Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

A LAGRANGIAN APPROACH TO DYNAMIC RESOURCE ALLOCATION

Yasin Gocgun
Archis Ghate

Industrial and Systems Engineering
University of Washington
BOX 352650, Seattle, WA 98195, USA

ABSTRACT

We define a class of discrete-time resource allocation problems where multiple renewable resources must be dynamically allocated to different types of jobs arriving randomly. Jobs have geometric service durations, demand resources, incur a holding cost while waiting in queue, a penalty cost of rejection when the queue is filled to capacity, and generate a reward on completion. The goal is to select which jobs to service in each time-period so as to maximize total infinite-horizon discounted expected profit. We present Markov Decision Process (MDP) models of these problems and apply a Lagrangian relaxation-based method that exploits the structure of the MDP models to approximate their optimal value functions. We then develop a dynamic programming technique to efficiently recover resource allocation decisions from this approximate value function on the fly. Numerical experiments demonstrate that these decisions outperform well-known heuristics by at least 35% but as much as 220% on an average.

1 Introduction

We consider the following class of problems henceforth termed Dynamic Resource Allocation Problems (DRAPs):

1. (Discrete-time infinite horizon) We work in a discrete-time infinite-horizon setting.
2. (Heterogeneous job types) The index set of job types is denoted $\mathscr{I} = \{1,2,\dots,I\}$.
3. (Random arrivals) Up to $K^i < \infty$ new jobs of type $i \in \mathscr{I}$ may arrive during a time-period for some positive integer K^i. The probability that $0 \le m \le K^i$ new jobs of type $i \in \mathscr{I}$ arrive during a time-period is $p^i(m)$. Arrivals across different types are independent.
4. (Preemption) An ongoing job can be interrupted and resumed later.
5. (Geometric job durations) Type i jobs need to be served for a geometrically distributed number of time-periods before they are complete. These geometric job durations are independent random variables with success probability $0 < q^i \le 1$. From a practical viewpoint, an important special case corresponds to $q^i = 1$ where type i jobs are completed in one time-period (see for instance recent work on patient service in healthcare (Green, Savin, and Wang 2006)).
6. (Multiple resource constraints) $\mathscr{J} = \{1,\dots,J\}$ denotes the set of resources and $b^j < \infty$ (positive integer) is the total quantity of resource $j \in \mathscr{J}$ available for consumption in each time-period. In order to perform a job of type $i \in \mathscr{I}$, a non-negative integer amount $a^{ij} \ge 0$ of resource $j \in \mathscr{J}$ is required. We assume that for each job type $i \in \mathscr{I}$, there is at least one resource $j \in \mathscr{J}$ such that $a^{ij} > 0$.
7. (Jobs in queue) Incomplete jobs form a queue and $W^i < \infty$ denotes the queue capacity for type $i \in \mathscr{I}$ jobs.
8. (Rewards and costs) The following rewards/costs are received/incurred:

- (Completion reward) Reward R^i is received on completing a type $i \in \mathscr{I}$ job at the end of a time-period.
- (Holding cost) Each type $i \in \mathscr{I}$ job in queue incurs a holding cost of H^i per period. We assume that this cost is charged at the beginning of a period on jobs remaining in queue after selecting the jobs to be served in that period.
- (Penalty cost) Type $i \in \mathscr{I}$ jobs that arrive when W^i jobs of that type are in queue are rejected incurring a penalty cost of G^i per rejected job. We assume that this cost is also charged at the end of a period.

9. (Total discounted profit objective) The goal is to decide which jobs to serve in each period so as to maximize the total discounted expected profit over an infinite horizon with discount factor $0 < \alpha < 1$.

DRAPs and their variations arise in several applications. In a flexible manufacturing system, jobs correspond to orders for different products and resources include manufacturing equipment (Tolio 2009). In high performance computing facilities such as a supercomputer or a multi-processor machine with an operating system that allows dynamic resource reconfiguration (e.g., Solaris 10 on Sun), jobs correspond to computing tasks and resources to CPUs, memory, bandwidth, and storage space (Vengerov 2007). In communication networks, jobs correspond to requests for data, voice, or video transmission and the resource is bandwidth (Ross and Tsang 1989, Altman 2002). In healthcare management, jobs relate to elective, that is, non-emergency patient requests for hospital admission and resources correspond to diagnostic/treatment equipment, hospital beds and personnel such as residents, doctors, and nurses (Nunes, de Carvalho, and de Cássia Meneses Rodrigues 2009).

Problems related to DRAPs have been studied in the literature. One class is termed Network Revenue Management Problems (NRMPs) and is described as follows (Topaloglu 2009). Itinerary requests for a set of flight legs arrive randomly over time. An accept/reject decision is made for each arriving request. Accepted requests generate revenue and consume capacity on the corresponding flight legs. A rejected request is lost. The goal is to maximize expected revenue over a planning horizon. The second group is called Dynamic Stochastic Knapsack Problems (DSKPs) (Perry and Hartman 2009, Kleywegt and Papastavrou 1998), where items with random weights and rewards arrive according to a stochastic process and are either accepted or rejected for inclusion in a knapsack. These classes differ from DRAPs in several ways. First, the resource, for instance, the flight leg capacity is non-renewable. Second, the complicating notion of "job duration" is irrelevant. Moreover, requests do not queue. In addition, in DRAPs, arriving jobs are never actively rejected unless the queue is full. Luh, Miao, Chang, and Castanon (1989) investigate job selection problems with random arrivals but allow only one renewable resource and deterministic job durations. Moreover, they work with the simplifying assumption that all newly arriving jobs are of distinct types. Arriving jobs leave the system if they are not completed by a deadline. This leads to mathematical models entirely different from those of DRAPs and owing to computational difficulties the authors are only able to solve problems with two job types. Finally, random job arrivals and stochastic durations distinguish DRAPs from static-deterministic renewable resource allocation problems that are more common in the operations research literature (Schwindt 2005).

The reader will also notice conceptual similarities between DRAPs and the now classic multi-class queuing model with Poisson arrivals (Harrison 1975) and some of its extensions. The main difference is that DRAPs explicitly include multiple resource constraints, and in addition, accommodate an essentially arbitrary discrete-time arrival process along with finite queue capacities and allow preemption. Some of these points also differentiate DRAPs from problems in which Gittins index-based (Gittins 1979, Sonin 2008) and other structured policies are known to be optimal (Megow, Uetz, and Vredeveld 2006, Pinedo 2008).

A different class of resource allocation problems arising in dynamic fleet management applications have also received considerable attention (Godfrey and Powell 2002a, Godfrey and Powell 2002b, Topaloglu and Powell 2005). In these problems, the resources correspond to transport vehicles such as trains and trucks which occupy different locations in a transportation network. Jobs correspond to shipping requests of distinct types, for instance, characterized by source-destination pairs. Decisions include either moving (empty) vehicles from one location to another or servicing shipping requests. These decisions are assumed to require a fixed number of time-periods to execute. The state of this system is captured by locations of all vehicles resulting in mathematical models structurally different from and computationally more challenging than those of DRAPs.

In this paper, we present a Markov Decision Process (MDP) (Puterman 1994) model of DRAPs. Exact solution of this model is intractable because its state- and action-spaces are both exponential in I. We therefore apply a Lagrangian relaxation-based technique (Adelman and Mersereau 2008) that exploits the structure of our MDP to approximate its optimal value function. We then design a dynamic programming approach to efficiently recover resource allocation decisions from this approximate value function. Numerical experiments on randomly generated problem instances indicate that these decisions outperform myopic and μc-type heuristics that are often used as benchmarks in the queuing literature (Cox and Smith 1961).

2 A special case with deterministic job durations

We first develop an MDP model for the special case of DRAPs where each job is completed in one time-period, that is, $q^i = 1$ for all $i \in \mathscr{I}$. As we shall see later in Section 3, this MDP model is structurally identical to that in the geometric durations case but needs much simpler notation to describe. In this special case, all jobs in queue at the beginning of a time-period are precisely those that arrived during the previous time-period. Thus the state of our MDP model is defined as $x = (x^1, x^2, \ldots, x^I)$, where x^i, for $i = 1, 2, \ldots, I$, is the number of type i jobs in queue at the beginning of a time-period. Let X_i denote the set $\{0, 1, \ldots, W^i\}$ for $i = 1, 2, \ldots, I$. The set X of all possible states is then defined as the Cartesian product $X = X_1 \times X_2 \times \ldots \times X_I$. The decision vector is represented by $u = (u^1, u^2, \ldots, u^I)$, where u^i, for $i = 1, 2, \ldots, I$, is the number of type i jobs that we choose to serve in a time-period after observing state x. Let $U^i(x^i) = \{0, 1, \ldots, x^i\}$ and $U(x)$ denotes the Cartesian product $U_1(x^1) \times U_2(x^2) \times \ldots \times U^I(x^I)$. The set $\bar{U}(x) \subseteq U(x)$ of all decision vectors feasible in state x is defined by the resource constraints

$$\bar{U}(x) = \left\{ (u^1, u^2, \ldots, u^i) \in U(x) : \sum_{i=1}^{I} a^{ij} u^i \leq b^j,\ j = 1, 2, \ldots, J \right\}. \tag{1}$$

The discounted expected profit accumulated during this period is given by

$$f(x,u) = \underbrace{\alpha \sum_{i=1}^{I} R^i u^i}_{\text{reward}} - \underbrace{\sum_{i=1}^{I} H^i (x^i - u^i)}_{\text{holding cost}} - \underbrace{\alpha \sum_{i=1}^{I} \sum_{n_i=0}^{K^i} p^i(n_i) G^i(\max\{(x^i - u^i) + n_i - W^i, 0\})}_{\text{penalty cost}}. \tag{2}$$

Specifically, $f(x,u)$ equals $\sum_{i=1}^{I} f_i(x^i, u^i)$, where

$$f_i(x^i, u^i) = \left\{ \alpha R^i(u^i) - H^i(x^i - u^i) - \alpha \sum_{n_i=0}^{K^i} p^i(n_i) G^i(\max\{(x^i - u^i) + n_i - W^i, 0\}) \right\}. \tag{3}$$

Let $V(x)$ be the maximum total infinite-horizon discounted expected profit obtained from the current and all future periods if the current state is x. This optimal value function can in theory be obtained as the solution of the following Bellman's equations for all $x \in X$:

$$V(x) = \max_{u \in \bar{U}(x)} \left\{ \sum_{i=1}^{I} f_i(x^i, u^i) + \alpha \sum_{n_1=0}^{K^1} \ldots \sum_{n_I=0}^{K^I} \left(\prod_{i=1}^{I} p^i(n_i) \right) V(x'^1, x'^2, \ldots, x'^I) \right\}, \tag{4}$$

where

$$x'^i = \min\{(x^i - u^i) + n_i, W^i\},\ \text{for } i = 1, 2, \ldots, I. \tag{5}$$

Common approaches for solving these implicit equations and obtaining a corresponding optimal policy, that is, an optimal decision vector in every possible state, include policy iteration, value iteration, or linear programming (Ross 1983, Puterman 1994, Bertsekas 2007). These exact methods however are intractable for most realistic-scale problems as they suffer from the curse-of-dimensionality stemming from exceedingly large state and action spaces. For instance, in our example above, the state-space and action-space are both exponential in number of job types I rendering exact computation of the optimal value function and a corresponding optimal policy intractable. We illustrate this briefly through the linear programming method.

Let $\beta(x) > 0$ for $x \in S$ be any numbers indexed by the states in our example such that $\sum_{x\in S} \beta(x) = 1$. Thus, $\beta(\cdot)$ can be interpreted as the initial state probabilities. Then the optimal state values $V(x)$ for all $x \in X$ can be simultaneously obtained as an optimal solution of the following linear program in variables $\nu(x)$ indexed by $x \in S$:

$$(LP)\ F(\beta) = \min_{\nu(\cdot)} \sum_{x\in S} \beta(x)\nu(x)$$

$$\text{s.t.}\ \ \nu(x) - \alpha \sum_{n_1=0}^{K^1} \sum_{n_2=0}^{K^2} \cdots \sum_{n_I=0}^{K^I} \Big(\prod_{i=1}^{I} p^i(n_i)\Big) \nu(x'^1, x'^2, \ldots, x'^I) \geq f(x,u),\ \forall x \in X,\ \forall u \in \bar{U}(x),$$

where $x'^1, x'^2, \ldots, x'^I$ are defined in Equation (5). Observe that $F(\beta)$ is used in (LP) to denote the weighted optimal value function $\sum_{x\in S} \beta(x)V(x)$. Note that the number of variables in (LP) equals the cardinality $|S| = \prod_{i=1}^{I}(W^i + 1)$, and the number of constraints is given by $\sum_{x\in S} |\bar{U}(x)|$. Both these are exponential in I and hence (LP) is intractable for problems with more than a few job types. For example, the linear program in a problem with ten job types and queue capacity of three for each type has over a million variables.

Fortunately, the above MDP model has a special structure. The state-space X is a *Cartesian product* over I job types — $X_1 \times X_2 \times \ldots \times X_I$; feasible decision vectors in state $x \in X$ belong to a subset $\bar{U}(x)$ of the *Cartesian product* $U_1(x^1) \times U_2(x^2) \ldots \times U^I(x^I)$ over the I components $x^1, x^2, \ldots, x^I$ of state x; the expected profit function $f(x,u)$ is additively separable over job types (recall Equations (2)-(3)); the "state transition probabilities" are multiplicatively separable over the I job types; and consequently, the only feature linking the I job types is the presence of the resource constraints that define the set $\bar{U}(x)$ as in Equation (1). In other words, this MDP model is *weakly coupled* (Adelman and Mersereau 2008, Bertsimas and Mersereau 2007, Hawkins 2003). As a result, a Lagrangian relaxation-based approximation technique specifically developed for weakly coupled MDPs (Adelman and Mersereau 2008) is applicable to our model.

2.1 Problem decomposition by Lagrangian relaxation

The general Lagrangian relaxation approach for approximate solution of weakly coupled MDPs is based on the intuition that if we dualize the linking constraints on the decision vectors, we should obtain I separate and much smaller sub-problems. More specifically, in our model above, let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_J) \geq 0$ be any Lagrange multipliers associated with the J resources, and consider the following modification of Bellman's equations (4) $\forall x \in X$:

$$V^{\lambda}(x) = \max_{u\in U(x)} \left\{ \sum_{i=1}^{I} f_i(x^i, u^i) + \sum_{j=1}^{J} \lambda_j \Big(b^j - \sum_{i=1}^{I} a^{ij}u^i\Big) + \alpha \sum_{n_1=0}^{K^1} \sum_{n_2=0}^{K^2} \cdots \sum_{n_I=0}^{K^I} \Big(\prod_{i=1}^{I} p^i(n_i)\Big) V^{\lambda}(x'^1, x'^2, \ldots, x'^I) \right\}. \tag{6}$$

It turns out that (see Proposition 1 of (Adelman and Mersereau 2008)) for all $x \in X$,

$$V^{\lambda}(x) = \frac{1}{1-\alpha}\sum_{j=1}^{J}\lambda_j b^j + \sum_{i=1}^{I} V_i^{\lambda}(x^i), \tag{7}$$

where

$$V_i^{\lambda}(x^i) = \max_{u^i \in U^i(x^i)} \left\{ f_i(x^i,u^i) - \sum_{j=1}^{J}\lambda_j a^{ij}u^i + \alpha \sum_{n_i=0}^{K^i} p^i(n_i) V_i^{\lambda}(\min\{(x^i-u^i)+n_i, W^i\}) \right\}, \ \forall x^i \in X_i. \tag{8}$$

Moreover, for all $x \in X$, $V^{\lambda}(x) \geq V(x)$, and $F^{\lambda}(\beta) = \sum\limits_{x\in S}\beta(x)V^{\lambda}(x) \geq F(\beta)$ for all $\lambda \geq 0$ (see Proposition 2 of (Adelman and Mersereau 2008)). Thus the tightest Lagrangian bound on the weighted value $F(\beta) = \sum\limits_{x\in S}\beta(x)V(x)$ is obtained by minimizing $F^{\lambda}(\beta)$ over $\lambda \geq 0$. Combining this with I linear programs for computing the componentwise value functions in Bellman's equations (8), we obtain the following Lagrangian linear program:

$$(LLP)\ F^{\lambda^*}(\beta) = \min_{v_i(\cdot),\lambda} \frac{\sum\limits_{j=1}^{J}\lambda_j b^j}{1-\alpha} + \sum_{i=1}^{I}\sum_{x^i\in X_i}\beta_i(x^i)v_i(x^i)$$

$$\text{s.t.}\ \ v_i(x^i) - \alpha\sum_{n_i=0}^{K^i} p^i(n_i)v_i(\min\{(x^i-u^i)+n_i, W^i\}) \geq f_i(x^i,u^i) - \sum_{j=1}^{J}\lambda_j a^{ij}u^i,$$

$$x^i \in X_i, u^i \in U^i(x^i), i = 1,2,\ldots,I;\ \ \lambda \geq 0.$$

Here $\beta_i(x^i) > 0$ are numbers indexed by state components $x^i \in X_i$ such that $\beta_i(x^i) = \sum\limits_{\{x':x'^i=x^i\}}\beta(x')$. That is, $\beta_i(\cdot)$ can be interpreted as the marginal initial state probabilities. The optimal values of $v_i(x^i)$ (these equal $V_i(x^i)$) and the optimal value of vector λ, denoted λ^*, obtained after solving (LLP) define approximate optimal state values $V^{\lambda^*}(x)$ through Equation (7). Also note that $F^{\lambda^*}(\beta) \geq F(\beta)$. Most importantly, (LLP) has only $J + \sum\limits_{i=1}^{I}(W^i+1)$ variables and at most $J + \sum\limits_{i=1}^{I}(W^i+1)W^i$ constraints. Thus, whereas the numbers of variables and constraints are both exponential in I in the exact linear program (LP), they are both linear in I in the approximate Lagrangian linear program (LLP). We next develop a dynamic programming approach to efficiently recover resource allocation decisions from the approximate value function $V^{\lambda^*}(\cdot)$ on the fly.

2.2 Retrieving resource allocation decisions from approximate value functions

When an approximate value function is available, it can in principle be plugged into the right hand side of Bellman's equation (4) (in place of the optimal value function $V(\cdot)$) to find a decision vector $u \in \bar{U}(x)$ that maximizes the right hand side for each $x \in X$. Such a policy is said to be *greedy with respect to the approximate value function*. But this approach is intractable owing to the size of X. In practice however, a whole policy, that is, a decision vector $u \in \bar{U}(x)$ for every $x \in X$, is not needed *a priori* — we need to select a decision vector in a particular state only if and when the system reaches that state during an actual system run. In short, it suffices to retrieve decisions on the fly. Nevertheless, it is often highly non-trivial to retrieve even *one* decision vector that is greedy with respect to an approximate value function because this requires solving a challenging deterministic combinatorial optimization problem. We develop a dynamic programming approach to achieve this.

After solving the Lagrangian linear program (LLP), a vector λ^* and the corresponding $V_i^{\lambda^*}(x^i)$ values for all $x^i \in X_i$ for $i = 1,2,\ldots,I$ are available to us. Now suppose for a *specific* state $x \in S$, we wish to compute a decision vector that is greedy with respect to the value function approximation $V^{\lambda^*}(\cdot)$. This involves solving:

$$\max_{u \in \bar{U}(x)} \left\{ \sum_{i=1}^{I} f_i(x^i,u^i) + \alpha \sum_{n_1=0}^{K^1} \sum_{n_2=0}^{K^2} \cdots \sum_{n_I=0}^{K^I} \Big(\prod_{i=1}^{I} p^i(n_i) \Big) V^{\lambda^*}(x'^1, x'^2, \ldots, x'^I) \right\},$$

where $x'^1, x'^2, \ldots, x'^I$ are defined in Equation (5). On substituting $V^{\lambda^*}(x'^1,x'^2,\ldots,x'^I) = \frac{1}{1-\alpha} \sum_{j=1}^{J} \lambda_j b^j + \sum_{i=1}^{I} V_i^{\lambda^*}(x'^i)$ from Equation (7), the discrete optimization problem algebraically simplifies to

$$\frac{1}{1-\alpha} \sum_{j=1}^{J} \lambda_j b^j + \max_{u \in \bar{U}(x)} \left\{ \sum_{i=1}^{I} \Big[f_i(x^i,u^i) + \alpha \sum_{n_i=0}^{K^i} p^i(n_i) V_i^{\lambda^*}(\min\{(x^i-u^i)+n_i, W^i\}) \Big] \right\}.$$

Because the term outside the maximization (that does not depend on decision u) is irrelevant to this policy retrieval problem, we ignore it in the discussion below, and then rewrite the problem in more detail as follows:

$$\max_{u} \left\{ \sum_{i=1}^{I} \Big[f_i(x^i,u^i) + \alpha \sum_{n_i=0}^{K^i} p^i(n_i) V_i^{\lambda^*}(\min\{(x^i-u^i)+n_i, W^i\}) \Big] \right\} \tag{9}$$

$$u^i \in U^i(x^i) = \{0,1,\ldots,x^i\},\ i = 1,2,\ldots,I;\ \ \sum_{i=1}^{I} a^{ij} u^i \le b^j,\ j = 1,2,\ldots,J.$$

Problem (9) is a generalized multi-dimensional integer knapsack problem. In general, problems of this type are very difficult to solve, however, in our case (9) can often be solved efficiently as an I-stage dynamic program. To see this, let $s_i = (s_i^1, s_i^2, \ldots, s_i^J)$ be the quantities of the J resources remaining to be allocated to job types $i, i+1, \ldots, I$. Let $d_i = \min_{j=1,\ldots,J} \{\lfloor s_i^j / a^{ij} \rfloor\}$ be the maximum number of type i jobs that can be served when the amounts of resources remaining are $(s_i^1, s_i^2, \ldots, s_i^J)$. We define the set $D_i = \{0,1,2,\ldots,d_i\}$. Then problem (9) can be solved using the following dynamic programming recursion:

$$\Psi_i(s_i^1,\ldots,s_i^J) = \max_{u^i \in D_i} \left(\Big[f_i(x^i,u^i) + \alpha \sum_{n_i=0}^{K^i} p^i(n_i) V_i^{\lambda^*}(\min\{(x^i-u^i)+n_i, W^i\}) \Big] + \Psi_{i+1}(s_i^1 - a^{i1}u^i, \ldots, s_i^J - a^{iJ}u^i) \right).$$

Many realistic problems of interest often involve a few resources, say two or three, thus making the above recursion tractable. Thus we now have a method to approximate the optimal value function of our MDP and efficiently retrieve corresponding greedy resource allocation decisions on the fly. We now generalize our MDP model to DRAPs, that is, problems with geometric job durations.

3 Back to geometric job durations

In this case, given that an incomplete type i job is served during a time-period, the probability that it will be completed at the end of that time-period is q^i independently of everything else in the past. Owing to this memoryless property of geometric random variables and our assumption that preemption is allowed, the state of the MDP model is still given by the number of jobs in queue for each type. As before, we now wish to compute $f(x,u)$, the expected profit generated on implementing a feasible decision $u \in \bar{U}(x)$ in a time-period that begins in state x. Let η^i denote the random number of type i jobs completed in this time-period. The probability distribution of η^i is binomial(u^i,q^i). Thus the expected number of type i jobs completed is $u^i q^i$. Consequently, $f(x,u) = \sum_{i=1}^{I} f_i(x^i,u^i)$, where

$$f_i(x^i,u^i) = \left\{ \alpha u^i q^i R^i - H^i(x^i - u^i) - \alpha \sum_{n_i=0}^{K^i} \sum_{\eta^i=0}^{u^i} p^i(n_i) \binom{u^i}{\eta^i} (q^i)^{\eta^i} (1-q^i)^{u^i-\eta^i} G^i(\max\{(x^i-\eta^i)+n_i-W^i,0\}) \right\}.$$

Thus Bellman's equations for all $x \in X$ are now given by

$$V(x) = \max_{u \in \bar{U}(x)} \left\{ \sum_{i=1}^{I} f_i(x^i,u^i) + \alpha \sum_{n_1=0}^{K^1} \cdots \sum_{n_I=0}^{K^I} \left(\prod_{i=1}^{I} p^i(n_i) \binom{u^i}{\eta^i} (q^i)^{\eta^i} (1-q^i)^{u^i-\eta^i} \right) V(x'^1, x'^2, \ldots, x'^I) \right\}, \qquad (10)$$

where $x'^i = \min\{(x^i - \eta^i) + n_i, W^i\}$, for $i = 1,2,\ldots,I$.

Most importantly, this MDP model is also weakly coupled and hence Lagrangian decomposition and decision retrieval by dynamic programming discussed in Section 2 naturally extend to this geometric case through minor changes in notation. We therefore do not repeat that discussion here.

4 Numerical experiments

We demonstrate the power of ideas discussed above through computational experiments on randomly generated problems. Results are presented both for the deterministic job durations case and the geometric case. We consider $I = 6,8,10,20,30,40,50$. The maximum number of job arrivals K^i in a time-period is set to a uniform random integer in $[1,5]$ for all i. Once such a K^i is sampled, the probability mass function $p_i(\cdot)$ is obtained by normalizing K^i+1 uniform $(0,1)$ random variables. The resource availabilities and consumption values were generated using a technique similar to a standard method for generating multi-dimensional knapsack problems (Chu and Beasely [1998]). Particularly, the unit resource consumption values for job type i are set to uniform random integers in $[i,3i]$. The resource availability is then obtained by adding unit resource consumption values across all job types and then multiplying this quantity by a tightness ratio. Tightness ratios of 0.7 and 0.9 were used. Queue capacities W^i were set to either 3 or 6 for all i thus our experiments include problems with state-space as large as 7^{50}. The discount factor was set to 0.8. In general, our experience suggests that interesting economic tradeoffs are generated when the ratios R^i/H^i and R^i/G^i are type-dependent. The completion reward per job of type i was set to a uniform random integer in $[50i, 50(i+1)]$. The holding cost as well as rejection cost per job of type i was set to a uniform random integer between $[15(i-1)+5, 15(i-1)+10]$. For geometric job durations, the success probabilities q^i are chosen from the interval $[0.5,1]$ and are decreasing in i; specifically, q^i is set to equal a uniform $(0.5+0.5(I-i)/I, 0.5+0.5(I+1-i)/I)$ random variable.

We compared the performance of decisions retrieved using the Lagrangian approach with that of two heuristic policies common in the literature . The first is a myopic heuristic that in state x simply chooses a feasible decision $u \in \bar{U}(x)$ that maximizes $f(x,u)$. Note however that this problem itself is computationally quite difficult in general and we solve it using a dynamic program similar to that in Section 2.2. The second is a μc-type of heuristic that prioritizes jobs based on ratios $q^i(R^i+H^i+G^i)/(\sum_{j=1}^{J} a^{ij})$ — the higher this ratio the higher the priority. These ratios relate to the reward per resource consumed per expected service duration. Linear programs were solved using CPLEX 11.0 via AMPL on a PC running Windows Vista with 4GB RAM. All performance estimates were obtained by averaging total discounted profit over $N = 20$ sample path repetitions of $T = 50$ time-steps each (0.8^{50} is small enough to ignore the effect of tail profits) starting with states $(x^1,x^2,\ldots,x^I)$ whose components $i \in \mathscr{I}$ are chosen uniformly at random from $X_i = \{0,1,\ldots,W^i\}$. The results are reported in Tables 1 and 2 below, which list the performance improvement obtained by the Lagrangian approach over the two heuristics. Table 1 focuses on problems with single-period job durations, whereas Table 2 on geometric job durations. These results show that Lagrangian policy performs far better than the two heuristics. The average improvement over the myopic heuristic for problems with single-period job durations is about 40% with tightness ratio 0.7 and about 35% with tightness ratio 0.9. These numbers equal 20% and 43% for problems with geometric job durations. The average improvement over the μc heuristic for problems with single-period job durations is about 220% with tightness ratio 0.7 and about 67% with tightness ratio 0.9. These numbers equal 34% and 66% for problems with geometric job durations.

5 FUTURE WORK

Our future work will focus on extensions of DRAPs to problems with general stochastic job durations. The states in the corresponding MDPs will include information on the total time spent servicing each job in queue in the past and the decisions will need to be modified accordingly. Fortunately these models will continue to be weakly coupled.

ACKNOWLEDGMENTS

Yasin Gocgun's work was partly funded by the Department of Radiology at the University of Washington School of Medicine. Archis Ghate appreciates summer support from the University of Washington.

REFERENCES

Adelman, D., and A. J. Mersereau. 2008. Relaxations of weakly coupled stochastic dynamic programs. *Operations Research* 56 (3): 712–727.

Altman, E. 2002. Applications of markov decision processes in communication networks. In *Handbook of Markov Decision Processes: Methods and Applications*, ed. E. Feinberg and A. Shwartz, Chapter 16, 489–536. Springer.

Bertsekas, D. P. 2007. *Dynamic programming and optimal control*. third ed, Volume 1 and 2. Nashua, NH: Athena Scientific.

Bertsimas, D., and A. J. Mersereau. 2007. A learning approach for interactive marketing to a customer segment. *Operations Research* 55 (6): 1120–1135.

Cox, D. R., and W. L. Smith. 1961. *Queues*. London, UK: Chapman and Hall.

Gittins, J. C. 1979. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society, Series B* 41 (2): 148–177.

Godfrey, G. A., and W. B. Powell. 2002a. An adaptive dynamic programming algorithm for dynamic fleet management, i: Single period travel times. *Transportation Science* 36 (1): 21–39.

Godfrey, G. A., and W. B. Powell. 2002b. An adaptive dynamic programming algorithm for dynamic fleet management, ii: Multiperiod travel times. *Transportation Science* 36 (1): 40–54.

Green, L. V., S. Savin, and B. Wang. 2006. Managing patient service in a diagnostic medical facility. *Operations Research* 54 (1): 11–25.

Harrison, M. J. 1975. Dynamic scheduling of a multiclass queue: Discount optimality. *Operations Research* 23 (2): 270–282.

Hawkins, J. 2003. *A lagrangian decomposition approach to weakly coupled dynamic optimization problems and its applications*. Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA.

Kleywegt, A. J., and J. D. Papastavrou. 1998. The dynamic and stochastic knapsack problem. *Operations Research* 46 (1): 17–35.

Luh, P. B., X. Miao, S. Chang, and D. A. Castanon. 1989. Stochastic task selection and renewable resource allocation. *IEEE Transactions on Automatic Control* 34 (3): 335–338.

Megow, N., M. Uetz, and T. Vredeveld. 2006. Models and algorithms for stochastic online scheduling. *Mathematics of Operations Research* 31 (3): 513–525.

Nunes, L. G. N., S. V. de Carvalho, and R. de Cássia Meneses Rodrigues. 2009. Markov decision process applied to the control of hospital elective admissions. *Artificial Intelligence in Medicine* 47 (2): 159 – 171.

Perry, T. C., and J. C. Hartman. 2009. An approximate dynamic programming approach to solving dynamic, stochastic multiple knapsack problem. *International Transactions in Operational Research* 16:347–359.

Pinedo, M. L. 2008. *Scheduling theory, algorithms and systems*. third ed. New York, USA: Springer.

Puterman, M. 1994. *Markov decision processes*. New Jersey: John Wiley and Sons.

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

Ross, S. M. 1983. *Introduction to stochastic dynamic programming*. New York: Academic Press.

Schwindt, C. 2005. *Resource allocation in project management*. Berlin, Germany: Springer.

Sonin, I. M. 2008. A generalized gittins index for a markov chain and its recursive calculation. *Statistics and Probability Letters* 78 (12): 1526 – 1533.

Tolio, T. 2009. *Design of flexible production systems – methodologies and tools*. Berlin, Germany: Springer.

Topaloglu, H. 2009. Using lagrangian relaxation to compute capacity-dependent bid prices in network revenue management. *Operations Research* 57 (3): 637–649.

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

Vengerov, D. 2007. A reinforcement learning approach to dynamic resource allocation. *Engineering Applications of Artificial Intelligence* 20 (3): 383–390.

AUTHOR BIOGRAPHIES

YASIN GOCGUN is a Ph.D. candidate in the Department of Industrial and Systems Engineering at the University of Washington, Seattle. His research interests focus on combinatorial and dynamic optimization with applications to healthcare and manufacturing. His email address is <gocgun@u.washington.edu>.

ARCHIS GHATE is an Assistant Professor in the in the Department of Industrial and Systems Engineering at the University of Washington, Seattle. His research interests focus on theory and algorithms for dynamic optimization under uncertainty with applications to medical treatment planning and service operations management. His email address is <archis@u.washington.edu>.

Table 1: Comparison between the performance of Lagrangian and heuristic decisions for problems with single-period job durations. The percentages listed in the last two columns equal 100×(Lagrangian- heuristic)/(|heuristic|), where "Lagrangian" represents the average discounted profit obtained by Lagrangian decisions over $N = 20$ independent sample path replications of $T = 50$ time-steps each, and "heuristic" represents the same for myopic and μc. Problems 1-25 use tightness = 0.7, 26-50 use tightness = 0.9.

#	I	J	W^i $\forall i$	% improvement over myopic	% improvement over μc
1	6	1	3	4.32	19.24
2	8	1	3	41.04	117.01
3	10	1	3	7.33	10.11
4	6	2	3	37.64	57.68
5	8	2	3	19.36	145.10
6	10	2	3	15.87	4438.89
7	6	1	6	21.89	53.40
8	8	1	6	21.45	14.71
9	10	1	6	29.20	27.73
10	6	2	6	50.88	54.73
11	8	2	6	17.85	14.44
12	10	2	6	11.20	21.39
13	6	3	3	10.48	38.52
14	8	3	3	31.77	105.33
15	10	3	3	246.22	143.75
16	6	3	6	51.66	61.08
17	8	3	6	23.99	24.21
18	20	1	3	6.22	3.45
19	30	1	3	5.34	4.29
20	40	1	3	16.44	7.02
21	50	1	3	13.88	6.07
22	20	1	6	184.78	71.57
23	30	1	6	72.47	44.96
24	40	1	6	26.06	14.63
25	50	1	6	42.91	17.20
26	6	1	3	3.92	14.61
27	8	1	3	5.88	8.65
28	10	1	3	1.81	3.62
29	6	2	3	5.80	13.91
30	8	2	3	3.12	17.55
31	10	2	3	1.11	30.32
32	6	1	6	14.31	28.43
33	8	1	6	139.18	523.38
34	10	1	6	15.66	13.03
35	6	2	6	126.94	109.42
36	8	2	6	70.59	62.89
37	10	2	6	61.91	75.46
38	6	3	3	10.01	39.59
39	8	3	3	4.42	14.31
40	10	3	3	9.94	32.12
41	6	3	6	81.94	525.99
42	8	3	6	150.35	131.09
43	20	1	3	0.76	0.61
44	30	1	3	4.70	3.34
45	40	1	3	9.65	3.58
46	50	1	3	6.13	2.50
47	20	1	6	8.64	4.79
48	30	1	6	20.44	8.51
49	40	1	6	94.87	21.54
50	50	1	6	38.87	9.33

Table 2: Comparison between the performance of Lagrangian and heuristic decisions for problems with geometric job durations. The percentages listed in the last two columns equal 100×(Lagrangian- heuristic)/(|heuristic|), where "Lagrangian" represents the average discounted profit obtained by Lagrangian decisions over $N = 20$ independent sample path replications of $T = 50$ time-steps each, and "heuristic" represents the same for myopic and μc. Problems 1-8 use tightness = 0.7, 9-16 use tightness = 0.9.

#	I	J	$W^i \; \forall i$	% improvement over myopic	% improvement over μc
1	6	1	3	-7.74	39.38
2	8	1	3	32.77	31.66
3	10	1	3	72.37	86.03
4	6	1	6	26.96	63.87
5	8	1	6	-0.65	3.67
6	10	1	6	-0.97	1.64
7	20	1	3	14.51	32.54
8	20	1	6	27.04	19.66
9	6	1	3	21.37	34.22
10	8	1	3	-6.65	191.88
11	10	1	3	51.27	64.15
12	6	1	6	-36.66	29.99
13	8	1	6	22.23	21.00
14	10	1	6	7.69	13.02
15	20	1	3	13.44	17.99
16	20	1	6	277.83	162.64