Approximate Dynamic Programming for Dynamic Capacity Allocation with Multiple Priority Levels

Alexander Erdelyi
School of Operations Research and Information Engineering,
Cornell University, Ithaca, NY 14853, USA
alex@orie.cornell.edu

Huseyin Topaloglu
School of Operations Research and Information Engineering,
Cornell University, Ithaca, NY 14853, USA
topaloglu@orie.cornell.edu

May 21, 2010

Abstract

In this paper, we consider a quite general dynamic capacity allocation problem. There is a fixed amount of daily processing capacity. On each day, jobs of different priorities arrive randomly and we need to decide which jobs should be scheduled for which days. Waiting jobs incur a holding cost depending on their priority levels. The objective is to minimize the total expected cost over a finite planning horizon. We begin by formulating the problem as a dynamic program, but this formulation is computationally difficult as it involves a high dimensional state vector. To address this difficulty, we use an approximate dynamic programming approach that decomposes the dynamic programming formulation by the different days in the planning horizon to construct separable approximations to the value functions. We use the value function approximations for two purposes. First, we show that the value function approximations can be used to obtain a lower bound on the optimal total expected cost. Second, the value function approximations can be used to make the job scheduling decisions over time. Computational experiments indicate that the job scheduling decisions made by our approach perform significantly better than a variety of benchmark strategies.

Many companies face the problem of allocating limited processing capacity to jobs that arrive randomly over time. This is a common problem in production environments, where a company utilizes a certain production resource to serve the demands from customers with different profit margins. Similarly, clinics and owners of medical diagnostics resources need to allocate appointment slots to patients with different needs and urgency levels. The essential tradeoff in these problems is between using the available processing capacity for a low priority job that is currently available and delaying the processing of the low priority job with the hope that a high priority job can use the available capacity later. This tradeoff is further complicated by the fact that the processing capacity is perishable. If the processing capacity that is available on a day is not used by that day, then it ends up being wasted.

In this paper, we consider a generic capacity allocation problem that captures the fundamental tradeoffs mentioned above. We consider a setting where we have a fixed amount of daily processing capacity. Jobs of different priorities arrive randomly over time and we need to decide which jobs should be scheduled for which days. The jobs that are waiting to be processed incur a holding cost depending on their priority levels and we are interested in finding a job scheduling policy that minimizes the total expected cost over a finite planning horizon. It is possible to formulate such a capacity allocation problem as a dynamic program, but this dynamic programming formulation ends up being computationally difficult to solve as it involves a high dimensional state vector. In this paper, we propose an approximate dynamic programming strategy that is based on decomposing the dynamic programming formulation by the different days in the planning horizon.

Our dynamic programming decomposition approach starts with a deterministic linear programming formulation for the capacity allocation problem. This deterministic linear program is constructed under the assumption that the job arrivals take on their expected values and we can schedule fractional numbers of jobs. In the linear program, there is one capacity constraint for each day in the planning horizon, ensuring that the jobs scheduled on a day cannot exceed the daily available capacity. We use Lagrangian relaxation to relax these capacity constraints for every day in the planning horizon except for one particular day, so that we obtain a deterministic linear program for a capacity allocation problem with limited capacity only on one day. This latter deterministic linear program, in turn, indicates how we can decompose the original dynamic programming formulation of the problem by the different days in the planning horizon. Ultimately, our dynamic programming decomposition strategy provides separable approximations to the value functions and these value function approximations can be used to make the job scheduling decisions.

There are several interesting properties of our dynamic programming decomposition method. To begin with, it is possible to show that our dynamic programming decomposition method provides a lower bound on the optimal total expected cost. Such a lower bound becomes useful when assessing the optimality gap of a suboptimal control policy. It turns out that the deterministic linear programming formulation mentioned above also provides such a lower bound, but the lower bound obtained by our dynamic programming decomposition method is provably tighter than the one obtained by the deterministic linear program. Furthermore, the job scheduling decisions made by our approach can be computed by solving a min cost network flow problem. This is a significant result since it allows us to obtain integer solutions without imposing integrality requirements on the decision variables. Finally, our computational experiments indicate that the job scheduling decisions that are made by our dynamic programming decomposition method can provide significant improvements over numerous benchmarks based on the deterministic linear programming formulation.

Two papers in the literature are particularly relevant to our work. First, Patrick et al. (2008) consider the dynamic allocation of appointment slots in a clinic and their problem is quite similar to ours. They provide a dynamic programming formulation, but motivated by the high dimensionality of the state vector, they use linear approximations to the value functions. Linear value function approximations imply that the marginal value of an additional capacity does not depend on how much capacity is left on a particular day. However, it is intuitive to expect that if we have fewer units of capacity left on a particular day, then the marginal value of an additional capacity should increase. The value function approximations generated by our dynamic programming decomposition method are nonlinear and convex, indeed capturing the intuitive expectation that the marginal value of an additional capacity increases as we have fewer units of capacity left on a particular day. Second, Erdelyi and Topaloglu (2009*a*) use protection level policies for capacity allocation problems. A protection level policy is similar to a prioritized first come first serve policy. As the jobs arrive over time, it commits the capacity starting from the highest priority jobs, but the protection levels dictate how much capacity should be left untouched for the future job arrivals. The authors visualize the total expected cost over the planning horizon as a function of the protection levels and use sampled gradients of the total expected cost to search for a good set of protection levels. The implicit assumption behind using a set of protection levels is that the job arrivals are stationary so that using the same protection levels every day is expected to perform reasonably well. In contrast, our dynamic programming decomposition approach does not assume or require stationary job arrivals.

Dynamic programming decomposition ideas first appeared in the network revenue management literature. In network revenue management problems, an airline operates a set of flight legs to serve the requests for itineraries that arrive randomly over time. Whenever an itinerary request arrives, the airline has to decide whether to accept or reject the itinerary request. The fundamental tradeoff is between accepting an inexpensive itinerary request right now and reserving the capacity for an expensive itinerary request that may arrive in the future. It is possible to formulate network revenue management problems as dynamic programs, but this approach requires keeping track of the remaining capacity on all of the flight legs and the resulting dynamic programs end up having high dimensional state vectors. Therefore, it is customary to try to decompose the problem by the flight legs. Williamson (1992) is one of the first to decompose the network revenue management problem by the flight legs to generalize the expected marginal seat revenue heuristic of Belobaba (1987). A discussion of early decomposition approaches is given in Talluri and van Ryzin (2005). Liu and van Ryzin (2008) and Kunnumkal and Topaloglu (2009) use decomposition ideas while incorporating customer choice behavior into network revenue management problems. Zhang and Adelman (2009) are the first to establish that one can use dynamic programming decomposition methods to obtain upper bounds on the optimal total expected revenue. Topaloglu (2009) shows that decomposition methods can be visualized as an application of Lagrangian relaxation to the dynamic programming formulation of the network revenue management problem. Erdelyi and Topaloglu

(2009*b*) use decomposition ideas to make overbooking decisions in network revenue management. It is worthwhile to note that our capacity allocation problem can be visualized as a revenue management problem as it involves making the most use out of limited and perishable capacity. However, traditional revenue management problems involve only the decision of whether to accept or reject a job, whereas our capacity allocation problem involves the decision of when to schedule a job, possibly along with whether to accept or reject a job.

Zhang (2010) develops a dynamic programming decomposition method that improves the upper bounds on the optimal total expected revenue provided by the methods in Liu and van Ryzin (2008), Kunnumkal and Topaloglu (2009) and Zhang and Adelman (2009). The development in Zhang (2010) is for the network revenue management setting and it does not immediately extend to our capacity allocation setting for two reasons. First, it is common in the network revenue management literature to assume that there is at most one customer arrival at each time period. Therefore, the random variable that is of interest at each time period corresponds to the itinerary that the arriving customer purchases. Such a random variable has a reasonably small number of possible realizations and one can compute expectations simply by enumerating over all of its possible realizations. In our capacity allocation setting, the random variable that is of interest on each day corresponds to the numbers of job arrivals at different priority levels. The number of possible realizations for such a random variable grows exponentially with the number of priority levels. For example, if we have four priority levels and the number of job arrivals at each priority level can take 100 different values, then number of possible realizations of job arrivals is on the order of one hundred million. Second, the approach proposed by Zhang (2010) provides nonseparable approximations to the value functions. Nonseparable value function approximations do not pose a difficulty in the network revenue management setting when finding a set of itineraries to make available for purchase. In contrast, our capacity allocation setting requires jointly deciding which arriving jobs with different priority levels should be scheduled for which days and the separable structure of the value function approximations that we obtain plays a crucial role in making the job scheduling decisions in a tractable fashion.

The idea of decomposing large scale dynamic programs by using duality arguments has a history in the literature. The links between dynamic programs and duality are studied in the book chapter by Ruszczynski (2003). Adelman and Mersereau (2008) use the term weakly coupled to refer to dynamic programs that would decompose if a few constraints did not link the different components of the state vector. They develop a duality framework for such dynamic programs. Under the broad umbrella of approximate dynamic programming, there is rising interest in approximate solutions of large scale dynamic programs. The books by Bertsekas and Tsitsiklis (1996) and Powell (2007) provide excellent coverage of this work. Although dynamic programming decomposition ideas are not covered in these books, our work in this paper and the success of these ideas in the network revenue management literature indicate their capability in dealing with large scale dynamic programs.

With rising interest in health care applications, there are a several dynamic capacity allocation problems that originate within this problem domain. The paper by Patrick et al. (2008) that we describe above is one such work. Gerchak et al. (1996) focus on a setting where limited operating room capacity has to be allocated between emergency and elective surgery. They characterize the structure of the optimal policy and show that protection level policies may not be optimal. Gupta and Wang (2008) present a model that considers customer choice behavior when the customers choose appointment slots. Similar to Gerchak et al. (1996), these authors also indicate that the form of the optimal policy may have complicated structure and they propose approximation strategies.

In this paper, we make the following research contributions. 1) We develop a method to make job scheduling decisions in a capacity allocation model with jobs of different priorities. Our approach decomposes the problem by the different days in the planning horizon and solves a sequence of dynamic programs with scalar state variables. 2) We show that our approach provides a lower bound on the optimal total expected cost and this lower bound is at least as tight as the one provided by a deterministic linear programming formulation. 3) Computational experiments demonstrate that our method performs significantly better than benchmark strategies based on linear programming formulations.

The rest of the paper is organized as follows. In Section 1, we formulate the capacity allocation problem as a dynamic program. This formulation is computationally difficult due to the size of the state vector. In Section 2, we give a deterministic linear program, which serves as a starting point for our decomposition strategy. In Section 3, we decompose the dynamic programming formulation of the problem by building on the deterministic linear program. In Section 4, we demonstrate how to overcome some computational hurdles in our dynamic programming decomposition method. In Section 5, we dwell on how we can apply the policy obtained by our dynamic programming decomposition method. In Section 6, we provide computational experiments. In Section 7, we conclude.

1 Problem Formulation

We have a fixed amount of daily processing capacity. At the beginning of each day, we observe the arrivals of jobs of different priorities and we need to decide which jobs should be scheduled for which days. The jobs that are waiting to be processed incur a holding cost depending on their priority levels, but we also have the option of rejecting a job immediately by incurring a penalty cost. The objective is to minimize the total expected cost over a finite planning horizon.

The problem takes place over the set of days $\mathcal{T} = \{1, \dots, \tau\}$. The set of possible priority levels for the jobs is $\mathcal{P}$. The number of priority p jobs that arrive on day t is given by the random variable D_t^p so that $D_t = \{D_t^p : p \in \mathcal{P}\}$ captures the job arrivals on day t. We assume that the job arrivals on different days are independent. While such an assumption is reasonable in many situations, there can be some settings where it would be problematic to assume that the job arrivals on different days are independent. For example, if the jobs that are rejected come back to the system, then there can be correlations among the job arrivals on different days. Even in such cases, if there are few rejected jobs that come back to the system when compared with the fresh job arrivals or if the rejected jobs get service elsewhere, then it can be reasonable to assume that the job arrivals on different days are independent. Furthermore, the quality of service that we provide on earlier days may influence the job arrivals in the future, creating correlations among the job arrivals on different days, but such an involved stochastic process for the job arrivals is not within the scope of our model. If we schedule a priority p job that arrives on day t for day j, then we incur a holding cost of h^p_{jt}. The penalty cost of rejecting a priority p job that arrives on day t is r^p_t. We use the convention that $h^p_{jt} = \infty$ whenever it is infeasible to schedule a priority p job arriving on day t for day j. For example, since it is infeasible to schedule a job for a day in the past, we naturally have $h^p_{jt} = \infty$ whenever $j < t$. Similarly, if it is not possible to schedule a job more than S days into the future, then we can capture this constraint by letting $h^p_{jt} = \infty$ whenever $j - t \geq S$. Once we schedule a job for a particular day, this decision is fixed and cannot be changed. For notational brevity, we assume that all jobs consume one unit of processing capacity, but it is possible to extend our approach to multiple units of capacity consumption.

Given that we are on day t, we let x_{jt} be the remaining capacity on day j so that $x_t = \{x_{jt} : j \in \mathcal{T}\}$ captures the state of the remaining capacities as observed at the beginning of day t. Naturally, given that we are on day t, we do not need to keep track of the remaining capacities on the days in the past and we can use $\tilde{x}_t = \{x_{jt} : j = t, t+1, \ldots, \tau\}$ as our state variable. However, we prefer to use $x_t = \{x_{jt} : j \in \mathcal{T}\}$ for notational uniformity so that the state variable on different days ends up having the same number of dimensions. We let u^p_{jt} be the number of priority p jobs that we schedule for day j on day t so that $u_t = \{u^p_{jt} : p \in \mathcal{P},\ j \in \mathcal{T}\}$ captures the decisions that we make on day t. In this case, the set of feasible decisions on day t is given by

$$\mathcal{U}(x_t, D_t) = \Big\{ u_t \in \mathbb{Z}_+^{|\mathcal{P}||\mathcal{T}|} : \sum_{p \in \mathcal{P}} u^p_{jt} \leq x_{jt} \quad j \in \mathcal{T}, \quad \sum_{j \in \mathcal{T}} u^p_{jt} \leq D^p_t \quad p \in \mathcal{P} \Big\}, \tag{1}$$

where the first set of constraints ensure that the decisions that we make do not violate the remaining capacities and the second set of constraints ensure that the decisions that we make do not violate the job arrivals. On the other hand, the total cost that we incur on day t is given by

$$\sum_{j \in \mathcal{T}} \sum_{p \in \mathcal{P}} h^p_{jt}\, u^p_{jt} + \sum_{p \in \mathcal{P}} r^p_t \Big[D^p_t - \sum_{j \in \mathcal{T}} u^p_{jt} \Big],$$

where the first term corresponds to the holding cost for the jobs that we schedule for different days in the planning horizon and the second term corresponds to the penalty cost for the jobs that we reject. We note that without loss of generality, it is possible to assume that $r^p_t = 0$ for all $p \in \mathcal{P},\ t \in \mathcal{T}$. To see this, we write the cost function above as $\sum_{j \in \mathcal{T}} \sum_{p \in \mathcal{P}} \big[h^p_{jt} - r^p_t\big] u^p_{jt} + \sum_{p \in \mathcal{P}} r^p_t\, D^p_t$. Since $\sum_{p \in \mathcal{P}} r^p_t\, D^p_t$ is a constant that is independent of the decisions, assuming that $r^p_t = 0$ for all $p \in \mathcal{P}$, $t \in \mathcal{T}$ is equivalent to letting $c^p_{jt} = h^p_{jt} - r^p_t$ and working with the holding cost c^p_{jt} instead of h^p_{jt}. By working with the holding cost $c^p_{jt} = h^p_{jt} - r^p_t$, we a priori assume that all of the job arrivals on a day are rejected and immediately charge the penalty cost $\sum_{p \in \mathcal{P}} r^p_t\, D^p_t$, but for each priority p job that we schedule for day j, we reimburse the penalty cost r^p_t back and charge the holding cost h^p_{jt} instead. Throughout the paper, we indeed assume that $r^p_t = 0$ for all $p \in \mathcal{P},\ t \in \mathcal{T}$ and work with the holding costs $\{c^p_{jt} : p \in \mathcal{P},\ j, t \in \mathcal{T}\}$.

We use $V_t(x_t)$ to denote the minimum possible total expected cost over days $\{t, \ldots, \tau\}$ given that the state of the remaining capacities on day t is x_t. Letting e_j be the $|\mathcal{T}|$ dimensional unit vector with a one in the element corresponding to $j \in \mathcal{T}$, we can compute the value functions $\{V_t(\cdot) : t \in \mathcal{T}\}$ by solving the optimality equation

$$V_t(x_t) = \mathbb{E}\Big\{ \min_{u_t \in \mathcal{U}(x_t, D_t)} \Big\{ \sum_{j \in \mathcal{T}} \sum_{p \in \mathcal{P}} c^p_{jt}\, u^p_{jt} + V_{t+1}(x_t - \sum_{j \in \mathcal{T}} \sum_{p \in \mathcal{P}} u^p_{jt}\, e_j) \Big\} \Big\} \tag{2}$$

with the boundary condition that $V_{\tau+1}(\cdot) = 0$. The optimal total expected cost over the whole planning horizon is $V_1(x_1)$, where x_1 captures the initial state of the capacities and it is a part of the problem data. The knowledge of $\{V_t(\cdot) : t \in \mathcal{T}\}$ can be used to find the optimal decisions on each day. In particular, if the state of the remaining capacities and the job arrivals on day t are respectively given by x_t and D_t, then we can solve the optimization problem inside the expectation in (2) to find the optimal decisions on this day.

Unfortunately, the number of dimensions of the state vector x_t is equal to the number of days, which can easily be on the order of hundreds for practical applications. Furthermore, solving the optimality equation in (2) requires taking an expectation over the job arrivals on each day and this expectation can be difficult to compute. These considerations render finding the exact solution to the optimality equation in (2) intractable. In the next two sections, we give two possible approaches for finding approximate solutions to the optimality equation in (2). The first approach involves solving a deterministic linear program, whereas the second approach builds on this deterministic linear program to construct separable approximations to the value functions.

2 Deterministic Linear Programming Formulation

One approach for finding approximate solutions to the optimality equation in (2) involves solving a deterministic linear program that is formulated under the assumption that the job arrivals take on their expected values and it is possible to schedule fractional numbers of jobs for different days. In particular, using the decision variables $\{u^p_{jt} : p \in \mathcal{P},\ j \in \mathcal{T},\ t \in \mathcal{T}\}$ with the same interpretation as in the previous section, we can solve the problem

$$\min \quad \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{T}} \sum_{p\in\mathcal{P}} c^p_{jt}\, u^p_{jt} \tag{3}$$

$$\text{subject to} \quad \sum_{t\in\mathcal{T}} \sum_{p\in\mathcal{P}} u^p_{jt} \le x_{j1} \qquad j \in \mathcal{T} \tag{4}$$

$$\sum_{j\in\mathcal{T}} u^p_{jt} \le \mathbb{E}\{D^p_t\} \qquad p \in \mathcal{P},\ t \in \mathcal{T} \tag{5}$$

$$u^p_{jt} \ge 0 \qquad p \in \mathcal{P},\ j \in \mathcal{T},\ t \in \mathcal{T} \tag{6}$$

to approximate the optimal total expected cost over the planning horizon. Constraints (4) in the problem above ensure that the decisions that we make over the planning horizon do not violate the remaining capacity on each day, whereas constraints (5) ensure that the decisions that we make on each day do not violate the expected numbers of job arrivals.

One important use of problem (3)-(6) occurs when we want to obtain lower bounds on the optimal total expected cost. In particular, letting z^*_{LP} be the optimal objective value of problem (3)-(6), the next proposition shows that we have $z^*_{LP} \le V_1(x_1)$ so that the optimal objective value of problem (3)-(6) provides a lower bound on the optimal total expected cost. Such lower bounds become useful when we want to gain insight into the optimality gap of any suboptimal control policy.

Proposition 1 *We have $z^*_{LP} \leq V_1(x_1)$.*

Proof Our proof is standard, but it allows us to demonstrate a useful inequality. Given that the job arrivals over the planning horizon are $D = \{D^p_t : p \in \mathcal{P},\ t \in \mathcal{T}\}$, we let $\pi^*(D)$ be the total cost incurred by the optimal policy and $z^*_{LP}(D)$ be the optimal objective value of problem (3)-(6) that is obtained after replacing the right side of constraints (5) with $\{D^p_t : p \in \mathcal{P},\ t \in \mathcal{T}\}$. Replacing the right side of constraints (5) with $\{D^p_t : p \in \mathcal{P},\ t \in \mathcal{T}\}$ and solving problem (3)-(6) corresponds to making the decisions after observing all of the job arrivals over the whole planning horizon. Furthermore, problem (3)-(6) allows scheduling fractional numbers of jobs. On the other hand, the optimal policy makes the decisions for a particular day after observing the job arrivals only on that day. Furthermore, the optimal policy schedules integer numbers of jobs. Therefore, it holds that $z^*_{LP}(D) \leq \pi^*(D)$. Taking expectations and noting that $z^*_{LP}(D)$ is a convex function of D, Jensen's inequality implies that

$$\mathbb{E}\{\pi^*(D)\} \geq \mathbb{E}\{z^*_{LP}(D)\} \geq z^*_{LP}(\mathbb{E}\{D\}) = z^*_{LP}, \tag{7}$$

where the equality follows from the definition of z^*_{LP}. The result follows from the fact that $\mathbb{E}\{\pi^*(D)\}$ gives the total expected cost incurred by the optimal policy and is equal to $V_1(x_1)$. □

Therefore, (7) implies that both z^*_{LP} and $\mathbb{E}\{z^*_{LP}(D)\}$ provide lower bounds on the optimal total expected cost. The lower bound provided by z^*_{LP} is looser, but this lower bound can be computed simply by solving problem (3)-(6). On the other hand, the lower bound provided by $\mathbb{E}\{z^*_{LP}(D)\}$ is tighter, but there is no closed form expression for the expectation in $\mathbb{E}\{z^*_{LP}(D)\}$ and computing this expectation requires estimation through Monte Carlo samples. Nevertheless, our computational experiments indicate that this extra computational burden usually pays off and the lower bound provided by $\mathbb{E}\{z^*_{LP}(D)\}$ can be significantly tighter than the lower bound provided by z^*_{LP}.

Another important use of problem (3)-(6) occurs when we want to construct an approximate policy to make the decisions on each day. In particular, letting $\{\mu^*_j : j \in \mathcal{T}\}$ be the optimal values of the dual variables associated with constraints (4), we can use μ^*_j to estimate the value of a unit of remaining capacity on day j. This idea suggests using the linear function $\sum_{j\in\mathcal{T}} \mu^*_j\, x_{jt}$ as an approximation to the value function $V_t(x_t)$. In this case, if the state of the remaining capacities and the job arrivals on day t are respectively given by x_t and D_t, then we can replace $V_{t+1}(x_t - \sum_{j\in\mathcal{T}} \sum_{p\in\mathcal{P}} u^p_{jt}\, e_j)$ on the right side of (2) with $\sum_{j\in\mathcal{T}} \mu^*_j \big[x_{jt} - \sum_{p\in\mathcal{P}} u^p_{jt}\big]$ and solve the problem

$$\min_{u_t \in \mathcal{U}(x_t, D_t)} \left\{ \sum_{j\in\mathcal{T}} \sum_{p\in\mathcal{P}} c^p_{jt}\, u^p_{jt} + \sum_{j\in\mathcal{T}} \mu^*_j \Big[x_{jt} - \sum_{p\in\mathcal{P}} u^p_{jt}\Big] \right\} \tag{8}$$

to make the decisions on day t. We refer to this decision rule as the LVF decision rule, standing for linear value functions. We use the LVF decision rule as a benchmark strategy later in our computational experiments. An immediate shortcoming of the LVF decision rule is that it constructs the value function approximations by only using expected values of the job arrivals. In the next section, we build on problem (3)-(6) to construct a value function approximation strategy that addresses the random nature of the job arrivals more carefully.

3 Dynamic Programming Decomposition

In this section, we develop a method that constructs separable approximations to the value functions by decomposing the optimality equation in (2) into a sequence of optimality equations with scalar state variables. We begin with a duality argument on problem (3)-(6). We let $\{\mu_j^* : j \in \mathcal{T}\}$ be the optimal values of the dual variables associated with constraints (4) in problem (3)-(6). We choose an arbitrary day i in the planning horizon and relax constraints (4) for all of the other days by associating the dual multipliers $\{\mu_j^* : j \in \mathcal{T} \setminus \{i\}\}$ with them. In this case, letting $\mathbf{1}(\cdot)$ be the indicator function, the objective function of problem (3)-(6) can be written as

$$\sum_{t\in\mathcal{T}}\sum_{j\in\mathcal{T}}\sum_{p\in\mathcal{P}} c_{jt}^p\, u_{jt}^p + \sum_{j\in\mathcal{T}\setminus\{i\}} \mu_j^* \Big[x_{j1} - \sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}} u_{jt}^p\Big] \\ = \sum_{t\in\mathcal{T}}\sum_{j\in\mathcal{T}}\sum_{p\in\mathcal{P}} \big[c_{jt}^p - \mathbf{1}(j \neq i)\,\mu_j^*\big] u_{jt}^p + \sum_{j\in\mathcal{T}\setminus\{i\}} \mu_j^*\, x_{j1}.$$

Therefore, by linear programming duality, the optimal objective value of problem (3)-(6) is the same as the optimal objective value of the problem

$$z_{LP}^* = \min \quad \sum_{t\in\mathcal{T}}\sum_{j\in\mathcal{T}}\sum_{p\in\mathcal{P}} \big[c_{jt}^p - \mathbf{1}(j \neq i)\,\mu_j^*\big] u_{jt}^p + \sum_{j\in\mathcal{T}\setminus\{i\}} \mu_j^*\, x_{j1} \tag{9}$$

$$\text{subject to} \quad \sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}} u_{it}^p \le x_{i1} \tag{10}$$

$$\sum_{j\in\mathcal{T}} u_{jt}^p \le \mathbb{E}\{D_t^p\} \qquad p \in \mathcal{P},\ t \in \mathcal{T} \tag{11}$$

$$u_{jt}^p \ge 0 \qquad p \in \mathcal{P},\ j \in \mathcal{T},\ t \in \mathcal{T}. \tag{12}$$

Comparing problem (9)-(12) with problem (3)-(6) and ignoring the constant term $\sum_{j\in\mathcal{T}\setminus\{i\}} \mu_j^*\, x_{j1}$ in the objective function, we observe that problem (9)-(12) is the deterministic linear programming formulation for a capacity allocation problem where there is limited capacity only on day i and the capacities on all of the other days are infinite. In this capacity allocation problem, if we schedule a priority p job that arrives on day t for day j, then we incur a holding cost of $c_{jt}^p - \mathbf{1}(j \neq i)\,\mu_j^*$. We refer to this problem as the capacity allocation problem focused on day i. Thus, the discussion in this paragraph and Proposition 1 imply that $z_{LP}^* - \sum_{j\in\mathcal{T}\setminus\{i\}} \mu_j^*\, x_{j1}$ provides a lower bound on the optimal total expected cost in the capacity allocation problem focused on day i.

On the other hand, if there is limited capacity only on day i and the capacities on all of the other days are infinite, then the set of feasible decisions can be written as

$$\mathcal{U}_i(x_{it}, D_t) = \Big\{ u_t \in \mathbb{Z}_+^{|\mathcal{P}||\mathcal{T}|} : \sum_{p\in\mathcal{P}} u_{it}^p \le x_{it}, \quad \sum_{j\in\mathcal{T}} u_{jt}^p \le D_t^p \quad p \in \mathcal{P} \Big\}. \tag{13}$$

The definition above is similar to the one in (1), but it only pays attention to the remaining capacity on day i. In this case, if there is limited capacity only on day i and the holding cost of scheduling a priority p job that arrives on day t for day j is $c_{jt}^p - \mathbf{1}(j \neq i)\,\mu_j^*$, then we can obtain the optimal total expected cost by computing the value functions $\{v_{it}(\cdot) : t \in \mathcal{T}\}$ through the optimality equation

$$v_{it}(x_{it}) = \mathbb{E}\bigg\{ \min_{u_t \in \mathcal{U}_i(x_{it}, D_t)} \bigg\{ \sum_{j \in \mathcal{T}} \sum_{p \in \mathcal{P}} \big[c^p_{jt} - \mathbf{1}(j \neq i)\, \mu^*_j\big] u^p_{jt} + v_{i,t+1}(x_{it} - \sum_{p \in \mathcal{P}} u^p_{it}) \bigg\} \bigg\} \tag{14}$$

with the boundary condition that $v_{i,\tau+1}(\cdot) = 0$. The optimality equation in (14) is similar to the one in (2), but it keeps track of the remaining capacity only on day i. This is emphasized by the subscript i in the value functions. The discussion in the previous paragraph implies that $z^*_{LP} - \sum_{j \in \mathcal{T} \setminus \{i\}} \mu^*_j \, x_{j1} \leq v_{i1}(x_{i1})$. Furthermore, the next proposition shows that $v_{i1}(x_{i1}) + \sum_{j \in \mathcal{T} \setminus \{i\}} \mu^*_j \, x_{j1} \leq V_1(x_1)$ and we obtain the chain of inequalities

$$z^*_{LP} \leq v_{i1}(x_{i1}) + \sum_{j \in \mathcal{T} \setminus \{i\}} \mu^*_j \, x_{j1} \leq V_1(x_1).$$

Therefore, we can solve the optimality equation in (14) to obtain a lower bound on the optimal total expected cost in the original capacity allocation problem and this lower bound is at least as tight as the one provided by the optimal objective value of problem (3)-(6).

Proposition 2 *We have $v_{it}(x_{it}) + \sum_{j \in \mathcal{T} \setminus \{i\}} \mu^*_j \, x_{jt} \leq V_t(x_t)$ for all $i \in \mathcal{T}$, $t \in \mathcal{T}$.*

Proof Since we have $v_{i,\tau+1}(\cdot) = V_{\tau+1}(\cdot) = 0$ for all $i \in \mathcal{T}$ and $\mu^*_j \leq 0$ for all $j \in \mathcal{T}$ by dual feasibility to problem (3)-(6), the result holds for day $\tau + 1$. We assume that the result holds for day $t+1$ and let $\{\hat{u}^p_{jt} : p \in \mathcal{P}, \ j \in \mathcal{T}\}$ be an optimal solution to the optimization problem inside the expectation in (2), $\vartheta_t(x_t, D_t)$ be the optimal objective value of this problem and $\vartheta_{it}(x_{it}, D_t)$ be the optimal objective value of the optimization problem inside the expectation in (14). We have

$$\begin{aligned}
\vartheta_t(x_t, D_t) &= \sum_{j \in \mathcal{T}} \sum_{p \in \mathcal{P}} c^p_{jt} \, \hat{u}^p_{jt} + V_{t+1}(x_t - \sum_{j \in \mathcal{T}} \sum_{p \in \mathcal{P}} \hat{u}^p_{jt} \, e_j) \\
&\geq \sum_{j \in \mathcal{T}} \sum_{p \in \mathcal{P}} c^p_{jt} \, \hat{u}^p_{jt} + v_{i,t+1}(x_{it} - \sum_{p \in \mathcal{P}} \hat{u}^p_{it}) + \sum_{j \in \mathcal{T} \setminus \{i\}} \mu^*_j \Big[x_{jt} - \sum_{p \in \mathcal{P}} \hat{u}^p_{jt} \Big] \\
&= \sum_{j \in \mathcal{T}} \sum_{p \in \mathcal{P}} \big[c^p_{jt} - \mathbf{1}(j \neq i)\, \mu^*_j\big] \hat{u}^p_{jt} + v_{i,t+1}(x_{it} - \sum_{p \in \mathcal{P}} \hat{u}^p_{it}) + \sum_{j \in \mathcal{T} \setminus \{i\}} \mu^*_j \, x_{jt} \geq \vartheta_{it}(x_{it}, D_t) + \sum_{j \in \mathcal{T} \setminus \{i\}} \mu^*_j \, x_{jt},
\end{aligned}$$

where the first inequality follows from the induction assumption and the second inequality follows from the fact that $\{\hat{u}^p_{jt} : p \in \mathcal{P}, \ j \in \mathcal{T}\}$ is a feasible but not necessarily an optimal solution to the optimization problem inside the expectation in (14). The result follows by taking expectations in the chain of inequalities above and noting that $V_t(x_t) = \mathbb{E}\{\vartheta_t(x_t, D_t)\}$ and $v_{it}(x_{it}) = \mathbb{E}\{\vartheta_{it}(x_{it}, D_t)\}$. $\Box$

Proposition 2 suggests using the lower bound $v_{it}(x_{it}) + \sum_{j \in \mathcal{T} \setminus \{i\}} \mu^*_j \, x_{jt}$ as an approximation to $V_t(x_t)$. In this approximation, given that we are on day t, the component $v_{it}(x_{it})$ captures the value of the remaining capacity on day i, whereas the component $\sum_{j \in \mathcal{T} \setminus \{i\}} \mu^*_j \, x_{jt}$ captures the values of the remaining capacities on days other than day i. We note that the component $\sum_{j \in \mathcal{T} \setminus \{i\}} \mu^*_j \, x_{jt}$ is somewhat trivial in the sense that it exactly corresponds to how the LVF decision rule assesses the values of the remaining capacities on days other than day i. To obtain more sophisticated estimates for the values of the remaining capacities on days other than day i, we propose computing $\{v_{it}(\cdot) : t \in \mathcal{T}\}$ for all $i \in \mathcal{T}$. In this case, given that we are on day t, the component $v_{it}(x_{it})$ captures the value of the remaining capacity on day i and we can use $\sum_{i \in \mathcal{T}} v_{it}(x_{it})$ as an approximation to $V_t(x_t)$. In the next section, we resolve the computational issues that are related to solving the optimality equation in (14).

4 Solving the Optimality Equation

The value functions $\{v_{it}(\cdot) : t \in \mathcal{T}\}$ computed through the optimality equation in (14) involve a scalar state variable. Therefore, it is tractable to store these value functions. However, solving the optimality equation in (14) still requires dealing with an optimization problem with $|\mathcal{P}||\mathcal{T}|$ decision variables and computing an expectation that involves the high dimensional random variable D_t, which are nontrivial tasks. In this section, we resolve the computational issues that are related to solving the optimality equation in (14). We begin with the next proposition, which shows that the optimization problem inside the expectation in (14) can be solved efficiently.

Proposition 3 *We can solve the optimization problem inside the expectation in* (14) *by a sort operation.*

Proof As mentioned above, the optimality equation in (14) assumes that there is limited capacity only on day i. Therefore, there are three things we can do with a job. We can schedule it for day i, schedule it for a day other than day i or reject it. If we decide to schedule the job for a day other than day i or reject it, then we simply follow the option with the smallest cost since there is no limit on the number of jobs we can schedule for a day other than day i or reject. To make the idea precise, we let $\bar{K} = \max_{j \in \mathcal{T}}\{x_{j1}\}$ so that the remaining capacities never exceed $\bar{K}$. By using induction over the days of the planning horizon, we can show that $\{v_{it}(\cdot) : t \in \mathcal{T}\}$ are convex in the sense that $2\, v_{it}(k) \leq v_{it}(k+1) + v_{it}(k-1)$ for all $k = 1, \ldots, \bar{K}-1$, $t \in \mathcal{T}$. Thus, letting $\mathcal{K} = \{0, \ldots, \bar{K}-1\}$ and $\Delta^k_{i,t+1} = v_{i,t+1}(k+1) - v_{i,t+1}(k)$ for all $k \in \mathcal{K}$, we associate the decision variables $\{w^k_{i,t+1} : k \in \mathcal{K}\}$ with the first differences $\{\Delta^k_{i,t+1} : k \in \mathcal{K}\}$ and write the optimization problem inside the expectation in (14) as

$$\begin{aligned}
\min \quad & \sum_{j \in \mathcal{T}} \sum_{p \in \mathcal{P}} \left[c^p_{jt} - \mathbf{1}(j \neq i)\, \mu^*_j\right] u^p_{jt} + \sum_{k \in \mathcal{K}} \Delta^k_{i,t+1}\, w^k_{i,t+1} + \ v_{i,t+1}(0) \\
\text{subject to} \quad & \sum_{p \in \mathcal{P}} u^p_{it} + \sum_{k \in \mathcal{K}} w^k_{i,t+1} = x_{it} \\
& \sum_{j \in \mathcal{T}} u^p_{jt} + y^p_t = D^p_t && p \in \mathcal{P} \\
& w^k_{i,t+1} \leq 1 && k \in \mathcal{K} \\
& u^p_{jt}, y^p_t, w^k_{i,t+1} \in \mathbb{Z}_+ && p \in \mathcal{P},\ j \in \mathcal{T},\ k \in \mathcal{K},
\end{aligned}$$

where we use the slack variables $\{y^p_t : p \in \mathcal{P}\}$ in the second set of constraints above.

For a particular priority level p, we consider the decision variables $\{u^p_{jt} : j \in \mathcal{T} \setminus \{i\}\}$ and y^p_t in the problem above. As far as the constraints are concerned, these decision variables appear only in the second set of constraints. Therefore, if any one of the decision variables $\{u^p_{jt} : j \in \mathcal{T} \setminus \{i\}\}$

and y_t^p takes a nonzero value in the optimal solution, then this decision variable has to be the one with the smallest objective function coefficient. This implies that we can replace all of these decision variables with a single decision variable, say z_t^p, and the objective function coefficient of z_t^p would be equal to the smallest of the objective function coefficients of the decision variables $\{u_{jt}^p : j \in \mathcal{T} \setminus \{i\}\}$ and y_t^p. Since the objective function coefficients of the decision variables $\{u_{jt}^p : j \in \mathcal{T} \setminus \{i\}\}$ are $\{c_{jt}^p - \mu_j^* : j \in \mathcal{T} \setminus \{i\}\}$ and the objective function coefficient of the decision variable y_t^p is zero, letting $\hat{c}_t^p = \min\{\min\{c_{jt}^p - \mu_j^* : j \in \mathcal{T} \setminus \{i\}\}, 0\}$, the last problem above becomes

$$\begin{aligned}
\min \quad & \sum_{p \in \mathcal{P}} c_{it}^p \, u_{it}^p + \sum_{p \in \mathcal{P}} \hat{c}_t^p \, z_t^p + \sum_{k \in \mathcal{K}} \Delta_{i,t+1}^k \, w_{i,t+1}^k + \; v_{i,t+1}(0) && && (15)\\
\text{subject to} \quad & \sum_{p \in \mathcal{P}} u_{it}^p + \sum_{k \in \mathcal{K}} w_{i,t+1}^k = x_{it} && && (16)\\
& u_{it}^p + z_t^p = D_t^p && p \in \mathcal{P} && (17)\\
& w_{i,t+1}^k \le 1 && k \in \mathcal{K} && (18)\\
& u_{it}^p, z_t^p, w_{i,t+1}^k \in \mathbb{Z}_+ && p \in \mathcal{P}, \; k \in \mathcal{K}. && (19)
\end{aligned}$$

We note that the decision variable z_t^p corresponds to the least cost option for a priority p job among the options of scheduling this job for a day other than day i and rejecting it. Using constraints (17), we write $z_t^p = D_t^p - u_{it}^p$ for all $p \in \mathcal{P}$ and plug them into the objective function to obtain

$$\begin{aligned}
& \sum_{p \in \mathcal{P}} c_{it}^p \, u_{it}^p + \sum_{p \in \mathcal{P}} \hat{c}_t^p \big[D_t^p - u_{it}^p \big] + \sum_{k \in \mathcal{K}} \Delta_{i,t+1}^k \, w_{i,t+1}^k + \; v_{i,t+1}(0) \\
& \qquad = \sum_{p \in \mathcal{P}} \big[c_{it}^p - \hat{c}_t^p \big] u_{it}^p + \sum_{p \in \mathcal{P}} \hat{c}_t^p \, D_t^p + \sum_{k \in \mathcal{K}} \Delta_{i,t+1}^k \, w_{i,t+1}^k + \; v_{i,t+1}(0).
\end{aligned}$$

Therefore, dropping the constant term $\sum_{p \in \mathcal{P}} \hat{c}_t^p \, D_t^p + v_{i,t+1}(0)$ from the objective function, problem (15)-(19) can be written as

$$\begin{aligned}
\min \quad & \sum_{p \in \mathcal{P}} \big[c_{it}^p - \hat{c}_t^p \big] \, u_{it}^p + \sum_{k \in \mathcal{K}} \Delta_{i,t+1}^k \, w_{i,t+1}^k && && (20)\\
\text{subject to} \quad & \sum_{p \in \mathcal{P}} u_{it}^p + \sum_{k \in \mathcal{K}} w_{i,t+1}^k = x_{it} && && (21)\\
& u_{it}^p \le D_t^p && p \in \mathcal{P} && (22)\\
& w_{i,t+1}^k \le 1 && k \in \mathcal{K} && (23)\\
& u_{it}^p, w_{i,t+1}^k \in \mathbb{Z}_+ && p \in \mathcal{P}, \; k \in \mathcal{K}. && (24)
\end{aligned}$$

Problem (20)-(24) is a knapsack problem with each item consuming one unit of capacity. The items are indexed by $p \in \mathcal{P}$ and $k \in \mathcal{K}$. The (dis)utilities of items $p \in \mathcal{P}$ and $k \in \mathcal{K}$ are respectively $c_{it}^p - \hat{c}_t^p$ and $\Delta_{i,t+1}^k$. The capacity of the knapsack is x_{it}. We can put at most D_t^p units of item $p \in \mathcal{P}$ and one unit of item $k \in \mathcal{K}$ into the knapsack. The result follows by the fact that a knapsack problem with each item consuming one unit of capacity can be solved by sorting the objective function coefficients and filling the knapsack starting from the item with the smallest objective function coefficient. $\Box$

Another difficulty with the optimality equation in (14) is that we need to compute the expectation of the optimal objective value of the optimization problem inside the first set of curly brackets. The proof of Proposition 3 shows that this optimization problem is equivalent to problem (20)-(24), which is, in turn, a knapsack problem where each item consumes one unit of capacity and there is a random upper bound on how many units of a particular item we can put into the knapsack. Powell and Cheung (1994) derive a closed form expression for the expectation of the optimal objective value of such a knapsack problem. We briefly describe how their result relates to our setting.

To reduce the notational clutter, we note that for appropriate choices of the set of items $\mathcal{N} = \{1, \ldots, N\}$, (dis)utilities $\{\beta_n : n \in \mathcal{N}\}$, integer valued knapsack capacity Q and integer valued random upper bounds $\alpha = \{\alpha_n : n \in \mathcal{N}\}$, problem (20)-(24) is a knapsack problem of the form

$$\begin{aligned} \min \quad & \sum_{n \in \mathcal{N}} \beta_n \, z_n && && (25) \\ \text{subject to} \quad & \sum_{n \in \mathcal{N}} z_n = Q && && (26) \\ & z_n \leq \alpha_n && n \in \mathcal{N} && (27) \\ & z_n \in \mathbb{Z}_+ && n \in \mathcal{N}. && (28) \end{aligned}$$

Using $\zeta(\alpha)$ to denote the optimal objective value of the problem above as a function of α, we are interested in computing $\mathbb{E}\{\zeta(\alpha)\}$. For a given realization of α, we can solve problem (25)-(28) by sorting the objective function coefficients of the decision variables and filling the knapsack starting from the item with the smallest objective function coefficient. We assume that $\beta_1 \leq \beta_2 \leq \ldots \leq \beta_N$, in which case it is optimal to start from the item with the smallest index.

We let $\phi(n, q)$ be the probability that the nth item uses the qth unit of the available knapsack capacity in the optimal solution. If we know $\phi(n, q)$ for all $n \in \mathcal{N}, \; q = 1, \ldots, Q$, then we can compute the expectation of the optimal objective value of problem (25)-(28) as

$$\mathbb{E}\{\zeta(\alpha)\} = \sum_{q=1}^{Q} \sum_{n \in \mathcal{N}} \beta_n \, \phi(n, q).$$

Computing $\phi(n, q)$ turns out to be not too difficult. Since the optimal solution fills the knapsack starting from the item with the smallest index, for the nth item to use the qth unit of capacity in the knapsack, the total capacity consumed by the first $n - 1$ items should be strictly less than q and the total capacity consumed by the first n items should be greater than or equal to q. Therefore, we have $\phi(n, q) = \mathbb{P}\{\alpha_1 + \ldots + \alpha_{n-1} < q \leq \alpha_1 + \ldots + \alpha_n\}$ and we can compute $\phi(n, q)$ as long as we can compute the convolutions of the distributions of $\{\alpha_n : n \in \mathcal{N}\}$.

The discussion in the previous two paragraphs provides a method to compute the expectation of the optimal objective value of the optimization problem inside the first set of curly brackets in (14). In the next section, we describe how we can use the value functions computed through the optimality equation in (14) to make the job scheduling decisions on each day.

5 Applying the Greedy Policies

At the end of Section 3, we propose using $\sum_{j \in \mathcal{T}} v_{jt}(x_{jt})$ as an approximation to $V_t(x_t)$. In this case, if the state of the remaining capacities and the job arrivals on day t are respectively given by x_t and D_t, then we can replace $V_{t+1}(x_t - \sum_{j\in\mathcal{T}} \sum_{p\in\mathcal{P}} u^p_{jt}\, e_j)$ on the right side of (2) with $\sum_{j\in\mathcal{T}} v_{j,t+1}(x_{jt} - \sum_{p\in\mathcal{P}} u^p_{jt})$ and solve the problem

$$\min_{u_t\in\mathcal{U}(x_t,D_t)} \left\{ \sum_{j\in\mathcal{T}} \sum_{p\in\mathcal{P}} c^p_{jt}\, u^p_{jt} + \sum_{j\in\mathcal{T}} v_{j,t+1}(x_{jt} - \sum_{p\in\mathcal{P}} u^p_{jt}) \right\} \tag{29}$$

to make the decisions on day t. We refer to this decision rule as the DPD decision rule, standing for dynamic programming decomposition. The problem above involves $|\mathcal{P}||\mathcal{T}|$ decision variables. The next proposition shows that we can efficiently solve this problem as a min cost network flow problem.

Proposition 4 *We can solve problem* (29) *as a min cost network flow problem.*

Proof As mentioned in the proof of Proposition 3, it is possible to show that $\{v_{jt}(\cdot) : j \in \mathcal{T},\ t \in \mathcal{T}\}$ are convex functions. In this case, letting $\mathcal{K}$ and $\Delta^k_{j,t+1}$ be defined as in the proof of Proposition 3, we associate the decision variables $\{w^k_{j,t+1} : k \in \mathcal{K},\ j \in \mathcal{T}\}$ with the first differences $\{\Delta^k_{j,t+1} : k \in \mathcal{K},\ j \in \mathcal{T}\}$ and write problem (29) as

$$\begin{aligned}
\min \quad & \sum_{j\in\mathcal{T}} \sum_{p\in\mathcal{P}} c^p_{jt}\, u^p_{jt} + \sum_{j\in\mathcal{T}} \sum_{k\in\mathcal{K}} \Delta^k_{j,t+1}\, w^k_{j,t+1} + \sum_{j\in\mathcal{T}} v_{j,t+1}(0) \\
\text{subject to} \quad & \sum_{p\in\mathcal{P}} u^p_{jt} + \sum_{k\in\mathcal{K}} w^k_{j,t+1} = x_{jt} && j \in \mathcal{T} \\
& \sum_{j\in\mathcal{T}} u^p_{jt} \le D^p_t && p \in \mathcal{P} \\
& w^k_{j,t+1} \le 1 && k \in \mathcal{K},\ j \in \mathcal{T} \\
& u^p_{jt}, w^k_{j,t+1} \in \mathbb{Z}_+ && p \in \mathcal{P},\ k \in \mathcal{K},\ j \in \mathcal{T}.
\end{aligned}$$

We define the new decision variables $\{y^p_t : p \in \mathcal{P}\}$ as $y^p_t = \sum_{j\in\mathcal{T}} u^p_{jt}$ for all $p \in \mathcal{P}$, in which case the problem above becomes

$$\begin{aligned}
\min \quad & \sum_{j\in\mathcal{T}} \sum_{p\in\mathcal{P}} c^p_{jt}\, u^p_{jt} + \sum_{j\in\mathcal{T}} \sum_{k\in\mathcal{K}} \Delta^k_{j,t+1}\, w^k_{j,t+1} + \sum_{j\in\mathcal{T}} v_{j,t+1}(0) && && (30) \\
\text{subject to} \quad & \sum_{p\in\mathcal{P}} u^p_{jt} + \sum_{k\in\mathcal{K}} w^k_{j,t+1} = x_{jt} && j \in \mathcal{T} && (31) \\
& \sum_{j\in\mathcal{T}} u^p_{jt} - y^p_t = 0 && p \in \mathcal{P} && (32) \\
& y^p_t \le D^p_t && p \in \mathcal{P} && (33) \\
& w^k_{j,t+1} \le 1 && k \in \mathcal{K},\ j \in \mathcal{T} && (34) \\
& u^p_{jt}, y^p_t, w^k_{j,t+1} \in \mathbb{Z}_+ && p \in \mathcal{P},\ k \in \mathcal{K},\ j \in \mathcal{T}. && (35)
\end{aligned}$$

Ignoring the constant term $\sum_{j\in\mathcal{T}} v_{j,t+1}(0)$ in the objective function, the problem above is a min cost network flow problem. To see this, we consider a network composed of two sets of nodes $\mathcal{N}_1 = \{j \in \mathcal{T}\}$ and $\mathcal{N}_2 = \{p \in \mathcal{P}\}$, along with a sink node. The decision variable u^p_{jt} corresponds to an arc connecting node $j \in \mathcal{N}_1$ to node $p \in \mathcal{N}_2$, the decision variable y^p_t corresponds to an arc connecting node $p \in \mathcal{N}_2$ to the sink node and the decision variable $w^k_{j,t+1}$ corresponds to an arc connecting node $j \in \mathcal{N}_1$ to the sink node. The supply of node $j \in \mathcal{N}_1$ is x_{jt} and constraints (31) are the flow balance constraints for the nodes in $\mathcal{N}_1$. The supply of node $p \in \mathcal{N}_2$ is zero and constraints (32) are the flow balance constraints for the nodes in $\mathcal{N}_2$. The flow balance constraint for the sink node is redundant and omitted. Constraints (33) and (34) act as simple upper bounds on the arcs. Figure 1 shows the structure of the network for the case where $\mathcal{T} = \{1, 2\}$, $\mathcal{P} = \{a, b\}$ and $\mathcal{K} = \{l, m\}$. $\Box$

Since problem (29) can be solved as a min cost network flow problem, its continuous relaxation naturally provides integer solutions. Therefore, we can solve problem (29) as a linear program.

6 Computational Experiments

In this section, we numerically compare the performance of the separable value function approximations constructed through the method in Section 3 with a number of benchmark strategies.

6.1 Experimental Setup

We present two sets of computational experiments. In the first set of computational experiments, we generate a base problem and modify its certain attributes to obtain test problems with different characteristics. In the base problem, the number of days in the planning horizon is $|\mathcal{T}| = 100$ and the set of priority levels is $\mathcal{P} = \{1, 2, 3\}$. We assume that it is not possible to schedule a job more than S days into the future so that $h^p_{jt} = \infty$ whenever we have $j - t \geq S$. We use the holding costs $h^p_{jt} = \phi^p \, 1.25^{j-t}$ and the penalty costs $r^p_t = \beta \, \phi^p \, 1.25^{S-1}$ for all $p \in \mathcal{P}$, $t \in \mathcal{T}$ and $t \leq j \leq t + S - 1$. We note that the priority level one has the lowest priority since it incurs the lowest holding and penalty costs. In the base problem, we use $S = 7$, $\phi = 2$ and $\beta = 5$. The expected numbers of daily job arrivals for the first, second and third priority levels are respectively 40, 20 and 10 and the coefficient of variation for the numbers of daily job arrivals is CV = 0.3. The daily processing capacity is 70 so that the daily expected job arrivals match the daily processing capacity.

In the second set of computational experiments, we work with test problems whose cost structures are inspired by the health care application in Patrick et al. (2008). In particular, we assume that there is a target duration for each priority level. If we can schedule a job within the target duration corresponding to its priority level, then we do not incur a holding cost. Otherwise, we incur a holding cost of f^p for each day that we run over the target duration for a priority p job. Similar to the first set of computational experiments, we assume that it is not possible to schedule a job more than S days into the future. Therefore, using b^p to denote the target duration for a priority p job, the holding costs are of the form

$$h^p_{jt} = \begin{cases} \infty & \text{if } j \leq t - 1 \\ 0 & \text{if } t \leq j \leq t + b^p - 1 \\ (j - t - b^p + 1)\, f^p & \text{if } t + b^p \leq j \leq t + S - 1 \\ \infty & \text{if } t + S \leq j \end{cases}$$

for all $p \in \mathcal{P}$, $j, t \in \mathcal{T}$. In the expression above, the first case corresponds to scheduling a job for a day in the past and the fourth case corresponds to scheduling a job more than S days into the future, which are both infeasible. The second case corresponds to scheduling a job within its target duration and the third case corresponds to scheduling a job beyond its target duration. The penalty costs are given by $r_t^p = (S - b^p + 1)\, f^p$ for all $p \in \mathcal{P},\ t \in \mathcal{T}$. Throughout the second set of computational experiments, the set of priority levels is $\mathcal{P} = \{1, 2, 3\}$ and we use $|\mathcal{T}| = 100$ and $S = 11$. Similar to the first set of computational experiments, the expected numbers of daily job arrivals for the first, second and third priority levels are respectively 40, 20 and 10 and the coefficient of variation for the numbers of daily job arrivals is CV = 0.3. We generate six test problems by using different combinations for $f = (f^1, f^2, f^3)$ and $b = (b^1, b^2, b^3)$. In Table 1, we show the values that we use for f and b to generate the six test problems in the second set of computational experiments.

6.2 Benchmark Strategies

We compare the DPD decision rule introduced in Section 5 with three benchmark strategies. Our first benchmark strategy is the LVF decision rule that we describe at the end of Section 2. In contrast to the basic version of this benchmark strategy, we test the performance of a dynamic version that updates the values of dual variables $\{\mu_j^* : j \in \mathcal{T}\}$ periodically over time. In particular, we divide the planning horizon into M equal segments and resolve problem (3)-(6) on days $t_m = 1 + (m-1)\tau/M$ for $m = 1, \ldots, M$. On day t_m, we replace the set of days $\mathcal{T}$ in problem (3)-(6) with $\mathcal{T}_m = \{t_m, \ldots, \tau\}$ and the right side of constraints (4) with the current remaining capacities $\{x_{j,t_m} : j \in \mathcal{T}_m\}$. In this way, we only focus on the remaining portion of the planning horizon with the current values for the remaining capacities. Letting $\{\mu_j^* : j \in \mathcal{T}_m\}$ be the optimal values of the dual variables associated with constraints (4), we use these dual variables in problem (8) until we reach the beginning of the next segment and solve problem (3)-(6) again. We use $M = 5$ in all of our test problems.

To motivate our second benchmark strategy, we note that the LVF decision rule uses the dual solution of problem (3)-(6). Naturally, it is possible to use the primal solution to problem (3)-(6) to make the job scheduling decisions. In particular, given that we are on day t, we replace the set of days $\mathcal{T}$ in problem (3)-(6) with $\{t, \ldots, \tau\}$ and the right side of constraints (4) with the current remaining capacities $\{x_{jt} : j = t, \ldots, \tau\}$. Furthermore, we replace the right side of constraints (5) for day t with the current realizations of the job arrivals. For the other days, we continue using the expected values of the job arrivals. In this way, we only focus on the remaining portion of the planning horizon with the current values for the remaining capacities and the current realization of the job arrivals for day t. After these modifications, we solve problem (3)-(6) and letting $\{\hat{u}_{jt'}^p : p \in \mathcal{P},\ j, t' = t, \ldots, \tau\}$ be an optimal solution, we apply the decisions given by $\{\hat{u}_{jt}^p : p \in \mathcal{P},\ j = t, \ldots, \tau\}$ on day t. We refer to this benchmark strategy as the PLP decision rule, standing for primal linear program.

The third benchmark strategy is a simple first come first serve policy. In particular, this benchmark strategy schedules the job arrivals on a particular day starting from the jobs with the highest priority, without reserving capacity for the future job arrivals. We do not expect this benchmark strategy to perform well, but we use it to demonstrate the benefit from carefully reserving capacity for the future job arrivals. We refer to this benchmark strategy as the FC decision rule, standing for first come.

6.3 Computational Results

We begin by presenting our results for the first set of computational experiments. In particular, Table 2 shows the results for the base problem in the first set of computational experiments. The second column in this table gives the lower bound on the optimal total expected cost provided by the optimal objective value of problem (3)-(6). The third column gives the lower bound on the optimal total expected cost provided by the expectation of the perfect hindsight total cost. Using the notation in the proof of Proposition 1, the two lower bounds in the second and third columns respectively correspond to $z^*_{LP}(\mathbb{E}\{D\})$ and $\mathbb{E}\{z^*_{LP}(D)\}$. Proposition 2 shows that the DPD decision rule also provides a lower bound on the optimal total expected cost, but this lower bound turns out to be only marginally better than the one provided by the optimal objective value of problem (3)-(6) and we do not report this bound. The fourth, fifth, sixth and seventh columns in Table 2 respectively give the total expected costs incurred by the DPD, LVF, PLP and FC decision rules. We estimate these total expected costs by simulating the performances of the different benchmark strategies under 100 job arrival trajectories with common random numbers. The eighth, ninth, tenth and eleventh columns give the percentage of jobs rejected by the four benchmark strategies. The twelfth column gives the percent gap between the total expected costs incurred by the DPD and LVF decision rules. Similarly, the thirteenth and fourteenth columns give the percent performance gaps between the DPD decision rule and the remaining two benchmark strategies. The last column gives the percent gap between the total expected cost incurred by the DPD decision rule and the tightest of the lower bounds reported in the second and third columns. Thus, the last column gives an upper bound on the optimality gap of the DPD decision rule.

Table 2 suggests that the DPD decision rule performs significantly better than all of the benchmark strategies for the base problem. The performance gaps between the DPD decision rule and the two linear programming based decision rules is 4 to 8%. It is interesting that although the lower bound provided by the DPD decision rule does not improve the one provided by problem (3)-(6) by much, the policy provided by the DPD decision rule performs significantly better than the policies that are based on problem (3)-(6). The FC decision rule performs substantially worse, with a performance gap of about 18%. In the proof of Proposition 1, we show that $z^*_{LP}(\mathbb{E}\{D\}) \leq \mathbb{E}\{z^*_{LP}(D)\}$ and the second and third columns indicate that the gap between these two lower bounds can be significant. Therefore, estimating the lower bound $\mathbb{E}\{z^*_{LP}(D)\}$ by using Monte Carlo samples can be worth the effort.

In Tables 3 through 8, we vary different attributes of the base problem to generate test problems with different characteristics. The first columns in these tables indicate the parameter that we modify and its value. The interpretations of other columns remain the same as in Table 2. In Table 3, we begin by varying the number of priority levels. The DPD decision rule performs consistently better than all of the benchmark strategies. The performance gap between the DPD and PLP decision rules tend to decrease with the increasing number of priority levels, but the performance gap is still about 5% even when we have five priority levels. The FC decision rule rejects the smallest number of jobs, but this does not translate into better performance.

In Table 4, we vary how far we are allowed to look into the future when scheduling the jobs. The DPD decision rule continues to provide improvements over the other benchmark strategies. For the longest booking horizon, the PLP decision rule outperforms the DPD decision rule by a slight margin, but the performance of the PLP decision rule is quite erratic in the sense that there are test problems where this benchmark strategy performs quite poorly. In Table 5, we vary the coefficient of variation for the job arrivals. We observe that the performance gaps between the DPD decision rule and the LVF and PLP decision rules increase as the coefficient of variation increases. This observation is encouraging since addressing the random nature of the job arrivals is one of the main goals behind the DPD decision rule. Since the lower bound $z^*_{LP}(\mathbb{E}\{D\})$ only uses the expected values of the job arrivals, its value does not depend on the coefficient of variation, but the lower bound $\mathbb{E}\{z^*_{LP}(D)\}$ gets larger as the coefficient of variation increases. This increase in the lower bound is in agreement with the intuitive expectation that the total expected cost incurred by the optimal policy should increase as the job arrivals have more uncertainty and it becomes more difficult to anticipate the job arrivals.

In Table 6, we vary the daily available capacity. If there is plenty of daily available capacity, then it is relatively straightforward to have capacity to accommodate the high priority jobs and the problem becomes less challenging. Therefore, for the test problems with large capacities, all of the benchmark strategies perform similarly. However, the DPD decision rule can provide noticeable improvements when the capacities become tight. For the test problem with the tightest capacities, the performance gap between the DPD and FC decision rules is about 60%. Finally, in Tables 7 and 8, we vary the holding and penalty costs. We recall that we work with holding costs of the form $h^p_{jt} = \phi^p \, 1.25^{j-t}$ and penalty costs of the form $r^p_t = \beta \, \phi^p \, 1.25^{S-1}$. The base problem corresponds to the case where we have $S = 7$, $\phi = 2$ and $\beta = 5$. We observe that the DPD decision rule generally outperforms all of the other benchmark strategies with noticeable margins.

Table 9 shows our results for the second set of computational experiments. The interpretations of the columns of this table are very similar to those of Tables 2 through 8. For the second set of test problems, the performance of the FC decision rule turns out to be not competitive and we do not provide detailed results for this decision rule. Our results indicate that the DPD decision rule continues to perform better than the LVF and PLP decision rules. It is interesting to observe that the total expected costs incurred by the DPD decision rule for the first three test problems are very close to the lower bounds on the optimal total expected cost. Therefore, the performance of the DPD decision rule is essentially optimal for these test problems. On average, the LVF and PLP decision rules lag behind the DPD decision rule respectively by about 6% and 26%.

To sum up our observations, the DPD decision rule can provide significant improvements over the other benchmark strategies. The FC decision rule generally performs poorly as it neglects to plan for the future job arrivals. There are a few test problems where the performances of the LVF and PLP decision rules can be close to that of the DPD decision rule, but the LVF and PLP decision rules are erratic in the sense that there are quite a few test problems where their performances are substantially inferior to that of the DPD decision rule. In Table 10, we give the CPU seconds for the DPD decision rule to compute the value functions $\{v_{it}(\cdot) : t \in \mathcal{T}\}$ for all $i \in \mathcal{T}$. The left and right portions of this table respectively give the CPU seconds with different values for $|\mathcal{P}|$ and S. The results indicate that the CPU seconds scale a bit worse than quadratically with the number of priority levels, whereas increasing the number of days over which we can schedule the jobs has a somewhat less noticeable effect on the CPU seconds. The longest CPU seconds are less than a minute. Overall, since the value function approximations are computed in an offline fashion, the CPU seconds reported in Table 10 are more than adequate for practical implementation. Thus, given its superior performance, the DPD decision rule appears to be a viable approach for dynamic capacity allocation.

7 Conclusion

In this paper, we considered a capacity allocation problem that involves allocating a fixed amount of daily processing capacity to jobs of different priority levels arriving randomly over time. While finding the optimal policy would require solving a dynamic program with a high dimensional state vector, we proposed a dynamic programming decomposition method that required solving dynamic programs with scalar state variables. We showed that our dynamic programming decomposition method provides a lower bound on the optimal total expected cost and the greedy policy induced by the value function approximations can be implemented by solving min cost network flow problems. Computational experiments indicated that our dynamic programming decomposition method can provide significant improvements when compared with alternative policies.

A worthwhile extension to the problem that we study in this paper involves jobs that consume multiple units of capacity. It is relatively simple to extend our model to handle multiple units of capacity consumption. Letting a^p be the capacity consumption of a priority p job, all we need to do is to modify the capacity constraints $\sum_{p \in \mathcal{P}} u_{jt}^p \leq x_{jt}$ for all $j \in \mathcal{T}$ in (1) as $\sum_{p \in \mathcal{P}} a^p \, u_{jt}^p \leq x_{jt}$ for all $j \in \mathcal{T}$. In this case, the dynamic program in Section 1, the deterministic linear program in Section 2 and the dynamic programming decomposition method in Section 3 go through with straightforward modifications. However, if a job consumes multiple units of capacity, then Proposition 3 does not go through and we may have to solve the optimization problem inside the expectation in (14) as an integer program. Similarly, if a job consumes multiple units of capacity, then we lose the min cost network flow structure in problem (29). In this case, Proposition 4 does not go through and we may have to solve integer programs to find the job scheduling decisions made by the DPD decision rule. Therefore, if we have multiple units of capacity consumption, then our general approach goes through, but its computational tractability suffers to a certain extent and we may need to solve integer programs instead of using sort operations or solving min cost network flow problems. Finding efficient methods to solve these integer programs is an interesting research direction to pursue.

References

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

Belobaba, P. P. (1987), Air Travel Demand and Airline Seat Inventory Control, PhD thesis, Massachusetts Institute of Technology, Cambridge, MA.

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

Erdelyi, A. and Topaloglu, H. (2009*a*), 'Computing protection level policies for dynamic capacity allocation problems by using stochastic approximation methods', *IIE Transactions* **41**(6), 498–510.

Erdelyi, A. and Topaloglu, H. (2009*b*), 'A dynamic programming decomposition method for making overbooking decisions over an airline network', *INFORMS Journal on Computing* (to appear).

Gerchak, Y., Gupta, D. and Henig, M. (1996), 'Reservation planning for elective surgery under uncertain demand for emergency surgery', *Management Science* **42**, 321–334.

Gupta, D. and Wang, L. (2008), 'Revenue management for a primary-care clinic in presence of patient choice', *Operations Research* **56**(3), 576–592.

Kunnumkal, S. and Topaloglu, H. (2009), 'A new dynamic programming decomposition method for the network revenue management problem with customer choice behavior', *Production and Operations Management* (to appear).

Liu, Q. and van Ryzin, G. (2008), 'On the choice-based linear programming model for network revenue management', *Manufacturing & Service Operations Management* **10**(2), 288–310.

Patrick, J., Puterman, M. and Queyranne, M. (2008), 'Dynamic multi-priority patient scheduling for a diagnostic resource', *Operations Research* **56**(6), 1507–1525.

Powell, W. B. (2007), *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, John Wiley & Sons, Hoboken, NJ.

Powell, W. B. and Cheung, R. K. (1994), 'Stochastic programs over trees with random arc capacities', *Networks* **24**, 161–175.

Ruszczynski, A. (2003), Decomposition methods, *in* A. Ruszczynski and A. Shapiro, eds, '*Handbook in Operations Research and Management Science*, Volume on *Stochastic Programming*', North Holland, Amsterdam.

Talluri, K. T. and van Ryzin, G. J. (2005), *The Theory and Practice of Revenue Management*, Springer, New York, NY.

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

Williamson, E. L. (1992), Airline Network Seat Control, PhD thesis, Massachusetts Institute of Technology, Cambridge, MA.

Zhang, D. (2010), 'An improved dynamic programming decomposition approach for network revenue management', *Manufacturing & Service Operations Management* (forthcoming).

Zhang, D. and Adelman, D. (2009), 'An approximate dynamic programming approach to network revenue management with customer choice', *Transportation Science* **42**(3), 381–394.

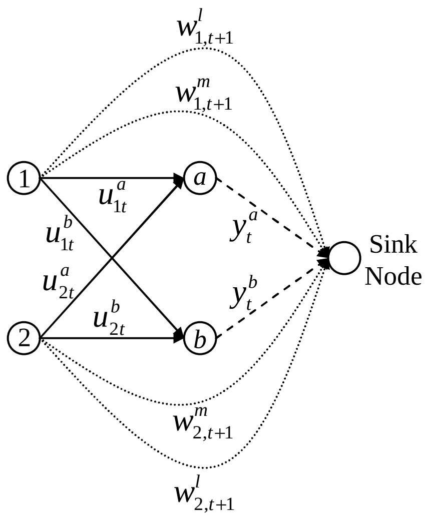

Figure 1: Problem (29) as a min cost network flow problem when we have $\mathcal{T} = \{1, 2\}$, $\mathcal{P} = \{a, b\}$ and $\mathcal{K} = \{l, m\}$.

Test Prb.	(f^1, f^2, f^3)	(b^1, b^2, b^3)
1	$(5, 10, 20)$	$(3, 3, 3)$
2	$(5, 10, 20)$	$(5, 5, 5)$
3	$(5, 10, 20)$	$(7, 7, 7)$
4	$(20, 20, 20)$	$(5, 4, 3)$
5	$(20, 20, 20)$	$(7, 5, 3)$
6	$(20, 20, 20)$	$(9, 6, 3)$

Table 1: Holding cost parameters and target durations for the test problems in the second set of computational experiments.

Test	Lower Bound		Total Expected Cost				% Rejected Jobs				% Gap with DPD			DPD
Prb.	$z(\mathbb{E}\{D\})$	$\mathbb{E}\{z(D)\}$	DPD	LVF	PLP	FC	DPD	LVF	PLP	FC	LVF	PLP	FC	gap
Base	13,979	14,771	15,885	16,528	17,238	18,747	2.05	2.46	4.23	0.02	4.05	8.52	18.02	7.54

Table 2: Results for the base problem in the first set of computational experiments.

	Lower Bound		Total Expected Cost				% Rejected Jobs				% Gap with DPD			DPD
$\lvert\mathcal{P}\rvert$	$z(\mathbb{E}\{D\})$	$\mathbb{E}\{z(D)\}$	DPD	LVF	PLP	FC	DPD	LVF	PLP	FC	LVF	PLP	FC	gap
2	11,654	12,574	13,804	14,233	15,184	15,545	2.10	2.53	4.41	0.13	3.11	10.00	12.61	9.78
3	13,979	14,771	15,885	16,528	17,238	18,747	2.05	2.46	4.23	0.02	4.05	8.52	18.02	7.54
4	17,234	17,893	18,964	19,966	20,108	23,577	2.05	2.58	4.04	0.02	5.28	6.03	24.33	5.99
5	19,838	20,520	21,581	22,832	22,656	27,594	2.19	2.63	4.07	0.00	5.80	4.98	27.86	5.17

Table 3: Results with varying $|\mathcal{P}|$ in the first set of computational experiments.

	Lower Bound		Total Expected Cost				% Rejected Jobs				% Gap with DPD			DPD
S	$z(\mathbb{E}\{D\})$	$\mathbb{E}\{z(D)\}$	DPD	LVF	PLP	FC	DPD	LVF	PLP	FC	LVF	PLP	FC	gap
3	11,994	13,018	14,012	14,217	15,063	14,390	1.04	1.31	6.89	0.72	1.46	7.50	2.70	7.64
5	12,566	13,590	14,722	15,197	16,426	16,417	1.32	1.65	6.15	0.21	3.23	11.57	11.51	8.33
7	13,979	14,771	15,885	16,528	17,238	18,747	2.05	2.46	4.23	0.02	4.05	8.52	18.02	7.54
9	16,527	16,936	17,885	20,075	17,810	21,390	1.96	1.68	2.63	0.00	12.24	-0.42	19.60	5.60
11	19,977	20,374	21,578	24,424	20,972	24,448	1.19	0.84	1.87	0.00	13.19	-2.81	13.30	5.91

Table 4: Results with varying S in the first set of computational experiments.

	Lower Bound		Total Expected Cost				% Rejected Jobs				% Gap with DPD			DPD
CV	$z(\mathbb{E}\{D\})$	$\mathbb{E}\{z(D)\}$	DPD	LVF	PLP	FC	DPD	LVF	PLP	FC	LVF	PLP	FC	gap
0.0	13,979	13,979	13,979	13,979	13,979	17,867	2.40	2.40	2.40	0.00	0.00	0.00	27.81	0.00
0.1	13,979	14,154	14,411	14,637	14,992	17,891	2.23	2.41	2.89	0.00	1.57	4.03	24.15	1.82
0.2	13,979	14,380	14,991	15,434	15,985	18,099	2.16	2.42	3.42	0.00	2.96	6.63	20.73	4.25
0.3	13,979	14,771	15,885	16,528	17,238	18,747	2.05	2.46	4.23	0.02	4.05	8.52	18.02	7.54
0.4	13,979	15,222	16,909	17,718	18,549	19,515	1.96	2.52	5.16	0.16	4.78	9.70	15.41	11.08
0.5	13,979	15,810	18,325	19,251	20,135	20,455	1.93	2.70	6.33	0.41	5.05	9.88	11.62	15.91

Table 5: Results with varying CV in the first set of computational experiments.

	Lower Bound		Total Expected Cost				% Rejected Jobs				% Gap with DPD			DPD
x_{j1}	$z(\mathbb{E}\{D\})$	$\mathbb{E}\{z(D)\}$	DPD	LVF	PLP	FC	DPD	LVF	PLP	FC	LVF	PLP	FC	gap
55	34,841	34,930	35,868	37,428	36,524	57,896	18.75	20.11	20.13	18.22	4.35	1.83	61.41	2.69
65	20,023	20,365	21,588	24,688	23,828	36,237	6.92	9.33	9.59	3.58	14.36	10.38	67.86	6.01
70	13,979	14,771	15,885	16,528	17,238	18,747	2.05	2.46	4.23	0.02	4.05	8.52	18.02	7.54
75	11,858	12,218	12,305	12,746	12,265	12,515	0.00	0.00	0.00	0.00	3.58	-0.33	1.71	0.71
85	11,342	11,389	11,397	11,459	11,396	11,448	0.00	0.00	0.00	0.00	0.54	-0.01	0.45	0.07

Table 6: Results with varying x_{j1} in the first set of computational experiments.

	Lower Bound		Total Expected Cost				% Rejected Jobs				% Gap with DPD			DPD
ϕ	$z(\mathbb{E}\{D\})$	$\mathbb{E}\{z(D)\}$	DPD	LVF	PLP	FC	DPD	LVF	PLP	FC	LVF	PLP	FC	gap
1.2	10,111	10,841	11,645	11,997	13,241	12,578	2.03	2.46	4.04	0.02	3.02	13.71	8.01	7.42
2.0	13,979	14,771	15,885	16,528	17,238	18,747	2.05	2.46	4.23	0.02	4.05	8.52	18.02	7.54
3.0	20,489	21,260	22,638	24,119	23,772	29,064	2.04	2.46	4.24	0.02	6.54	5.01	28.39	6.48
4.0	28,859	29,595	31,131	33,850	32,150	42,278	2.03	2.46	4.24	0.02	8.73	3.27	35.81	5.19

Table 7: Results with varying ϕ in the first set of computational experiments.

	Lower Bound		Total Expected Cost				% Rejected Jobs				% Gap with DPD			DPD
β	$z(\mathbb{E}\{D\})$	$\mathbb{E}\{z(D)\}$	DPD	LVF	PLP	FC	DPD	LVF	PLP	FC	LVF	PLP	FC	gap
2.0	12,194	13,074	14,372	14,697	14,944	18,729	2.13	2.46	8.24	0.02	2.26	3.98	30.32	9.93
4.0	13,384	14,286	15,383	15,918	17,064	18,741	2.06	2.46	5.63	0.02	3.48	10.93	21.83	7.68
5.0	13,979	14,771	15,885	16,528	17,238	18,747	2.05	2.46	4.23	0.02	4.05	8.52	18.02	7.54
6.0	14,575	15,195	16,378	17,138	16,813	18,753	2.03	2.46	2.95	0.02	4.64	2.66	14.50	7.79
8.0	15,413	15,904	16,829	19,107	16,771	18,765	1.15	0.81	1.80	0.02	13.54	-0.34	11.50	5.82

Table 8: Results with varying β in the first set of computational experiments.

Test	Lower Bound		Total Expected Cost			% Rejected Jobs			% Gap with DPD		DPD
Prb.	$z(\mathbb{E}\{D\})$	$\mathbb{E}\{z(D)\}$	DPD	LVF	PLP	DPD	LVF	PLP	LVF	PLP	gap
1	40,987	41,004	41,013	42,365	45,373	12.42	13.42	14.40	3.30	10.63	0.02
2	27,959	28,000	28,030	28,805	35,472	10.87	11.71	14.48	2.76	26.55	0.11
3	17,220	17,227	17,326	17,618	24,946	9.65	10.01	14.25	1.69	43.98	0.57
4	141,455	141,847	149,251	163,650	191,048	11.81	12.99	15.16	9.65	28.00	4.96
5	131,695	132,352	148,785	158,937	181,486	11.80	12.60	14.40	6.82	21.98	11.04
6	114,557	115,447	142,695	158,705	177,309	11.33	12.59	14.07	11.22	24.26	19.10

Table 9: Results for the second set of computational experiments.

$\|\mathcal{P}\|$	CPU secs.
2	5.5
3	7.4
4	32.1
5	59.8

S	CPU secs.
5	5.4
7	7.4
9	10.1
11	12.8

Table 10: CPU seconds for the DPD decision rule to compute the value functions $\{v_{it}(\cdot) : t \in \mathcal{T}\}$ for all $i \in \mathcal{T}$.