INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 16 (2009) 347–359
DOI: 10.1111/j.1475-3995.2008.00679

An approximate dynamic programming approach to solving a dynamic, stochastic multiple knapsack problem

Thomas C. Perry[a] and Joseph C. Hartman[b]

[a]*LSI Logic, Allentown, PA, USA,*
[b]*Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA*
E-mail: hartman@ise.ufl.edu [Hartman]

Received 31 August 2005; received in revised form 14 August 2006; accepted 7 January 2008

Abstract

We model a multiperiod, single resource capacity reservation problem as a dynamic, stochastic, multiple knapsack problem with stochastic dynamic programming. As the state space grows exponentially in the number of knapsacks and the decision set grows exponentially in the number of order arrivals per period, the recursion is computationally intractable for large-scale problems, including those with long horizons. Our goal is to ensure optimal, or near optimal, decisions at time zero when maximizing the net present value of returns from accepted orders, but solving problems with short horizons introduces end-of-study effects which may prohibit finding good solutions at time zero. Thus, we propose an approximation approach which utilizes simulation and deterministic dynamic programming in order to allow for the solution of longer horizon problems and ensure good time zero decisions. Our computational results illustrate the effectiveness of the approximation scheme.

Keywords: stochastic dynamic programming; approximate dynamic programming; simulation

1. Introduction

The dynamic, stochastic knapsack problem can be described as follows: items arrive over time according to a stochastic process. The periodic capacity requirements and reward for each item are made known upon arrival. If accepted, the reward is received and capacity is allocated to the item. An item can only be accepted if there is sufficient capacity in the knapsack. This model has a number of applications, including capacity reservation, bandwidth allocation, and capital budgeting problems. Kleywegt and Papastavrou (1998, 2001), Lu (2005), Lu et al. (1999), Papastavrou et al. (1996), Ross and Tsang (1989), Ross and Yao (1990) and Van Slyke and Young (2000) study different versions of the problem. These variants include equal and variable sized items and deadlines.

Published by Blackwell Publishing, 9600 Garsington Road, Oxford, OX4 2DQ, UK and 350 Main St, Malden, MA 02148, USA.

We model the case where items arrive and may "stay" in the knapsack for a number of periods before exiting, freeing capacity for future arrivals. We model this with multiple knapsacks to capture multiple time periods, as it is assumed that decisions are made periodically. This model is motivated by the problem of reserving capacity for orders in continuous manufacturing systems. That is, accepted orders are allocated capacity over a number of periods. Upon completion of the order, the capacity is freed for future use. As orders arrive dynamically in time, the tradeoff (as in the single knapsack problem) is between accepting rewards of orders that have arrived versus saving the capacity for future arrivals.

Ross and Tsang (1989) study an analogous problem when allocating bandwidth in a communications switching network to randomly arriving calls of random length. They model this problem as a Markov Decision Process and determine a threshold policy for two given classes of calls. We provide a different modeling and solution approach as our manufacturing application assumes decisions about arriving orders are made periodically, such as monthly or quarterly. That is, our decisions are not made in real-time as in the communications network allocation problem. Furthermore, we consider multiple classes of arrivals.

Our goal is to maximize expected, discounted revenues over an infinite horizon, as we expect the system being studied to continue operations indefinitely into the future. However, as we are computationally restricted to solving a finite horizon problem, we limit ourselves to identifying the optimal time zero decision, which in practice is nearly as good as finding an optimal sequence of decisions as only the first decision is implemented immediately (Bean and Smith, 1984). The approach is to solve a long finite-horizon problem to mitigate end-of-study effects such that the time-zero decision does not change for any longer horizon length, including infinite, due to discounting. The solution horizon is often referred to as a planning or decision horizon (Bes and Sethi, 1988).

Unfortunately, long horizon problems are computationally challenging as the state space grows exponentially in the number of knapsacks and the number of feasible decisions each period is exponential in the number of arrivals. Thus, we approximate a longer horizon of analysis by estimating terminal state values for the finite-horizon stochastic dynamic program (SDP). The hope is to mitigate the end-of-study effects and identify the optimal time zero decision for an infinite horizon.

We have previously modeled this problem with SDP (Perry and Hartman, 2004). Motivated by the approximation techniques of Bertsimas and Demir (2002) and Topaloglu and Powell (2005), among others, we examined linear programming relaxation and dual approaches to estimate the value function (Hartman and Perry, 2006). In this paper, we examine the use of simulation and deterministic dynamic programming to approximate the cost-to-go, or value, function of the SDP. It should be noted that simulation has been successfully used in dynamic programming approximation schemes in various applications (e.g., Topaloglu and Powell, 2006, and Godfrey and Powell, 2002).

This paper proceeds as follows. In the following section, we present the SDP for the dynamic, stochastic, multiple knapsack problem. In Section 3, we present the approximation scheme, including the simulation approach and associated deterministic dynamic program. In Section 4 we illustrate the effectiveness of the approach for making time zero decisions. This is followed by our conclusions and directions for future research.

2. Dynamic, stochastic multiple knapsack problem

It is assumed that capacity is utilized continuously in this setting, but reservation decisions are made on a periodic basis, such as every quarter or month. Under this assumption, the state of the system is defined by the utilization of capacity at each period of time. Note that this is equivalent to modeling the available capacity of the system. We describe the components of the model in the following before defining the SDP.

2.1. System state

The state of the system is defined as the amount of utilized, or reserved, capacity over a specified number of time periods. This is also referred to as a "capacity window," which can be represented as a capacity utilization vector (also referred to as the state vector or the state of the system):

$$S_t = [b_t, b_{t+1}, b_{t+2}, \ldots, b_{t+K-1}],$$

where b_t is the utilized capacity in period t and capacity can be reserved up to K periods into the future. As each period can be viewed as a knapsack of capacity, the value of K defines the number of knapsacks in our model. For implementation, we discretize the capacity in each period, forming a grid with a total of β "capacity buckets."

2.2. Order arrival process

We assume that a maximum of N orders may arrive in a period. Each order is independent with periodic capacity requirements and revenue becoming known upon arrival. (The interarrival times of the orders are also independent.) The probability of a given order n arriving in a given period t is defined as $p_t(n)$. With N possible arrivals each period, there are 2^N possible combinations of orders each period. Define the set of all possible combinations as Ψ and a given realization (one combination) as ψ. The probability of a set of orders arriving in period t is then defined as

$$p_t(\psi) = \prod_{n \in \psi} p_t(n) \prod_{n \notin \psi} (1 - p_t(n)),$$

assuming independence. Note that our solution method is not based on whether arrivals are independent or not. We merely require the definition of $p_t(\psi)$, where ψ defines the orders to be analyzed.

2.3. Feasible decisions

Given the set of arrivals ψ in a given period, an accept or reject decision must be made about each order $n \in \psi$. Define $\delta \subseteq \psi$ as the set of accepted orders from the arrival set. If $r_t(n)$ defines the revenue for order n in period t, then the total revenue from the decision set δ is

$$R_t(\delta) = \sum_{n \in \delta} r_t(n),$$

given that the capacity constraint is obeyed in each period. If $w_t(n)$ is the capacity required by order n in period t, then:

$$C_{t+\tau}(\delta) = b_{t+\tau} + \sum_{n \in \delta} w_{t+\tau}(n) \leq B_{t+\tau} \quad \forall \tau = 0, 1, 2, \ldots, K-1,$$

where B_t is the capacity of the system in period t and $C_t(\delta)$ is the required capacity as a result of implementing δ. Note that we track capacity through the capacity window K, regardless of the solution horizon T, for solution purposes, as described later.

2.4. State transitions

The state vector represents the amount of utilized (reserved) capacity at the end of the period. Thus, the transition of the state vector is dependent on the decision set δ such that the amount of capacity reserved in a given period is dependent on previous reservations and on the orders accepted in that period. We defined the state of the system at time t as

$$S_t = [b_t, b_{t+1}, b_{t+2}, \ldots, b_{t+K-1}].$$

If we define the state of the system at time $t+1$ as

$$S_{t+1}(\delta) = \left[b'_{t+1}, b'_{t+2}, b'_{t+3}, \ldots, b'_{t+K}\right],$$

then it should be clear that the amount of reserved capacity is equal to the amount reserved previously, plus any new requirements:

$$b'_{t+1}(\delta) = b_{t+1} + \sum_{n \in \delta} w_{t+1}(n).$$

Similar transitions occur in all periods through $t+K$.

2.5. SDP recursion

Our goal is to maximize expected, discounted revenue over time. Define $V_t(S)$ as the maximum expected net present value of revenue when making optimal accept/reject decisions for orders from time period t through the horizon T. With α representing the one period discount factor, $V_t(S)$ is defined in the following recursion:

$$V_t(S_t) = \sum_{\psi \in \Psi} p_t(\psi) \left[\max_{\delta \subseteq \psi | C_{t+\tau}(\delta) \leq B_{t+\tau} \forall \tau} \{R_t(\delta) + \alpha V_{t+1}(S_{t+1}(\delta))\} \right], \quad t = 0, 1, \ldots, T. \tag{1}$$

The recursion maximizes revenue and the discounted value of future states given an arrival set ψ. We solve the recursion over T periods, assuming the terminal condition:

$$V_{T+1}(S_{T+1}) = 0 \quad \forall S_{T+1}. \tag{2}$$

More specifically, our goal is to make the optimal allocation decision at $t = 0$, as that is the decision to be implemented immediately and the problem can be solved again with updated information for future decisions. To mitigate end-of-study effects on the time zero solution, we require T to be large, which poses solution difficulties, as discussed in the following section.

2.6. Implementation limitations

As with many dynamic programming algorithms, this one suffers from the "curse of dimensionality" as coined by Bellman (1957). The number of states in a given period is defined by the granularity of discretized capacity β. As there are K periods of capacity tracked, this translates to a maximum of β^K states in a given period. For a given state, there are 2^N possible arrivals and thus 2^N possible decisions, resulting in a maximum of 2^{2N} possible paths from a given state. In all, recursion (1) can be solved in $O(T2^{2N}\beta^K)$ time. Note that this run time is based on the worst case.

We can drastically reduce the number of states to evaluate by only examining those that are "reachable." This is best explained by the network in Fig. 1. In the network, the square nodes represent states while circular nodes represent points in time where probabilistic events occur. The arcs represent feasible decisions if they are labeled with δ values while random events are identified with ψ values.

As the state of the system and all arrivals at time zero are known, the decision-maker need only determine which orders to accept. Figure 1 has three decision arcs emanating from the initial state node. Assuming two orders have arrived but there is only capacity available to accept one, these decisions represent accepting either order (δ_1 and δ_2) or rejecting both orders (δ_3). The decision results in a state transition. This in turn is followed by a random event (arrival of orders) for consideration at the next time period. We can build a "reachable" network of possible future states as in Fig. 1 from a given initial state with all possible random events and feasible decisions over the finite horizon. This network can then be traversed backwards with our recursion (1) in

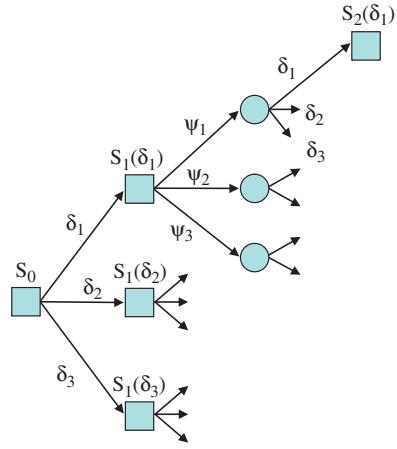

Fig. 1. Dynamic programming network considering reachable states from time zero.

order to determine the optimal decision at time zero, saving memory by only visiting states that may be on the optimal path.

Even with this implementation, execution times for long horizon problems may be prohibitive. This is a concern because it is necessary to solve a long horizon problem in order to mitigate end-of-study effects when finding good time zero decisions. The end-of-study effects are a result of the terminal condition (2) defining all states values as zero. As this assumption does not differentiate between states, initial time decisions may be altered from those that would be found with a longer horizon. Thus, we turn to approximation techniques to estimate state values for our terminal condition when we cannot solve problems with sufficiently long horizons.

3. Approximation approach

The SDP defined earlier was described by the network in Fig. 1. To describe our approximation approach, we expand the network to the one given in Fig. 2. The network illustrates the SDP up through time $T+1$. The remaining periods, through $T'+1$, represent our approximation network.

Recall that the SDP assumes that the values of each state at period $T+1$ are zero. The goal of our approximation approach is to estimate the value of each state at period $T+1$ under the

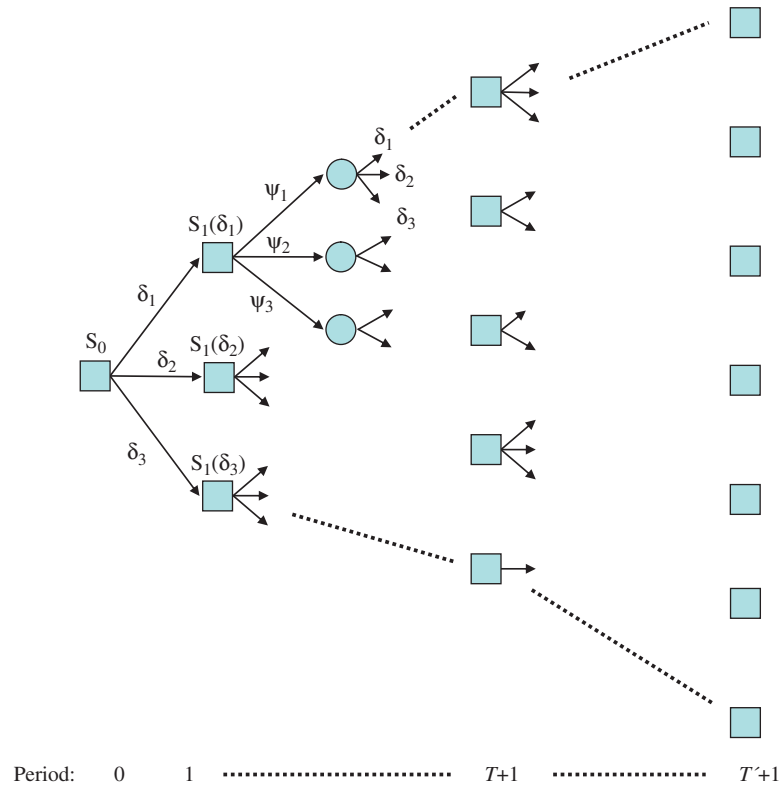

Fig. 2. Expanded network of reachable states for approximation approach.

assumption that the problem continues indefinitely. This is accomplished by solving an approximate problem over periods $T+1$ through T', the horizon for our approximation.

If we define the estimate of the value of state S in period $T+1$ from the approximation as $\tilde{V}_{T+1}(S)$, then we proceed with solving the SDP using the terminal condition $V_{T+1}(S) = \tilde{V}_{T+1}(S)$, as opposed to $V_{T+1}(S) = 0\ \forall S$. Note that this assumes that the SDP is tractable over T periods.

One approach to approximate the value function for each reachable state S at time $T+1$ is to simulate order arrivals from time $T+1$ through T' and solve the corresponding deterministic dynamic program. This process can be repeated in order to develop an estimate for the value of each reachable state S in the SDP at time $T+1$, as described in the following sections.

3.1. Simulation and deterministic dynamic programming

Assuming that we know the number of potential orders and the arrival rate, revenue and capacity requirements for each possible order in each period $T+1$ through T', we can simulate arrivals over that time period. For a given simulation scenario s, the set ψ_t' defines the set of orders that have arrived for consideration in each period t. Thus, the SDP defined by the recursion in (1) can be replaced by its deterministic counterpart for scenario s as

$$V_t^s(S_t) = \max_{\delta \subseteq \psi_t' | C_{t+\tau}(\delta) \le B_{t+\tau} \forall \tau} \left\{ R_t(\delta) + \alpha V_{t+1}^s(S_{t+1}(\delta)) \right\}, \quad t = T+1, T+2, \ldots, T'. \tag{3}$$

We implement the simulation and solution procedure as follows: for the reachable states S defined by the SDP at time $T+1$, we build a reachable network of states from time $T+1$ through time T'. The deterministic recursion (3) is then solved backwards by simulating arrivals at a given time t, solving the recursion, and then repeating at time $t-1$. However, the initialization process (defining $V_{T'+1}^s(S)$) is altered depending on whether the arrival process of orders is stationary or nonstationary, as described below.

3.1.1. Nonstationary implementation

If arrivals are nonstationary, then the deterministic recursion with simulated arrivals is solved repeatedly with the following end condition:

$$V_{T'+1}^s(S) = 0 \quad \forall S, s. \tag{4}$$

Once the simulation-solution process is complete, the state values at time $T+1$ are averaged with previous iterations, as

$$\tilde{V}_{T+1}^s(S) = \frac{(s-1)\,\tilde{V}_{T+1}^{s-1}(S) + V_{T+1}^s(S)}{s} \quad \forall S, s > 0. \tag{5}$$

We use the term $\tilde{V}_{T+1}(S)$ because these are the *estimated* state values at time $T+1$ which are used as the terminal condition of the SDP. That is, at the completion of s scenarios:

$$V_{T+1}(S) = \tilde{V}_{T+1}^s(S) \quad \forall S.$$

Thus, we repeat the simulation and recursion (3) procedure until the value of $\tilde{V}_{T+1}^s(S)$ stabilizes. Specifically, the procedure can be terminated after s scenarios when the change in the

value of $\tilde{V}^{s}_{T+1}(S)$ is less than some specified error, ε. The stochastic recursion (1) can then be solved, with the hopes that end-of-study effects have been diminished.

3.1.2. Stationary implementation

If the data is stationary, such that arrival information does not change with time, we can proceed differently through the simulation process in order to simulate an even longer horizon. We initiate the deterministic recursion as previously:

$$V^{0}_{T'+1}(S) = 0 \quad \forall S, \tag{6}$$

however, this is only used for the first scenario.

Once we have completed solving (3) for a given simulation scenario, we average the state values at time $T+1$ with the previous iterations as in (5). However, unlike our nonstationary implementation, for the ensuing scenarios, we include information from previous simulations to the next simulation via the terminal condition:

$$V^{s}_{T'+1}(S) = \tilde{V}^{s-1}_{T+1}(S) \quad \forall S, s > 0. \tag{7}$$

In essence, this approach implements a rolling horizon procedure such that the result is the solution of an $s(T' - T)$ horizon problem, where s is the total number of simulation scenarios.

Once the simulation procedure is complete, this information is passed to the SDP, replacing the terminal condition (2) with:

$$V_{T+1}(S) = \tilde{V}^{s}_{T+1}(S) \quad \forall S.$$

3.2. Approximation implementation

We previously described the implementation of the approximation and its integration into the SDP which is solved optimally over the T period horizon. In practice, it is assumed that the SDP will be re-solved whenever decisions are to be made (monthly or quarterly). This allows the decision-maker to update information about expected arrivals, returns, etc. Thus, the approximation procedure should also be re-run with new information. However, if future arrival information does not change between decision epochs, the results from previous approximation solutions may be utilized to solve SPD, speeding the process.

This is an important note because the approximation approach must still consider all combinations of order arrivals in the deterministic dynamic program. Thus, while it is easier to solve than the SDP, it is not trivial.

4. Implementation examples

To illustrate our approach, consider a problem with the following parameters: $T = 10$, $K = 4$, $\beta = 26$ (capacity is divided into 4% increments) with $N = 4$ possible arrivals each period. Furthermore, we assume a stationary arrival process for this example in that the four possible arrivals in each period

over time are described by the data defined in Table 1, including the probability of arrival in a given period. The classes are divided according to capacity requirements and revenue expectations. As one would expect, high volume orders will tend to have lower profit margins, but are assumed to arrive more frequently while low volume orders with high margins are expected to arrive less frequently. Note that capacity is reserved for the number of consecutive periods defined by "Life."

The goal in our problem is to identify the optimal time zero decision under the assumption of an infinite horizon. We examine the ability to find the time zero solution using various horizons with the SDP, both with and without the approximation approach.

4.1. Single arrival at time zero

To evaluate the effectiveness of the approximation, consider the situation where an order is present at time zero that requires 25% of system capacity for four consecutive periods (beginning immediately) for a revenue of \$1,500 and the initial state of the system is [6, 6, 6, 6], or roughly 25% full for the next four periods. This problem is solved over a number of periods both with (ASDP) and without (SDP) the simulation approximation. The resulting decision concerning the lone arrival and value of the state at time zero for each procedure are given in Table 2.

The solution of the SDP with terminal values of zero leads to different time zero decisions for different solution horizons due to the impact of end-of-study effects. These are clearly mitigated when examining the solutions of solving the SDP with approximate terminal values. We performed 100 simulation/deterministic dynamic programming iterations to generate these approximation values (taking nearly 6 h).

4.2. Multiple arrivals at time zero

We present an extreme case where the steady state of the system assumes a maximum of four arrivals over time as in the previous section, but there are *nine* different arrivals at time zero. We chose this larger problem to illustrate the need for decision support, as one could argue that a computational approach is not necessary for the analysis of single arrivals over time. The orders, with revenue and capacity requirements, are given in Table 3.

Again, we solved this problem over a number of periods with and without the approximation values assuming nine different initial states (capacities) from empty to nearly full. Table 4 provides the time zero decisions for each combination of initial state and horizon T both with and without

Table 1
Four classes of possible arrivals over time

Arrival	Volume class	Margin class	Probability of arrival (%)	Revenue	Required capacity (%)	Life
1	High	Low	75	\$2570	30	4
2	Medium	Medium	50	\$4000	20	4
3	High	Medium	35	\$6000	30	4
4	Low	High	25	\$4667	10	4

Table 2
Time zero decisions for various solution approaches and horizons

Solution method	Horizon T	Horizon $T' - T$	$V_0(S)$	Decision	Solution time (s)
SDP	1	–	\$1,500	Accept	0.21
SDP	2	–	\$6,609	Reject	0.96
SDP	3	–	\$10,522	Reject	2.33
SDP	4	–	\$14,840	Reject	5.16
SDP	5	–	\$17,282	Accept	12.67
SDP	6	–	\$21,182	Accept	26.16
SDP	9	–	\$31,286	Reject	72.78
ASDP	1	10	\$84,619	Reject	0.33
ASDP	5	10	\$92,271	Reject	12.13
ASDP	9	10	\$99,146	Reject	74.52

Table 3
Revenue and capacity requirements for nine orders arriving at time zero

Order	Revenue	Required capacity (%)	Life
1	\$2750	32	3
2	\$4000	23	3
3	\$4667	11	3
4	\$6000	30	4
5	\$960	17	2
6	\$931	12	1
7	\$1111	13	1
8	\$961	6	2
9	\$994	10	2

the approximation. The digits in the table refer to the orders that are accepted for the given problem instance. For example, the number 23478 refers to accepting orders 2, 3, 4, 7 and 8 while rejecting orders 1, 5, 6 and 9. A bold digit (solution) in the table identifies the horizon for which all decisions remain consistent for all longer horizons for a given solution procedure and initial state.

Examining the time zero decisions when using the approximation shows that the decisions are stable for nearly all solution horizons and initial states. For the final eight scenarios of differing initial capacities, the time zero decision does not change for solutions over 2 through 10 periods. Only in the first scenario does the first decision differ for horizons of 2 and 3 periods. However, the same decision is arrived at in the remaining horizons (4 through 10 periods) of analysis. This gives the decision-maker assurance that they have identified the optimal time zero decision.

This consistency is not present when solving the SDP without the approximation. For the empty system, three different time zero decisions are found over various horizons while for the nearly empty system (state [0, 1, 2, 3]), four time zero decisions are found with different solution horizons. Of the nine capacity scenarios, only two have consistent decisions over all solution horizons. Furthermore, the longest horizon solutions (10 periods) for initial states of [0, 0, 0, 0] and [3, 2, 1, 0] do not coincide with the solutions from the approximation approach. Thus, unlike

Table 4
Accepted orders from nine arrivals at time zero when solving SDP with and without approximation with horizons between 2 and 10 periods and various initial state capacities

State\horizon	2	3	4	5	6	7	8	9	10
SDP solutions									
[0, 0, 0, 0]	234689	23467	346789	234689	234689	23467	23467	**234689**	234689
[0, 1, 2, 3]	23479	23467	346789	236789	23467	23467	346789	**23467**	23467
[3, 2, 1, 0]	23478	23478	346789	236789	**23489**	23489	23489	23489	23489
[5, 5, 5, 5]	2349	3467	34678	23679	2346	**34678**	34678	34678	34678
[5, 11, 17, 23]	**6789**	6789	6789	6789	6789	6789	6789	6789	6789
[11, 11, 11, 11]	349	346	3679	3679	346	346	**3679**	3679	3679
[17, 17, 17, 17]	389	37	368	368	368	37	**368**	368	368
[23, 17, 11, 5]	389	**368**	368	368	368	368	368	368	368
[23, 23, 23, 23]	8	8	8	8	8	8	8	8	8
SDP with approximation solutions									
[0, 0, 0, 0]	23467	234689	**23467**	23467	23467	23467	23467	23467	23467
[0, 1, 2, 3]	**23467**	23467	23467	23467	23467	23467	23467	23467	23467
[3, 2, 1, 0]	**23478**	23478	23478	23478	23,478	23478	23478	23478	23478
[5, 5, 5, 5]	**34678**	34678	34678	34678	34678	34678	34678	34678	34678
[5, 11, 17, 23]	**6789**	6789	6789	6789	6789	6789	6789	6789	6789
[11, 11, 11, 11]	**3679**	3679	3679	3679	3679	3679	3679	3679	3679
[17, 17, 17, 17]	**368**	368	368	368	368	368	368	368	368
[23, 17, 11, 5]	**368**	368	368	368	368	368	368	368	368
[23, 23, 23, 23]	**8**	8	8	8	8	8	8	8	8

the approximation approach, this does not provide assurance that the decision-maker is making the best decision at time zero.

The solution times for each of these problems varied between 6.5 s (two period problem with capacity of [23, 23, 23, 23]) to 103.9 s for the 10 period problem with an empty system. The simulation results used in the previous section were also used here (recall that 100 iterations took just under 6 h). While the approximation problem takes time, recall that it can be solved offline between decision epochs.

5. Conclusions and directions for future research

We have presented a SDP recursion to a dynamic, stochastic, multiple knapsack problem. The state space grows exponentially in the number of knapsacks and potential order arrivals each period, leading to computational difficulties when finding optimal time zero decisions as end-of-study effects may be present when solving the model over small horizons. However, these effects can be mitigated by using approximate end-of-horizon state values (terminal conditions). We combine simulation and deterministic dynamic programming to approximate these state values. This approach approximates solving an infinite horizon problem, providing assurance to the decision-maker that they are making good time zero decisions, which are to be implemented immediately.

There are a number of avenues for future research. Clearly, this approach with simulation can be refined such that the amount of work spent simulating and solving the approximation can be reduced. We did not overly concern ourselves with this issue as the simulation was performed offline. However, real-time applications might require that the approximation be updated repeatedly, thereby forcing faster solution times.

The model could also be extended for various additional decisions, most notably, capacity expansion. If one has estimates of the possible growth in future orders, then an additional decision can be made as to whether capacity should be increased or not. While this would add to the complexity of the model, it would clearly fit into this presented solution framework.

Acknowledgments

We would like to recognize National Science Foundation funding for this research through grants DMI-9984891 and DMI-0121395.

References

Bean, J.C., Smith, R.L., 1984. Conditions for the existence of planning horizons. *Mathematics of Operations Research* 9, 391–401.

Bellman, R.E., 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Bertsimas, D., Demir, R., 2002. An approximate dynamic programming approach to multidimensional knapsack problems. *Management Science* 48, 4, 550–565.

Bes, C., Sethi, S., 1988. Concepts of forecast and decision horizons: applications to dynamic stochastic optimization problems. *Mathematics of Operations Research* 13, 295–310.

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.

Hartman, J.C., Perry, T.C., 2006. Approximating the solution of a dynamic, stochastic, multiple knapsack problem. *Journal of Control and Cybernetics* 35, 3, 535–550.

Kleywegt, A.J., Papastavrou, J.D., 1998. The dynamic and stochastic knapsack problem. *Operations Research* 46, 17–35.

Kleywegt, A.J., Papastavrou, J.D., 2001. The dynamic and stochastic knapsack problem with random sized items. *Operations Research* 49, 1, 26–41.

Lu, L.L., Chiu, S.Y., Cox, L.A. Jr., 1999. Optimal project selection: Stochastic knapsack with finite time horizon. *Journal of Operations Research Society* 50, 645–650.

Lu, Y., 2005. Solving a dynamic resource allocation problem through continuous optimization. *IEEE Transaction on Automatic Control* 50, 6, 890–894.

Papastavrou, J.D., Rajagopalan, S., Kleywegt, A.J., 1996. The dynamic and stochastic knapsack problem with deadlines. *Management Science* 42, 12, 1706–1718.

Perry, T.C., Hartman, J.C., 2004. Allocating manufacturing capacity by solving a dynamic, stochastic, multiple knapsack problem. Technical Report T004-009, Industrial and Systems Engineering, Lehigh University, Bethlehem, PA.

Ross, K.W., Tsang, D., 1989. The stochastic knapsack problem. *IEEE Transactions on Communications* 37, 7, 740–747.

Ross, K.W., Yao, D.D., 1990. Monotonicity properties stochastic knapsack. *IEEE Transactions on Information Theory* 36, 1173–1179.

Topaloglu, H., Powell, W.B., 2005. A distributed decision making structure for dynamic resource allocation using nonlinear functional approximations. *Operations Research* 53, 2, 281–297.

Topaloglu, H., Powell, W.B., 2006. Dynamic programming approximations for stochastic, time-staged integer multicommodity flow problems. *INFORMS Journal on Computing*, 18, 31–42.

Van Slyke, R., Young, Y., 2000. Finite horizon stochastic knapsacks with applications in yield management. *Operations Research* 48, 155–172.