteration and the on-line decision making, we propose to make a decision in an “implicit subaction space”—$U^{X(k)}$, a set of actions taken by the heuristics during the heuristic simulation—for each state in the confined state space. Thus, the number of the states in “subaction space” is the same as the number of states in the restricted state space. Instead of a set of aggregated actions, the set of heuristics that visited the state in the simulation are recorded for the implicit subaction space. In this way, we can avoid distorted state transitions introduced by action aggregation. The definition of action in Eq. 2 is kept the same, other than that implicit heuristic rule h_i is applied for the state $X(k)$, to reproduce a set of possible actions to be evaluated for the state:

$$u_i = h_i[X(k)] \qquad \text{for } i = i_1, \ldots, i_N \tag{8}$$

$$U^{X(k)} = \{u_{i_1}, \ldots, u_{i_N}\} \tag{9}$$

where i is the index of the heuristics that visited the state $X(k)$ in the simulation. Restricting the decision in the subaction space may prohibit choosing the optimal decision (action) for the given state. However, it ensures that a chosen action is feasible and increases the probability of the next state being in the restricted state space.

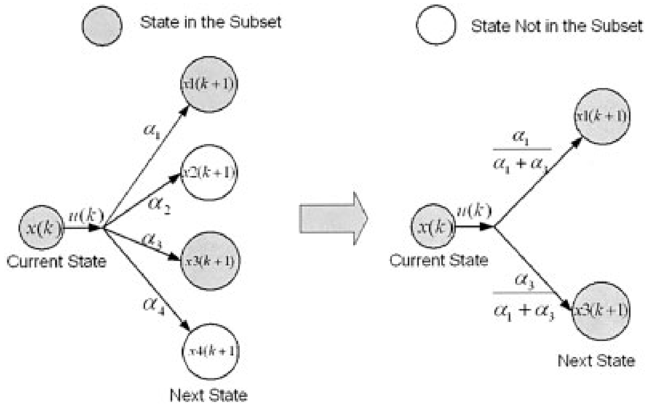

Figure 3. Profit-to-go approximation.

Bellman iteration over the confined state space

The Bellman iteration step of the proposed approach is identical to that in the conventional DP described in the previous section except for the following two details. First, it is done over the restricted state space instead of the entire state space. Second, in each iteration, the entire action space U is replaced with implicit subaction space $U^{X(k)}$, to provide possible actions to be evaluated:

$$J^{i+1}[X(k)] = \max_{u(k)\in U^{X(k)}} E\{\phi[X(k), u(k)] + \alpha J^i[X(k+1)|X(k), u(k)]\} \tag{10}$$

In calculation of Eq. 10, the current cost, $\phi[X(k), u(k)]$, is deterministic for a given state $X(k)$ and an action $u(k)$. The profit-to-go of the next state, $J[X(k+1) \mid X(k), u(k)]$ is stochastic as a consequence of the uncertain demand and price parameters. The exact expected value of $J[X(k+1) \mid X(k), u(k)]$ may not be calculated because all the possible next states may not be included in the restricted state space because of the limited number of realizations simulated. To obtain an approximate value of the $J[X(k+1) \mid X(k), u(k)]$, the cost-to-go approximation method, in which normalized weighting factor of the

Table 1. Probabilities and Parameters of the Markov Chains in the Illustrative Example

Markov Chain	Demand and Price	Probability Matrix
MC1 (Product A)	$\begin{bmatrix} 50 & 54 \\ 32 & 41 \\ 20 & 37 \end{bmatrix}$	$\begin{bmatrix} 0.60 & 0.10 & 0.10 \\ 0.30 & 0.50 & 0.20 \\ 0.10 & 0.40 & 0.70 \end{bmatrix}$
MC2 (Product B)	$\begin{bmatrix} 47 & 75 \\ 33 & 73 \\ 25 & 64 \end{bmatrix}$	$\begin{bmatrix} 0.50 & 0.30 & 0.15 \\ 0.35 & 0.50 & 0.25 \\ 0.15 & 0.20 & 0.60 \end{bmatrix}$
MC3 (Product C)	$\begin{bmatrix} 58 & 42 \\ 38 & 36 \end{bmatrix}$	$\begin{bmatrix} 0.8 & 0.3 \\ 0.2 & 0.7 \end{bmatrix}$
Markov Chain	RM Price	Probability Matrix
MC4 (RM for A)	$\begin{bmatrix} 18 \\ 25 \end{bmatrix}$	$\begin{bmatrix} 0.82 & 0.25 \\ 0.18 & 0.75 \end{bmatrix}$
MC5 (RM for B and C)	$\begin{bmatrix} 30 \\ 40 \end{bmatrix}$	$\begin{bmatrix} 0.85 & 0.40 \\ 0.15 & 0.60 \end{bmatrix}$

profit-to-go (or cost-to-go), is applied for the state in the restricted state space as shown in Figure 3. A detailed description of the approximation method is given in our previous article.[17]

Real-time decision making

The "converged" profit-to-go values obtained in the previous step are implemented for real-time decision making as follows:

$$u^*(k) = \arg \max_{u(k) \in U^{X(k)}} E\{\phi[X(k), u(k)] + \alpha J^*[X(k+1)|X(k), u(k)]\} \quad (11)$$

The calculation in on-line decision making is the same as that in the Bellman iteration except that it is needed only for the specific encountered state at the time. The decision gets implemented and the actual system (rather than the model) provides the next state. Because the decision belongs to the subaction space, the next state will be in the restricted state space, if enough simulations have been performed to reach all allowable outcomes. However, because exhaustive simulation is computationally infeasible, an unvisited state not in the restricted state space could be encountered. The state transitions can also be misled by the distortion induced by the state aggregation, arising from aggregation and disaggregation of the state in the state transition calculation. In this case,

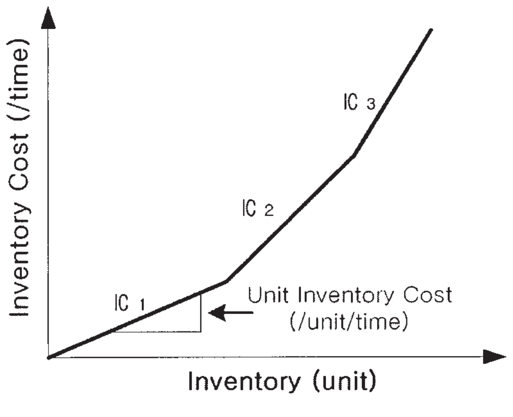

Figure 4. Piecewise linear inventory cost.

Table 2. Inventory Cost Parameters for the Illustrative Example

Inventory	Parameter			
$IA1$	Range	$0 \le i \le 60$	$60 < i$	
	Inv. Cost(/unit)	1.5	2.4	
$IA2$	Range	$0 \le i \le 40$	$40 < i \le 100$	$100 < i$
	Inv. Cost(/unit)	1.3	1.8	2.2
$IA3$	Range	$0 \le i \le 30$	$30 < i \le 50$	$50 < i$
	Inv. Cost(/unit)	1.5	2.0	3.0
$IB1$	Range	$0 \le i \le 60$	$60 < i \le 100$	$100 < i$
	Inv. Cost(/unit)	1.1	1.7	2.3
$IB2$	Range	$0 \le i \le 60$	$60 < i$	
	Inv. Cost(/unit)	1.4	2.0	
$IB3$	Range	$0 \le i \le 60$	$60 < i \le 100$	$100 < i$
	Inv. Cost(/unit)	1.6	2.2	2.9
$IC1$	Range	$0 \le i \le 60$	$60 < i$	
	Inv. Cost(/unit)	1.5	2.5	
$IC2$	Range	$0 \le i \le 60$	$60 < i \le 100$	$100 < i$
	Inv. Cost(/unit)	1.7	2.4	3.0
$IC3$	Range	$0 \le i \le 50$	$50 < i \le 90$	$90 < i$
	Inv. Cost(/unit)	1.9	2.5	3.3

the current state is replaced with the most "similar" state in the restricted state space according to the "state similarity criteria." In the given problem, the state is defined with three types of state variables, as represented in Eq. 1:

$$X(k) = [\text{Inventory Level, Ongoing Production, Information State}] \quad (12)$$

In finding the most similar state in the confined state space with the current state, priorities of the state variables are given in the order of information state variables, ongoing production state variables, and then inventory level state variables. The information state variables are considered the most important have to be matched first. Because the information state variables are not aggregated or disaggregated in state transitions, existence of the states with exactly the same information state variables in the restricted state space is ensured under the assumption that the restricted state space was obtained with a sufficient number of simulations. The "similar state" in the confined state space is searched for using the following criteria:

(1) States in the restricted state space with the same information state as the information state in the current state are searched.

(2) Among the states obtained in the previous step, states with the same ongoing production of the current state are searched.

• If none of the states obtained in the previous step has the same ongoing production state variables with the current state variable, find a state that has the minimum infinity norm of the difference from the current state.

Table 3. Plant Parameters for the Illustrative Example

Parameter	Plant A	Plant B	Plant C
P. cost(/unit)	5	9	4
Max. capacity	120	70	70
P. time	3	2	2
Fixed cost(/batch)	120	100	95
Ratio (R:P)*	1:1	1:1	1:2

*Raw material to product conversion rate.

Table 4. State Aggregation

Rep. Index*	1	2	3	4	5
Range of $IA1$**	0–25	25–50	50–75	75–100	100–∞
Rep. $IA1$[†]	13	38	63	88	113
Range of $IA2$	0–25	25–50	50–75	75–100	100–∞
Rep. $IA2$	13	38	63	88	113
Range of $IA3$	0–25	25–50	50–75	75–100	100–∞
Rep. $IA3$	13	38	63	88	113
Range of $IB1$	0–25	25–50	50–75	75–100	100–∞
Rep. $IB1$	13	38	63	88	113
Range of $IB2$	0–25	25–50	50–75	75–100	100–∞
Rep. $IB2$	13	38	63	88	113
Range of $IB3$	0–25	25–50	50–75	75–100	100–∞
Rep. $IB3$	13	38	63	88	113
Range of $IC1$	0–25	25–50	50–75	75–100	100–∞
Rep. $IC1$	13	38	63	88	113
Range of $IC2$	0–25	25–50	50–75	75–100	100–∞
Rep. $IC2$	13	38	63	88	113
Range of $IC3$	0–25	25–50	50–75	75–100	100–∞
Rep. $IC3$	13	38	63	88	113
Rep. Index	0	1	2	3	4
Range of $TA2$	0	0–35	35–70	70–105	105–∞
Rep. $TA2$	0	18	43	78	120
Range of $TB2$	0	0–20	20–40	40–60	60–∞
Rep. $TB2$	0	10	30	50	70
Range of $TC2$	0	0–20	20–40	40–60	60–∞
Rep. $TC2$	0	10	30	50	70

*Representative index in site representation.
**Aggregation range of the inventory level.
[†]Representative inventory level in reverse-aggregation.

Table 5. Heuristic 1

s_a1	S_a1	s_a2	S_a2	s_a3	S_a3
80	120	100	200	80	150
s_b1	S_b1	s_b2	S_b2	s_b3	S_b3
70	150	90	150	70	150
s_c1	S_c1	s_c2	S_c2	s_c3	S_c3
90	130	80	95	70	120

(3) Among the states obtained in the previous step, find a state that has the minimum infinity norm of the difference in inventory from the current state.

Illustrative Example

As an illustrative example of the SCM problem with uncertainty, we consider a supply chain with three products and multiple inventories linked to external suppliers, plants, and markets (customers) as shown in Figure 1. As stated earlier, the objective is to maximize the overall profit by controlling all the relevant material flows in the system. Demand and price of each product are uncertain in the market and evolve according to underlying Markov chains. Besides the demand and price, the raw material price is also assumed to follow an independent Markov chain. Five Markov chains (three for the products and two for the raw materials) are introduced in the example to represent the uncertain parameter variations. All transition probability matrices and uncertain parameters for the Markov chains are summarized in Table 1. In this case we have assumed that markets are driven by exogenous factors—such as competitor marketing plans and capacity swings, and general macroeconomic factors—which cause the market demand and price to "soften" in a correlated way. This is unlike the case where we have control over the price and thus might see changes in demand as the price we offer changes. The details of the Markov parameters do not affect the methodology, only the specific policy that will result from the computation.

Inventory cost arises from every inventory in the system and the cost is assumed to be a piecewise linear function shown in Figure 4. The cost can be a complex function of the state and the framework does not rely on linearity or convexity of the function, which is an advantage over traditional mathematical programming formulations. It is assumed that outsourcing is available for every inventory and thus the capacity of each inventory is infinite. However, as the amount of product in an inventory increases, the total inventory cost increases more rapidly as a result of the higher inventory cost parameters (that is, $IC_1 < IC_2 < IC_3$ in Figure 4). The inventory parameters of the problem are summarized in Table 2.

Three batch plants are involved in the example and each plant has certain maximum production capacity (/unit time), production time, and minimum production cost for a single batch of production. Those parameters for the three plants are summarized in Table 3. Besides the parameters mentioned above, we set a fixed internal transaction cost (=33) for product A in material flows $TA3$ and $TA4$. The internal transaction price will be used to evaluate the profit of the different product lines classified in Figure 1. Note that the overall profit of the entire system is identical regardless of the internal transaction price for product A.

Definition and aggregation of the state and action

The state for the illustrative example is defined as follows:

$$X(k) = [\underbrace{IA1(k), IA2(k), IA3(k), IB1(k), IB2(k), IB3(k), IC1(k), IC2(k), IC3(k),}_{\text{Inventory Levels}}$$
$$\underbrace{TA2(k-2), TA2(k-1), TB2(k-1), TC2(k-1),}_{\text{Ongoing Production Amounts}} \underbrace{Z1(k), Z2(k), Z3(k), Z4(k), Z5(k)}_{\text{Information State}}] \quad (13)$$

As discussed earlier, the state has to be aggregated to avoid an unmanageably large state space. Table 4 shows parameters used for the aggregation of the inventory levels and the ongoing production state variables $TA2(k-2)$, $TA2(k-1)$, $TB2(k-1)$, and $TC2(k-1)$.

Based on the state aggregation table and the state definition of the problem, the total number of states in the entire state space can be calculated to be $8.79 \times 10^{10} = 5^9 \times 5^4 \times (3^2 \times 2^3)$. For the illustrative example, action consists of the 11 relevant material flows in the system:

Table 6. Heuristic 2

s_a1	S_a1	s_a2	S_a2	s_a3	S_a3
80	100	120	220	60	130
s_b1	S_b1	s_b2	S_b2	s_b3	S_b3
80	120	70	100	80	120
s_c1	S_c1	s_c2	S_c2	s_c3	S_c3
100	150	60	110	100	150

$$\underline{U}(k) = [\underbrace{TA1(k),\, TA2(k),\, TA3(k),\, TA4(k),\, TA5(k),}_{\text{Product A Decisions}}$$
$$\underbrace{TB1(k),\, TB2(k),\, TB3(k),}_{\text{Product B Decisions}}\underbrace{TC1(k),\, TC2(k),\, TC3(k)}_{\text{Product C Decisions}}] \tag{14}$$

State Transition Rules. Next state transition rules are defined. As discussed earlier, there are two types of state variables: controllable and uncontrollable state variables. For the controllable state variables, the state transition rules are simply represented with the following material balance equations.

- State transition rules of the controllable state variables

$$IA1(k+1) = IA1(k) - TA2(k) + TA1(k) \tag{15}$$

$$IA2(k+1) = IA2(k) - TA5(k) - TA4(k) - TA3(k) + TA2(k-2) \tag{16}$$

$$IA3(k+1) = IA3(k) - DA(k) + TA5(k) \quad \text{if } IA3(k+1) > 0 \tag{17}$$

$$IA3(k+1) = 0 \qquad \text{if } IA3(k+1) \leq 0 \tag{18}$$

$$IB1(k+1) = IB1(k) - TB2(k) + TB1(k) + TA4(k) \tag{19}$$

$$IB2(k+1) = IB2(k) - TB3(k) + TB2(k-1) \tag{20}$$

$$IB3(k+1) = IB3(k) - DB(k) + TB3(k) \quad \text{if } IB3(k+1) > 0 \tag{21}$$

$$IB3(k+1) = 0 \qquad \text{if } IB3(k+1) \leq 0 \tag{22}$$

$$IC1(k+1) = IC1(k) - TC2(k) + TC1(k) + TA3(k) \tag{23}$$

$$IC2(k+1) = IC2(k) - TC3(k) + 2TC2(k-1) \tag{24}$$

Table 7. Heuristic 3

s_a1	S_a1	s_a2	S_a2	s_a3	S_a3
50	100	180	250	100	130
s_b1	S_b1	s_b2	S_b2	s_b3	S_b3
50	120	100	160	100	120
s_c1	S_c1	s_c2	S_c2	s_c3	S_c3
50	120	120	180	100	150

Table 8. Heuristic 4

s_a1	S_a1	s_a2	S_a2	s_a3	S_a3
80	100	120	220	70	130
s_b1	S_b1	s_b2	S_b2	s_b3	S_b3
80	120	100	160	100	150
s_c1	S_c1	s_c2	S_c2	s_c3	S_c3
70	120	100	250	80	150

$$IC3(k+1) = IC3(k) - DC(k) + TC3(k) \quad \text{if } IC3(k+1) > 0 \tag{25}$$

$$IC3(k+1) = 0 \qquad \text{if } IC3(k+1) \leq 0 \tag{26}$$

Among the controllable state transition equations, the retail inventory levels $IA3(k)$, $IB3(k)$, and $IC3(k)$ are only "partially" controllable because demands of the products $DA(k)$, $DB(k)$, and $DC(k)$ are uncertain. However, the state transition equations of those retained inventory levels are deterministic once the product demands are known from the Markov chain realization. The state transitions of the uncontrollable (information) state variables depend solely on the realization of the five underlying Markov chains.

Simulation results for the heuristics

Heuristics can be created by combining static inventory control policies for all the inventories in the system to control the supply chain; in this case there are nine inventory variables. Six heuristics are proposed, where each heuristic consists of nine static inventory control policies. The stationary inventory control parameters *s* and *S* of each heuristic are summarized in Tables 5 to 10. In the heuristics, the two raw material inventories of products *B* and *C*, *IC*1 and *IB*1, are replenished by the external supplier and the inventory *IA*2 with the ratio of 8:2 (80% from the supplier, 20% from the inventory *IA*2). In distributing the internal material flows, *TA*3 and *TA*4, in all of the heuristics, the required *TA*4 (to *IB*2) is satisfied first and the required flow *TA*3 (to *IC*2) is satisfied if *IA*2 still has surplus inventory after fulfilling the required *TA*4 according to the given heuristic. As an initial step of the algorithmic framework, the behavior of the supply chain under the six heuristics is simulated for a large number of realizations of Markov chains. Table 11 shows the performance of the heuristics for a certain set of realizations (30,000 time horizon).

From the simulation results shown in the Table 11, Heuristic #3 is better than the other heuristics in terms of the total profit. On the other hand, Heuristic #4 is the best in terms of the average customer service level of the products. The customer service level of a product is defined as the amount of fulfilled demand over the total demand for the product. The results shown in Table 11 indicate the possibility of improving the solutions obtained by the six heuristics by searching actions

Table 9. Heuristic 5

s_a1	S_a1	s_a2	S_a2	s_a3	S_a3
80	100	120	220	70	130
s_b1	S_b1	s_b2	S_b2	s_b3	S_b3
80	120	100	160	100	150
s_c1	S_c1	s_c2	S_c2	s_c3	S_c3
50	120	100	180	80	130

Table 10. Heuristic 6

s_a1	S_a1	s_a2	S_a2	s_a3	S_a3
80	140	110	230	80	110
s_b1	S_b1	s_b2	S_b2	s_b3	S_b3
90	140	90	160	100	130
s_c1	S_c1	s_c2	S_c2	s_c3	S_c3
40	80	100	170	120	160

over the restricted state space visited by the heuristics in the simulation because none of the heuristics is universally best for all cases.

Restricted state space and subaction space

In the previous section, the simulation is performed over 20 sets of realizations, which correspond to 600,000 realizations (20 sets with 30,000 realizations), which are used to obtain the restricted state space for the given problem. For each heuristic and each set of realizations, the simulation is carried out with three different initial inventory conditions—low, medium, and high—to capture different transient state trajectories until the supply chain operation reaches a stable pattern. Thus, the total number of individual realizations in the series of simulations is 10,800,000. For each individual realization, its profit-to-go value for the visited state [$\hat{J}(X)_{sim}$] can be calculated from simulation data by the profit-to-go approximation (Eq. 4) with $H = 100$ and $\alpha = 0.95$. If the state is already in the storage, the profit-to-go value of the state in the confined state space [$\hat{J}(X)_{cs}^{old}$] is updated by the following equation:

$$\hat{J}(X)^{new} = \frac{n\hat{J}(X)^{old} + \hat{J}(X)_{sim}}{n+1} \tag{27}$$

where n is the total number of times of the state X was visited in the series of simulations. On the other hand, if the state is not among the stored states, it is added as a new state in the storage with $n = 1$ along with the profit-to-go value $\hat{J}(X)_{sim}$. During the simulation, 1,433,694 states are visited and those states are recorded to define the restricted state space. The size (number of states) of the restricted state space increases as more and more simulation data are added. Figure 5 shows the increase of the states space size with the total number of realizations. As can be seen from the figure, the restricted state space appears "saturated" with 1,433,694 states, which is the criterion we apply to stop the simulation.

The subaction space is also obtained along with each state in the restricted state space. By restricting the action space for an individual state, in the Bellman iteration and in the real-time decision making, we can dramatically reduce the computational load because the action space is represented with only six

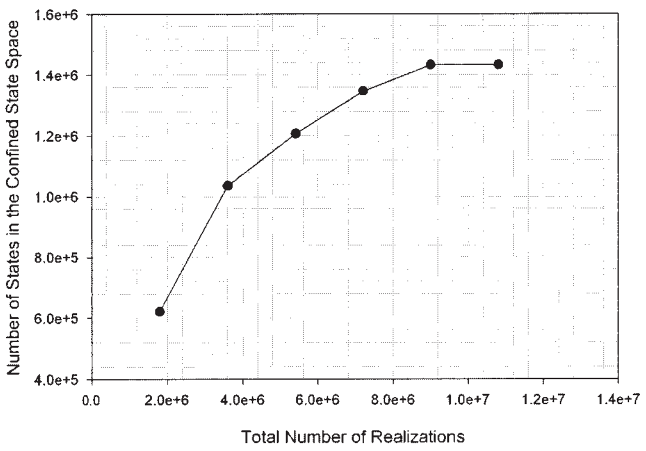

Figure 5. Increase in number of the states in the restricted state space with the number of realizations.

heuristics applied for the given problem rather than the enormous number represented by all possible combinations of the individual actions. Out of 1,433,694 stored states, 1,335,784 states have a subaction space defined by only one heuristic as a result of the limited overlap in the ranges of inventories in which the various heuristics operated. For such states, decision making is trivial because there is only one action to take. Table 12 shows the distribution of the number of actions (taken by the heuristics) over the stored states.

Rollout approach: on-line decision making with initial profit-to-go

The restricted state space and the subaction space can be used directly for on-line decision making even without the Bellman iteration. This way of approximating the "profit-to-go" is called the *rollout approach*, which is particularly well suited for deterministic combinatorial problems.[12] Based on the hypothesis that the heuristic simulation is done for enough number of realizations, the initial profit-to-go, $\hat{J}^0$, naturally contains a complex stochastic variation of the system and is good enough to be used for the real-time decision making as shown in the following equation:

$$u(k) = \arg \max_{u(k)\in U^{X(k)}} E\{\phi[X(k), u(k)] + \alpha\hat{J}^0[X(k+1)|X(k), u(k)]\} \tag{28}$$

The on-line policy obtained by the rollout approach is tested for the same set of realizations used for the results shown in

Table 11. Results of Simulating the Heuristics for Realization (30,000 Time Horizon): Set #1

	Profit A	Profit B	Profit C	Total Profit	CSLA*	CSLB	CSLC
H1	2.340e+6	2.107e+7	1.306e+7	3.647e+7	0.9441	0.9761	0.8882
H2	4.637e+6	1.585e+7	1.144e+7	3.193e+7	0.9546	0.7168	0.9199
H3	3.628e+6	2.078e+7	1.692e+7	4.133e+7	0.9796	0.8602	0.9525
H4	4.684e+6	2.004e+7	1.382e+7	3.854+7	0.9608	0.9836	0.9249
H5	4.794e+6	2.002e+7	1.429e+7	3.910e+7	0.9618	0.9835	0.8009
H6	2.438e+6	2.001e+7	1.608e+7	3.863e+7	0.9164	0.9751	0.9287

*Customer service level of the product A.

Table 12. Distribution of the Number of Actions in Subaction Space

Number of Heuristics	Number of States
1	1,335,784
2	807,807
3	13,903
4	2,759
5	449
6	19

Table 11. Computational results of the performance of the policy represented by Eq. 28 are summarized in Table 13. A comparison of Table 13 with Table 11 shows that the solution obtained by the rollout approach is slightly better (about 2.1%) than the best heuristic policy, Heuristic #3. The on-line decision making with the initial profit-to-go can be a quick alternative solution method when the Bellman iteration is not computationally feasible because of the large state space. Average computational time of the decision making for each realization (unit time) was only 0.3 s when implemented in MATLAB 6.3 on an Intel windows machine with dual Xeon 2.66-GHz CPUs and 2GB RAM.

Bellman iteration over the restricted state space

The Bellman iteration proposed in Eq. 10 is much faster than the conventional Bellman iteration in Eq. 6, for the given example, because of the small subaction space. As shown in Table 12, for more than 93% (1,335,784 out of 1,433,694) of the states in the state space, the decision making is trivial because only one action exists in the subaction space. At every iteration, 1,433,694 profit-to-go values are updated for the states in the restricted set. The Bellman iteration is performed until a certain convergence criterion [that is, $\| (J^{i+1} - J^i)/J^i \|_\infty < 0.01$] is met. According to the Bellman iteration in Eq. 10, a new set of profit-to-go values are obtained at each step of the iteration. With the intermediate profit-to-go values, on-line decision policy can be constructed by replacing the converged profit-to-go J^* with the intermediate profit-to-go $\hat{J}^i$:

$$u(k) = \arg \max_{u(k) \in U^{X(k)}} E\{\phi[X(k), u(k)] + \alpha \hat{J}^i[X(k+1)|X(k), u(k)]\} \tag{29}$$

Figure 6 shows the improvement in the total profit with the intermediate on-line policies when tested on the 30,000 realizations used for the simulation of the heuristics.

As the profit-to-go values are updated by the Bellman iteration, total profits of the intermediate solution are gradually improved. It should be noted that the profit improvement may be smaller for new sets of realizations that were not experienced during the simulation. Comprehensive computational analysis of the on-line policy with the converged profit-to-go will be shown in the next section for various sets of realizations

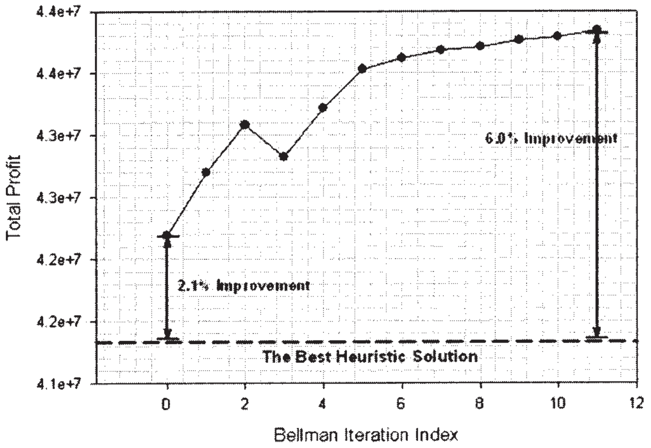

Figure 6. Total profit improvement with intermediate profit-to-go values.

to verify the robustness of the proposed approach. The iteration was converged with the error tolerance of $\| (J^{i+1} - J^i)/J^i \|_\infty < 0.01$ after the 15th iteration and took 8.2 days when implemented in MATLAB on a machine with two 2.66-GHz Xeon CPUs and 2GB RAM. Figure 7 shows the maximum relative error, $\| (J^{i+1} - J^i)/J^i \|_\infty < 0.01$, of the profit-to-go values in the Bellman iteration.

On-line decision making with the converged profit-to-go

On-line decision making with the subaction space and the converged profit-to-go defined in Eq. 11 is also computationally efficient compared to the conventional one in Eq. 7. If the state space is obtained with enough realizations under the simulated heuristics, the on-line decision making is guaranteed to be superior to any of the heuristics tested. To test the performance of the on-line supply operating policy based on the converged profit-to-go, 50 new sets of realizations were generated with the underlying Markov chain. Each set of realizations corresponds to a 1000-unit time horizon. Figure 8 shows the total profit improvement for the supply chain system with the resulting operating policy compared to the best heuristic, Heuristic #3.

The average improvement is 4.53%, with 1.32% and 7.84% being the minimum and maximum improvements, respectively. The results in Figure 8 demonstrate that the on-line policy with the converged profit-to-go performs well even for new sets of realizations that are not used in the training stage. However, the performance improvements are irregular and depend on the realization because the profit-to-go is approximated over a relatively long time horizon with $H = 100$ and $\alpha = 0.95$.

Figure 9 shows one-stage total profits obtained by the best heuristic and the DP policy over a certain time horizon, 400 to 500, in one of the realization sets (index #2) generated for the test shown in Figure 8. As we can see in the figure, the new

Table 13. Online Decision Making with Initial Profit-to-Go

	Profit A	Profit B	Profit C	Total Profit	CSLA	CSLB	CSLC
Rollout approach	3.664e+6	2.109e+7	1.744e+7	4.219e+7	0.9809	0.8624	0.9578
Best heuristic	3.628e+6	2.078e+7	1.692e+7	4.133e+7	0.9796	0.8602	0.9525

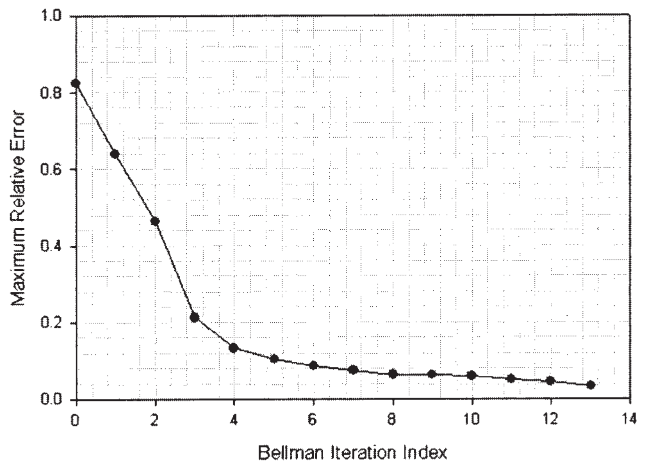

Figure 7. Maximum relative error, $\| (J^{i+1} - J^i)/J^i \|_\infty$, of the Bellman iteration.

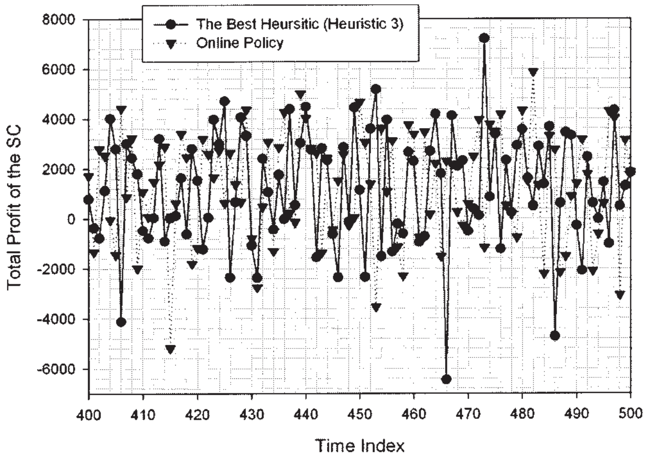

Figure 9. One-stage total profit (cost) comparison.

policy acts more "conservatively" than the best heuristic. The one-stage total profit of the DP policy neither results in as much profit nor incurs as much cost as the best heuristic. Indeed, for the given set of realization #2, standard deviations of the one-stage total profits are 2396.44 and 2301.26 for the best heuristic and the DP policy, respectively. The "conservative" behavior of the DP policy mainly arises from its ability to blend future information into the decision. The profit-to-go value represents an approximate value of future profit and the on-line decision making equation considers all possible next states and their realization probabilities in the decision-making action. Thus, the DP policy does not take an extreme action if a high loss is expected in the future.

We hypothesized that the DP policy will be a combination of the heuristics used in the simulation. Figure 10 shows the "shape" of the DP policy for the same realization set shown in Figure 9. Because the action space is implicitly represented by the subaction space introduced in the *Implicit subaction space for a state* subsection, every discrete action taken by the DP policy corresponds to a specific heuristic. Thus, the DP policy can be viewed as a piecewise combination of the heuristics. It chooses the best heuristic (Heuristic #3) most of the time, but switches to other heuristics for brief periods, thus bringing an improvement over any single heuristic.

Conclusions

Problems in supply chain management have assumed increasing importance in the continuous process industries as cost competitiveness intensifies, becoming global in scope. Solution methods for these problems need to address both the potentially complex models of the individual manufacturing facilities and the uncertainty in the information surrounding their operation, such as market demands. We have developed an approach to solve, approximately, the stochastic dynamic programming problem that results from considering the evolution of the uncertain parameters as a Markov chain and allowing the decisions to be based on the information available at each time step. The approach is based on the idea of performing rigorous dynamic programming over a state space that is deliberately restricted. The restricted space is constructed by simulating a set of heuristics and storing the states that are visited during the simulation. The optimal trajectory constructed from these states can be significantly better than the individual heuristics that generated them; in our case study the performance was improved by 4.5%. This results from the

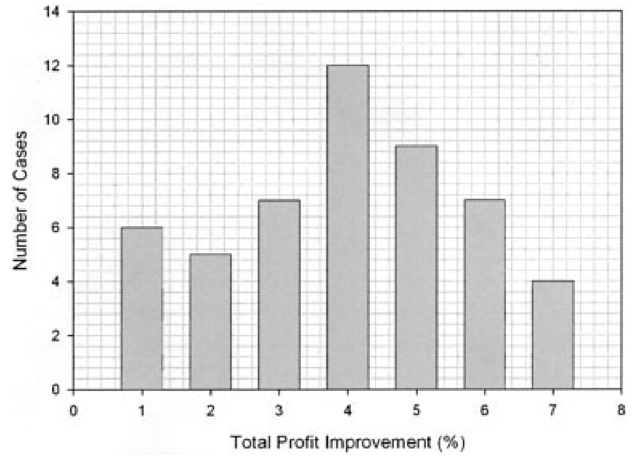

Figure 8. Improvement in the total profit with the policy based on the converged profit-to-go.

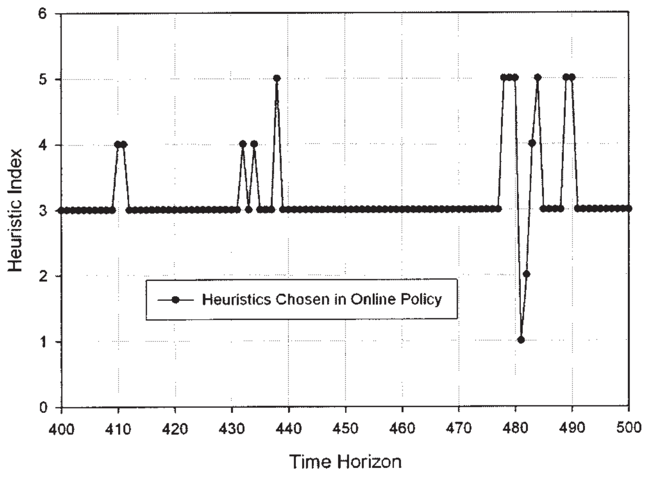

Figure 10. The on-line policy.

ability to choose the action based on a good estimate of the cost-to-go and context-sensitive information. The question of how much more improvement could be achieved with an optimal policy remains open. The computation of optimal policies appears daunting in these problems because of the size of the state and action spaces that must be considered. The computation of performance bounds is complicated by the multistage nature of the problem,[25] and deterministic bounds based on knowing the demand profile before solving for the policy are highly optimistic.

The approach presented herein can be applied to any problem formulated as a stochastic dynamic program, provided that there are reasonable heuristics available for simulation. The major effort is in building the necessary simulation program and the Bellman iteration over the restricted space, which can still be very large for realistic problems. The sensitivity of the solution quality both to errors in the Markov chain parameters and to the quality of the heuristics used to find the restricted state space is an important open question. Robust dynamic programming is in its early stages and has not yet provided results that can be applied to practically sized problems.[26]

Acknowledgments

J. H. Lee gratefully acknowledges the financial support from the National Science Foundation.

Literature Cited

1. Ierapetritou MG, Pistikopoilos EN. Batch plant design and operations under uncertainty. *Comput Chem Eng.* 2000;24:2613-2621.
2. Shah N. Single- and multisite planning and scheduling: Current status and future challenges. 3rd International Conference on Foundations of Computer-Aided Process Operations. *AIChE Symp Ser.* 1998;94:75-90.
3. Sabri EH, Beamon BM. A multi-objective apporach to simultaneous strategic and operational planning in supply chain design omega-international. *J Manage Sci.* 2000;28:581-598.
4. Tsiakis P, Shah N, Pantelides CC. Design of multi-echelon supply chain networks under demand uncertainty. *Ind Eng Chem Res.* 2001; 40:3585-3604.
5. Grahovac J, Chakravarty A. Sharing and lateral transshipment of inventory in a supply chain with expensive low-demand items. *Manage Sci.* 2001;47:579-594.
6. Belvaux G, Wolsey LA. Modelling practical lot-sizing problems as mixed-integer programs. *Manage Sci.* 2001;47:993-1007.
7. Gupta A, Maranas CD. Managing demand uncertainty in supply chain planning. *Comput Chem Eng.* 2003;27:1219-1227.
8. Clark AJ, Scarf H. Optimal policies for a multi-echelon inventory problem. *Manage Sci.* 1960;6:475-490.
9. Chen F. Optimal policies for multi-echelon inventory problems with batch ordering. *Oper Res.* 2000;48:376-389.
10. Verrijdt JHCM, Kok AG. Distribution planning for a divergent N-echelon network without intermediate stocks under service restrictions. *Int J Product Econ.* 1995;38:225-243.
11. Diks DB, Kok AG. Optimal control of a divergent multi-echelon inventory system. *Eur J Oper Res.* 1998;111:75-97.
12. Bertsekas DP. *Dynamic Programming and Optimal Control 1.2*. 2nd Edition. Belmont, MA: Athena Scientific; 1995.
13. Bhattacharjee S, Ramesh R. A multi-period profit maximizing model for retail supply chain management: An itegration of demand and supply-side mechanisms. *Eur J Oper Res.* 2000;122:584-601.
14. Cheng TCE, Kovalyov MY. Single supplier scheduling for multiple deliveries. *Ann Oper Res.* 2001;107:51-63.
15. Hall NG, Potts CN. Supply chain scheduling: Batching and delivery. *Oper Res.* 2003;51:566-584.
16. Choi J, Lee JH, Realff MJ. An algorithmic framework for improving heuristic solutions. Part II: A new version of stochastic traveling salesman problem. *Comput Chem Eng.* 2004;28:1297-1307.
17. Choi J, Lee JH, Realff MJ. Dynamic programming in a heuristically confined state space: A stochastic resource-constrained project scheduling application. *Comput Chem Eng.* 2004;28:1039-1058.
18. Choi J, Realff MJ, Lee JH. An algorithmic framework for improving heuristic solutions. Part I: A deterministic discount coupon traveling salesman problem. *Comput Chem Eng.* 2004;28:1285-1296.
19. Yin KK, Liu H, Johnson NE. Markovian inventroy policy with application to the paper industry. *Comput Chem Eng.* 2002;26:1399-1413.
20. Gupta A, Maranas CD, McDonald CM. Mid-term supply chain planning under demand uncertainty: Customer demand satisfaction and inventory management. *Comput Chem Eng.* 2000;24:2613-2621.
21. Weng ZK, McClurg T. Coordinated ordering decisions for short life cycle products with uncertainty in delivery time and demand. *Eur J Oper Res.* 2003;151:12-24.
22. Petkov DB, Maranas CD. Multiperiod planning and scheduling of multiproduct batch plants under uncertainty. *Ind Eng Chem Res.* 1997; 36:4864-4881.
23. Buchan J, Koenigsberg E. *Scientific Inventory Management*. 1st Edition. Englewood Cliffs, NJ: Prentice-Hall; 1963.
24. Barto AG, Bandtke SJ, Singh SP. Learning to act using real-time dynamic programming. *Artif Intell.* 1995;73:81-107.
25. Shapiro A. Inference of statistical bounds for multistage stochastic programming problems. *Math Methods Oper Res.* 2003;58:57-68.
26. Iyengar GN. Robust dynamic programming. *Math Oper Res.* 2005;30: 257-280.

Manuscript received Mar. 21, 2005, revision received Dec. 26, 2005, and final revision received Feb. 24, 2006.